# XIII ENCUENTROS DE GEOMETRÍA COMPUTACIONAL

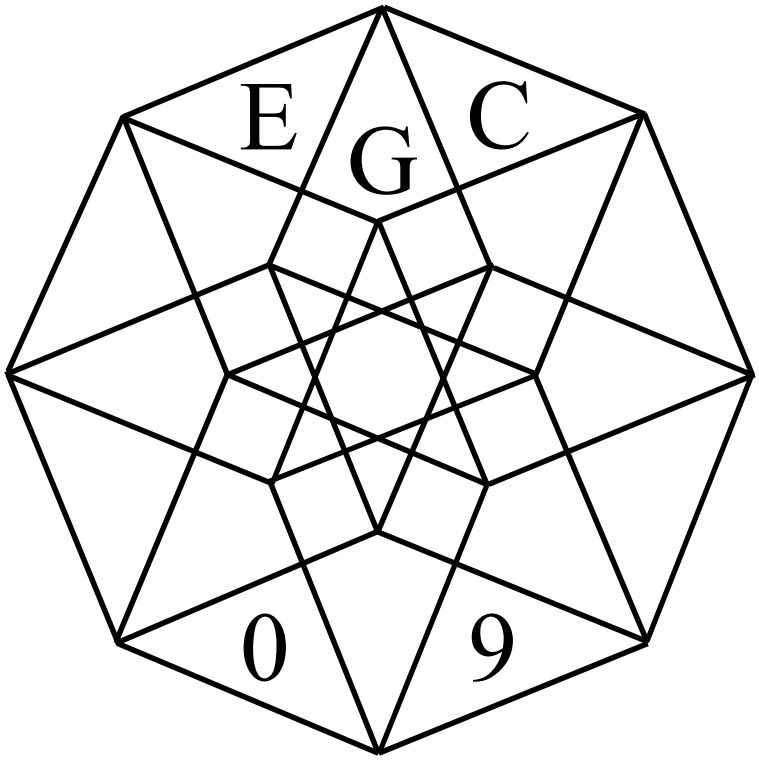
## ZARAGOZA, 29 JUNIO - 1 JULIO, 2009

ALFREDO GARCÍA
JAVIER TEJEL
(editores)

Prensas Universitarias de Zaragoza

# XIII Encuentros de Geometría Computacional

**Zaragoza, del 29 de junio al 1 de julio de 2009**




**Alfredo García**
**Javier Tejel**
**Editores**

Entidades colaboradoras







Prensas Universitarias de Zaragoza. Edificio de Ciencias Geológicas, c/ Pedro Cerbuna, 12, 50009 Zaragoza, España.
Tel.: 976 761 330. Fax: 976 761 063
puz@unizar.es http://puz.unizar.es

Prensas Universitarias de Zaragoza es la editorial de la Universidad de Zaragoza, que edita e imprime libros desde su fundación en 1542.


1.ª edición, 2009

ISBN: 978-84-92774-11-1
D.L.: Z-2581/2009
Impreso en España
Imprime: Servicio de Publicaciones de la Universidad de Zaragoza

# Presentación

Dando continuidad a la larga serie de Encuentros ya celebrados en diferentes ciudades españolas, los XIII Encuentros de Geometría Computacional se han celebrado en Zaragoza, del 29 de junio al 1 de julio de 2009. El objetivo fundamental ha sido potenciar al máximo la colaboración entre los investigadores del área de Geometría Computacional, ofreciendo un marco adecuado para la difusión de sus trabajos más recientes.

En estas actas se recogen las cinco conferencias invitadas y las veintinueve comunicaciones presentadas en dichos Encuentros. Con éllas, se pretende dejar constancia del desarrollo y consolidación de la disciplina en España, así como reflejar el creciente interés que suscitan estos Encuentros entre la comunidad internacional, especialmente entre la comunidad hispanohablante.

Desde el comité organizador queremos expresar nuestro más sincero agradecimiento a todos aquellos que con su esfuerzo y colaboración han hecho posible la realización de estos Encuentros, y muy especialmente a los participantes, sin cuyos trabajos y aportaciones hubiese sido imposible alcanzar los objetivos de estos Encuentros.

Por último, agradecer también a la Universidad de Zaragoza, al Departamento de Métodos Estadísticos, la Facultad de Ciencias y el Instituto Universitario de Matemáticas y Aplicaciones de dicha Universidad, a la Real Sociedad Matemática Española, al proyecto Ingenio Mathematica, a la Diputación General de Aragón y al Ministerio de Ciencia y Tecnología los diferentes apoyos recibidos.

*Los editores*

# Índice

# Conferencias invitadas

# On Modem Illumination Problems

R. Fabila-Monroy* Andres Ruiz Vargas ** Jorge Urrutia ***

**Resumen**

In this paper we review recent results on a new variation of the *Art Gallery* problem. A common problem we face nowadays, is that of placing a set of wireless modems in a building in such a way that a computer placed anywhere within the building receives a signal strong enough to connect to the Web. In most buildings, the main limitation for this problem is not the distance of a computer to a wireless modem, but rather the number of walls that separate them. We study variations of the following problem: Let $P$ be a simple polygon with $n$ vertices. How many points $p_1, \ldots, p_k$ (representing wireless modems) are always sufficient such that for any other point $p$ in $P$, there is a $p_i$ such that the line segment joining $p$ to $p_i$ crosses at most $k$ edges of $P$? The parameter $k$ represents the *strenght* of the signal emited by the modems. We study variations of this problem for families of line segments, families of lines, orthogonal polygons, and sets of horizontal or vertical disjoint segments, or sets of lines.

## 1. Introducción

Let $p_0, \ldots, p_{n-1}$ be a set of points on the plane, and $e_i$ the line segment joining $p_i$ to $p_{i+1}$, $i = 0, \ldots, n-1$, addition taken *mod* $n$. We say that $e_1, \ldots, e_{n-1}$ form a simple polygon $P$ if $e_i$ and $e_j$ do not intersect, except perhaps at a common end point, $i \neq j$. The elements of $\{e_0, \ldots, e_{n-1}\}$ are called the edges of $P$. All polygons considered here are simple. Abusing a bit our terminology, in this paper a polygon will refer to the closed region of the plane bounded by its edges. We say that two points $p$ and $q$ of a polygon $P$ are visible if the line segment joining them is contained in $P$. A set of points $g_1, \ldots, g_k$ guards $P$ if any point in $P$ is visible from at least one $g_i$. In 1975, V. Klee posed the following problem, known as *The Art Gallery Problem*:

*How many guards are always sufficient to guard any simple polygon with n vertices?*

The Art Gallery problem was solved by V. Chvatal [3], he proved that $\lfloor \frac{n}{3} \rfloor$ guards are always sufficient and sometimes necessary to guard any polygon with $n$ vertices.

Since then, there has been a plethora of very interesting results and variations to the original problem, the interested reader can consult [8, 11, 12]. A new variation in which the edges of our *polygons* are allowed to be arcs of convex curves has been studied in and [2, 6, 7]. In this paper we deal with a new variation of the Art Gallery Problem arising from the following everyday and practical problem: How to place wireless modems in a building in such a way that at any point within the building a laptop with a wireless card receives a signal strong enough to have a stable connection to navigate in the Web?

Experience dictates that in most buildings, the distance of our laptop to a wireless modem is not a limiting factor to obtaining a good signal, the main limiting factor seems to be the number of walls that separate us from a wireless modem.

This inspired us to study the following problem which we cast as follows: Let $k \geq 0$ be an integer, and $P$ a polygon with $n$ vertices. We say that a $k$-modem $\mathcal{M}$ represented by a point on the plane covers a point $p \in P$ if the line segment joining $p$ to $\mathcal{M}$ *crosses* the boundary of $P$ at most $k$ times.

*Facultad de Ciencias, Universidad Nacional Autónoma de México, ruy@ciencias.unam.mx

**Facultad de Ciencias, Universidad Nacional Autónoma de México, andresruiz1904@gmail.com

***Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx. Research supported in part by MTM2006-03909 (Spain) and CONACYT of México, Proyecto SEP-2004-Co1-45876.

In this paper we review some of the results known in this area, and present some new results. In section 3 we study several variations of the following problem:

**The k-modem Art Gallery Problem:** *How many k-modems are always sufficient to cover all the points of a polygon $P$ with $n$ vertices?*

In Figure 1 we show a polygon for which we need two 2-modems to cover it, one is not sufficient.

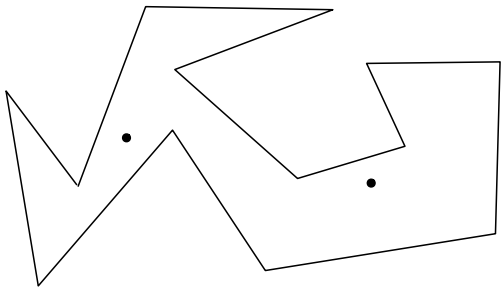


Figura 1: Two 2-modems placed at the points shown as small circles cover the polygon shown here.

In section 2 we study the following problem: Suppose then that we have a set of obstacles represented by a set $L = \{l_1, \ldots, l_n\}$ of $n$ disjoint line segments. As above, we say that a point $p$ is covered by a $k$-modem $\mathcal{M}$ if the line segment connecting $p$ to $\mathcal{M}$ intersects at most $k$ elements of $L$.

**The k-modem Covering Problem of the Plane:** *How many k-modems are always sufficient to cover all the points of the plane in the presence of n obstacles, represented by a set of n disjoint lines?*

In section 3 we review briefly the known results on covering polygons with modems.

In section 4 we study covering problems for polygons, orthogonal polygons, and sets disjoint horizontal or vertical line segments using few modems.

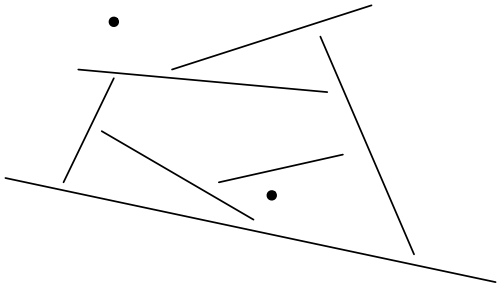


Figura 2: Two 2-modems located at the points shown as small circles are sufficient to cover the plane.

## 2. Arrangements of Lines

In this section we consider a variation of our problem involving families of lines. Let $\mathcal{A} = \{\ell_1, \ldots, \ell_n\}$ be a set of lines such that no three of them intersect at a point, and containing no parallel lines, see Figure 10. The elements of $\mathcal{A}$ divide the plane into a set of faces bounded by the elements of $\mathcal{A}$. The set $\mathcal{A}$ together with the set of faces they produce is known as an *arrangements of lines*, and will be simply denoted as $\mathcal{A}$. For technical reasons we will assume that the faces of $\mathcal{A}$ are open.

We say that a $k$-modem $\mathcal{M}$ coves a face $\mathcal{F}$ of $\mathcal{A}$ if any line joining $\mathcal{M}$ to any point in $\mathcal{F}$ crosses at most $k$ elements of $\mathcal{A}$. In this case, we also say that $\mathcal{M}$ is at distance at most $k$ from $\mathcal{F}$. In general we will assume that the modems are located in the interior of the faces of $\mathcal{A}$. In a few instances we will

allow them to be located on the lines of $\mathcal{A}$. In such case we will assume that a ray emanating from a point on a line $\ell$ does not cross $\ell$, and the bounds obtained will drop by one. A set of $k$-modems $\mathcal{H} = \{\mathcal{M}_1, \ldots, \mathcal{M}_i\}$ *covers an arrangement* $\mathcal{A}$ if every face of $\mathcal{A}$ is covered by at least one of the modems. In this section we study the following problem:

**Arrangement Illumination Problem:** *How many $k$-modems are always sufficient to cover any arrangement $\mathcal{A}$ with $n$ lines?*

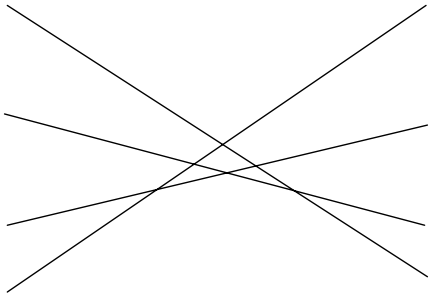


Figura 3: An arrangement of lines.

It is clear that a $n$-modem $\mathcal{M}$ placed anywhere on the plane always covers all the faces of an arrangement with $n$ lines. It is also easy to see that if a $k$-modem $\mathcal{M}$ covers all the faces of $\mathcal{A}$, then $k \geq \lceil \frac{n}{2} \rceil$. On the other hand, it is easy to construct arrangements with $n$ lines such that no $\lceil \frac{n}{2} \rceil$-modem covers them. For example if we have an arrangement $\mathcal{A}$ with three lines $\ell_1, \ldots, \ell_3$, then a modem placed in the interior of the bounded face of the arrangement will necessarily cross two lines to reach some faces of $\mathcal{A}$. A modem placed in the interior of any other face of $\mathcal{A}$ has to cross three lines to reach some faces of $\mathcal{A}$.

Replace now each of the lines $\ell_1, \ldots, \ell_3$ of $\mathcal{A}$ by an arrangement of $n$ lines $\mathcal{A}_i$ such that:

1. all the lines of $\mathcal{A}_i$ have a slope within $\epsilon$ from that of $l_i$, $i = 1, 2, 3$,
2. all of the elements of $\mathcal{A}_i$ pass within an $\delta$ distance from the middle point of the segment contained in $\ell_i$ whose endpoints are the intersection points of $\ell_i$ with the other elements of $\mathcal{A}$, $\epsilon$ and $\delta$ small enough.

It is easy to see that the arrangement $\mathcal{A}^* = \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_3$ is such that any modem placed in the interior of any face of $\mathcal{A}^*$ that covers it has power at least $\lceil \frac{2n}{3} \rceil$, see Figure 4. In this section we give a new proof of the next result first proved in [5], we also review their proof.

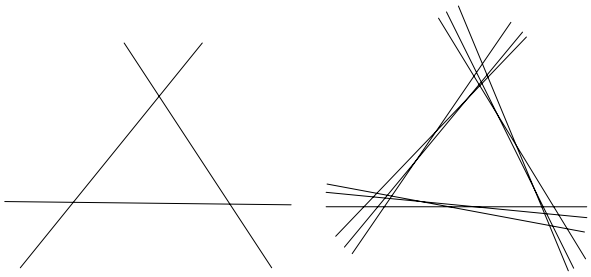


Figura 4: An arrangement of lines.

**Theorem 2.1.** *[5] For every arrangement of $n$ lines on the plane, there is a point $p$ such that a $\lceil \frac{2n}{3} \rceil$-modem placed at $p$ covers the plane, $\lceil \frac{2n}{3} \rceil$ power is sometimes necesary.*

Let $p$ be a point in the interior of a face of an arrangement $\mathcal{A}$ of $n$ lines. The depth of $p$ is the minimum number of lines in $\mathcal{A}$ that we have to cross to reach an unbounded face of $\mathcal{A}$. Rousseew and

Hubert [9] proved that for any arrangement of $n$ lines there is always a point $p$ with depth at least $\frac{n}{3}$, that is any ray emanating from $p$ crosses at least $\frac{n}{3}$ lines of $\mathcal{A}$. It is now straightforward to see that if we place a $\frac{2n}{3}$ modem at $p$, it will cover $\mathcal{A}$.

We develop now some Voronoi-like properties on arrangements of lines that among other things, will give us a new proof of Theorem 2.1. Some preliminary results will be needed. Let $\mathcal{F}$ and $\mathcal{F}'$ be two faces of an arrangement $\mathcal{A}$. We say that $\mathcal{F}'$ is at distance $k$ from $\mathcal{F}$ if a line segment joining a point in $\mathcal{F}$ to a point in $\mathcal{F}'$ crosses $k$ elements of $\mathcal{A}$, the distance from $\mathcal{F}$ to $\mathcal{F}'$ will be denoted as $d(\mathcal{F}, \mathcal{F}')$. It is easy to see that $d(\mathcal{F}, \mathcal{F}')$ is well defined.

Let $\mathcal{A}$ be an arrangement of $n$ lines. Observe that $\mathcal{A}$ has $2n$ unbounded faces. Suppose that we label these faces $\mathcal{F}_0, \ldots, \mathcal{F}_{2n-1}$ in the clockwise direction starting at any of these faces. Let $\mathcal{F}_i$ be an infinite face of $\mathcal{A}$. We call $\mathcal{F}_{i+n}$ the face opposite to $\mathcal{F}_i$, addition *mod* $2n$. It is clear that $\mathcal{F}_i$ and $\mathcal{F}_{i+n}$ are separated by all the lines of $\mathcal{A}$.

We say that a face $\mathcal{Q}$ of $\mathcal{A}$ is equidistant to two faces $\mathcal{F}_i$ and $\mathcal{F}_j$ of $\mathcal{A}$ if $d(\mathcal{F}_i, \mathcal{Q}) = d(\mathcal{Q}, \mathcal{F}_j)$. Suppose that the distance between two faces $\mathcal{F}_i$ and $\mathcal{F}_j$ of an arrangement $\mathcal{A}$ is even. We define the *bisector* of $\mathcal{F}_i$ and $\mathcal{F}_j$, to be the union of the set of faces of $\mathcal{A}$ (together with their boundaries) equidistant to $\mathcal{F}_i$ and $\mathcal{F}_j$, call this set by $Bis(\mathcal{F}_i, \mathcal{F}_j)$, see Figure 5(a).

It is worth noticing that if $d(\mathcal{F}_i, \mathcal{F}_j)$ is odd, then there is no face of $\mathcal{A}$ equidistant to $\mathcal{F}_i$ and $\mathcal{F}_j$. In this case it is easy to see that the set of points equidistant from $\mathcal{F}_i$ and $\mathcal{F}_j$ are points on a polygonal line contained in the lines of $\mathcal{A}$, see Figure 5(b). To obtain our results, we will be mainly concerned with bisectors of faces at even distance, and thus the set of faces equidistant to them will always be non-empty.

We now study some properties of the bisector of $Bis(\mathcal{F}_i, \mathcal{F}_j)$.

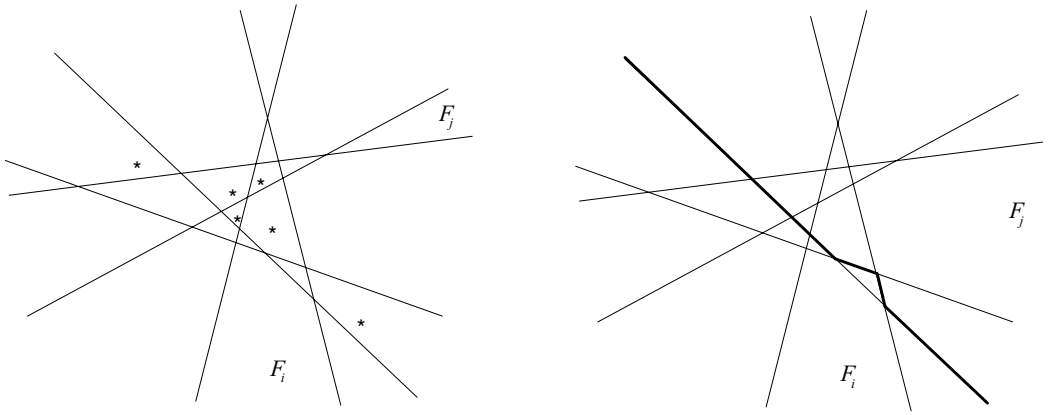



Figura 5: The set of faces equidistant to $\mathcal{F}_i$ and $\mathcal{F}_j$ in (a) are marked with an *.

Let $\mathcal{A}$ be an arrangement with an even number of lines, and let $\mathcal{F}_i$ and $\mathcal{F}_j$ be the unbounded opposite faces of $\mathcal{A}$ such that all the lines in $\mathcal{A}$ are above $\mathcal{F}_i$, and all the lines in $\mathcal{A}$ are below $\mathcal{F}_j$. In this case, it is easy to see that $Bis(\mathcal{F}_i, \mathcal{F}_j)$ contains what is known as the mid-level of the arrangement $\mathcal{A}$, that is $Bis(\mathcal{F}_i, \mathcal{F}_j)$ contains all the faces of $\mathcal{A}$ such that for any point in a face in $Bis(\mathcal{F}_i, \mathcal{F}_j)$, there are exactly half of the lines of $\mathcal{A}$ below it, and half of the elements of $\mathcal{A}$ above it. For this reason, when $\mathcal{F}_i$ and $\mathcal{F}_j$ are infinite and opposite faces of an arrangement $\mathcal{A}$ with an even number of lines, the bisector $Bis(\mathcal{F}, \mathcal{F}')$ will be refered too as to the *mid-level* of $\mathcal{A}$ with respect to $\mathcal{F}_i$ and $\mathcal{F}_j$, see Figure 6.

Suppose now that the unbounded faces of an arrangement $\mathcal{A}$ with $n = 2m$ lines are labeled $\mathcal{F}_0, \ldots, \mathcal{F}_{4m-1}$, in the counterclockwise order, and that $\mathcal{F}_0$ is the region below all the lines of $\mathcal{A}$. Then faces $\mathcal{F}_m$ and $\mathcal{F}_{3m}$ are in $Bis(\mathcal{F}_0, \mathcal{F}_{2m})$. What will be more important to our purposes, is the fact that there is a simple curve $f_{0,2m}$ (which in fact can be chosen so as to be the graph a real function

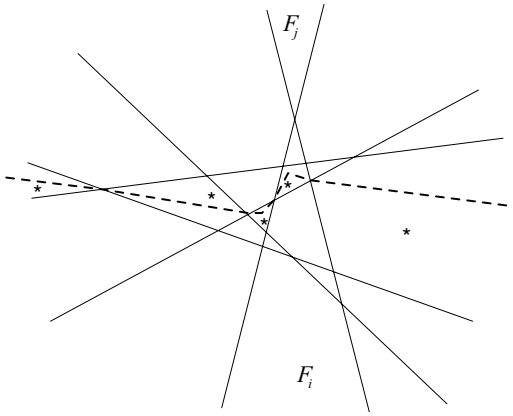



Figura 6: The set of faces equidistant to $\mathcal{F}_i$ and $\mathcal{F}_j$ are marked with an *.

$f : R \to R$) starting in $\mathcal{F}_{3m}$ and ending in $\mathcal{F}_m$ contained in $Bis(\mathcal{F}_0, \mathcal{F}_{2m})$, see Figure 6. Thus any point on $f_{0,2m}$ is equidistant to $\mathcal{F}_0$ and $\mathcal{F}_{2m}$. Clearly for each pair of opposite faces $\mathcal{F}_i$ and $\mathcal{F}_{i+2m}$ of $\mathcal{A}$, we can choose one such a curve, which we will denote as $f_{i,i+2m}$, addition taken *mod* $4m$.

Suppose now that $\mathcal{F}$ and $\mathcal{F}'$ are faces in $\mathcal{A}$ at even distance, but that they are not necessarily opposite faces in $\mathcal{A}$, nor necessarily unbounded. Split the lines of $\mathcal{A}$ into two subsets, $S_1$ containing all the lines of $\mathcal{A}$ that separate $\mathcal{F}$ and $\mathcal{F}'$, and $S_2$ the remaining elements of $\mathcal{A}$. Let $\mathcal{A}'$ be the arrangement defined by the elements of $S_1$. Clearly $\mathcal{F}$ and $\mathcal{F}'$ are opposite faces in $\mathcal{A}'$. The following lemma describes the union of the set of faces equidistant from $\mathcal{F}$ and $\mathcal{F}'$.

**Lemma 2.2.** *The union of the faces of $\mathcal{A}$ equidistant from $\mathcal{F}$ and $\mathcal{F}'$ is exactly the mid-level of $\mathcal{A}'$ with respect to $\mathcal{F}$ and $\mathcal{F}'$.*

*Proof:* Let $\ell_i$ be an element of $S_2$. When we add $\ell_i$ to $S_1$, it will create some new faces in the arrangement defined by $S_1 \cup \{\ell_i\}$. Let $p$ be a point in one such face $\mathcal{F}_j$. If $p$ belongs to the mid-level of $\mathcal{A}'$ with respect to $\mathcal{F}$ and $\mathcal{F}'$, then any line segment $\ell'$ joining $p$ to a point $r$ in $\mathcal{F}$ crosses the same number of lines in $\mathcal{A}'$ than any line segment $\ell''$ joining $p$ to a point in $\mathcal{F}'$. Since $\ell_i \in S_2$, it does not separate $\mathcal{F}$ and $\mathcal{F}'$, either both of $\ell'$ and $\ell$" cross $\ell_i$, or both of them do not cross $\ell_i$. It follows that $p$ remains equidistant to $\mathcal{F}$ and $\mathcal{F}'$. In a similar way, we can prove that if no point in $\mathcal{F}_j$ belongs to the mid-level of $\mathcal{A}'$ with respect to $\mathcal{F}$ and $\mathcal{F}'$, then $\mathcal{F}_j$ is not equidistant to $\mathcal{F}$ and $\mathcal{F}'$. It follows easily that when we add all the elements of $S_2$ to $S_1$ any face equidistant to $\mathcal{F}$ and $\mathcal{F}'$ is contained in the mid-level of $\mathcal{A}'$ with respect to $\mathcal{F}$ and $\mathcal{F}'$. See Figure 7.

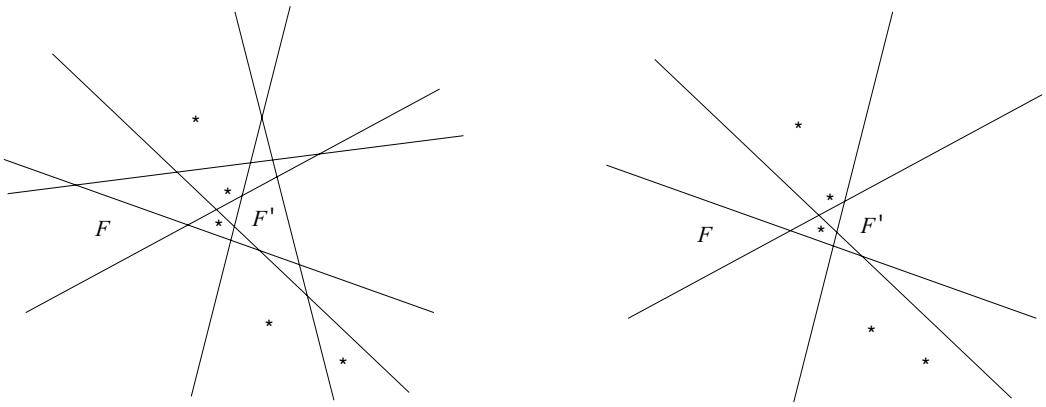



Figura 7: The bisector of two faces at evn distance.

As before, let $\mathcal{A}$ be an arrangement of $n$ lines with its unbounded faces labelled $\mathcal{F}_0, \ldots, \mathcal{F}_{2n-1}$, and consider two unbounded faces $\mathcal{F}_i$ and $\mathcal{F}_j$ of $\mathcal{A}$ at even distance. By using the previous lemma, we

can now define a curve $f_{i,j}$ contained in the mid-level of $\mathcal{F}$ and $\mathcal{F}'$ with respect to the arrangement $\mathcal{A}'$ defined by the lines of $\mathcal{A}$ that separate $\mathcal{F}_i$ and $\mathcal{F}_j$. It is easy to see that $f_{i,j}$ start and end in the unbounded faces $\mathcal{F}_k$ and $\mathcal{F}_{k+n}$ of $\mathcal{A}$, were $k = \frac{i+j}{2}$, addition taken *mod* $2n$.

Let $\mathcal{A}'$ be an arrangement with $3m$ lines, and let us assume as before that the unbounded faces of $\mathcal{A}$ are labeled $\mathcal{F}_1, \ldots, \mathcal{F}_{6m-1}$. Consider now the unbounded faces $\mathcal{F}_0$, $\mathcal{F}_{2m}$, and $\mathcal{F}_{4m}$. It is clear that $d(\mathcal{F}_0, \mathcal{F}_{2m}) = d(\mathcal{F}_{2m}, \mathcal{F}_{4m}) = d(\mathcal{F}_{4m}, \mathcal{F}_0) = 2m$.

Consider the curves $f_{0,2m}$ and $f_{2m,4m}$. It is easy to see that faces $\mathcal{F}_m$ and $\mathcal{F}_{4m}$ are in $Bis(\mathcal{F}_0, \mathcal{F}_{2m})$, and that $\mathcal{F}_{3m}$ and $\mathcal{F}_0$ are in $Bis(\mathcal{F}_{2m}, \mathcal{F}_{4m})$. It follows now that $f_{0,2m}$ start and end in faces $\mathcal{F}_m$ and $\mathcal{F}_{4m}$, and that $f_{2m,4m}$ starts in $\mathcal{F}_{3m}$ and ends in $\mathcal{F}_0$. Therefore $f_{0,2m}$ and $f_{2m,4m}$ intersect at a point $p$, see Figure 8. It could happen that $p$ lies on the intersection of two lines in $\mathcal{A}$, this is the case in Figure 8. For our purposes, the worst case arises when $p$ belongs to the interior of a face in $\mathcal{A}$, we analyze only this case. Since $p$ belongs to $f_{0,2m}$ and $\mathcal{F}_{2m}$, it is equidistant to $\mathcal{F}_0$, $\mathcal{F}_{2m}$, and to $\mathcal{F}_{4m}$. Suppose that $p$ is at distance $r$ from $\mathcal{F}_0$. Since $\mathcal{F}_0$ and $\mathcal{F}_{3m}$ are opposite faces, then the sum of the distances of $p$ to $F_0$ and to $\mathcal{F}_{3m}$ equals the number of lines of $\mathcal{A}'$, that is they add to 3m. It now follows that the distance from $p$ to $\mathcal{F}_{3m}$ is $3m - r$. Similarly we can prove that the distance from $p$ to $\mathcal{F}_m$ and to $\mathcal{F}_{5m}$ is $3m - r$, and thus $p$ is also equidistant to $\mathcal{F}_m$, $\mathcal{F}_{3m}$, and $\mathcal{F}_{5m}$.

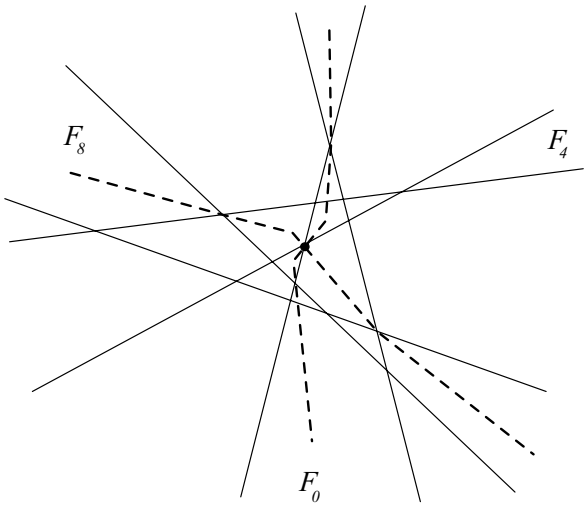



Figura 8: The curves $f_{0,4}$ and $f_{4,8}$ intersect at the point indicated with a small circle.

We now prove:

**Lemma 2.3.** *The distance of $p$ to any face of $\mathcal{A}'$ is at most $2m$.*

*Proof:* Take a point $q_i$ in $\mathcal{F}_i$, $i = 0, m, 2m, 3m, 4m, 5m$ such that they are the vertices of a convex hexagon $\mathcal{H}$ that contains in its interior all the intersection points of pairs of lines of $\mathcal{A}'$.

We now prove that any point in $\mathcal{H}$ is at distance at most $2m$ from $p$. Consider the triangle $\mathcal{T}$ with vertices $p$, $q_0$, and $q_m$. We now prove that exactly $2m$ lines of $\mathcal{A}'$ intersect the boundary of $\mathcal{T}$. Observe first that since the distance from $\mathcal{F}_0$ to $\mathcal{F}_m$ the line segment joining $q_0$ to $q_m$ crosses exactly $m$ lines of $\mathcal{A}$. Since the distance from $p$ to $\mathcal{F}_0$ and to $\mathcal{F}_m$ are respectively $r$ and $3m - r$, then the lines joining $p$ to $q_0$, and $p$ to $q_m$ crossed respectively exactly $r$ and $3m - r$ lines of $\mathcal{A}'$. Thus the edges of $\mathcal{T}$ are crossed exactly $m + r + (3m - r) = 4m$ times by the lines of $\mathcal{A}'$. Since each line that crosses the boundary of $\mathcal{T}$, crosses it twice, it follows that $2m$ lines of $\mathcal{A}'$ enter $\mathcal{T}$, and thus the distance from any point in $\mathcal{T}$ to $p$ is at most $2m$. In a similar way we prove that any point in each of the six triangles formed by $p$ and each of the edges of $\mathcal{H}$ is at distance at most $2m$ from $p$. Our result follows.

Theorem 2.1 follows directly from Lemma 2.3.

### 2.0.1. Two Modems

It is clear the if we place two modems with power $\lceil \frac{n}{2} \rceil$ in two unbounded opposite faces of an arrangement with $n$ lines, they will cover the whole plane. The natural question now is: By placing them more carefully, can we improve on the $\lceil \frac{n}{2} \rceil$ previous bound? This is not possible. We now prove:

**Theorem 2.4.** *For every arrangement $\mathcal{A}$ of $n$ lines on the plane, two $\lceil \frac{n}{2} \rceil$-modems are always sufficient and necessary to cover the plane.*

*Proof:* Let $\mathcal{M}_1$ and $\mathcal{M}_2$ two $k$-modems placed on the plane that cover the plane. Suppose that they are placed at two faces $\mathcal{F}_1$ and $\mathcal{F}_2$ of $\mathcal{A}$, and that the line joining them is horizontal. Divide the lines of $\mathcal{A}$ into two subsets, $S_1$ containing the lines of $\mathcal{A}$ that separate $\mathcal{F}_1$ and $\mathcal{F}_2$, and $S_2$ the remaining lines of $\mathcal{A}$. Several cases arise, depending on the number of lines in $\mathcal{A}$, and the distance between $\mathcal{F}_1$ and $\mathcal{F}_2$. We analyze the case when $n = 2m$, and the distance between $\mathcal{F}_1$ and $\mathcal{F}_2$ is even. The other cases follow in a similar way.

Let $\mathcal{A}'$ be the arrangement induced by $S_1$, and suppose that the distance between $\mathcal{F}_1$ and $\mathcal{F}_2$ is $2k$. Then $S_1$ has $2k$ elements, and $S_2$ has $2r$ elements, $k + r = m$. Clearly $\mathcal{M}_1$ and $\mathcal{M}_2$ are in opposite unbounded faces of $\mathcal{A}'$. Thus there are two unbounded faces $\mathcal{H}_1$ and $\mathcal{H}_2$ of $\mathcal{A}'$ at distance $k$ from $\mathcal{F}_1$ and $\mathcal{F}_2$. It is easy to see that one of them, say $\mathcal{H}_1$ contains a point $p_1$ below all the lines in $S_2$, while $\mathcal{H}_2$ contains a point $p_2$ above all the elements in $S_2$. Since the lines in $S_2$ do not separate $\mathcal{M}_1$ from $\mathcal{M}_2$, for each $\ell_i \in S_2$, $\mathcal{M}_1$ and $\mathcal{M}_2$ are above $\ell_i$ or below it. Then at least half of the elements of $S_2$ have both of $\mathcal{M}_1$ and $\mathcal{M}_2$ above them, or at least half of them have $\mathcal{M}_1$ and $\mathcal{M}_2$ below them. Thus the distance of $p_1$ or $p_2$ to both of $\mathcal{M}_1$ and $\mathcal{M}_2$ is at least $k + \frac{|S_2|}{2} \geq k + r = \lceil \frac{n}{2} \rceil$, see Figure 9.

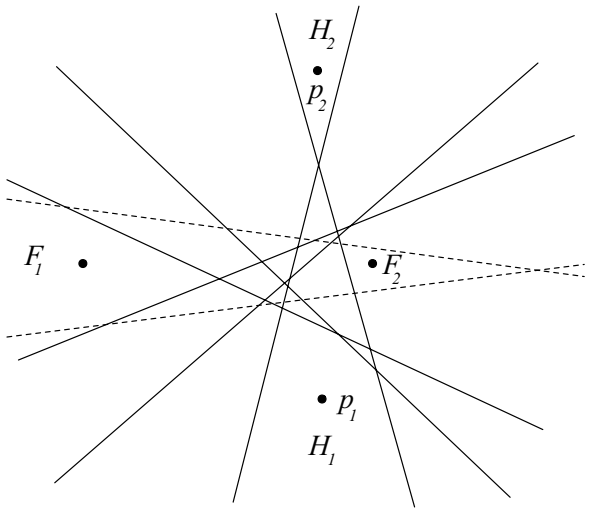



Figura 9: The elements of $S_1$ are the solid lines, the elements of $S_2$ are the dotted lines.

Using arguments similar to those used in the proof of Theorem 2.1, we can also prove:

**Theorem 2.5.** *Any arrangement of $n$ lines, can be covered with four $\frac{5}{12}$-modems.*

The proof of this result will be given in a forthcoming paper. We believe that this bound is not tight, and we pose the following open problem:

**Problem 2.6.** Is it true that any arrangement $\mathcal{A}$ with $n$ lines can always be covered with four $\frac{n}{3} + c$ modems, $c$ a constant?

## 3. Covering Polygons

When $k = 0$, the probelm of covering polygons with $k$-modem is equivalent to the original Art Gallery Problem. However for $k \geq 1$, the $k$-modem illumination problem has turned out to be very

difficult to solve. So far we only have significant results concerning *x-monotone* polygons, that is polygons such that the intersection of any any vertical line with them is empty or an interval. In [1] the following result is proved:

**Theorem 3.1.** *[1] Every monotone polygon with $n$ vertices can be covered with at most $\lceil \frac{n}{2k} \rceil$ $k$-modems.There are There re monotone $n$-gons that require at least $\lceil \frac{n}{2k-2} \rceil$ $k$-modems to cover them.*

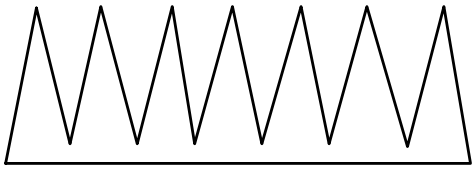


Figura 10: A monotone polygon that needs $\lceil \frac{n}{2k-2} \rceil$ $k$-modems to be covered.

The proof of the previous result is based on the following Lemmas which are given without proof. The interested reader is refered to [1]. Let $P$ be a monotone polygon, and suppose that its vertices are labelled $p_1, \ldots, p_n$ from left to right. We have:

**Lemma 3.2.** *(The Splitting Lemma) Let $P$ be a polygon with vertices $p_1, \ldots, p_n$, then for every $m < n$ there is a vertical line segment $\ell$ and two polygons $R$ and $L$ such that $L$ has $m$ vertices, $R$ has $n-m+2$ vertices, and:*

- *Either $\ell$ is a chord of $L$ and an edge of $L$, or viceversa*
- *$p_m$ or $p_{m+1}$ is an endpoint of $\ell$*
- *Let $L'$ be the polygon containing all the points of $L$ on, or to the left of $\ell$ and $R'$ the polygon containing all the points of $R$ to the right of $\ell$. Then $P = L' \cup R'$.*

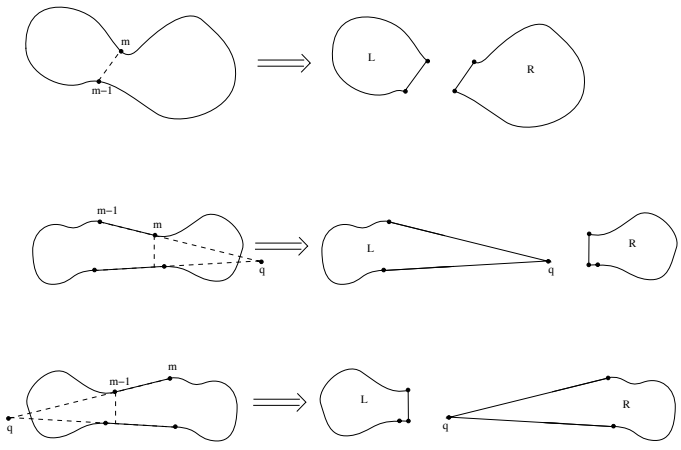



Figura 11: Illustrating the Splitting Lemma.

**Lemma 3.3.** *Any $k+2$ gon can be covered with a $k$-modem placed anywhere in the interior of $P$.*

**Lemma 3.4.** *Every $2k+2$-monotone polygon can be covered with a $k$-modem placed at either of its $k+1$-th or $k+2$-th vertex.*

The proof of Theorem 3.1 follows by applying the Splitting Lemma recursively to $P$ to obtain $\lceil \frac{n}{2k} \rceil$ $2k+2$-monotone polygons, and then applying Lemma 3.4 to each of them.

# 4. Covering Polygons With Few Modems With High Power

In this section we study the problem of covering simple polygons using few modems with high power. In a recent paper Fulek, Holmsen, and Pach [5] proved the following result which will prove useful to us:

**Theorem 4.1.** *Let $\mathcal{F}$ be a family of $n$ pairwise convex sets in $\mathbb{R}^d$, then there is a point $p$ in the plane such that any ray emanating from $p$ intersects at most $\frac{dn+1}{d+1}$ elements of $\mathcal{F}$.*

In the same paper, they proved that for any $n$, there is a set $T_{3m}$ of $3m$ disjoint segments such that for any point $p$ on the plane there is a ray emanating from $p$ that intersects at least $2n-2$ elements of $T_{3m}$.

It can be shown that we can add $3m$ segments to the elements of $T_{3m}$ to obtain a simple polygon $P_{6m}$ with $6m$ vertices such that for every point $p$ on the plane, there is a ray emanating from $p$ intersects at least $4m-c$ edges of $P_{6m}$, that is to cover $P_{6m}$ we need a $(4m-c)$-modem, $c$ a constant. Using these results we can prove:

**Theorem 4.2.** *Any simple polygon $P$ with $n$ edges can always be covered with a modem of power $\lceil\frac{2n+1}{3}\rceil$, the bound is tight up to an additive constant.*

*Proof:* The sufficiency follows directly from Fulek, Holmsen, and Pach's Theorem 4.1. The proof that sometimes we need a $(\lceil\frac{2n+1}{3}\rceil - c)$-modem, follows from the fact that to cover $P_{6m}$ we need a $(4m-c)$-modem.

### 4.0.2. Orthogonal Arrangements of Lines, Line Segments, and Polygons

In this section we study the problem of covering orthogonal arrangements of lines and orthogonal polygons, that is simple polygons such that its edges are all horizontal or vertical.

An orthogonal arrangement of lines is defined as in Section 2, except that $\mathcal{A}$ is now allowed to contain parallel lines, and all of them are horizontal and vertical. We prove first:

**Theorem 4.3.** *Let $\mathcal{F}$ be a family of disjoint segments, $m$ horizontal, $n$ vertical. Then there is a point $p$ in the plane such that any ray emanating from $p$ intersects at most $\lceil\frac{m}{2}\rceil + \lceil\frac{n}{2}\rceil$ elements of $\mathcal{F}$.*

*Proof:* It is easy to see that the worst case happens when no two elements of $\mathcal{F}$ are co-linear. Let $\ell_1$ and $\ell_2$ be a horizontal and a vertical line respectively, such that there are exactly $\lceil\frac{m}{2}\rceil$ horizontal elements of $\mathcal{F}$ above $\ell_1$, and $\lceil\frac{m}{2}\rceil$ vertical elements of $\mathcal{F}$ to the left of $\ell_1$. Let $p$ be the intersection point of $\ell_1$ with $\ell_2$. It is straightforward to verify that any ray emanating from$p$ intersects at most $\lceil\frac{m}{2}\rceil + \lceil\frac{n}{2}\rceil$ elements of $\mathcal{F}$. The sufficiency of our result follows from a set of lines as shown in Figure 12.

In a similar way we can prove now:

**Theorem 4.4.** *Let $\mathcal{A}$ be an orthogonal arrangement of $m$ horizontal and $n$ vertical lines. Then there is a point $p$ in the plane such that any ray emanating from $p$ intersects at most $\lceil\frac{m}{2}\rceil + \lceil\frac{n}{2}\rceil$ elements of $\mathcal{A}$.*

Given a point $p=(a,b)$ on the plane let $\mathcal{C}_+^+(p) = \{q=(x,y) : a \leq x, b \leq y\}$. When $p$ is the origin, $C(p)_+^+$ is the positive quadrant of the plane. Similarly we define $\mathcal{C}_+^-(p) = \{q=(x,y) : a \leq x, b \geq y\}$, $\mathcal{C}_-^-(p) = \{q=(x,y) : a \geq x, b \geq y\}$, and $\mathcal{C}_-^+(p) = \{q=(x,y) : a \geq x, b \leq y\}$ The following lemma will be used:

**Lemma 4.5.** *Let $P$ be an orthogonal polygon and $p$ be a point on the plane. Suppose that $k$ vertices of $P$ belong to the interior of $\mathcal{C}_+^+(p)$. Then any ray emanating from $p$ contained in $\mathcal{C}_+^+(p)$ crosses at most $k$ edges of $P$.*

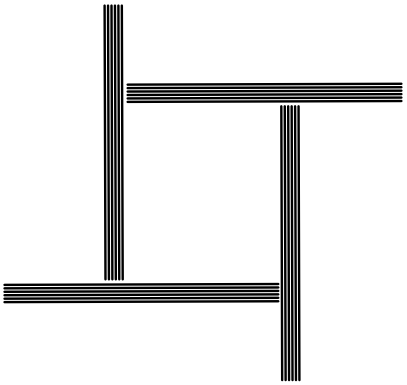


Figura 12:

*Proof:* Suppose that $p$ is the origin, and let $P$ be an orthogonal polygon such that $P$ has $k$ vertices in $\mathcal{C}_+^+(p)$. Let $\vec{p}$ be a ray contained in $\mathcal{C}_+^+(p)$ that emanates from $p$. If $\vec{p}$ intersects a horizontal edge $e$ of $P$, charge this intersection to the right endvertex of $e$. If $\vec{p}$ intersects a vertical edge $f$ of $P$, charge this intersection to the top endvertex of $f$. We can picture this as orienting the horizontal edges of $P$ from left to right, and the vertical edges from bottom to top. It is now easy to see any ray $\vec{p}$ can charge at most one intersection to any vertex of $P$ in $\mathcal{C}_+^+(p)$.

Clearly this lemma also holds for $\mathcal{C}_+^-(p)$, $\mathcal{C}_-^-(p)$, and $\mathcal{C}_-^+(p)$. We now prove

**Theorem 4.6.** *Let $P$ be an orthogonal polygon with $2m$ edges. Then if $m$ is even, $P$ can always be covered with an $m-1$ modem located* in the interior *of $P$. The bound is tight. If $m$ is odd, an $m$-modem is always sufficient. For $m$ even, the bound is tight.*

*Proof:* Let $P$ be an orthogonal polygon with $2m$ edges. Suppose that $m$ is even. Let $\ell$ be a line that leaves $\frac{m}{2}$ horizontal edges of $P$ in the interior of each of the half-planes it defines. Thus $P$ has exactly $m$ vertices above $\ell$. Choose a point $p$ on $\ell$ and in the interior of $P$. It is easy to see that since $p$ belongs to the interior of $P$, each of $\mathcal{C}_+^+(p)$, $\mathcal{C}_+^-(p)$, $\mathcal{C}_-^-(p)$, and $\mathcal{C}_-^+(p)$ contains at most $m-1$ vertices in $P$. The upper bound now follows from the previous Lemma. The case when $m$ is odd follows in a similar way.

Examples when an $(m-1)$-modem is required are shown in Figure 13.

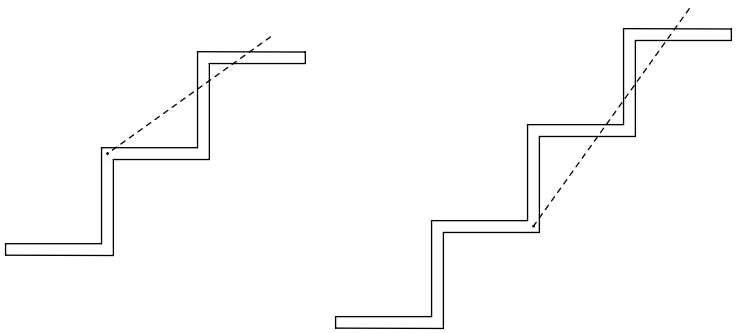


Figura 13:

When we allow our modems to be anywhere on the plane we can prove:

**Theorem 4.7.** *Any orthogonal polygon $P$ with $n$ edges can be covered with a $\lceil \frac{n}{3} \rceil$-modem.*

We only give a sketch of the proof. Suppose that $P$ is contained in the unit square. Pick a point on the vertical band above the unit square, and high enough so that any ray emanating from $p$ intersects

at most one vertical edge of $P$. Let $k$ be the maximum number of horizontal edges of $P$ that any ray emanating from $p$ intersects. If $k \leq \lceil \frac{n}{3} \rceil$ we are done as any such ray intersects at most one vertical edge of $P$. Suppose then that a ray $\vec{p}$ emanating from $p$ intersects at least $k \geq \lceil \frac{n}{3} \rceil$ horizontal edges of $P$. Assume without loss of generality that $\vec{p}$ is vertical. Then the line $\ell$ containing $\vec{p}$ has at least $k \geq \frac{n}{3}$ vertices of $P$ on each of its sides. Sweep a horizontal line $h_\ell$ from top to bottom until one of the two quadrants (above $h_\ell$) defined by $\ell$ and $h_\ell$ contains exactly $\lceil \frac{n}{3} \rceil$ vertices.

It now follows easily that since $k \geq \lceil \frac{n}{3} \rceil$, each of the four quadrants defined by $\ell$ and $h_\ell$ contains at most $\lceil \frac{n}{3} \rceil$ vertices of $P$. Our result now follows from Lemma 4.5.

It is left as an open problem to determine the minimum $k$ such that it is always possible to cover any orthogonal polygon with $n$ edges with a $k$-modem. We conjecture that such $k$ is within a constant factor of $\frac{n}{4}$.

# Referencias


[1] O. Aichholzer, R. Fabila-Monroy, D. Flores-Peñaloza, T. Hackl, C. Huemer, and J. Urrutia, *Modem Illumination of Monotone Polygons*,

[2] J. Cano-Vila, J. Espinosa-Longi, and J. Urrutia, *Vigilancia en Galerías de Arte Curvilíneas*, in Proceedings XIII Encuentros de Geometr'a Computacional, Zaragoza, España, 2009, por aparecer.

[3] V. Chvátal, *A Combinatorial Theorem in Plane Geometry*, J. Combin. Theory Ser B 18 (1975), 39-41

[4] S. Fisk, *A Short Oroof of Chvátal's Watchman Theorem*, J. Combin. Theory Ser B 24 (1978), 374

[5] R. Fulek, A. Holmsen, and J. Pach, *Intersecting convex sets by rays*, in: Proceedings 24th annual Symposium on Computational Geometry, ACM Press, 2008, 385–391.

[6] M. I. Karavelas and E. P. Tsigaridas, *Guarding Curvilinear Art Galleries with Vertex or Point Guards*, Research Report 6132, INRIA, 2007

[7] M. I. Karavelas, Cs. D. Tóth and E. P. Tsigaridas, *Guarding Curvilinear Art Galleries with Vertex or Point Guards*, Computational Geometry: Theory and Applications, to appear.

[8] J. O'Rourke, *Art Gallery Theorems and Algorithms.* The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.

[9] P.J. Rousseew, and M. Hubert, *Depth in an Arrangement of Hyperplanes*, Discrete and Computational Geometry, Vol. 22, pp. 167-176, 1999.

[10] S. Ramaswami, P. Ramos and G. Toussaint. *Converting Triangulations to Quadrangulations*, Computational Geometry: Theory And Applications, vol. 9, pp 257-276, 1998.

[11] T. C. Shermer. *Recent Results in Art Galleries.* Proc. IEEE, 80(9):1384-1399, 1992.

[12] J. Urrutia, *Art Gallery and Illumination Problems*, J.-R. Sack and J. Urrutia editors, *Handbook of Computational Geometry*, 973-1027. North-Holland, 2000.

# Basics on Geometric Constraint Solving

Robert Joan-Arinyo *

**Abstract**

We survey the current state of the art in geometric constraint solving. Both 2D and 3D constraint solving is considered, and different approaches are characterized.

**Keywords**: Geometric constraints, constraint solving, parametric design.

## 1 Introduction and Scope

2D geometric constraint solving is arguably a core technology of computer-aided design (CAD) and, by extension, of managing product design data. Since the introduction of parametric design by Pro/Engineer in the 1980s, every major CAD system has adopted geometric constraint solving into its design interface. Most prominently, 2D constraint solving has become an integral component of sketchers on which most systems base feature design.

Beyond applications in CAD and, by extension, in manufacturing, geometric constraint solving is also applicable in virtual reality and is closely related in a technical sense to geometric theorem proving. For solution techniques, geometric constraint solving also borrows heavily from symbolic algebraic computation and matroid theory.

In this paper, we review basic techniques that are widely available for solving 2D and 3D geometric constraint problems. We focus primarily on the basics of 2D solving and touch lightly on spatial constraint solving and the various ways in which geometric constraint solvers can be extended with relations, external variables, and parameter value enclosures. These and other extensions and problem variants have been published in the literature. They are recommended to the interested reader as follow-on material for study.

## 2 The Geometric Constraint Solving Problem

A geometric constraint problem can be characterized by means of a tuple $(E, O, X, C)$ where $E$ is the geometric space constituting a reference framework into which the problem is embedded. $E$ is usually Euclidean, $O$ is the set of specific geometric objects which define the problem. They are chosen from a fixed repertoire including points, lines, circles and the like, and $C$ is the set of geometric constraints. They are relationships between geometric elements chosen from a predefined set, e.g., distance, angle, tangency, etc.

The geometric constraint solving problem can now be stated as follows: Given a set $O$ with $n$ geometric elements and a set $C$ with $m$ geometric constraints defined on them

1. Is there a placement of the $n$ geometric elements such that the $m$ constraints are fulfilled? If the answer is positive,

*Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Universitat Politècnica de Catalunya robert@lsi.upc.edu

2. given an assignment of values to the $m$ constraints tags, is there an actual construction of the $n$ geometric elements satisfying the constraints?

When dealing with geometric constraint solving, the first issue that needs to be settled is the dimension of the embedding space $E$. In 2D Euclidean space, $E = \mathcal{R}^2$, a number of techniques have been developed that successfully solve the geometric constraint solving problem. For an in-depth review see Jermann, [29]. However, there remain open questions such as characterizing the competence (also called domain) of the known techniques.

Spatial constraint solving, where $E = \mathcal{R}^3$, include problems in fields like molecular modeling, robotics, and terrain modeling. Here, both a good conceptualization and an effective solving methodology for the geometric constraint problem has proved to be difficult. Pioneering work has been reported by Hoffmann and Vermeer, [27] and by Durand, [13].

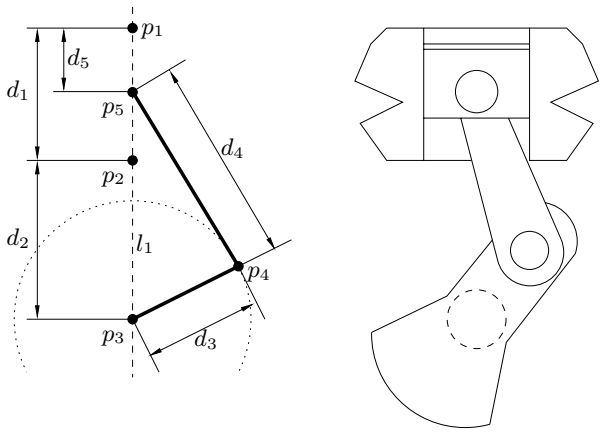



Figure 1: Piston, crankshaft and connecting rod mechanism.

Figure 1 depicts a piston-crankshaft mechanism, [12], a geometric constraint solving problem in 2D. The left side shows the geometric problem, the right side shows the actual mechanism so abstracted. The mechanism transforms the translational motion of point $p_5$ along the straight line $l_1$ into a rotational motion of point $p_4$, on a circular path with center $p_3$ and radius $d_3$. The piston-crankshaft mechanism can be abstracted as a geometric constraint solving problem comprising five points $p_i$, $1 \leq i \leq 5$, and a straight line $l_1$. The set of constraints is given in Figure 2 and includes point-point distances, *dpp*(), and coincidences, *on*().

## 2.1 Problem Categorization

The CAD/CAM community focuses on the design and manufacture of rigid objects, that is, objects that are fully determined up to a global coordinate system. Similarly, we seek solutions to a constraint problem that are determined up to a global coordinate system, that is, where solutions are congruent under the rigid-body transformations of translation and rotation. We call a configuration of geometric objects in Euclidean space *rigid* when all objects are fixed with respect to each other up to translation and rotation.

| | | | |
|---|---|---|---|
| 1. | $\text{dpp}(p_1, p_2) = d_1$ | 6. | $\text{on}(p_1, l_1)$ |
| 2. | $\text{dpp}(p_2, p_3) = d_2$ | 7. | $\text{on}(p_2, l_1)$ |
| 3. | $\text{dpp}(p_3, p_4) = d_3$ | 8. | $\text{on}(p_3, l_1)$ |
| 4. | $\text{dpp}(p_4, p_5) = d_4$ | 9. | $\text{on}(p_5, l_1)$ |
| 5. | $\text{dpp}(p_1, p_5) = d_5$ | | |

Figure 2: A set of geometric constraints for the piston-crankshaft mechanism.

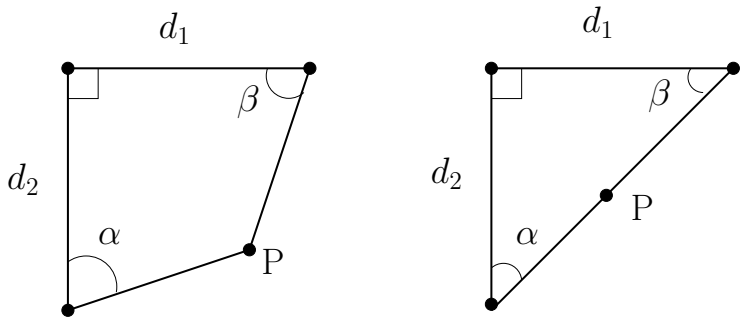



Figure 3: Left: General configuration. Right: Degenerate configuration for $\alpha + \beta = 90°$. Fudos and Hoffmann, [15].

An intuitive way to introduce rigidity comes from considering the number of solutions that a geometric constraint problem has. There are three categories: A problem is *structurally under constrained* if there are infinitely many solutions that are not congruent under rigid transformation, *structurally well-constrained*, if there are finitely many solutions modulo rigid transformation, and *structurally over constrained* if the deletion of one or more constraints results in a well-constrained problem. A constraint problem naturally corresponds to a set of (usually nonlinear) algebraic equations.

Defined in this way, the concept of rigidity appears to be simple but it is not quite in accord with the intuition about rigidity. The categories so defined only refer to the problem's structure and do not account for other issues such as inconsistencies that could originate from specific values assigned to the constraints. Clearly a problem that is structurally well-constrained could actually be underconstrained for specific values of the constraints. For example, consider the *structurally* well constrained problem given in Figure 3, see Fudos and Hoffmann, [15]. Point $P$ is properly placed whenever $\alpha + \beta \neq 90°$ and the problem is well-constrained. But if $\alpha + \beta = 90°$, then the placement for point $P$ is undetermined and, therefore, the problem is no longer well constrained.

Different formal definitions of rigidity have been explored in the literature. See, for example, the work by Henneberg, [19], and Laman, [40], or the more recent works by Graver *et al.* [17], Fudos and Hoffmann, [15], Hoffmann *et al.*, [24], and Whitley, [64].

# 3 Major Approaches

Geometric constraint solving methods can be roughly classified as graph-based, logic-based, or algebraic. For 2D solvers, the graph-based approach has become dominant in CAD. A problem closely related to geometric constraint solving is Automated Theorem Proving.

## 3.1 Graph-Based Approach

In the graph-based approach, the constraint problem is translated into a graph (or hyper graph) whose vertices represent the geometric elements and whose edges the constraints upon them . The solver analyzes the graph and formulates a solution strategy whereby subproblems are isolated and their solutions suitably combined. A subsequent phase then solves all subproblems and combines them. The advantage of this type of solver is that the subproblems often are very small and fall into a few simple categories. The disadvantage is that the graph analysis of a fully competent solver is rather complicated. The graph-based approach can be further subdivided into constructive, degree of freedom analysis, and propagation.

### 3.1.1 Constructive Approach

The constructive approach generates the solution to a geometric constraint problem as a symbolic sequence of basic construction steps. Each step is a rule taken form a predefined set of operations that position a subset of the geometric elements. For example, the operations may restrict to ruler-and-compass constructions. Clearly, this approach preserves the geometric sense of each operation involved in the solution. Note that the sequence of construction steps allows to compactly represent a possibly exponential number of solution instances. However, the constructive approach cannot solve problems with symbolic constraints or external variables.

Depending on the technique used to analyze the problem, two different categories of constructive approaches can be distinguished: *top-down* and *bottom-up*.

The top-down technique recursively splits the problem until it has isolated simpler, basic problems whose solutions are known. In this category, Todd, [60], defines the *r-tree* concept and derives a geometric constraint solving algorithm. Owen, [51], describes a more general method based on the recursive decomposition of the constraints graph into triconnected components. Inspired by Owen's work, Fudos reported a new decomposition method in [15]. Efficient algorithms with a running time $O(n^2)$, where $n$ is the number of geometric elements, are known for the methods of Owen, [51], and Fudos, [15].

In the bottom-up approach, the solution is built by suitably combining recursively solutions to subproblems already computed, starting from the constraints in the given set, considering each constraint as a single element.

Constraints may be represented implicitly as a collection of sets of geometric elements where the elements of each set are placed with respect to a local framework. Sets are merged; e.g., by application of rewriting rules until all the geometric elements are included in just one set. The advantage of this representation is that the sets of constraints capture the relationships between geometric elements compactly. Fudos *et al.*, in the method described in [15], use one type of sets of constraints, called *cluster*, and one generic rule that merges three clusters which pairwise share two elements.

Lee *et al.*, [43], describe a constructive method that associates with each vertex in the graph a status which can be defined, half defined or not defined. Inference rules are used to modify the status of the vertices.

Efficient algorithms with a running time $O(n^2)$, where $n$ is the number of geometric elements, are known for the methods listed above. However, the constructive approach is not complete, therefore assessing the competence of solvers in this category is an important issue. Verroust, [63], partially characterizes the set of relevant problems solved by the solver described. Joan-Arinyo *et al.*, [32], describe a formalization that unifies the methods reported by Fudos, [15], and Owen, [51]. In [33], Joan-Arinyo *et al.* show that the sets of problems solved by Fudos' and Owen's approaches are the same.

In [23] Hoffmann and Joan-Arinyo describe a technique that extends constructive methods with the capability of managing functional relationships between geometric and externals variables. Essentially, the technique combines two methods: one is a constructive constraint solving method, and the other is a systems of equations analysis method. Joan-Arinyo and Soto-Riera, [31], further improved the technique and formalized it as a rewriting system.

Constructive methods work well in 2-space. Several attempts to extend them to 3-space have been reported. See, for example, Brüderlin [6], Verroust [62], and Hoffmann and Vermeer [27].

### 3.1.2 Degrees of Freedom Analysis

Degrees of Freedom Analysis assigns degrees of freedom to the geometric elements by labeling the vertices of the graph of the problem. Each edge of the graph is labeled with the number of degrees of freedom canceled by the associated constraint. Then the method solves the problem by analyzing the resulting labeled graph.

Kramer, [38], developed a method to solve specific problems from the kinematics of mechanisms. The method applies techniques borrowed from the process planning field to yield a symbolic solution. Since the set of rules used to generate the plan preserves geometric sense, the entire method also preserves it. Kramer, [38], proves that his method is correct by showing that the set of rules together with the labeled graph is a canonical rewriting system. The method runs in time $O(nm)$, where $n$ is the number of geometric elements and $m$ the number of constraints in the problem. Since $m$ is typically $O(n)$, the method has the same complexity as the constructive approach. Bhansali *et al.*, [3], describe a method that generates automatically segments of the symbolic solution in Kramer's approach.

Salomons *et al.*, [52], represent objects and constraints as a graph and apply geometric and equational reasoning following the lines given by Kramer's method.

In [28], Hsu reports a method with two phases. First, a symbolic solution is generated. Then, the actual construction is carried out. The method applies geometric reasoning and, if this fails, numerical computation.

Latham *et al.*, [42], decompose the labeled graph in minimal connected components called *balanced sets*. If a balanced set corresponds to one of the predefined specific geometric constructions, then it can be solved. Otherwise the underlying equations are solved numerically. The method also deals with symbolic constraints and identifies over- and under constrained problems. Assigning priorities to the constraints allows them to solve over constrained problems. A proof of correctness is also given.

Hoffmann *et al.*, [25], have developed a flow-based method for decomposing graphs of geometric constraint problems. The method generically iterates to obtain a decomposition of the underlying algebraic system into small subsystems called *minimal dense* subgraphs. The method fully generalizes degree-of-freedom calculations, the approaches based on matching specific subgraphs patterns, as well as the prior flow-based approaches. However, the decomposition rendered does not necessarily have geometric sense since minimal dense subgraphs can be of arbitrary complexity far exceeding problems that yield to classical geometric construction.

### 3.1.3 Propagation Approach

Propagation methods represent the set of algebraic equations with a symmetric graph whose vertices are variables and equations and whose edges are labeled with the occurrences of the variables in the equations.

Propagation methods try to orient the edges in the graph in such a way that each equation vertex is a sink for all the edges incident on it except one. If the process succeeds, then there is a general incremental solution. That is, the system of equation can be transformed into a triangular system and solved using back substitution.

Among the techniques to orient a graph we find in the literature degrees of freedom propagation and propagation of known values, [14, 53, 61]. Propagation methods do not guarantee finding a solution whenever one exists. They fail when the orientation algorithm finds a loop. Propagation methods can combined with numerical methods for equation solving to ameliorate circularity, [5, 39, 56, 59]. Veltkamp and Arbab, [61], apply other techniques to break loops created while orienting the graph.

Leler, [44], describes propagation methods in depth and proposes *augmented rewriting terms*, a tool which consists of a classical rewriting system along with an association of atomic-value and object-type. This tool has had success in solving certain systems of nonlinear equations.

In [4], Borning *et al.*, describe an local propagation algorithm that can deal with inequalities.

## 3.2 Logic-Based Approach

In the logic-based approach, the problem is translated into a set of assertions and axioms characterizing the constraints and the geometric objects. By employing reasoning steps, the assertions are transformed

in ways that expose solution steps in a stereotypical way and special solvers then compute coordinate assignments.

Aldefeld, [1], Brüderlin, [7], Sohrt, [55] and, Yamaguchi *et al.*, [66], use first order logic to derive geometric information applying a set of axioms from Hilbert's geometry. Essentially these methods yield geometric loci at which the elements must be.

Sunde, [58], and Verroust, [63], consider two different types of sets of constraints: sets of points placed with respect to a local framework, and sets of straight line segments whose directions are fixed with respect to a local framework. The reasoning is basically performed by means of a rewriting system on the sets of constraints. The problem is solved when all the geometric elements belong to a unique set. Joan-Arinyo and Soto-Riera, [30], extended these sets of constraints with a third type consisting of sets containing one point and one straight line such that the perpendicular point-line distance is fixed.

## 3.3 Algebraic Methods

In the algebraic approach, the constraint problem is translated directly into a set of nonlinear equations and is solved using any of the available methods for solving nonlinear equations. The main advantages of algebraic solvers are their generality, dimension independence and the ability to deal with symbolic constraints naturally.

In principle, an algebraic solver can be fully competent. However, algebraic solvers may have low efficiency or may have difficulty constructing solutions reliably. When used to pre-process and study specific constraint systems, however, algebraic techniques can be extremely useful and very practical.

As a result of mapping the geometric domain problem into an equational one, the geometric sense of the solutions rendered is lost. Moreover, well constrained problems are mapped to under constrained systems of equations because constraints fix the placement for each geometric element with respect each other only modulo translation and rotation. Therefore a set of additional equations must be joined to cancel these remaining degrees of freedom.

Algebraic methods can be further classified according to the specific technique used to solve the system of equations, namely into numerical, symbolic, and analysis of systems of equations.

### 3.3.1 Numerical Methods

Numerical methods provide powerful tools to solve iteratively large systems of equations. In general, a good approximation of the intended solution should be supplied to guarantee convergence. This means that if, as it is customary, the starting point is taken from the sketch defined by the user, then the sketch should be close to the intended solution. The numerical methods may offer little control over the solution in which the user is interested. To achieve robustness, numerical iterative methods must be carefully designed and implemented.

Borning, [5], Hillary and Braid, [21], and Sutherland, [59] use a *relaxation* method. This method is an alternative to the propagation method. Basically, the method perturbs the values assigned to the variables and minimizes some measure of the global error. In general, convergence to a solution is slow.

The method most widely used is the well-known *Newton-Raphson*, [34] iteration. It is used in the solvers described in [20, 45, 46, 50]. Newton-Raphson is a local method and converges much faster than relaxation. The method does not apply to consistently over constrained systems of equations unless special provisions are made such as combining it with least-squares techniques.

*Homotopy* or *continuation*, [2], is a family of methods with a growing popularity. These methods are global and guarantee convergence. Moreover, they are exhaustive and allow to determine all solutions of a constraint problem. However, their efficiency is worse than that of Newton-Raphson. Lamure and Michelucci, [41], and Durand, [13], apply this method to geometric constraint solving.

Other, less conventional methods have also been proposed. For example, in [18], Hel-Or *et al.*, introduced the *relaxed parametric design* method where the constraints are soft, that is, they do not have to be met exactly, the problem is modeled as a static stochastic process, and the resulting system of probabilistic equations is solved using the Kalman filter familiar from control theory. The Kalman filter was developed to efficiently compute linear estimators and when applied to nonlinear systems, it does not necessarily finds a solution even if one exists.

Kin *et al.*, [35], reported on a numerical method based on extended Boltzmann machines which are a sort of neural network whose goal is to minimize a given polynomial that measures the energy of the system.

### 3.3.2 Symbolic Methods

Symbolic algebraic methods compute a Gröbner basis for the given system of equations. Algorithms to compute these bases include those by Buchberger [9], and by Wu-Ritt [10, 65]. These methods, essentially, transform the system of polynomial equations into a triangular system whose solutions are those of the given system. In effect, triangularization reduces solving a simultaneous, nonlinear system to univariate root finding. Forward or a backward substitution must be used.

Buchanan *et al.*, [8], describe a solver built on top of the Buchberger's algorithm. In [36], Kondo reports a symbolic algebraic method. In [37], Kondo improves that work by generating a polynomial that summarizes the changes undergone by the system of equations.

### 3.3.3 Analysis of Systems of Equations

Methods based on the analysis of systems of equations determine whether a system is under-, well- or over-constrained from the system structure. These methods can be extended to decompose systems of equations into a set of minimal graphs which can be solved independently, [49, 57]. They can be used as a pre-processing phase for any other method, reducing the number of variables and equations that must be solved simultaneously.

Serrano, [54], applies analysis of systems of equations to select from a set of candidate constraints a well constrained, solvable subsets of equations.

## 3.4 Theorem Proving

Solving a geometric constraint problem can be seen as automatically proving a geometric theorem. However, automatic geometric theorem proving requires more general techniques and, therefore, methods which are much more complex than those required by geometric constraint solving.

Wu Wen Tsün pioneered the Wu-Ritt method, an algebraic-based geometric constraint solving method. In [65], he uses it to prove geometric theorems. The method automatically finds necessary conditions to obtain non-degenerated solutions.

In [10], Chou applies Wu's method to prove novel geometric theorems. Chou *et al.*, in [11], report on a work in automatic geometric theorem proving which allows to interpret, from a geometric point of view, the proof generated by computation.

# 4 Spatial Constraint Solving

In 2D sketching applications, a major application area of constraint solving, graph-based solvers have become dominant. The underlying reason is that for the constructs common in 2D sketching a small set of subgraphs suffices, and that the associated algebraic solution problems are rather simple. In

constrast, spatial constraint solving does not appear to have a simple and small core that suffices for most practical applications.

When approached using graph decomposition, the problem of spatial constraint solving is that simple subgraphs of, even up to 6 vertices, are numerous and many of them correspond to associated algebraic problems that are rather difficult. Note that a subgraph with 6 vertices is the smallest simultaneous spatial constraint problem involving only points and planes.

There is also the problem that no consensus has arisen of the characteristic spatial constraint problems of relevance to, say, CAD, so that there is little guidance on how to select a subset from the subgraph patterns to arrive at a compact solver that is widely applicable. In this section we review some of these issues in the context of a graph decomposition solver architecture. We begin with the problem of solving the equations associated with a selection of spatial constraint problems.

## 4.1 Sequential Construction Problems

The simplest 3-space constraint problems require placing a single geometric element (point, plane or line) with respect to a set of geometric elements whose position and orientation are known. We call such problems *sequential*, since the elements are placed one-by-one sequentially.

Many, but not all, sequential problems are easy to solve. For example, placing a point with respect to three known points requires the intersection of three spheres. Elementary algebraic manipulation reduces this task to solving a univariate quadratic equation. On the other hand, a difficult sequential problem is placing a line such that it is at prescribed distance from four known points in 3-space. Geometrically, this is equivalent to finding common tangents to four given spheres.

A lower bound on the number of tangents to four spheres was established by Macdonald *et al.*, [47], who exhibited four unit spheres that have 12 distinct, common tangents. That this bound is sharp was proved by Hoffmann and Yuan in [22] by deriving a nonlinear system of equations whose solutions give all common tangents. It was shown that the geometric degree of the system is 12, thus establishing the missing upper bound. Note that solving this system is not trivial.

## 4.2 Algebraic and Geometric Solutions

In the algebraic approach to solving specific constraint problems, one formulates a system of algebraic equations. The system is then simplified using algebraic manipulation and geometric reasoning, and if the resulting system is simple enough, its solutions can be computed with high reliability and accuracy. For spatial constraint problems, it is not necessarily easy to find such simple systems, and so computing solutions may require sophisticated algorithms, e.g., [13]. As we have seen, sequential problems may already require solving algebraic systems of degree 12.

A geometric aspproach to finding the solutions of a nonlinear equation system is the *locus method*, [16]. Instead of relying only on root finding techniques, a geometric idea is introduced:

> Drop one constraint from the problem, say a dimensional constraint $c$, resulting in an underconstrained problem. Evaluate the underconstrained configuration and measure the actual value of the dimension. Different configurations lead to different values, resulting in a curve that can be traced. This curve is the *locus* of $c$. Intersect the locus of $c$ with the nominal constraint value, usually a straight line. The resulting intersection configurations are solutions to the original system.

The method can be extended to computing the locus of more than one cut constraint, resulting in geometric manifolds of higher dimensions. Cutting two constraints we would obtain a surface that would be intersected with two planes, cutting three constraints a spatial manifold is obtained, and so on. We will illustrate the idea for octahedral problems.

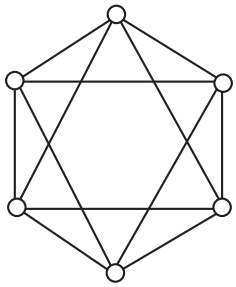


Figure 4: Octahedral graph.

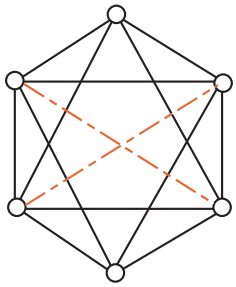


Figure 5: Octahedral graph. In red, dependent distances.

## 4.3 Octahedral Problems

Consider a constraint graph with six vertices and with edges arranged as shown in Figure 4. We call such a graph *octahedral* on account of the fact that the topology is that of the vertices and edges of a regular octahedron. The vertices represent points or planes in 3-space, and the edges therefore a distance between two points, a distance between a point and a plane, and/or an angle between two planes. There are seven major configurations according to the number of planes in the problem. Two of them, namely configurations with 5 or 6 planes, are structurally under determined. The other five configurations can be solved algebraically; [26].

An elegant approach to formulating the equation system is due to Michelucci, [48][1] who employs the Cayley-Menger determinant as follows. The determinant expresses the distances between the five points:

$$\begin{vmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & d_{12} & d_{13} & d_{14} & d_{15} \\ 1 & d_{12} & 0 & d_{23} & d_{24} & d_{25} \\ 1 & d_{13} & d_{23} & 0 & d_{34} & d_{35} \\ 1 & d_{14} & d_{24} & d_{34} & 0 & d_{45} \\ 1 & d_{15} & d_{25} & d_{35} & d_{45} & 0 \end{vmatrix} = 0$$

where the $d_{ik} = \|p_i - p_k\|^2$ is the squared distance between the points $p_i$ and $p_k$. Choosing two sets of five points in the configuration, we can express the squared distances indicated by a red, dashed line in Figure 5 as function of the given distances. This results in two bivariate equations of degree 4 each in the two unknown distances, indicating at most 16 distinct solutions. That the bound of 16 solutions is sharp was shown earlier by Hoffmann and Vermeer in [26]. Suitable extensions of the Cayley-Menger determinant can be used for the remaining cases comprising both points and planes.

A solution of the two quartic equations can be obtained by the locus method, for example. Each

[1]The URL is no longer valid.

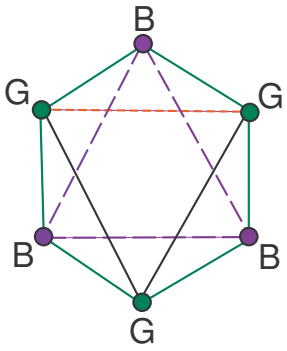



Figure 6: Octahedral problem. The locus method.

quartic equation taken separately yields a plane curve. Tracing these curves can then give initial values for an iterative numerical method that isolates the up to 16 roots of the system.

The locus method can also be used to plot directly the value of a cut constraint, and we explain the approach assuming only points in the configuration. Consider Figure 6. Select the three blue vertices, labeled B, and cut the red, dotted-line constraint (between two green vertices labeled G). The three blue vertices B can be positioned using the blue constraints only, indicated by dashed lines, in the $xy$-plane for example.

Each of the green vertices, labeled G in the figure, together with the adjacent blue vertices, labeled B, can be thought of as a rigid triangle that pivots about a blue, dashed edge, and so each green vertex moves on a circle whose plane is perpendicular to the $xy$-plane. Select one green vertex and parameterize its position on the circle on which it moves, say by $\theta$. For a given $\theta$ value, we can now compute the position of the remaining two green vertices using the black constraints, indicated by solid lines between the G vertices, intersecting a circle and a sphere. Once these vertices have been placed, we can compute the distance $d$ between the two vertices adjacent to the cut, red constraint, so plotting a curve in the $\theta d$-plane. Intersecting the curve with the line $d = d_0$, where $d_0$ is the distance stipulated by the red constraint, we obtain all solutions to the original problem.

## 4.4 Line Configurations

In the discussion of sequential problems before, we have seen that sequential line problems in 3-space may yield algebraic equation systems of considerable complexity. In addition, there is a large number of individual problems involving only a small number of geometric elements. We illustrate the combinatorial explosion of cases involving constraint graphs with 6 or fewer vertices representing points, lines and planes.

Note that between two planes, between to points, and between a point and a plane we can have at most one constraint, namely of angle, of distance, and of distance, respectively. Hoffmann and Vermeer show in [26] that the octahedron problem is the simplest, nonsequential problem involving only points and planes. As shown in that paper, there are exactly seven distinct major[2] configurations, two of them underconstrained.

We can have at most a single constraint between a point and a line (distance), and a single constraint between a plane and a line (angle). However, between two lines we can have up to two constraints (distance and angle). As Gao *et al.* show in [16], there are two distinct nonsequential constraint problems with four lines, shown in Figure 7.

[2]Some of the major configurations have several sub configurations. For instance, in a problem with two planes, the planes could have an angle constraint between them, or no constraint them, leading to two sub configurations for this case.

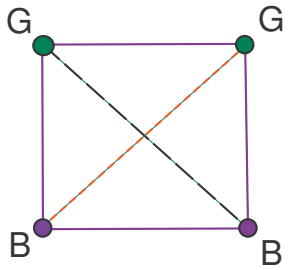



Figure 7: The four lines problem.

The blue, solid lines represent two constraints, of angle and distance, and the black, dashed line represents a distance constraint. The two configurations differ in the red, dotted line constraint, a distance constraint in one and an angle constraint in the other configuration.

In [16], Gao *et al.* also establish that there are 17 distinct configurations with 5 geometric elements, including lines, but more than 680 configurations with 6 geometric elements. These numbers show that a solver, based on decomposing a spatial constraint problem into a (recursive) set of small subproblems must have a very large repertoire of subproblem patterns, even when allowing only up to six geometric elements. Moreover, the algebraic structure of the many configurations involving lines remain unexplored.

## Acknowledgements

I wish to acknowledge my collaborators over the years, especially M. Freixas, I. Fudos, C. M. Hoffmann, A. Soto, M. Tarrès, Sebastià Vila. Much of the material reviewed has been to their credit. R. Joan-Arinyo has been supported by The Ministerio de Educación y Ciencia and by FEDER under grant TIN2007-67474-C03-01.

## References

[1] B. Aldefeld. Variation of geometric based on a geometric-reasoning method. *Computer-Aided Design*, 20(3):117–126, April 1988.

[2] Eugene L. Allgower and Kurt Georg. Continuation and path following. *Acta Numerica*, 7:1–64, 1993.

[3] S. Bhansali, G.A. Kramer, and T.J. Hoar. A principled approach towards symbolic geometric constraint satisfaction. *Journal of Artificial Intelligence Research*, (4):419–443, 1996.

[4] A. Borning, R. Anderson, and B. Freeman-Benson. Indigo: A local propagation algorithm for inequality constraints. In *UIST'96*, pages 129–136, Seattle, Washington, USA, November 6-8 1996. ACM.

[5] A.H. Borning. The programming language aspects of ThingLab, a constrained oriented simulation laboratory. *ACM Trans. on Prog. Lang. and Systems*, 3(4):353–387, October 1981.

[6] B.D. Brüderlin. Constructing three-dimensional geometric objects defined by constraints. In *Workshop on Interactive 3D Graphics*, pages 111–129. ACM, October 23–24 1986.

[7] B.D. Brüderlin. Using geometric rewrite rules for solving geometric problems symbolically. In *Theoretical Computer Science 116*, pages 291–303. Elsevier Science Publishers B.V., 1993.

[8] S.A. Buchanan and A. de Penington. Constraint definition system: a computer-algebra based approach to solving geometric-constraint problems. *Computer-Aided Design*, 25(12):741–750, December 1993.

[9] B. Buchberger. *Multidimensional Systems Theory*, chapter Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory, pages 184–232. D. Reidel Publishing Theory, 1985.

[10] S.-C. Chou. An introduction to Wu's method for mechanical theorem proving in geometry. *Journal of Automated Reasoning*, 4:237–267, 1988.

[11] S.-C. Chou, X.-S. Gao, and J.-Z. Zhang. Automated generation of readable proofs with geometric invariants: Multiple and shortest proof generation. *Journal of Automated Reasoning*, 7:325–347, 1996.

[12] H.M. Cundy and A.P. Rollet. *Mathematical Models*. Oxford University Press, 1961. Second edition.

[13] C. Durand. *Symbolic and Numerical Techniques for Constraint Solving*. PhD thesis, Computer Science, Purdue University, December 1998.

[14] B. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. *Communications of the ACM*, 33(1):54–63, 1990.

[15] I. Fudos and C.M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, April 1997.

[16] X.-S. Gao, C. Hoffmann, and W.-Q. Yang. Solving spatial basic geometric constraint configurations with locus intersection. In *Solid Modeling SM'02*, Saarbrücken, Germany, 2002.

[17] J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*. American Mathematical Society, 1993.

[18] Y. Hel-Or, A. Rapoport, and M.Werman. Relaxed parametric design with probabilistic constraints. In J. Rossignac, J. Turner, and G. Allen, editors, *Second Symposium on Solid Modeling and Applications*, pages 261–270, Montreal, Canada, May 19-21 1993. ACM Press.

[19] L. Henneberg. *Die Graphische Statik de Starren Systeme*. Leipzig, 1911.

[20] A. Heydon and G. Nelson. The Juno-2 constraint-based drawing editor. Research Report 131a, Digital Systems Research Center, December 1994.

[21] R.C. Hillyard and I.C. Braid. Characterizing non-ideal shapes in terms of dimensions and tolerances. In *ACM Computer Graphics*, pages 234–238, 1978.

[22] C. Hoffmann and B. Yuan. There are 12 common tangents to four spheres. http://www.cs.purdue.edu /homes/cmh/distribution/ SphereTangents.htm, 2000.

[23] C.M. Hoffmann and R. Joan-Arinyo. Symbolic constraints in constructive geometric constraint solving. *Journal of Symbolic Computation*, 23:287–300, 1997.

[24] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In *Principles and Practice of Constraint Programming*, pages 463–477, Schloss Hagenberg, Austria, October 29 - November 1 1977.

[25] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Decompostion Plans for Geometric Constraint Systems, Part I: Performance Measurements for CAD. *Journal of Symbolic Computation*, 31:367–408, 2001.

[26] C.M. Hoffmann and P.J. Vermeer. Geometric constraint solving in $R^2$ and $R^3$. In D.-Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*, pages 266–298. World Scientific Publishing, 1995.

[27] C.M. Hoffmann and P.J. Vermeer. A spatial constraint problem. In J.P. Merlet and B. Ravani, editors, *Computational Kinematics'95*, pages 83–92. Kluwer Academic Publ., 1995.

[28] C.-Y. Hsu and B.D. Brüderlin. A hybrid constraint solver using exact and iterative geometric constructions. In D. Roller and P. Brunet, editors, *CAD Systems Development: Tools and Methods*, pages 265–279, Berlin, 1997. Springer-Verlag.

[29] C. Jermann. *Résolution de constraintes géométriques par rigidification récursive et propagation d'intervalles*. PhD thesis, Université de Nice-Sophia Antipolis, 2002. (Written in French).

[30] R. Joan-Arinyo and A. Soto. A correct rule-based geometric constraint solver. *Computer & Graphics*, 21(5):599–609, 1997.

[31] R. Joan-Arinyo and A. Soto-Riera. Combining constructive and equational geometric constraint solving techniques. *ACM Transactions on Graphics*, 18(1):35–55, January 1999.

[32] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana. Declarative characterization of a general architecture for constructive geometric constraint solvers. In D. Plemenos, editor, *The Fifth International Conference on Computer Graphics and Artificial Intelligence*, pages 63–76, Limoges, France, 14-15 May 2002. Université de Limoges.

[33] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana. Revisiting decomposition analysis of geometric constraint graphs. In K. Lee and N. Patrikalaikis, editors, *Seventh Symposium on Solid Modeling and Applications*, pages 105–115, Saarbrücken, Germany, June 19-21 2002. ACM Press.

[34] L. W. Johnson and R. D. Riess. *Numerical analysis*. Addison-Wesley, 1982. Second edition.

[35] N. Kin, Y. Takai, and T.L. Kunii. A connectionist approach to geometrical constraint-solving. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics*. Springer Verlag, 1993.

[36] K. Kondo. PIGMOD : Parametric and interactive geometric modeller for mechanical design. *Computer Aided Design*, 22(10):633–644, December 1990.

[37] K. Kondo. Algebraic method for manipulation of dimensional relationships in geometric models. *Computer Aided Design*, 24(3):141–147, March 1992.

[38] G. Kramer. *Solving Geometric Constraints Systems*. MIT Press, 1992.

[39] G. Kwaiter, V. Gaildrat, and R. Caubet. Interactive constraint system for solid modeling objects. In C. Hoffmann and W. Bronsvoort, editors, *Fourth Symposium on Solid Modeling and Applications*, pages 265–270. ACM SIGGRAPH, May 1997.

[40] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4(4):331–340, October 1970.

[41] H. Lamure and D. Michelucci. Solving geometric constraints by homotopy. In C. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 263–269, Salt Lake City, Utah USA, May 17-19 1995. ACM Press.

[42] R.S. Latham and A.E. Middleditch. Connectivity analysis: a tool for processing geometric constraints. *Computer Aided Design*, 28(11):917–928, November 1996.

[43] J.Y. Lee and K. Kim. Geometric reasoning for knowledge-based parametric design using graph representations. *Computer-Aided Design*, 28(10):831–841, 1996.

[44] W. Leler. *Constraint Programming Languages: Their Specification and Generation*. Addison Wesley, 1988.

[45] R. Light and D. Gossard. Modification of geometric models through variational geometry. *Computer Aided Design*, 14:209–214, July 1982.

[46] V.C. Lin, D.C. Gossard, and R.A. Light. Variational geometry in computer-aided design. *ACM Computer Graphics*, 15(3):171–177, August 1981.

[47] I. Macdonald, J. Pach, and T. Theobald. Common tangents to four unit balls. *Discr. Comp. Geometry*, 26:1–17, 2001.

[48] D. Michelucci. Using Cayley-Menger determinants. http://www.emse.fr/~micheluc/MENGER/. approx. from 2000; URL no longer valid.

[49] K. Murota. *Systems Analysis by Graphs and Matroids*. Algorithms and Combinatorics 3. Springer-Verlag, 1987.

[50] G. Nelson. Juno, a constraint-based graphics system. *SIGGRAPH*, pages 235–243, San Francisco, July 22–26 1985.

[51] J.C. Owen. Algebraic solution for geometry from dimensional constraints. In R. Rossignac and J. Turner, editors, *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 397–407, Austin, TX, June 5-7 1991. ACM Press.

[52] O.W. Salomons, F. van Slooten, F.J.A.M van Houten, and H.J.J. Kals. Conceptual graphs in constraint based re-design. In C. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 55–64, Salt Lake City, Utah USA, May 17-19 1995. ACM Press.

[53] M. Sannella. The SkyBlue constraint solver. Technical Report 92-07-02, University of Washington, Dep of Computer Science and Engineering, 1993.

[54] D. Serrano. Automatic dimensioning in design for manufacturing. In J. Rossignac and J. Turner, editors, *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 379–386, Austin, TX, June 5-7 1991. ACM Press.

[55] W. Sohrt and B.D. Brüderlin. Interaction with constraints in 3D modeling. *International Journal of Computational Geometry & Applications*, 1(4):405–425, 1991.

[56] L. Solano and P. Brunet. Constructive constraint-based model for parametric CAD systems. *Computer Aided Design*, 26(8):614–621, August 1994.

[57] N. Sridhar, R. Agrawal, and G.L. Kinzel. Algorithms for the structural diagnosis and decomposition of sparse, underconstrained design systems. *Computer-Aided Design*, 28(4):237–249, 1996.

[58] G. Sunde. A CAD system with declarative specification of shape. *Eurographics Workshop on Intelligent CAD Systems*, pages 90–105, April 1987.

[59] I. Sutherland. Sketchpad, a man-machine graphical communication system. In *Proc. of the Spring Joint Comp. Conference*, pages 329–345. IFIPS, 1963.

[60] P. Todd. A k-tree generalization that characterizes consistency of dimensioned engineering drawings. *SIAM J. Disc. Math*, 2(2):255–261, 1989.

[61] R.C. Veltkamp and F. Arbab. Geometric constraint propagation with quantum labels. In B. Falcidieno, I. Herman, and C. Pienovi, editors, *Eurographics Workshop on Computer Graphics and Mathematics*, pages 211–228. Springer, 1992.

[62] A. Verroust. *Etude de Problèmes Liés à la Définition, la Visualisation et l'Animation d'Objects Complexes en Informatique Graphique*. PhD thesis, Universite de Paris-Sud, Centre d'Orsay, 1990. (Written in French).

[63] A. Verroust, F. Schonek, and D. Roller. Rule-oriented method for parameterized computer-aided design. *Computer Aided Design*, 24(10):531–540, October 1992.

[64] W. Whiteley. Rigidity and scene analysis. In J.E. Goodman and J. O'Rourke, editors, *Handbook for discrete and computational geometry*, pages 893–916. CRC Press LLC, 1998.

[65] W.-T. Wu. Mechanical theorem proving in geometries. In B. Buchberger and G. E. Collins, editors, *Texts and monographs in symbolic computations*. Springer-Verlag, 1994.

[66] Y. Yamaguchi and F. Kimura. A constraint modeling system for variational geometry. In J.U. Turner M.J. Wozny and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 221–233. Elsevier North Holland, 1990.

# Algorithms for graphs embedded on surfaces

Sergio Cabello*

**Abstract**

Graphs embedded on surfaces are a natural generalization of plane graphs. Any cycle in an embedded graph defines a closed curve in the surface and we can therefore consider topological properties of such curve. When the graph is equipped with edge-weights, we can also talk about the length of cycles. We will give an overview of the algorithms and techniques used to find shortest cycles with certain topological properties in embedded graphs, like for example a shortest non-contractible cycle.

## 1 Introduction

Let $G$ be a graph embedded in a surface $\Sigma$. Closed walks in $G$ correspond to closed curves in $\Sigma$, and hence we may ask for shortest closed walks with certain topological properties, such as being non-contractible. The term 'cycle' has different meanings in topology and graph theory. For the rest of this paper the term *cycle* will be used for a *closed walk without repeated vertices*. We will use *closed curve* for any closed curve in the surface, possibly crossing along faces of the embedded graph.

We shall concentrate our discussion on shortest cycles and shortest closed walks with prescribed topological properties. The input to our problems will be a graph embedded in a surface of genus $g$. We will use $E$ to denote the number of edges of the input graph, which can be seen as the size of the input.

**Applications.** A mesh representing the boundary of a 3-dimensional object is a graph embedded on a surface. Cutting the surface into a topologically trivial surface is useful for parameterization or adding texture to an object [12]. When the mesh is obtained as a reconstruction of a surface from sample points, it often contains topological noise. Techniques for removing this noise in the graphical model or for finding topological meaningful parts share several ideas with the problem of finding shortest non-trivial cycles in embedded graphs [10, 15, 24]. However, it is important to note that algorithms for surfaces do not use or assume an ambient space, whereas this is an important feature to exploit in several applications.

The crossing number of a graph is the minimum $k$ such that the graph can be drawn in the plane with at most $k$ crossings. (An edge may be represented by arbitrary curves, not only straight-line segments.) If a graph has crossing number $k$, then it can be embedded in a surface of genus $k$ by placing a handle in each crossing. The reversed implication is not true in general. However, when a graph is embedded in a surface densely enough (large *face-width* is the precise term), one can obtain lower bounds on the crossing numbers. Following this approach, Kawarabayashi and Reed [17] have shown the following: for each constant $k > 0$ there exists a linear-time algorithm that decides if the crossing number of an input graph is bounded by $k$. A key subroutine in their algorithm is based on computing (a 2-approximation to) the shortest non-contractible cycle in an embedded graph.

Algorithms have been proposed for optimization problems when the input graph is embedded in a surface [1, 9]. It seems that there is a catch in the model: the input is an embedded graph, but how

*Department of Mathematics, IMFM, and Department of Mathematics, FMF, University of Ljubljana, Slovenia. E-mail: sergio.cabello@fmf.uni-lj.si. Research partially supported by the Slovenian Research Agency, program P1-0297.

| | Cycle | Closed walk |
|---|---|---|
| Contractible | $O(E^2 \log E)$ [2] | $O(E \log^2 E)$ [4] |
| Separating | NP-hard [2] | ??? FPT wrt $g$ |
| Non-contractible | $O(\min\{g^3, E\}E \log E)$ [3, 12] | $\leftarrow$ same |
| Non-separating | $O(\min\{g^3, E\}E \log E)$ [3, 12] | $\leftarrow$ same |
| Tight | $\uparrow$ same | $O(E \log E)$ [4] |
| Splitting | NP-hard [6] | NP-hard, FPT wrt $g$ [6] |
| Prescribed homotopy | ??? | $O(gE \log E + gEk_{in} \log(nk_{in}))$ [8, 4] |

Table 1: Summary of results to find shortest cycles or closed walks.

do we obtain the embedding? In general, finding the surface with smallest genus where a graph can be embedded is NP-hard. However, for any fixed surface there is a linear-time algorithm to decide if an input graph can be embedded in the surface [20]. Therefore, these algorithms also work for graphs with bounded genus or bounded crossing number, even when no embedding is provided.

### 1.1 Summary of results

In table 1 there is a summary of the known results for finding shortest cycles or closed walks with prescribed topological properties.

## 2 Background

Our presentation will be using terminology mostly used for graphs embedded in surfaces [21]. An alternative option would be using the terminology of combinatorial surfaces [8].

**Topology.** We review some basic topology of surfaces. See any of the books [16, 19, 22] for a comprehensive treatment. Two topologial spaces $X, Y$ are homeomorphic if there is bijection between $X$ and $Y$ which is continous and whose inverse is also continous. We think of homeomorphic spaces as topologically equivalent: any topological property holds for $X$ if and only if it holds for $Y$.

A *surface* (or 2-manifold) $\Sigma$ is a compact topological space where each point has a neighborhood homeomorphic to the plane. A surface is *non-orientable* if it contains a subset homeomorphic to the Möbius band, and *orientable* otherwise. An orientable surface is homeomorphic to a sphere with $g$ handles attached to it, for a unique number $g \geq 0$. A non-orientable surface is homeomorphic to the connected sum of $g$ projective planes, for a unique $g \geq 0$. In both cases, $g$ is the called the *genus* of the surface. For the rest of the presentation, we will consider only orientable surfaces.

A *path* in $\Sigma$ is a continuous mapping from $[0, 1]$ to $\Sigma$; its endpoints are the image of 0 and 1. If the endpoints agree, the path is a *loop*, and the endpoint is its *basepoint*. A *closed curve* is a continuous mapping from $\mathbb{S}^1$ to $\Sigma$. A loop naturally defines a cycle. The concatenation of two paths $\alpha$ and $\beta$ that fullfill $\alpha(1) = \beta(0)$ is the path $\alpha \cdot \beta$ defined by

$$(\alpha \cdot \beta)(t) = \mathbf{1}_{[0,1/2]}(t)\, \alpha(2t) + \mathbf{1}_{(1/2,1]}(t)\, \alpha(2t-1).$$

The reversal $\alpha^R$ of a path $\alpha$ is $\alpha^R(t) = \alpha(1-t)$ A path or a closed curve is *simple* when the mapping is injective.

Two closed curves $\alpha, \beta$ are (freely) *homotopic* if there is a continuous function $h : [0, 1] \times \mathbb{S}^1 \to \Sigma$ such that $\alpha(\cdot) = h(0, \cdot)$ and $\beta(\cdot) = h(1, \cdot)$. A closed curve is *contractible* if it is homotopic to a constant mapping. A simple closed curve is identified by its image, up to homotopy and reversal.

Cutting the surface along a simple contractible closed curve gives two connected components, and one of them is a topological disk. A simple closed curve $\alpha$ is *separating* if cutting the surface along (the image of) $\alpha$ gives rise to two connected components. The concept of separating is meaningfull

Figure 1: Cutting a surface along a cycle.

only for simple curves. For general curves, it is convenient to deal with $\mathbb{Z}_2$-homology. A simple closed curve is separating if and only if it is zero-homologous. A simple, contractible, closed curve is also a separating curve.

Two curves $\alpha, \beta$ *cross* $c$ *times* if the following two conditions hold: (i) there exist infinitesimal continuous perturbations of $\alpha, \beta$ that cross transversally $c$ times; and (ii) any infinitesimal continuous perturbations of $\alpha, \beta$ have at least $c$ points in common.

**Graphs and embeddings.** For a spanning tree $T$ of a graph $G$ and an edge $e \in E(G) \setminus E(T)$, we use $\texttt{cycle}(T, e)$ to denote the unique cycle contained in $T + e$.

An embedding of a graph in a surface $\Sigma$ assigns a distinct point of $\Sigma$ to each vertex of the graph, and assigns a simple path in $\Sigma$ to each edge, such that the endpoints of the path agree with the point assigned to the vertices of the edge. Moreover, paths assigned to distinct edges are disjoint, except at common endpoints. It is costumary to consider only *cellular embeddings* of a graph, where the removal of the image of the graph leaves a set of topological disks.

We assume that the embedding is represented in a suitable way, like for example the gem representation discussed by Eppstein [11]. For orientable surfaces, one can also use the DCEL that is costumary in Computational Geometry.

**Walks and surgery.** As mentioned before, the term *cycle* shall refer to a closed walk without repeated vertices. Homotopy of closed walks and cycles is the same as homotopy of closed curves. We say that a walk in the graph is simple when it can be infinitessimally deformed to a simple path or cycle.

Let $G$ be an embedded graph and let $\alpha$ be a cycle in $G$. We use the notation $G \not| \alpha$ to denote the embedded graph obtained after cutting the surface that receives the embedding along the edges of $\alpha$. This can be seen as unglueing the two faces adjacent to an edge $e$ for each edge $e$ in $\alpha$. One may then attach topological disks to keep having a surface, or keep working with a surface with boundary. See Figure 1. The representation of the embedding should allow to cut along a cycle in time proportional to the length of the cycle.

**Lengths.** We assume that a graph $G$ has positive edge-weights. The length $|\alpha|$ of a walk $\alpha$ is defined as the sum of the weights of the edges along $\alpha$, counted with multiplicity. A walk is *tight* if it is shortest in its homotopy class. Some arguments can be simplified if we are willing to assume that that there is a unique shortest path between any two vertices of the graph. This condition can be probabilistically ensured adding random noise to each edge length [12].

# 3 Some relevant techniques

We shall review some standard techniques that are used in the area.

## 3.1 3-path condition

The following concept was introduced and explored by Thomassen [23]; see also the monograph by Mohar and Thomassen [21, Section 4.3].

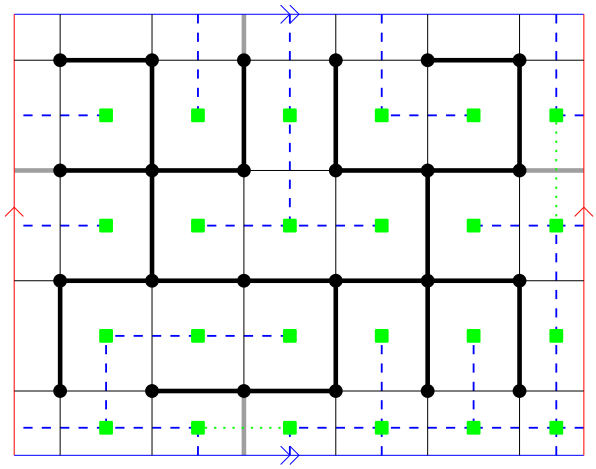


Figure 2: A tree-cotree decomposition of a regular grid in the torus.

**Definition 3.1.** A family of closed walks $\mathcal{W}$ satisfies the *3-path condition* if the following holds: for any three walks $\alpha_1, \alpha_2, \alpha_3$ with the same starting vertex and the same ending vertex, if the closed walks $\alpha_1 \cdot \alpha_2^R$ and $\alpha_2 \cdot \alpha_3^R$ are not in $\mathcal{W}$, then the closed walk $\alpha_1 \cdot \alpha_3^R$ is also not in $\mathcal{W}$.

The family of closed walks with an odd number of edges, the family of contractible closed walks, and the family of zero-homologous closed walks satisfy the 3-path condition.

When $\mathcal{W}$ satisfies the 3-path condition, then a shortest element of $\mathcal{W}$ is a cycle. Therefore, the problems of finding a shortest closed walk and finding a shortest cycle in $\mathcal{W}$ are equivalent. Moreover, such shortest cycle has no chords, that is, the shortest path between any two points of the cycle has to be a subpath of the cycle.

If $\mathcal{W}$ satisfies the 3-path condition and membership in $\mathcal{W}$ can be tested in polynomial time, then a shortest cycle in $\mathcal{W}$ can be found in polynomial time using the so-called *fundamental cycle method* [23, 21]. This method consists on considering as candidate cycles the cycles $\texttt{cycle}(T, e)$, where $T$ iterates over the shortest-path trees of the graph and $e$ iterates over the set of edges $E(G) \setminus E(T)$, and taking the shortest one that is contained in the family $\mathcal{W}$. As particular cases, one obtains polynomial-time algorithms to find a shortest non-contractible cycle and a shortest non-separating cycle in an embedded graph.

## 3.2 Tree-cotree

Let $G$ be a (multi)graph embedded in a surface. The dual (multi) graph $G^*$ is defined as follows. Each face $f$ of $G$ defines a node of $G^*$, and each edge $e$ that defines the boundary of two (possibly equal) faces $f_0, f_1$ of $G$ defines an edge $e^*$ in $G^*$ connecting the nodes $(f_0)^*, (f_1)^*$. A *cotree* of $G$ is a subgraph $C$ of $G$ whose dual edges $\{e^* \mid e \in C\}$ define a spanning tree of the dual graph $G^*$.

Let $T$ be a spanning tree of $G$, and let $C$ be a cotree that is edge-disjoint from $T$. If $G$ is a planar graph, then $E(G) = E(T) \cup E(C)$, and $C$ is uniquely defined by $T$. When $G$ is a graph embedded in a surface of genus $g$, there are the "leftover" edges $X = E(G) \setminus (E(T) \cup E(C))$. The triple $(T, C, X)$ forms a tree-cotree decomposition of $G$ [11]. Euler's formula shows that $X$ has $2g$ edges when the surface is orientable. Figure 2 shows an example in the torus.

Consider a tree-cotree decomposition $(T, C, X)$ of $G$. The cycles $\texttt{cycle}(T, e)$, $e \in X$, form a cutset: cutting the surface along them leaves a topological disk [11], possibly with holes. Let $H$ be the subgraph of $G$ with edges $E(C) \cup X$. With a slight abuse of notation, we denote by $H^*$ the subgraph of $G^*$ that contains the edges $\{e^* \mid e \in E(H)\}$. The graph $H^*$ is always connected because $T$ does not contain any cycle. For any edge $e \in E(H)$, the cycle $\texttt{cycle}(T, e)$ is separating if and only if $H^* - e^*$ has two connected components. Moreover, $\texttt{cycle}(T, e)$ is contractible if and only if $H^* - e^*$ has two components, and one of them is a tree.

Typically, one considers tree-cotree decompositions $(T, C, X)$ where $T$ is a shortest path tree or a minimum spanning tree. Tree-cotree decompositions can be used to simplify previous algorithms to find shortest non-separating or non-contractible cycles, and to construct data structures that quickly decide if $\texttt{cycle}(T, e)$ has certain topological property [14]. They have also been used to construct a shortest set of loops with given basepoint [13], and to dynamically maintain a shortest path tree as the root of the tree moves along the graph [3].

## 3.3 Crossing patterns

Let $\mathcal{W}$ be a family of shortest closed walks or cycles with certain topological property. In many cases, it is possible to show that there is a shortest element in $\mathcal{W}$ that crosses certain curves according to certain patterns, which narrows down the search for shortest elements in $\mathcal{W}$. This property was first used by Erickson and Har-Peled [12] to obtain a 2-approximation to the shortest non-contractible or non-separating cycle. Here we show some examples of this technique.

A closed curve in the surface hosting the embedding is separating if and only if it crosses any other closed curve an even number of times. The following statement shows that to decide if a closed curve is non-separating it is enough to check how many times it crosses a finite family of closed curves.

**Lemma 3.2** ([5]). *Let $(T, C, X)$ be a tree-cotree decomposition of $G$ and consider the set of cycles $\mathcal{Q}_{(T,U,X)} = \{\texttt{cycle}(T, e) \mid e \in X\}$. A cycle $Q$ in $G$ is non-separating if and only if there is some cycle $Q_e \in \mathcal{Q}_{(T,U,X)}$ such that $Q$ and $Q_e$ cross an odd number of times.*

If we choose $T$ to be a shortest path tree, then each cycle $Q_e \in \mathcal{Q}_{(T,U,X)}$ is made of two shortest paths plus an edge. Using the 3-path condition one can then argue that a shortest non-separating cycle crosses $Q_e$ at most twice. Together with the previous lemma leads to the following property: there is a shortest non-separating cycle in $G$ that crosses some $Q_e \in \mathcal{Q}_{(T,U,X)}$ exactly once.

Here we have another statement that has been useful to find a tight closed walk in a surface.

**Lemma 3.3** ([4]). *Let $\ell_x$ be a shortest non-separating closed walk in $G$ through a given basevertex $x$. There exists a shortest closed walk homotopic to $\ell_x$ that does not cross $\ell_x$.*

Finally, in some cases the crossing pattern is more complicated. The following characterization was used to show that finding a shortest splitting closed walk is fixed-parameter tractable with respect to the genus.

**Lemma 3.4** ([6]). *Let $P$ be a family of shortest paths in $G$ that are interior disjoint. There is a shortest splitting closed walk that crosses each path in $P$ at most $O(g)$ times.*

The authors also show that this statement is best possible. In some cases, a shortest path may be crossed several times by a shortest splitting closed walk.

## 3.4 Distances

Let $\alpha$ be a cycle in an embedded graph. How does it look a shortest cycle $Q_{\texttt{short}}$ that crosses $\alpha$ exactly once? If $Q_{\texttt{short}}$ crosses $\alpha$ in a vertex $v$, then $Q_{\texttt{short}}$ corresponds to the shortest path in $G\not\!\!\!\!\alpha$ between the two copies $v', v''$ of $v$. Therefore, if we denote by $u', u''$ the two copies in $G\not\!\!\!\!\alpha$ of any vertex $u \in V(\alpha)$, it is clear that it is convenient to have a tool to obtain the distances between vertices $u', u''$. The key observation is that the vertices $u', u''$ are contained in two faces of $G\not\!\!\!\!\alpha$, which allows for better data structures. The following result extends a result by Klein [18], which was only for plane graphs.

**Theorem 3.5** ([3]). *Let $G$ be a graph with $n$ vertices embedded in a surface of genus $g$, and let be $f$ a face of $G$. With $O(g^2 n \log n)$ preprocessing time, a shortest path distance from any vertex on $f$ to any other vertex can be found in $O(\log n)$ time.*

# 4 Further work

There are several problems where progress would be welcome. Here are some examples:

- Computing the genus of a graph is NP-hard. Is there a constant-factor approximation?
- Computing the crossing number of a graph is NP-hard, but fixed-parameter tractable. Is the geometric (or rectilinear) crossing number fixed-parameter tractable?
- It has been shown that flow problems can be solved faster in embedded graphs [7]. Can max-flow problems be solved in $O(E \log E)$ time for graphs embedded in surfaces of bounded genus?
- The edge-width of an embedded graph $G$ is the shortest non-contractible cycle of $G$ when $G$ has unit edge-weights. In this case, shortest path trees are BFS trees, and can be computed in linear time easily. Can the edge-width be computed in linear time for graphs embedded on surfaces of bounded genus?

# References

[1] G. Borradaile, E. D. Demaine, and S. Tazari. Polynomial-time approximation schemes for subset-connectivity problems in bounded-genus graphs. In S. Albers and J.-Y. Marion, editors, *STACS*, volume 09001 of *Dagstuhl Seminar Proceedings*, pages 171–182. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2009.

[2] S. Cabello. Finding shortest contractible and shortest separating cycles in embedded graphs. *ACM Trans. Algorithms*, in press.

[3] S. Cabello and E. W. Chambers. Multiple source shortest paths in a genus $g$ graph. In *SODA '07: Proc. 18th Symp. Discrete Algorithms*, pages 89–97, 2007.

[4] S. Cabello, M. DeVos, J. Erickson, and B. Mohar. Finding one tight cycle. Manuscript available at `http://www.imfm.si/preprinti/PDF/01047.pdf`; preliminary version in SODA '08: Proc. 19th Symp. Discrete Algorithms.

[5] S. Cabello and B. Mohar. Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. *DCG*, 37:213–235, 2007. Preliminary version in ESA '05.

[6] E. W. Chambers, É. Colin de Verdière, J. Erickson, F. Lazarus, and K. Whittlesey. Splitting (complicated) surfaces is hard. *Computational Geometry: Theory & Applications*, 41:94–110, 2008. Preliminary version in SOCG '06.

[7] E. W. Chambers, J. Erickson, and A. Nayyeri. Homology flows, cohomology cuts. In *STOC 2009*, page to appear, 2009.

[8] É. Colin de Verdière and J. Erickson. Tightening non-simple paths and cycles on surfaces. In *SODA '06: Proc. 17th Symp. Discrete Algorithms*, pages 192–201, 2006.

[9] E. D. Demaine, M. Hajiaghayi, and B. Mohar. Approximation algorithms via contraction decomposition. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 278–287, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[10] T. K. Dey, K. Li, J. Sun, and D. Cohen-Steiner. Computing geometry-aware handle and tunnel loops in 3d models. *ACM Trans. Graph.*, 27(3):1–9, 2008.

[11] D. Eppstein. Dynamic generators of topologically embedded graphs. In *SODA '03: Proc. 14th Symp. Discrete Algorithms*, pages 599–608, 2003.

[12] J. Erickson and S. Har-Peled. Optimally cutting a surface into a disk. *DCG*, 31:37–59, 2004. Preliminary version in SOCG '02.

[13] J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. In *SODA '03: Proc. 16th Symp. Discrete Algorithms*, pages 1038–1046, 2005.

[14] J. Erickson and P. Worah. Computing the shortest essential cycle. Available at `http://compgeom.cs.uiuc.edu/~jeffe/pubs/essential.html`, 2009.

[15] I. Guskov and Z. Wood. Topological noise removal. In *Graphics Interface*, 2001.

[16] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001. Available at `http://www.math.cornell.edu/~hatcher/`.

[17] K. Kawarabayashi and B. Reed. Computing crossing number in linear time. In *STOC '07: Proc. 39th ACM Symp. Theory Comput.*, pages 382–390, 2007.

[18] P. N. Klein. Multiple-source shortest paths in planar graphs. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 146–155, Philadelphia, PA, USA, 2005. SIAM.

[19] W. S. Massey. *Algebraic Topology: An Introduction*. Springer Verlag, 1967.

[20] B. Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discret. Math.*, 12(1):6–26, 1999.

[21] B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, Baltimore, 2001.

[22] J. Stillwell. *Classical Topology and Combinatorial Group Theory*. Springer-Verlag, New York, 1993.

[23] C. Thomassen. Embeddings of graphs with no short noncontractible cycles. *J. Combin. Theory, Ser. B*, 48:155–177, 1990.

[24] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Removing excess topology from isosurfaces. *ACM Trans. Graph.*, 23(2):190–208, 2004.

# [Empty][colored] $k$-gons - Recent results on some Erdős-Szekeres type problems

Oswin Aichholzer*

**Abstract**

We consider a family of problems which are based on a question posed by Erdős and Szekeres in 1935: "What is the smallest integer $g(k)$ such that any set of $g(k)$ points in the plane contains at least one convex $k$-gon?" In the mathematical history this has become well known as the "Happy End Problem". There are several variations of this problem: The $k$-gons might be required to be empty, that is, to not contain any points of the set in their interior. In addition the points can be colored, and we look for monochromatic $k$-gons, meaning polygons spanned by points of the same color. Beside the pure existence question we are also interested in the asymptotic behavior, for example whether there are super-linear many $k$-gons of some type. And finally, for several of these problems even small non-convex $k$-gons are of interest.

We will survey recent progress and discuss open questions for this class of problems.

## 1 Introduction

In 1935 Erdős and Szekeres [22] considered a problem about the existence of a number $g(k)$ such that any set $S$ of $g(k)$ points in general position in the plane has a subset of $k$ points that are the vertices of a convex $k$-gon. Later Erdős and Guy [21] stated the following more general question. "What is the least number of convex $k$-gons determined by any set of $n$ points in the plane?".

Both versions turned out to be rather challenging and have attracted many researchers. Meanwhile there exists a whole family of problems based on these questions and we will survey recent results and pose open problems for some of them. More specifically we will discuss the following variants:

- **General vs. empty $k$-gons:** A $k$-gon is called empty, or for short a $k$-*hole*, if it does not contain any points of the set in its interior.
- **Convex vs. non-convex:** We will consider different levels of non-convexity for $k$-gons and $k$-holes using the recent notion of $j$-convexity.
- **Colored point sets:** If the underlying point set is colored, $k$-gons and $k$-holes are required to be monochromatic, that is, they are spanned by points of the same color.

The collection of problems in this paper is by no means exhaustive, but rather reflects the subjective preference of the author. For a history of this class of problems and an exhaustive list of references we refer the reader to the surveys [10, 39, 48] and Chapter 8 of [13].

Throughout this paper all point sets in the plane are assumed to be in *general position*, that is, no three points in the set are collinear. When a subset of a point set $S$ is the vertex set of a polygon $P$, we say that $P$ is *spanned* by points in $S$. Further we will only consider simple (non self-intersecting) polygons.

*Institute for Software Technology, Graz University of Technology, Graz, Austria, oaich@ist.tugraz.at

## 2 $k$-gons

Around 1933 Esther Klein raised the following question which was partially answered in a classical paper by Erdős and Szekeres [22] 1935: "Is it true that for any $k$ there is a smallest integer $g(k)$ such that any set of $g(k)$ points contains at least one convex $k$-gon?" In the mathematical history this problem is also known as the "Happy End Problem", since Szekeres and Klein became engaged while collaborating on this topic and married shortly afterwards [28, 13]. Klein observed that $g(4) = 5$ and Kalbfleisch et al. [33] solved the more involved case of $g(5) = 9$.

More than 70 years after the problem was posed and more then 35 years after the proof for $g(5) = 9$, the case $k = 6$ has been settled. In 2006 Szekeres and Peters [46] showed that $g(6) = 17$ by an exhaustive computer search. The approach is based on the order type of a point set – introduced by Goodman and Pollack in 1983 [26] – which assigns an orientation to each triple of points. Thus only a finite number of configurations needs to be considered when combinatorial problems on point sets are investigated, see e.g. [8] for various applications. Using several observations, Szekeres and Peters significantly reduced the number of configurations to be computed to make the problem tractable.

The well-known Erdős–Szekeres Theorem [22] states that $g(k)$ is finite for any $k$. The currently best bounds are

$$2^{k-2} + 1 \leq g(k) \leq \binom{2k-5}{k-2} + 1,$$

where the lower bound goes back to Erdős and Szekeres [23] and is conjectured to be tight.

**Problem 2.1.** [23] Prove or disprove that $g(k) = 2^{k-2} + 1$. It is known to be true for $k \leq 6$.

The upper bound of Erdős–Szekeres was $g(k) \leq \binom{2k-4}{k-2} + 1$. Subsequently Chung and Graham [14] removed the additive +1, Kleitman and Pachter [36] improved to $g(k) \leq \binom{2k-4}{k-2} + 7 - 2k$. The bound $\binom{2k-5}{k-2} + 2$ was given by G. Tóth, P. Valtr [47] in 1998, who finally reduced the +2 to +1 in 2005 to obtain the currently best upper bound [48].

Erdős and Guy [21] posed the following generalization: "What is the least number of convex $k$-gons determined by any set $S$ of $n$ points in the plane?" The trivial solution for the case $k = 3$ is $\binom{n}{3}$. But already for convex 4-gons this question is related to the search for the rectilinear crossing number $\bar{cr}(S)$ of $S$. This is the number of proper intersections in the drawing of the complete straight line graph on $S$. The number of convex 4-gons is equal to $\bar{cr}(S)$ and is thus minimized by sets minimizing the rectilinear crossing number, a well known, difficult problem in discrete geometry, see [13] and [21] for details. Asymptotically the number of convex $k$-gons is $c_k\binom{n}{k} = \Theta(n^k)$ for sufficiently large $n$ and a constant $\frac{1}{\binom{g(k)}{k}} \leq c_k < 1$. Since $c_4$ equals the rectilinear crossing constant we get $0.37968 \leq c_4 \leq 0.38054$, and tight values for the number of convex 4-gons are known for $n \leq 27$ points, see e.g. [1]. Table 1 gives the minimum number of convex $k$-gons every $n$-point set $S$ determines, for $n \leq 15$ and $k = 3, 4, 5$.

| $n$ | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3-gon | 1 | 4 | 10 | 20 | 35 | 56 | 84 | 120 | 165 | 220 | 286 | 364 | 455 |
| 4-gon | - | 0 | 1 | 3 | 9 | 19 | 36 | 62 | 102 | 153 | 229 | 324 | 447 |
| 5-gon | - | - | 0 | 0 | 0 | 0 | 1 | 2 | 7 | $\leq 13$ | $\leq 34$ | $\leq 62$ | $\leq 113$ |

Table 1: Minimum numbers of convex $k$-gons [8].

## 3 $k$-holes

A *$k$-hole* is an empty $k$-gon, that is, a $k$-gon which does not contain any points of the underlying set in its interior. In 1978 Erdős [19] raised the following question for convex $k$-holes: "What is the smallest integer $h(k)$ such that any set of $h(k)$ points in the plane contains at least one convex $k$-hole?" For $k \leq 5$ exact values for $h(k)$ are known, see Table 2. As already observed by Esther Klein, every set of

| $k$ | 3 | 4 | 5 | 6 | $\geq 7$ |
|---|---|---|---|---|---|
| $h(k)$ | 3 | 5 | 10 | $\geq 30$<br>$\leq 1717$ | $\infty$ |

Table 2: Bounds for $h(k)$.

5 points determines a convex 4-hole, and 10 points always contain a convex 5-hole, a fact proved by Harborth [27]. However, in 1983 and on the contrary to what was conjectured, Horton showed that there exist arbitrarily large sets of points not containing any convex 7-hole [29].

It again took almost a quarter of a century after Hortons construction to answer the existence question for 6-holes. In 2007/08 Nicolás [40] and independently Gerken [25] proved that every sufficiently large point set contains a convex 6-hole. In both approaches the result has been obtained by deriving a relation between convex 6-holes and larger convex $k$-gons, and by considering nested convex hull layers inside the convex $k$-gons. While Nicolás provided a simpler proof for $h(6) \leq g(25)$, Gerken succeeded with a rather exhaustive case analysis (57 cases) to show $h(6) \leq g(9)$. In fact he shows that every point set that spans a convex 9-gon also contains a convex 6-hole. This is best possible in the sense that there exist point sets without convex 6-holes that have a convex hull of size 8 [41].

From the bounds known for $g(k)$ it follows that any set of at least 1717 points contains a convex 6-hole. Moreover, if the conjecture $g(k) = 2^{k-2} + 1$ of Erdős and Szekeres is true, then this bound drops down to 129 points. A better bound of $h(6) \leq \max\{g(8), 400\} \leq 463$ (respectively $\leq 65$ if the conjecture in Problem 2.1 is true) has been claimed [37, 38], but not been properly published yet.

Valtr [50] provides a simpler and more general version of Gerken's proof, but requires more points, namely $h(6) \leq g(15)$. As for a lower bound it is known that at least 30 points are needed, that is, there exists a set of 29 points without convex 6-hole. This set was found by an extensive computer search, including heuristical point insertion and removal techniques [41].

**Problem 3.1.** What is the minimum cardinality $h(6)$ such that any set of at least $h(6)$ points determines a convex 6-hole? It is known that $30 \leq h(6) \leq 1717$.

| $n$ | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3-hole | 1 | 3 | 7 | 13 | 21 | 31 | 43 | 58 | 75 | 94 | 114...116 | 136...141 | 160...169 |
| 4-hole | - | 0 | 1 | 3 | 6 | 10 | 15 | 23 | 32 | 42 | 51...55 | 61...71 | 72...90 |
| 5-hole | - | - | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 3...5 | 3...8 | 3...12 |

Table 3: Minimum number of convex $k$-holes [8, 15, 27].

Varying the problem of Erdős and Guy from Section 2 we get the following question [20]. "What is the least number $h_k(n)$ of convex $k$-holes determined by any set of $n$ points in the plane?" Table 3 gives exact values for small sets, and we know by Hortons construction that $h_k(n) = 0$ for $k \geq 7$. Table 4 summarizes the currently best general lower and upper bounds for $k = 3 \ldots 6$.

$$
\begin{aligned}
n^2 - O(n \log n) \leq h_3(n) &\leq 1.6196n^2 + o(n^2) \\
\tfrac{n^2}{2} - O(n) \leq h_4(n) &\leq 1.9397n^2 + o(n^2) \\
3\lfloor \tfrac{n-4}{8} \rfloor \leq h_5(n) &\leq 1.0207n^2 + o(n^2) \\
\lfloor \tfrac{n-5}{1712} \rfloor \leq h_6(n) &\leq 0.2006n^2 + o(n^2)
\end{aligned}
$$

Table 4: Lower and upper bounds on the number $h_k(n)$ of $k$-holes.

All upper bounds in Table 4 are taken from [11], improving over previous bounds: For 3-holes Katchalski and Meir [35] showed that for all $n \geq 3$ a lower bound is given by $\binom{n-1}{2}$ and that there exists a constant $c > 0$ such that there exist sets with at most $cn^2$ 3-holes. Around the same time, Bárány and Füredi [9] showed that any set of $n$ points has at least $n^2 - O(n \log n)$ 3-holes. They also

gave examples with at most $2n^2$ 3-holes if $n$ is a power of 2. Valtr [49] described a configuration of $n$ points related to Horton sets [29] with fewer than $1.8n^2$ 3-holes and also provided examples with small numbers of convex $k$-holes, e.g. with at most $2.42n^2$ convex 4-holes. Later Dumitrescu [18] improved these constructions to a configuration with $\approx 1.68n^2$ 3-holes, which then consequently was further improved by Bárány and Valtr [11], obtaining the currently best bounds shown in the table. For 3-holes it is still unknown whether the constant could be smaller than 1, that is, whether there exists a family of $n$-element sets with fewer than $n^2$ 3-holes.

**Problem 3.2.** [11] Prove or disprove that $h_3(n) \geq (1+\varepsilon)n^2$ for sufficiently large $n$ and some fixed $\varepsilon \geq 0$.

Concerning lower bounds, we already mentioned the result from [9] for 3-holes. The lower bound for 4-holes can be found in [18, 49]. For convex 5-holes the existence of at least three convex 5-holes in every set of 12 points (cf. Table 3) leads two the lower bound of $h_5(n) \geq 3\lfloor \frac{n-4}{8} \rfloor$ [30], improving over the previous bound $h_5(n) \geq \lfloor \frac{n-4}{6} \rfloor$ of Bárány and Károlyi [10]. To obtain this bound simply sort the point set from left to right, split it into groups of 8 points, and, for each group, reuse the rightmost 4 points from the group to its left. In a similar way we obtain the lower bound $h_6(n) \geq \lfloor \frac{n-5}{1712} \rfloor$ by using the upper bound $h(6) \leq 1717$.

**Problem 3.3.** Show a super-linear lower bound for $h_5(n)$ and/or give a sub-quadratic upper bound for $h_5(n)$. Similar for $h_6(n)$.

Note that proving $h_5(n) \geq \varepsilon n^2$ for some $\varepsilon > 0$ is equivalent to a positive answer of Problem 3.2 by results in [44].

## 3.1 Non-convex $k$-holes

As we know that there are arbitrarily large point sets which do not contain convex $k$-holes of a certain size, we now relax the problem by skipping the convexity requirement. Of course we still want the holes to be as 'near-convex' as possible, as otherwise any polygonization (spanning cycle) of the point set will be an $n$-hole.

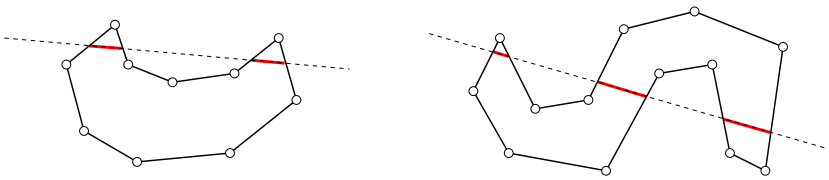


Figure 1: 2-convex polygon (left) and 3-convex (but not 2-convex) polygon (right).

Several ways of measuring the convexity or non-convexity of a given polygon have been proposed in the literature. See [4] for a short overview, and a generalization of convexity which will prove useful in our context: A polygon $P$ is called *$j$-convex* if there exists no straight line that intersects $P$ in more than $j$ connected components. Thus 1-convexity refers to convexity in its standard meaning, and among non-convex polygons 2-convex ones can be considered to be as convex as possible. Figure 1 shows examples of 2-convex and 3-convex polygons. Typical examples for 2-convex polygons are also pseudo-triangles. These are simple polygons with precisely three convex vertices (so-called corners) with internal angles less than $\pi$; see the recent survey [45] for details and applications.

Obviously 4-gons and 5-gons are always 2-convex, while a 6-gon might be not. Moreover for any set of up to 9 points there always exists a 2-convex polygonization [3], which implies that 2-convex $k$-holes exist for $k \leq 9$ for any $n \geq k$. Utilizing convex chains of logarithmic length, whose existence follow from the exponential upper bound for $g(k)$, it is easy to see that every set of $n$ points determines a 2-convex hole of size $\Omega(\log n)$. Moreover there exist sets of $n$ points such that the largest spanned 2-convex polygon has size $O(\log^2 n)$ [4].

**Problem 3.4.** [4] Prove or disprove that for any $n$ there are sets of $n$ points in $\mathcal{R}^2$ in general position where every 2-convex $k$-hole is of size at most $k = O(\log n)$.

**Problem 3.5.** [4] Prove or disprove that any set of $n$ points in $\mathcal{R}^2$ in general position spans a 2-convex $k$-gon of size $k = \Omega(\log^2 n)$.

Another variation is to ask for the existence of 2-convex $k$-gons of a special type. For example, what is the minimum cardinality $p(k)$ such that any point set of size $p(k)$ spans either a convex $k$-gon, or a pseudo-triangle of size at least $k$? It is not hard to obtain $p(k)$ for $k \leq 5$ and it is known that $p(6) = 12$ and $21 \leq p(7) \leq 23$ [2].

**Problem 3.6.** [2] Determine $p(k)$. What if we consider $k$-holes and empty pseudo-triangles?

### 3.2 4-gons and 4-holes

Considering 2-convex $k$-holes is an interesting question even for small, constant values of $k$. For small point sets Table 5 shows the minimum number of convex 4-holes, the maximum number of non-convex 4-holes, and the minimum and maximum number of 2-convex 4-holes, and, for easy comparison, the number of 4-tuples (which is identical to the maximum number of convex 4-holes).

| $n$ | convex | non-convex | 2-convex | | $\binom{n}{4}$ |
|---|---|---|---|---|---|
| | min | max | min | max | |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 3 | 1 | 3 | 1 |
| 5 | 1 | 8 | 5 | 9 | 5 |
| 6 | 3 | 18 | 15 | 22 | 15 |
| 7 | 6 | 36 | 35 | 43 | 35 |
| 8 | 10 | 64 | 66 | 77 | 70 |
| 9 | 15 | 100 | 102 | 126 | 126 |
| 10 | 23 | 150 | 147 | 210 | 210 |
| 11 | 32 | 216 | 203 | 330 | 330 |

Table 5: 2-convex 4-holes

For $n = 1...7$ it can be seen that the minimum number of 2-convex 4-holes is $\binom{n}{4}$, while it seems that the maximum number of 2-convex 4-holes is $\binom{n}{4}$ for $n \geq 9$. That is, convex sets minimize the cardinality of 2-convex 4-holes for $n \leq 7$ but seem to maximize it for $n \geq 9$.

**Problem 3.7.** Show that point sets in convex position maximize the cardinality of 2-convex 4-holes for $n \geq 9$ points.

**Problem 3.8.** Which family of point sets (a) minimizes the number of convex 4-holes? (b) maximizes the number of non-convex 4-holes? (c) minimizes the number of 2-convex 4-holes?

Note that if for a point set $S$ we consider 4-gons instead of 4-holes, then the number of 2-convex 4-gons is related to the rectilinear crossing number $\bar{cr}(S)$ of $S$ mentioned in Section 2. The number of convex 4-gons equals $\bar{cr}(S)$ while the number of non-convex 4-gons is $3\left(\binom{n}{4} - \bar{cr}(S)\right)$. So the number of 2-convex 4-gons is $3\binom{n}{4} - 2\bar{cr}(S)$, and therefore minimized for point sets in convex position and maximized for sets minimizing the rectilinear crossing number.

## 4 Monochromatic $k$-holes

In this section we consider variations of the above problems where the points of the given set $S$ belong to different classes – usually described as *colors*. We say that a (not necessarily convex) $k$-gon or $k$-hole is monochromatic if all its vertices have the same color. This colorful family of problems was introduced in 2003 by Devillers et al. [16]. They showed for example that any bichromatic set of $n$ points determines at least $\lceil \frac{n}{4} \rceil - 2$ monochromatic 3-holes with pairwise disjoint interiors, which is tight. In fact all these 3-holes are of the same color.

It is natural to wonder whether results similar to the just mentioned one are possible when there are more than two colors: What is the minimum number of colors such that there exist sets of $n$ points which we can color in a way so that they do not determine any convex $k$-hole? In [16] (Theorem 3.3) this question has been settled by showing that already for three colors there exists a coloring of the Horton set so that it does not span any monochromatic 3-hole.

Note that the according question for $k$-gons can be reduced to the uncolored version. It is not hard to see that any $c$-colored set of at least $c \cdot (g(k) - 1) + 1$ points contains at least one monochromatic $k$-gon [16]. Variations of the problem, where the vertices of the $k$-gon/$k$-hole can be colored differently (polychromatic), or where all vertices must have different colors (heterochromatic), also exist. See [16] for details.

Another result Devillers et al. provide in [16] is that for $k \geq 5$ and any $n$ there are bichromatic sets of $n$ points without convex monochromatic $k$-holes. The proof is based on the 3-coloring of the Horton set mentioned above. So the most interesting remaining cases in this family of problems are monochromatic 3-holes and 4-holes in bichromatic sets, which we will consider in the next two sections. A possible relaxation of the problem is to allow $j$-convex holes, see also Section 4.2.

**Problem 4.1.** For which combinations of $j \geq 1$ and $k \geq 4$ does any sufficiently large bichromatic point set in $\mathbb{R}^2$ in general position determine a $j$-convex $k$-hole?

Note that from the results in [16] it is clear that there are point sets with more than two colors where no $j$-convex $k$-holes exist for any $j \geq 1$ and $k \geq 3$. See also [13] and [34] for a number of related problems on colored point sets.

## 4.1 Monochromatic 3-holes

It is easy to see that any bichromatic set of at least 10 points contains a monochromatic 3-hole: From Section 3 we know that any (uncolored) set of 10 points contains a convex 5-hole. Hence, however these 5 points are colored, we will obtain a monochromatic 3-hole. In fact it can be argued that any bichromatic set of $n \geq 7$ points contains at least $\lfloor \frac{n-7}{2} \rfloor$ monochromatic 3-holes. Therefore, already bichromatic point sets of size 9 are sufficient to determine a monochromatic 3-hole. From the point set order type data base [8] it is also known that there are bichromatic sets of 9 and 10 points which only contain a unique monochromatic 3-hole. Moreover there are sets of 8 points without monochromatic 3-holes, see Figure 2.

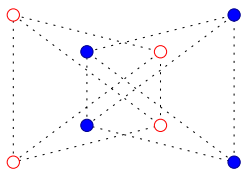


Figure 2: bichromatic point set without monochromatic 3-hole.

As we can split any large set of points into groups of constant cardinality the above observation already implies that there is a linear number of monochromatic 3-holes. As mentioned above, a stronger result has been shown by Devillers et al. [16], namely that any bichromatic set of $n$ points determines at least $\lceil \frac{n}{4} \rceil - 2$ monochromatic 3-holes with pairwise disjoint interiors, which is tight.

So the question arises whether there exist super-linear many monochromatic 3-holes, where it is of course not required that the 3-holes are disjoint. In contrast to the race for the best constant in the uncolored case described in Section 2, there are only recent results for the asymptotic number of monochromatic 3-holes in bichromatic sets, and no tight bounds are known yet. The first super-linear bound of $\Omega(n^{5/4})$ was given in [5]. The proof is based on a so-called discrepancy lemma, which implies that if there is a significant difference in the cardinality of the two color classes then the number of monochromatic 3-holes is sufficiently large. Combining this with special triangulations of the point set obtained by using Dilworth's Theorem [17], the bound follows.

By refining the techniques used in [5], this result has been improved by Pach and Tóth [43] to the currently best known bound of $\Omega(n^{4/3})$ monochromatic 3-holes. Unfortunately, the conjecture in [5] that any bichromatic set of $n$ points spans a quadratic number of monochromatic 3-holes is still unsettled.

**Problem 4.2.** [5] Prove or disprove that any bichromatic set of $n$ points in $\mathbb{R}^2$ in general position determines $\Omega(n^2)$ monochromatic 3-holes.

An affirmative answer would follow from showing that $h_5(n) \geq \varepsilon n^2$ for some $\varepsilon > 0$, cf. Problem 3.3.

## 4.2 Monochromatic 4-holes

From the above discussion it follows that for the existence question of monochromatic $k$-holes the most interesting remaining case is the existence of monochromatic 4-holes in bichromatic point sets.

Figure 3(a) shows a set with 18 points which does not contain a convex monochromatic 4-hole, and larger examples with 20 [12], 30 [24], 32 [51] and most recently 36 [31] points (Fig 3(b)) have been found. However, all larger examples do contain non-convex monochromatic 4-holes, while the one in Figure 3(a) does not.

**Problem 4.3.** Find large examples of bichromatic point sets which do not contain (convex) monochromatic 4-holes.

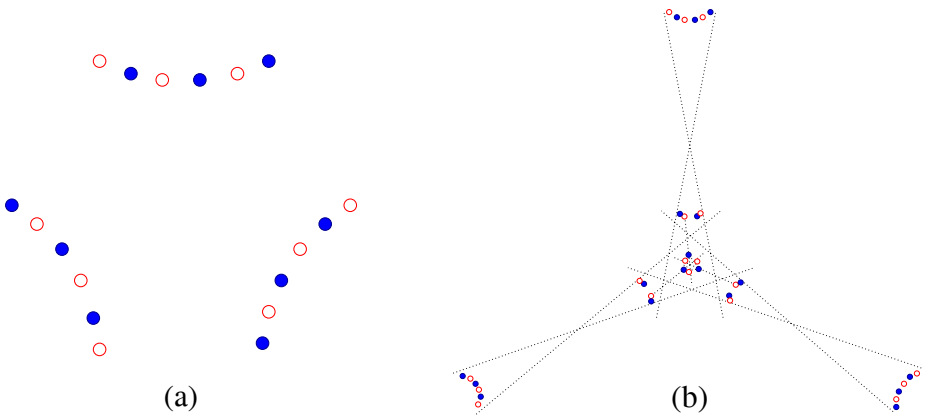



Figure 3: (a) 18 points without 2-convex monochromatic 4-holes. (b) 36 points without convex monochromatic 4-holes (sketch, see [31] for details).

Notice that every (uncolored) point set that admits a convex 7-hole will contain a convex monochromatic 4-hole for any bicoloration, because at least four of the vertices of the heptagon will have the same color. However, it has been shown that that for $n \geq 64$ any bichromatic Horton set contains convex monochromatic 4-holes [16], and thus the authors of this paper conjecture that for sufficiently large $n$ any bichromatic point set contains at least one convex monochromatic 4-hole.

Until recently this conjecture has not been settled even for 2-convex monochromatic 4-holes, that is, 4-holes which are not required to be convex. This weaker version of the problem arose [32, 42] as no progress for the original question had been obtained. It was considered to be an important step towards solving the initial problem. The current situation is that in [7] this relaxed version of the conjecture has been shown to be true. If the cardinality of the bichromatic point set $S$ is sufficiently large, there always exists a 2-convex monochromatic 4-gon spanned by $S$. In [7] the given lower bound on the cardinality was $n \geq 5044$. Using observations on vertex degree parity constraints for triangulations of $S$ this bound has most recently been lowered to 2760 points [6].

**Problem 4.4.** [16] Prove or disprove that any sufficiently large bichromatic set of points in $\mathcal{R}^2$ in general position determines at least one convex monochromatic 4-hole.

## Acknowledgments

We thank Thomas Hackl and Birgit Vogtenhuber for valuable discussions. Research supported by the FWF [Austrian Fonds zur Förderung der Wissenschaftlichen Forschung] under grant S9205-N12, NFN Industrial Geometry.

## References

[1] O. Aichholzer On the Rectilinear Crossing Number. `http://www.ist.tugraz.at/aichholzer/research/rp/triangulations/crossing/`

[2] O. Aichholzer, F. Aurenhammer, S. Bereg, T. Hackl, T. Krasser, On $k$-gons and $k$-pseudo-triangles. manuscript, 2009.

[3] O. Aichholzer, F. Aurenhammer, T. Hackl, F. Hurtado, P. Ramos, J. Urrutia, B. Vogtenhuber, $k$-convex point sets. manuscript, 2009.

[4] O. Aichholzer, F. Aurenhammer, F. Hurtado, P. Ramos, J. Urrutia, Two-convex polygons. *In Proc. 25th European Workshop on Computational Geometry EuroCG '09*, 25:117-120, Brussels, Belgium, 2009.

[5] O. Aichholzer, R. Fabila-Monroy, D. Flores-Peñaloza, T. Hackl, C. Huemer, J. Urrutia, Empty Monochromatic Triangles. *Proc.* $20^{th}$ *Canadian Conference on Computational Geometry CCCG'08*, 20:75–78, Montreal, Quebec, Canada, 2008, journal version to appear in CGTA 2009 (in press).

[6] O. Aichholzer, T. Hackl, M. Hoffmann, A. Pilz, G. Rote, B. Speckmann, B. Vogtenhuber, Plane Graphs with Parity Constraints. *Proc. Algorithms And Data Structures Symposium - WADS (LNCS)*, Banff, Alberta, Canada, 2009 (to appear).

[7] O. Aichholzer, T. Hackl, C. Huemer, F. Hurtado, B. Vogtenhuber, Large bichromatic point sets admit empty monochromatic 4-gons. *In Proc. 25th European Workshop on Computational Geometry EuroCG '09*, 25:133-136, Brussels, Belgium, 2009.

[8] O. Aichholzer and H. Krasser, The point set order type data base: A collection of applications and results, *Proc. 13th Canadian Conference on Computational Geometry CCCG'01*, 13:17–20, Waterloo, Ontario, Canada, 2001.

[9] I. Bárány, Z. Füredi, Empty simplices in Euclidean space. *Canad. Math. Bull.*, 30:436–445, 1987.

[10] I. Bárány, Gy. Károlyi, Problems and Results around the Erdős–Szekeres Convex Polygon Theorem. In *Discrete and Combinatorial Geometry*, LNCS 2098, 91–105, 2001.

[11] I. Bárány, P. Valtr, Planar point sets with a small number of empty convex polygons. *Studia Scientiarum Mathematicarum Hungarica*, 41(2):243-269, 2004.

[12] P. Brass, Empty monochromatic fourgons in two-colored points sets. *Geombinatorics*, XIV(1):5–7, 2004.

[13] P. Brass, W. Moser, J. Pach, *Research Problems in Discrete Geometry*, Springer, 2005.

[14] F.R.K. Chung, R.L. Graham, Forced convex $n$-gons in the plane. *Discrete & Computational Geometry*, 19:367–371, 1998.

[15] K. Dehnhardt, Leere konvexe Vielecke in ebenen Punktmengen. PhD Thesis, TU Braunschweig, Germany, 1987.

[16] O. Devillers, F. Hurtado, G. Károlyi, C. Seara, Chromatic variants of the Erdős-Szekeres Theorem. *Computational Geometry, Theory and Applications*, 26(3):193–208, 2003.

[17] R.P. Dilworth, A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–166, 1950.

[18] A. Dumitrescu, Planar sets with few empty convex polygons. *Studia Scientiarum Mathematicarum Hungarica*, 36(1-2):93-109, 2000.

[19] P. Erdős, Some more problems on elementary geometry. *Austral. Math. Soc. Gaz.*, 5:52–54, 1978.

[20] P. Erdős, Some old and new problems in combinatorial geometry. In *Convexity and Graph Theory*, M. Rosenfeld et al., eds., Annals Discrete Math., 20:129–136, 1984.

[21] P. Erdős, R.K. Guy, Crossing number problems. *Amer. Math. Monthly*, 88:52–58, 1973.

[22] P. Erdős, G. Szekeres, A combinatorial problem in geometry. *Compositio Math. 2*, 463–470, 1935.

[23] P. Erdős, G. Szekeres, On some extremum problems in elementary geometry. *Ann. Univ. Sci. Budapest. Eötvös, Sect. Math.*, 3/4:53–62, 1960.

[24] E. Friedmann, 30 two-colored points with no empty monochromatic convex fourgons. *Geombinatorics*, XIV,(2):53–54, 2004.

[25] T. Gerken, Empty convex hexagons in planar point sets. *Discrete and Computational Geometry*, 39(1-3):239–272, 2008.

[26] J.E.Goodman, R.Pollack, Multidimensional sorting. *SIAM J. Computing*, 12:484-507, 1983.

[27] H. Harborth, Konvexe Fünfecke in ebenen Punktmengen. *Elemente Math.*, 33:116–118, 1978.

[28] P. Hoffmann, The man who loved only numbers. Hyperion, New York, 1998.

[29] J.D. Horton, Sets with no empty convex 7-gons. *Canad. Math. Bull.*, 26(4):482–484, 1983.

[30] C. Huemer, Personal communication, Barcelona, Feb. 2009.

[31] C. Huemer, C. Seara, 36 two-colored points with no empty monochromatic convex fourgons. To appear in Geombinatorics, 2009.

[32] F. Hurtado, Open Problem Session. $21^{st}$ *European Workshop on Computational Geometry, EuroCG'05*, Eindhoven, The Netherlands, 2005.

[33] J.D. Kalbfleisch, J.G. Kalbfleisch, R.G. Stanton, A combinatorial problem on convex $n$-gons. *Louisiana Conference on Combinatorics, Graph Theory, and Computing*, Baton Rouge, 180–188, 1970.

[34] A. Kaneko, M. Kano, Discrete geometry on red and blue points in the plane – a survey. In *Discrete and Computational Geometry, Algorithms Combin.*, 25:551–570, Springer, Berlin, 2003.

[35] M. Katchalski, A. Meir, On empty triangles determined by points in the plane. *Acta Math. Hung.*, 51(3-4):323–328, 1988.

[36] D. Kleitman, L. Pachter, Finding convex sets among points in the plane. *Discrete and Computational Geometry* 19:405—410, 1998.

[37] V.A. Koshelev, On the Erdős–Szekeres problem in combinatorial geometry. *Electronic Notes in Discrete Mathematics*, 29:175-177, 2007.

[38] V.A. Koshelev, On the Erdős–Szekeres problem. *Doklady Mathematics*, 76:603–605, 2007.

[39] W. Morris, V. Soltan, The Erdős–Szekeres problem on points in convex position – a survey. *Bull. Amer. Math. Soc.*, 37:437–458, 2000.

[40] C.M. Nicolás, The empty hexagon theorem. *Discrete and Computational Geometry*, 38(2):389–397, 2007.

[41] M.H. Overmars, Finding sets of points without empty convex 6-gons. *Discrete and Computational Geometry*, 29:153–158, 2003.

[42] J. Pach, On simplices embracing a point (invited talk). *Topological & Geometric Graph Theory (TGGT)*, Paris, France, 2008.

[43] J. Pach, G. Tóth, Monochromatic empty triangles in two-colored point sets. *Geometry, Games, Graphs and Education: the Joe Malkevitch Festschrift*, COMAP, Bedford, MA, 195–198, 2008.

[44] R. Pinchasi, R. Radoičić, M. Sharir, *On empty convex polygons in a planar point set*, J. Combinat. Theory, Ser. A 113:385–419, 2006.

[45] G. Rote, F. Santos, I. Streinu, *Pseudo-Triangulations – a Survey.* Surveys on Discrete and Computational Geometry – Twenty years later: Contemporary Mathematics 453:343–410, 2008.

[46] G. Szekeres, L. Peters, Computer solution to the 17-point Erdős–Szekeres problem. *The Anziam Journal*, 48(2):151–164, 2006.

[47] G. Tóth, P. Valtr, Note on the Erdős–Szekeres theorem, *Discrete and Computational Geometry*, 19:457-459, 1998.

[48] G. Tóth, P. Valtr, The Erdős–Szekeres theorem: upper bounds and related results. In J.E. Goodman, J. Pach, and E. Welzl (Eds.), *Combinatorial and Computational Geometry, MSRI Publications*, 52:557-568, 2005.

[49] P. Valtr, On the minimum number of empty polygons in planar point sets. *Studia. Sci. Math. Hungar.* 30:155–163, 1995.

[50] P. Valtr, On empty hexagons. In: J. E. Goodman, J. Pach, and R. Pollack, *Surveys on Discrete and Computational Geometry, Twenty Years Later*, Contemp. Math. 453, AMS, 433–441, 2008.

[51] R. Van Gulik, 32 two-colored points with no empty monochromatic convex fourgons. *Geombinatorics*, XV(1):32–33, 2005.

# Empty convex polygons in planar point sets

Pavel Valtr*

## Extended abstract

Let $X$ be a finite set of points in the plane. We say that $X$ is *in general position* if no three points of $X$ lie on a line. The convex hull of $X$ is denoted by $\mathrm{conv}X$. We say that $X$ is *in convex position*, if each point of $X$ is a vertex of $\mathrm{conv}X$.

A classical result in discrete geometry and also in Ramsey theory is the following theorem:

**Theorem 1 (Erdős–Szekeres Theorem [4]).** *For every $k \geq 3$ there is a (smallest) integer $\mathrm{ES}(k)$ such that any set of at least $\mathrm{ES}(k)$ points in general position in the plane contains $k$ points in convex position.*

The Erdős–Szekeres theorem inspired a lot of research. A frequent topic in this area is the study of the existence of so-called empty convex polygons in finite planar point sets. Let $P$ be a finite set of points in general position in the plane. A convex $k$-gon $G$ is called a *$k$-hole (*or *empty convex $k$-gon) of $P$*, if all vertices of $G$ lie in $P$ and no point of $P$ lies inside $G$. Frequently we will mean by a $k$-hole the vertex set of $G$. Erdős [3] asked if, for a fixed $k$, any sufficiently large point set in general position has a $k$-hole. Already many years ago, this was known to be true for $k \leq 5$ [6] and false for $k \geq 7$ [7]. The remaining case $k = 6$ became a well-known open problem. Gerken [5] (see also [14]) has solved it in the affirmative. Somewhat later the same result was also obtained independently by Nicolás [10]. Thus, we have

**Theorem 2 (The Empty Hexagon Theorem).** *There is an integer $g$ such that any set of at least $g$ points in general position in the plane has a 6-hole.*

Gerken [5] proved the Empty hexagon theorem with $g = \mathrm{ES}(9) \leq 1717$. Koshelev [8] has recently announced a proof with $g = \max\{400, \mathrm{ES}(8)\} \leq 463$. A lower bound $g \geq 30$ follows from Overmars' construction [11] of 29 points in general position with no 6-hole. Overmars' lower bound seems to be closer to the best value of $g$ than the above upper bounds.

Also in higher dimensions one can find arbitrarily large finite point sets in general position with no holes with many vertices (depending on the dimension). In $\mathbf{R}^d$, a *$k$-hole* of a point set $P$ is a set of $k$ convexly independent points of $P$ such that their convex hull contains no other points of $P$. It was shown in [13] that for any $d \geq 3$ there is a constant $h(d)$ such that there are arbitrarily large finite point sets in general position in $\mathbf{R}^d$ with no $h(d)$-hole. The best known upper bound [13] on $h(d)$ is somewhat bigger than exponential in $d$, while the best known lower bound [13] is only linear in $d$: $h(d) \geq 2d + 1$.

Further research related to Theorems 1 and 2 is described in several survey papers [1, 2, 9, 12].

In my talk I will also mention various other questions and results about empty hexagons, e.g., the minimum number of empty $k$-gons ($3 \leq k \leq 6$), linear dependencies for the numbers of $k$-holes in a finite point set, sufficient conditions for the existence of large holes, colored versions, (degenerate) sets with no empty triangles and no empty quadrilaterals, respectively, the modular version of the Erdős–Szekeres theorem, prescribed or bounded number of interior points, etc.

*Department of Applied Mathematics and Institute for Theoretical Computer Science (ITI), Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic

## References

[1] I. Bárány and Gy. Károlyi, Problems and results around the Erdős–Szekeres convex polygon theorem, *Japanese Conference on Discrete and Computational Geometry* (Tokyo, 2000), *Lecture Notes in Comput. Sci.* **2098** (2001), 91–105.

[2] P. Brass, W. Moser, and J. Pach, Convex polygons and the Erdős-Szekeres problem, Chapter 8.2 in the book *Research problems in discrete geometry*, Springer, 2005.

[3] P. Erdős, On some problems of elementary and combinatorial geometry, *Ann. Mat. Pura. Appl.* **103** (1975), 99–108.

[4] P. Erdős and G. Szekeres, A combinatorial problem in geometry, *Compositio Mathematica* **2** (1935), 463–470.

[5] T. Gerken, Empty convex hexagons in planar point sets, *Discrete Comput. Geom.* **39** (2008), 239–272.

[6] H. Harborth, Konvexe Fünfecke in ebenen Punktmengen, *Elem. Math.* **33** (1978), 116–118.

[7] J. D. Horton, Sets with no empty convex 7-gons, *Canadian Math. Bull.* **26** (1983), 482–484.

[8] V. A. Koshelev, On the Erdős–Szekeres problem, *Doklady Mathematics* **76** (2007), 603–605.

[9] W. Morris and V. Soltan, The Erdős-Szekeres problem on points in convex position – a survey. *Bull. Amer. Math. Soc.* **37** (2000), 437–458.

[10] C. M. Nicolás, The empty hexagon theorem, *Discrete Comput. Geom.* **38** (2007), 389–397.

[11] M.H. Overmars, Finding sets of points without empty convex 6-gons, *Discrete and Computational Geometry* **29** (2003), 153–158.

[12] G. Tóth and P. Valtr, The Erdős-Szekeres theorem: upper bounds and related results, Combinatorial and Computational Geometry, MSRI Publications 52 (2005), 557–568.

[13] P. Valtr, Sets in $\mathbf{R}^d$ with no large empty convex subsets, *Discrete mathematics* 108 (1992), 115–124.

[14] P. Valtr, On empty hexagons, Contemp. Math. **453** (2008), 433–441, Surveys on discrete and computational geometry.

# Lunes, 29 de junio

## Sesión 1, 11:50–13:30

# Vigilancia en Galerías de Arte Curvilíneas

Javier Cano-Vila* Joel Espinosa Longi ** Jorge Urrutia ***

**Resumen**

En este artículo consideramos el problema de vigilar una galería de arte curvilínea. Las aristas de este tipo de galerías, pueden ser arcos de curvas convexas. Es fácil ver que en general, el número de guardias necesarios para vigilar una galería de arte curvilínea, no es acotado. Sin embargo cuando todas las aristas de una galería curvilínea son *cóncavas* o *convexas* con respecto al interior de la galería, el número de guardias necesarios, resulta ser una función lineal del número de vértices de la galería. Recientemente M. I. Karavelas, E. P. Tsigaridas y Cs. D. Tóth [3, 4] probaron que $\lfloor \frac{2n}{3} \rfloor$ guardias colocados en vértices de la galería son suficientes y a veces necesarios para vigilar una galería de arte curvilínea en la cual, todas sus aristas son convexas con respecto al interior. Sin embargo, cuando se permite que los guardias estén en cualquier parte de la galería, no se conoce un ejemplo en el que se necesiten más de $\lceil \frac{n}{2} \rceil$ guardias para vigilarla. En este trabajo demostramos que cualquier galería con aristas convexas respecto a su interior, siempre se puede vigilar usando a lo más $\lfloor \frac{5n}{8} \rfloor$ guardias colocados en cualquier parte de la galería.

## 1. Introducción

Sea $a$ una curva que une dos puntos $v$ y $v'$. Decimos que $a$ es un *arco convexo* si y solo si la unión de $a$ con el segmento de recta que une a $v$ y $v'$ es una curva simple que acota a una región convexa del plano, sea $C_a$ dicha región.

Sea $V = \{v_0, v_2, \ldots, v_{n-1}\}$ un conjunto de $n$ puntos en el plano, y $A = \{a_0, ..., a_{n-1}\}$ un conjunto de $n$ arcos convexos que no se intersecan excepto, posiblemente, en sus extremos, donde los extremos de $a_i$ son exactamente $v_i$ y $v_{i+1}$ (en suma módulo $n$). Los elementos de $A$ los llamaremos las aristas de $P$. La región cerrada y acotada por la unión de los elementos de $A$, la llamaremos un *Polígono Curvilíneo* cuyos conjuntos de aristas y vértices son respectivamente $A$ y $V$.

Sea $P$ un polígono curvilíneo, y $a_i$ una arista de $P$. Diremos que $a_i$ es *convexa* con respecto a $P$ si y solo si para cualquier punto $p$ de $a_i$ existe una $\varepsilon > 0$ tal que cualquier punto al interior de $C_{a_i}$ a distancia menor o igual a $\varepsilon$ de $p$ está en $P$. Si $a_i$ no es convexa con respecto a $P$, diremos que es *cóncava*. A partir de ahora asumiremos que todas las aristas de $P$ son convexas, o que todas son cóncavas con respecto a $P$. En el primer caso, $P$ será un *polígono arista-convexo*, en el segundo *polígono arista-cóncavo*, ver la Figura 1.

Uno de los problemas clásicos de la Geometría Computacional es el *Problema de la Galería de Arte*, donde una galería de arte, es una región cerrada del plano acotada por un polígono simple $P$. Sean $g$ y $p$ puntos en $P$. Decimos que $g$ ve a $p$ si el segmento $[g, p]$ de recta que los une está totalmente contenido en $P$. El problema consiste entonces, en encontrar un conjunto $G$ de puntos de $P$ llamados guardias, de tal forma que cualquier punto $p \in P$ es visto por algún guardia $g \in G$. En 1975, V. Chvátal probó que cualquier galería de arte con $n$ vértices siempre puede ser vigilada con a lo más $\lfloor \frac{n}{3} \rfloor$ guardias. A partir de entonces, surgieron muchas variantes del problema, al lector interesado se le refiere a los compendios [5, 7, 8] dedicados a este problema y sus variantes.

*Posgrado en Ciencias e Ingeniería de la Computación, Universidad Nacional Autónoma de México, himura.kno@gmail.com

**Posgrado en Ciencias e Ingeniería de la Computación, Universidad Nacional Autónoma de México, j.longi@gmail.com

***Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx. Research supported in part by MTM2006-03909 (Spain) and CONACYT of México, Proyecto SEP-2004-Co1-45876.

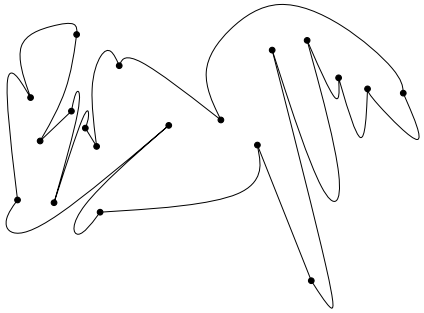


Figura 1: $P$, un polígono arista-convexo.

Recientemente Karavelas y Tsigaridas [3] propusieron una nueva variante del problema en la cual la frontera de las galerías a vigilar, están acotadas por polígonos arista-convexos, o arista-cóncavos. Dichas galerías, serán llamadas *galerías arista-convexas*, o *galerías arista-cóncavas* respectivamente. Sorprendentemente, el número de guardias requeridos para vigilar cualquier galería arista-convexa, o arista-cóncava está siempre acotado por una función lineal en el número de vértices de $P$.

Karavelas y Tsigaridas [3] prueban que $\lfloor \frac{2n}{3} \rfloor$ vértices guardia son siempre son suficientes para vigilar cualquier galería arista-convexa con $n$ vértices, además, prueban que $2n-4$ puntos guardia son siempre suficientes y en ocasiones necesarios para vigilar cualquier galería arista-cóncava con $n$ vértices. Posteriormente junto a Cs. D. Tóth [4] muestran una familia de polígonos arista-convexos que requieren al menos $\lfloor \frac{2n}{3} \rfloor$ vértices guardia para ser vigilados. Para puntos guardia, $\lfloor \frac{2n}{3} \rfloor$ siguen siendo siempre suficientes, sin embargo no se conoce ningún ejemplo donde se necesiten más de $\lceil \frac{n}{2} \rceil$ puntos guardia.

El objetivo principal de este artículo, es probar que $\lfloor \frac{5n}{8} \rfloor$ puntos guardia son siempre suficientes para vigilar cualquier galería arista-convexa con $n$ vértices. Este es un resultado interesante dado que, es la primera variante del problema de la galería de arte, en la cual se tienen cotas diferentes para el número de puntos guardia y el número de vértices guardia necesarios para vigilar una galería de arte. Para probar el resultado anterior, probamos que cualquier polígono simple con $n$ vértice, $n$ par, siempre puede cuadrilaterizarse usando a lo más $\frac{n-2}{4}$ puntos Steiner colocados al interior del polígono.

Comenzaremos este trabajo dando una prueba más sencilla del resultado de Karavelas y Tsigardas que $\lfloor \frac{2n}{3} \rfloor$ vértices guardia siempre son suficientes para vigilar una galería arista-convexa con $n$ vértices.

## 2. El Polígono Rectilíneo Subyacente

Sea $P$ un polígono arista-convexo, definimos a $\bar{P}$, el polígono rectilíneo subyacente de $P$, como sigue: sean $\bar{V} = V$ los vértices de $\bar{P}$ y definimos a $\bar{E}$, las aristas de $\bar{P}$, como sigue: Sea $E_i$ el conjunto de aristas

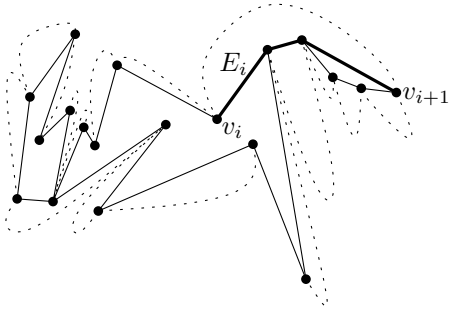



Figura 2: El polígono rectilíneo subyacente de $P$.

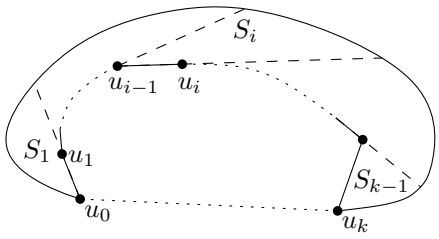



Figura 3: Ilustración de la prueba del Lema 2.1.

de la poligonal definida por el camino más corto contenido en $P$ entre $v_i$ y $v_{i+1}$ para $i = 0, \ldots, n-1$, como se muestra en la Figura 2, entonces $\bar{E} = \cup_{i=0}^{n-1} E_i$. Como se puede ver en la Figura 2, $\bar{P}$ es una gráfica geométrica conexa formada por componentes 2-conexas y trayectorias que las conectan. Es fácil de ver que cada componente 2-conexa de $\bar{P}$ es un ciclo. A las regiones que acotan estos ciclos les llamaremos las componentes poligonales de $\bar{P}$. No es difícil ver que cada arista de $\bar{P}$ está en a lo más una componente poligonal de $\bar{P}$.

Sea $a_i$ una arista de $P$. La *habitación* $R_i$ de $a_i$ es la región cerrada acotada por la unión de $a_i$ y $E_i$, ver la Figura 3. Decimos que un punto $p \in P$ ve a un segmento de recta $\ell$ si para todo punto $q \in \ell$ el segmento de recta que une a $p$ con $q$ está contenido en $P$. Diremos que un punto $g$ ve a un conjunto $S \subset P$ si para todo elemento $s \in S$ el segmento de recta que une a $g$ con $s$ está contenido en $P$. El siguiente resultado será de gran utilidad.

**Lema 2.1.** *Para cualquier punto $p$ en una habitación $R_i$, existe una arista $e$ de $\bar{E}_i$, tal que $p$ ve a $e$.*

*Demostración.* Si $E_i$ contiene únicamente una arista es claro que $p$ ve a esa arista. Supongamos entonces que $E_i$ contiene varias aristas de $\bar{P}$, digamos $e_1, e_2, \ldots, e_k$ tales que $e_1$ contenga a $v_i$ y $e_k$ a $v_{i+1}$. Reetiquetemos los vértices de $E_i$ como sigue: $v_i = u_0, u_1, \ldots, u_k = v_{i+1}$. Ahora extendemos cada $e_j$, a partir de $u_j$ hasta que toque a $a_i$, para $j = 1, \ldots, k-1$. Esto define una partición de $R_i$ en regiones convexas $S_1, S_2, \ldots, S_{k-1}$ tal que $e_j$ está totalmente contenida en la frontera de $S_j$, Figura 3. Claramente $p$ estará en alguna $S_j$, y por lo tanto $p$ ve a $e_j$. □

# 3. Vértices Guardia

A continuación damos una prueba directa del resultado de Karavelas, Tóth and Tsigaridas [4].

**Teorema 3.1.** *Sea $P$ un polígono arista-convexo con $n$ vértices, entonces $\lfloor \frac{2n}{3} \rfloor$ vértices guardia son siempre suficientes y algunas veces necesarios para vigilar $P$.*

*Demostración.* Para la parte de suficiencia sea $T(\bar{P})$ una *triangulación* de $\bar{P}$, obtenida al triangular las componentes poligonales de $\bar{P}$. Claramente los vértices de $T(\bar{P})$ pueden ser 3-coloreados de forma que cualquier par de vértices adyacentes en $T(\bar{P})$ tengan colores distintos, ver la Figura 4. Se sigue del Lema 2.1 y de la prueba de Fisk [2], que cualesquiera dos clases cromáticas de esta tres coloración de $T(\bar{P})$ vigilan $P$, por lo que tomando las dos clases más pequeñas obtenemos un conjunto de a lo más $\lfloor \frac{2n}{3} \rfloor$ guardias que vigilan a $P$.

Por razones de completez construimos una familia de polígonos arista-convexos con $3n$ vértices tales que cualquier conjunto de guardias colocados en vértices de los mismos, tiene al menos $\frac{2n}{3}$ elementos. Nuestra construcción es muy similar a la presentada en [4]. Sean $v_1, \ldots, v_3$ y $r_1, \ldots, r_3$ los vértices y regiones mostradas en la Figura 5. Para cada $i \in \{1, 2, 3\}$ el interior de la región $r_i$ no es vigilada por $v_i$. Por lo que es claro que un guardia colocado en cualquiera de $v_1, \ldots, v_3$ no basta para vigilar las habitaciones de estas aristas. Es fácil ver que el ángulo entre $l_1$ y $l_2$ puede ser tan cercano a $\pi$ como queramos, conservando la propiedad anterior, por lo que podemos tomar cualquier polígono convexo y en cada vértice colocar una curva como la mostrada en la la Figura 5, esto nos resulta en un

polígono arista-convexo como el que se muestra en la Figura 6. Esta construcción demuestra la parte de necesidad del teorema.

□

# 4. Puntos Guardia

Cuando eliminamos la restricción de que los guardias estén solo en los vértices de un polígono arista-convexo, el número de guardias necesarios (llamados *puntos guardia*) para vigilar la familia mostrada en la sección anterior se reduce a $\frac{n}{3}+1$. Es fácil ver en la Figura 7 que cualquier punto en la región sombreada de la misma, vigila ese sector de la galería.

Hasta ahora no conocemos ningún ejemplo de polígonos arista-convexos que requieran más de $\lceil \frac{n}{2} \rceil$ puntos guardia para ser vigilados. En la Figura 8 se muestra un polígono que requiere $\lceil \frac{n}{2} \rceil$ puntos guardia para ser vigilado. En esta sección probamos que $\lfloor \frac{5n}{8} \rfloor$ puntos guardia son siempre suficientes para vigilar cualquier polígono arista-convexo.

Sea $Q$ un polígono simple con $n$ vértices, $n$ par. Decimos que $Q$ es cuadrilaterizable, si podemos encontrar $\frac{n-4}{2}$ diagonales ajenas que unen parejas de vértices de $Q$ que lo dividen en $\frac{n-4}{2}+1$ cuadriláteros (no necesariamente convexos) con interiores ajenos. Es fácil encontrar para cualquier $n$ par, $n \geq 6$, polígonos con $n$ vértices que que no son cuadrilaterizables. No es difícil ver que si $Q$ tiene un número par de vértices, siempre podemos agregar algunos puntos Steiner en el interior de $Q$ de tal manera que existe una cuadrilaterización de $Q$ tal que los vértices de dichos cuadriláteros son vértices o puntos Steiner.

El siguiente teorema, que por falta de espacio no probamos aquí, es la clave para probar el resultado principal de esta sección. Esto cierra un problema abierto planteado en [6].

**Teorema 4.1.** *Sea $P$ un polígono simple con $n \geq 4$ vértices. Entonces si $n$ es par, $\lfloor \frac{n-2}{4} \rfloor$ puntos Steiner escogidos en el interior de $P$ son siempre suficientes y en ocasiones necesarios para cuadrilaterizar a $P$. Cuando $n$ es impar, $\lfloor \frac{n-1}{4} \rfloor - 1$ puntos Steiner escogidos en el interior de $P$, y uno más sobre cualquier arista del mismo, son siempre suficientes y en ocasiones necesarios para cuadrilaterizar a $P$.*

No es difícil ver que si los vértices y aristas de $\bar{P}$ forman una gráfica bipartita, entonces al colorear los vértices de $\bar{P}$ con dos colores de tal forma que vértices adyacentes reciban colores distintos, los vértices de cualquier clase cromática en esta coloración, siempre bastan para vigilar las habitaciones de $P$. Sin embargo, podría ser que el interior de $\bar{P}$ no esté vigilado. Observemos que si las componentes poligonales de $\bar{P}$ fueran cuadrilaterizables, al bicolorear los vértices de la gráfica bipartita inducida por dicha cuadrilaterización, los vértices de cualquier clase cromática de dicha coloración, vigilarían las habitaciones de $P$ y el interior de $\bar{P}$. En tal caso, $\lfloor \frac{n}{2} \rfloor$ guardias bastan para vigilar $P$.

Supongamos que $\bar{P}$ contiene sólo una componente poligonal y que esta no es un triángulo. Por el Teorema 4.1 podemos obtener una cuadrilaterización del interior de $\bar{P}$ agregando a lo mas $\lfloor \frac{n-1}{4} \rfloor$

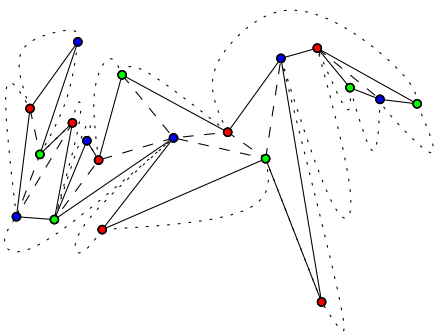


Figura 4: Triangulación y 3-coloración de $\bar{P}$.

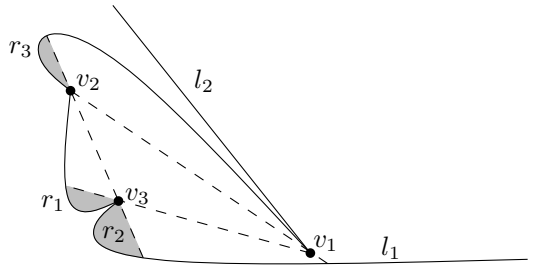



Figura 5: Bloque básico para la construcción de la cota inferior.

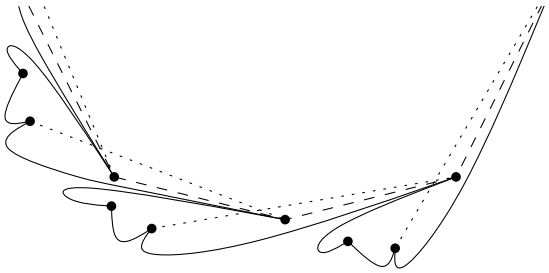


Figura 6: Una parte de un polígono que necesita $\frac{2n}{3}$ vértices guardia para ser vigilado.

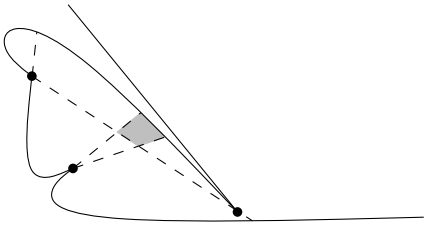


Figura 7: Un solo punto basta para vigilar este bloque.

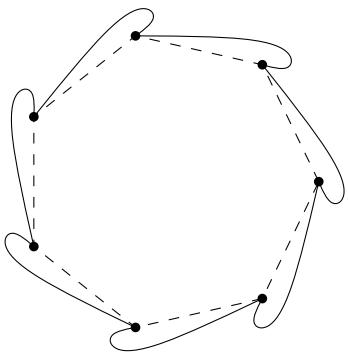


Figura 8: Un polígono arista-convexo que necesita $\lceil \frac{n}{2} \rceil$ puntos guardia para ser vigilado.

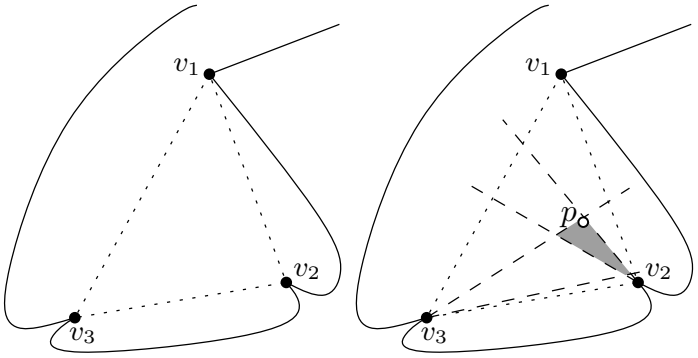



Figura 9: Ilustración del caso 1.

puntos Steiner. Dicha cuadrilaterización induce una gráfica bipartita con a lo más $\lfloor \frac{5n}{4} \rfloor$ vértices, la cual puede ser 2-coloreada. Por la observación anterior cualquier clase cromática de dicha coloración vigila $P$. Tomando la clase cromática más pequeña, obtenemos un conjunto con a lo más $\lfloor \frac{5n}{8} \rfloor$ puntos guardia (ya que algunos pueden ser puntos Steiner) que vigilan $P$.

En general, $\bar{P}$ puede tener varias componentes poligonales. Es fácil ver que cualesquiera dos componentes poligonales de $\bar{P}$ comparten a lo más un solo vértice. De esto no es difícil mostrar que si $\bar{P}$ contiene más de una componente poligonal y ninguna es un triángulo, entonces podemos cuadrilaterizar todas las componentes poligonales de $\bar{P}$ agregando a lo más $\lfloor \frac{n-1}{4} \rfloor$ puntos Steiner. Por tanto podemos encontrar un conjunto de a lo más $\lfloor \frac{5n}{8} \rfloor$ puntos guardia que vigilan $P$.

La suposición de que no haya triángulos en $\bar{P}$ se debe a que cuadrilaterizar cada triángulo de $\bar{P}$ nos forzaría a agregar más de $\lfloor \frac{n-1}{4} \rfloor$ puntos Steiner para poder cuadrilaterizar a $\bar{P}$. Por otra parte dejar los triángulos como tales y cuadrilaterizar el resto de las componentes poligonales de $\bar{P}$ no induce una gráfica bipartita. Ahora veremos que los triángulos pueden ser vigilados sin incrementar el número de guardias requeridos.

Supongamos entonces que $\bar{P}$ tiene un solo triángulo $T$. Reconocemos los siguientes casos:

1. $T$ tiene un único vértice con grado mayor a 2 en $\bar{P}$.
2. $T$ tiene más de un vértice con grado mayor a 2 en $\bar{P}$.

Para el primer caso es fácil ver que los vértices de $T$ deben ser consecutivos en $P$, por lo que sin perdida de generalidad podemos decir que son $v_1, v_2, v_3$, y que $v_1$ es el vértice de grado mayor a 2 en $\bar{P}$, ver la Figura 9. Sea $v_4$ el segundo vértice de $a_3$. Es fácil ver que en este caso siempre podemos escoger un punto $p$, como se muestra en la Figura 9 que vigila a $T$, las habitaciones de $a_1$ y $a_2$, así como parte de la habitación de $a_3$. Ahora podemos substituir $a_3$ por un nuevo arco convexo que contenga la parte no iluminada de la habitación de $a_3$ y que una a $v_1$ con $v_4$, obteniendo un polígono arista-convexo con $n-2$ vértices. De esta forma, podemos eliminar todos los triángulos de $\bar{P}$ con dos vértices de grado dos.

Antes de trabajar el segundo caso hagamos la siguiente observación que será de utilidad. Sea $v_i$ un vértice de grado mayor o igual a 3. Claramente $v_i$ está en los extremos de $E_i$ y $E_{i-1}$, al tener grado mayor a 2, $v_i$ incide al menos en tres aristas distintas, $e_1, e_2$ y $e_3$ de $\bar{P}$. Sin pérdida de generalidad podemos asumir que $e_1 \in E_i$ y $e_2 \in E_{i-1}$. Por tanto $v_i$ debe aparecer en una componente poligonal de $\bar{P}$. Claramente $e_1$ y $e_2$ no están en la misma componente poligonal, de lo contrario $v_i$ necesariamente tendría grado 2. Es fácil ver que $e_3 \in E_j$, para alguna $j \neq i$, $j \neq i-1$, de esto se sigue la siguiente Observación:

**Observación 4.2.** Todo vértice $v_i$ de $P$, con grado mayor a 2 en $\bar{P}$, necesariamente incide en dos aristas de una $E_j$, además al menos una de esas aristas pertenece a una componente poligonal de $\bar{P}$.

Supongamos ahora que $T$ tiene más de un vértice con grado mayor a 2 en $\bar{P}$. Sean $u, v$ y $w$, los vértices de $T$, y digamos que $u$ y $v$ tienen grado mayor a 2. Sea $\bar{P}'$ la subgráfica de $\bar{P}$ obtenida al remover la arista $uv$ de $\bar{P}$. Dado que $\bar{P}'$ ya no contiene triángulos, podemos obtener una cuadrilaterización $Q(\bar{P}')$ de $\bar{P}'$, agregando a lo más $\lfloor \frac{n-1}{4} \rfloor$ puntos Steiner. Sea $C$ una 2-coloración de $Q(\bar{P}')$ tal que vértices adyacentes tengan distintos colores. Falta probar que cualquiera de las clases cromáticas de $C$ vigila a $P$.

Es claro que $u$ y $v$ pertenecerán a la misma clase cromática en $C$, digamos la clase azul, y $w$ a la otra, digamos la roja. Claramente el interior de $\bar{P}$ es vigilado por cualquiera de las dos clases cromáticas. Por el lema 2.1 sabemos que la clase azul vigila todas las habitaciones de $P$ ya que tendremos al menos un guardia por cada arista de $\bar{P}$, sin embargo, no es directo que la clase roja vigile también las habitaciones de $P$.

Notemos que la arista $uv$ necesariamente pertenece a alguna $E_i$. Distinguimos los siguientes subcasos.

**Caso 2.1.** $|E_i| = 1$. En este caso $u$ y $v$ son los extremos de $a_i$, y por la Observación 4.2 existen dos cadenas $E_j$ y $E_s$ tales que $uw \in E_j$ y $vw \in E_s$. Dado que $E_j$ y $E_s$ son cadenas convexas, $w$ vigila completamente la habitación de $a_i$, que era la única habitación que no podíamos garantizar que era vigilada por la clase cromática roja. Figura 10a.

**Caso 2.2.** $|E_i| > 1$, y $u$ o $v$ está en algún extremo de $E_i$, Figura 10b. Sin pérdida de generalidad podemos asumir que $u$ es el extremo de $E_i$, nuevamente tenemos que la arista $uw$ pertenece a una cadena $E_j$, donde $u$ no está en ningún extremo de $E_j$. Extendamos la arista $vw$ desde $v$ hasta que toque a $a_i$, sea $v'$ el punto donde la toca, definamos el arco $a_i'$ como la unión del segmento de $a_i$ que conecta a $v'$ con $u$ y el segmento de recta que une a $v'$ con $v$. Análogamente, sea $a_i''$ el arco obtenido al unir el segmento de $a_i$ que conecta a $v'$ con el otro extremo de $a_i$ y el segmento de recta que une a $v'$ con $v$. Entonces podemos partir $P$ en dos polígonos arista-convexos $P_1$ y $P_2$, con la característica de sus conjuntos de vértices son subconjuntos propios de los vértices de $P$, y solo comparten el vértice $v$. No es difícil probar que $\bar{P}_1 \cup \bar{P}_2 = \bar{P}$. Supongamos que $P_1$ es el polígono que contiene a $w$, entonces cualquier clase cromática de $C$ vigila completamente a $P_2$, y todo $P_1$ a excepción posiblemente de la habitación de $a_i'$. Sin embargo, por la observación 4.2, y la forma en que se construyó $a_i'$, el punto $w$ vigila completamente tal habitación, por lo que cualquier clase cromática vigila también a $P_1$, y por tanto a $P$.

**Caso 2.3.** $|E_i| > 1$, y ni $u$ ni $v$ están en algún extremo de $E_i$, Figura 10c. Observamos que al extender las aristas $uw$ y $vw$ en $u$ y $v$ respectivamente hasta que toquen a $a_i$, dividimos $P$ en tres polígonos arista convexos $P_1, P_2$ y $P_3$ de manera similar al caso anterior. Nuevamente $\bar{P}_1 \cup \bar{P}_2 \cup \bar{P}_3 = \bar{P}$, por lo que si $P_1$ es el polígono que contiene a $w$, entonces $P_2$ y $P_3$ no contienen triángulos, por lo que cualquier clase cromática de $C$ los vigila completamente. Vemos que la clase roja también vigila a $P_1$ ya que la única habitación que no podría vigilar sería la de la arista que une a $u$ con $v$ en $P_1$, la cual, debido a como se construyó, está completamente vigilada por $w$. Por tanto cualquier clase cromática vigila a $P$.

De forma análoga podemos eliminar todos los triángulos de $\bar{P}$. Por tanto hemos probado el siguiente resultado:

**Teorema 4.3.** *Sea $P$ un polígono arista-convexo con $n$ vértices, entonces $\lfloor \frac{5n}{8} \rfloor$ puntos guardia son siempre suficientes para vigilar a $P$.*

# 5. Conclusiones

En este artículo demostramos que siempre es posible vigilar un polígono arista-convexo $P$ colocando a lo más $\lfloor \frac{5n}{8} \rfloor$ guardias en cualquier parte de $P$. Cuando se restringen los guardias a estar solo en vértices, $\lfloor \frac{2n}{3} \rfloor$ son siempre suficientes y a veces necesarios.

No conocemos ningún polígono arista-convexo que necesite más de $\lceil \frac{n}{2} \rceil$ puntos guardia para ser vigilado. Conjeturamos que $\lceil \frac{n}{2} \rceil$ puntos guardia son siempre suficientes para vigilar cualquier polígono arista-convexo. Nuestro resultado es ya interesante, porque esta es la primera variante que conocemos

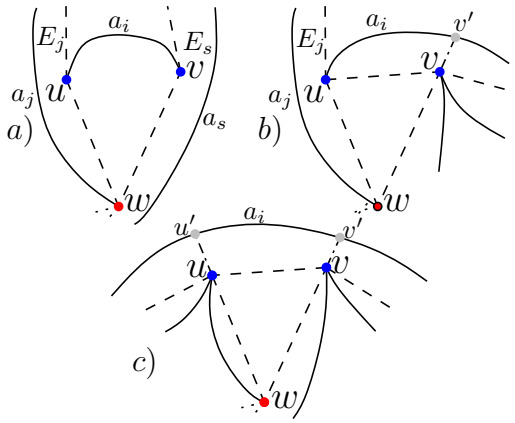



Figura 10: Ilustración del caso 2.

del problema de la galería de arte en la que se tienen cotas distintas sobre el número de vértices guardia y puntos guardia suficientes para vigilar nuestra galería de arte. En la versión completa de este artículo, probamos que si el polígono subyacente de $P$ es un *polígono ortogonal*, esto es cada una de sus sus aristas es paralela al eje $x$ o $y$, $\frac{n}{2}$ guardias son siempre suficientes y a veces necesarios para vigilar $P$.

## Referencias

[1] V. Chvátal, *A combinatorial theorem in plane geometry*, J. Combin. Theory Ser B 18 (1975), 39-41

[2] S. Fisk, *A short proof of Chvátal's watchman theorem*, J. Combin. Theory Ser B 24 (1978), 374

[3] M. I. Karavelas and E. P. Tsigaridas, *Guarding curvilinear art galleries with vertex or point guards*, Research Report 6132, INRIA, 2007

[4] M. I. Karavelas, Cs. D. Tóth and E. P. Tsigaridas *Guarding curvilinear art galleries with vertex or point guards*, Computational Geometry: Theory and Applications, to appear.

[5] J. O'Rourke. *Art Gallery Theorems and Algorithms.* The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.

[6] S. Ramaswami, P. Ramos and G. Toussaint. *Converting triangulations to quadrangulations.*, Computational Geometry: Theory And Applications, vol. 9, pp 257-276, 1998.

[7] T. C. Shermer. *Recent results in art galleries.* Proc. IEEE, 80(9):1384-1399, 1992.

[8] J. Urrutia, *Art gallery and illumination problems*, J.-R. Sack and J. Urrutia editors, *Handbook of Computational Geometry*, 973-1027. North-Holland, 2000.

# Aproximando la iluminación por módems*

A.L. Bajuelos† S. Canales, ‡ G. Hernández§ A. M. Martins¶

**Resumen**

En este artículo obtenemos soluciones aproximadas a una generalización del problema clásico de iluminación de polígonos. En lugar del foco de luz habitual, consideramos un dispositivo inalámbrico cuya señal puede atravesar un determinado número número $k$ de paredes. Estos dispositivos son designados por *k-módems*. Presentamos un algoritmo para construir la región poligonal cubierta por un $k$-módem y analizamos con técnicas metaheurísticas el problema de minimizar el número de $k$-módems, situados en vértices, que son necesarios para cubrir un polígono de $n$ lados. Se obtienen resultados para $k = 2$ y $k = 4$ tanto en polígonos generales como ortogonales.

## 1 Introducción

La iluminación y los problemas de visibilidad han sido siempre áreas de estudio en Matemáticas e Informática, especialmente en *Geometría Computacional*. La definición clásica de visibilidad asegura que dos puntos $x$ e $y$ de un conjunto $D$ en $\Re^2$ son visibles si el segmento $\overline{xy}$ está completamente contenido en $D$ (ver [7]). Sin embargo, el desarrollo de internet y de las redes inalámbricas inspiran nuevas investigaciones en este área de la *Geometría Computacional*, como se muestra en [3, 4, 8].

Recientemente Aichholzer y otros definen en [5] el problema de *iluminación con módem* para regiones poligonales. Existen dos factores clave a la hora de conectar un ordenador a una red inalámbrica: la distancia al módem inalámbrico y el número de paredes que separan nuestro ordenador de dicho módem. Así, se dice que un punto $x$ en un polígono $P$ está cubierto o iluminado por un $k$-módem $w$ en $P$ si el segmento $\overline{xw}$ cruza a lo sumo $k$ paredes (aristas) de $P$, (ver figura 1). Entonces podemos plantear el siguiente problema: *Dado un polígono $P$ con $n$ vértices, ¿cuál es el número mínimo de $k$-módems (situados sobre puntos de $P$) a veces necesarios y siempre suficientes para cubrir $P$?* Es fácil observar que $(i)$ para el caso en que $k = 0$ este problema se reduce al problema clásico de la Galería de Arte y que $(ii)$ para $k = n$ bastaría un único $n$-módem situado en cualquier punto para cubrir $P$ (solución trivial).

En [5] se obtienen cotas combinatorias ajustadas de este problema para polígonos monótonos, quedando abierto el problema para polígonos generales y ortogonales. Partiendo del presupuesto de que este problema es $\mathcal{NP}$-*duro* (como muchas variantes de los problemas de Galería de Arte) tiene sentido abordarlo aplicando técnicas de resolución aproximadas.

Este trabajo esta organizado de la siguiente forma: en la sección 2 se presentan dos algoritmos que calculan la región cubierta por un $k$-módem situado en un punto de un polígono con $n$ lados. Uno para $k$ fijo y otro que construye todas las regiones de visibilidad (cubiertas) para todos los posibles valores de $k$ $(0 \leq k \leq n)$. En la sección 3 se discuten las técnicas metaheurísticas diseñadas para resolver de forma aproximada el problema de la minimización del número de $k$-módems, basadas en un método

*Parcialmente subvencionado por los proyectos MEC MTM2008-05043, Acción Integrada HP2008-0060, CEOC con *Programa* POCTI, FAC, cofinanciado por EC fondos FEDER y apoyado por la beca FCT SFRH/BD/19138/2004

†Departament of Mathematics & CEOC. University of Aveiro. leslie@ua.pt

‡Escuela Técnica Superior de Ingeniería, ICAI. Universidad Pontificia Comillas de Madrid. scanales@dmc.icai.upcomillas.es

§Facultad de Informática. Universidad Politécnica de Madrid. gregorio@fi.upm.es

¶Departament of Mathematics & CEOC. University of Aveiro. ana.martins@ua.pt

híbrido que utiliza tanto *algoritmos genéticos* como *simulated annealing*. Presentamos a continuación en la sección 4 los resultados experimentales obtenidos para $k = 2$ y $k = 4$. Se realizó un estudio detallado del número mínimo de $k$-módems que producen en media dichas técnicas sobre conjuntos de polígonos generados aleatoriamente distinguiendo dos casos: polígonos simples arbitrarios ó generales y polígonos ortogonales. Los resultados obtenidos muestran que, aproximadamente y en media, el número de 2-módems necesarios para cubrir un polígono $P$ con $n$ lados es $\lfloor \frac{n}{26.10} \rfloor$, siendo $\lfloor \frac{n}{52.35} \rfloor$ si consideramos 4-módems. Si los polígonos son ortogonales, los valores obtenidos son $\lfloor \frac{n}{27.39} \rfloor$ y $\lfloor \frac{n}{57.47} \rfloor$, respectivamente.

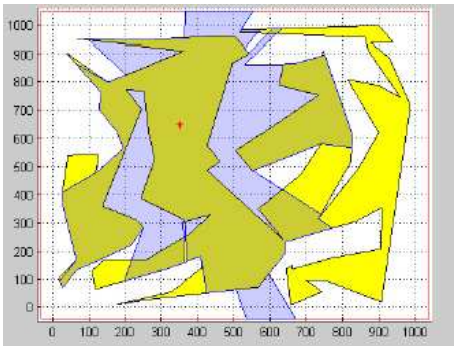



Figura 1: Zona cubierta por un 2-módem en un polígono con 100 vértices

# 2 Región de visibilidad de un $k$-módem

Sea $P$ un polígono simple con $n$ vértices $\{v_0, v_1, \ldots, v_{n-1}\}$, y $x$ un punto de $P$ donde se ha ubicado un $k$-módem. Construiremos la región iluminada desde $x$ atravesando a lo sumo $k$ paredes. Esta región tendrá zonas del interior de $P$ y zonas del exterior. Por razones de simplicidad consideramos $P$ contenido en una caja rectangular $R$ y construiremos la región visible en el interior de la caja.

## 2.1 Región visible para $k$ fijo

En primer lugar describimos un algoritmo que construye la región visible $Q$ desde un $k$-módem situado en un punto $x$, con $k$ fijo. Suponemos que $x$ está en posición general y que los vértices de $P$ están ordenados en sentido positivo. Los pasos del algoritmo son:

1. **Haz de semirrectas desde $x$.**

   Se trazan las semirrectas con origen en $x$ y que pasan por los vértices críticos de $P$ para $x$. Un vértice $v$ es crítico para $x$ si los vértices anterior y posterior a $v$ se encuentran en el mismo semiplano respecto de $\overline{vx}$, (ver figura 3(a)). Se ordenan angularmente desde $x$ estas semirrectas.

2. **Trazado del borde de la región visible $Q$**

   (a) Primero detectamos un punto $z$ en ese borde. Consideramos la semirrecta horizontal (a la derecha de $x$) y detectamos el punto de corte $z$ con $P$ tras atravesar $k$ paredes. En este punto $z$ comenzamos a trazar el borde de $Q$ en el sentido positivo de $P$. (Si se alcanza el rectángulo $R$ sin llegar a atravesar $k$ paredes, se toma el punto de corte con $R$ como punto $z$).

   (b) Si $k$ es par se avanza por $P$ (en sentido positivo) hasta alcanzar la siguiente semirrecta del haz. Si $k$ es impar se retrocede por el borde de $P$ hasta alcanzar la siguiente semirrecta.

   (c) Si $t$ es el punto de corte con dicha semirrecta, se distinguen dos casos: $(i)$ $t$ está más cerca de $x$ que el vértice crítico, en cuyo caso se prosigue por el borde de $P$ y $(ii)$ $t$ está más lejos de $x$ que el punto crítico. Entonces se avanza por la semirrecta, o bien hacia el punto crítico (casos 1 y 2), o bien en sentido contrario (casos 3 y 4) según se indica en la figura 2. El siguiente vértice del polígono de visibilidad $Q$ es, o bien el punto crítico (casos 1 y 2), o bien el segundo punto posterior de corte de la semirrecta con $P$.

3. Se continua la construcción del borde de $Q$ siguiendo la nueva arista hasta alcanzar la siguiente semirrecta. Y se repite el análisis anterior. Se prosigue hasta cerrar $Q$, polígono de visibilidad desde el $k$-módem $x$.

*Complejidad del algoritmo.* La complejidad de este algoritmo proviene del paso de ordenación de las semirrectas en tiempo $\mathcal{O}(n \log n)$. La construcción posterior del polígono $Q$ se realiza en tiempo lineal, pues cada lado de $P$ sólo interviene $\mathcal{O}(1)$ veces.

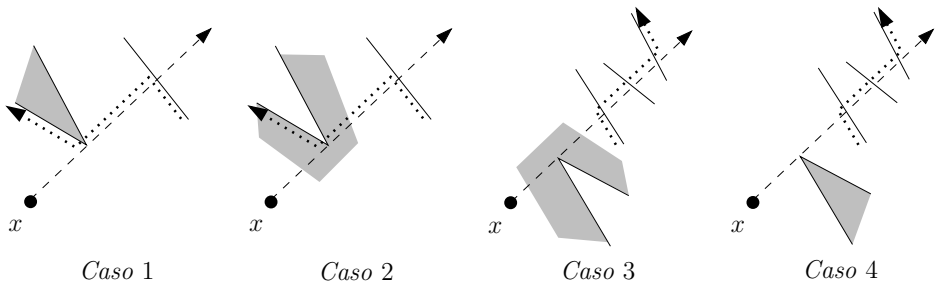



Figura 2: Casos posibles para $k$ fijo

## 2.2 Regiones visibles para $k$ variable

Si se desea calcular la región visible desde $x$ con varios $k$-módems y diferentes valores de $k$, el algoritmo anterior permite calcular cada uno en $\mathcal{O}(n \log n)$. Así si se quiere la región $k$-visible para $t$ valores diferentes de $k$, el coste sería $\mathcal{O}(tn \log n)$. Sin embargo, se pueden calcular todas las regiones de visibilidad desde $x$ para todos los posibles valores de $k$ en tiempo $\mathcal{O}(n^2)$ con el algoritmo cuyos pasos se describen a continuación:

1. Trazar todas las semirrectas con origen en $x$, y que pasan por los vértices críticos (ver figura 3(a)). Determinar los puntos de intersección de las semirrectas con cada uno de los lados del polígono $P$.

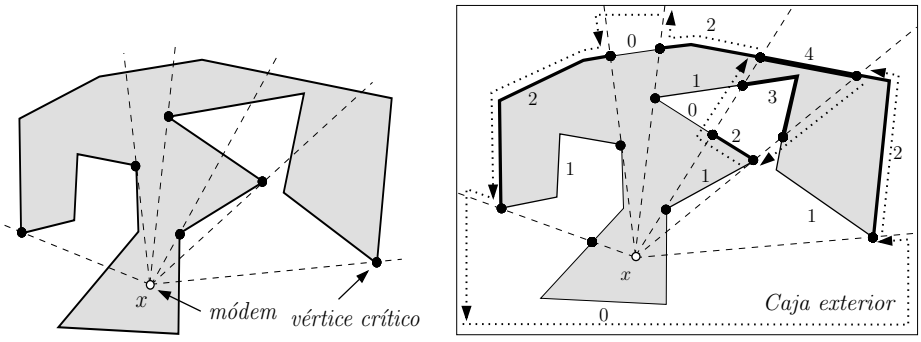



Figura 3: (a) Semirrectas y uno de los vértices críticos de $P$; (b) Etiquetado de segmentos y ruta para la construcción de la región cubierta por un 2-módem

2. Las semirrectas dividen cada lado del polígono en uno o varios segmentos. Etiquetamos cada segmento con el número de paredes atravesadas por un rayo que partiendo de $x$ alcance el segmento, (ver figura 3(b)).

3. La región visible desde $x$ con un $k$-módem se construye conectando los segmentos con etiqueta $k$, utilizando para ello las semirrectas incidentes en sus extremos. Si en uno de los sectores determinado por dos semirrectas consecutivas no hay segmento de etiqueta $k$, la conexión se efectúa por el borde de la caja que contiene al polígono y evidentemente si no hay ningún

segmento con etiqueta $k$ toda la caja que contiene al polígono está iluminada. En la figura 3(b) podemos ver los segmentos etiquetados y la conexión de los segmentos etiquetados con un 2, para construir la región cubierta por un 2-módem.

*Complejidad del algoritmo.* Cada semirrecta que parte del módem puede cortar a todos los lados del polígono, por lo que el número total de etiquetas de los segmentos es cuadrático. Por tanto, la fase de etiquetado tiene complejidad $O(n^2)$. La construcción del polígono de visibilidad para cada valor de $k$ fijo se puede hacer en tiempo lineal usando el algoritmo descrito en las sección 2.1, por lo que la construcción de todos los polígonos de $k$-visibilidad se efectúa en tiempo cuadrático.

# 3 Técnicas Metaheurísticas

Un conjunto recubridor de $k$-módems vértice $G$ es un subconjunto de vértices de $P$ tal que $\bigcup_{r\in G} Vis(r) = P$, siendo $Vis(r)$ la *región de visibilidad* del módem $r$. Desarrollamos una técnica metaheurística híbrida que conjuga los algoritmos genéticos y la heurística *simulated annealing*, para determinar el menor conjunto de $k$-módems vértice recubridores.

Según se muestra en [1], después de utilizar una técnica heurística sobre un problema de optimización, (en ese caso el problema estudiado era el Minimum Vertex Guard Set, para el cálculo del número mínimo de luces que iluminan un polígono), es preciso en algunas ocasiones aplicar un postproceso que permita afinar más la solución obtenida. Basándonos en esta experiencia desarrollamos para el problema una técnica híbrida, que fundamentalmente recurre a un *algoritmo genético* pero donde además de los operadores clásicos de *cruce* y *mutación*, se añade un nuevo operador basado en la heurística *simulated annealing (SA)*. Básicamente el proceso consiste en aplicar *SA* tras utilizar el operador de *cruce*, con el objetivo de refinar la solución que produce dicho operador. Tras esta operación se puede aplicar el operador de *mutación* correspondiente al algoritmo genético. Veamos a continuación la adaptación de *simulated annealing* a nuestro problema y después explicaremos la adaptación general del algoritmo genético.

## 3.1 Simulated Annealing

La principal ventaja de la heurística *simulated annealing (SA)* con respecto a otros métodos de aproximación es su capacidad para evitar quedar atrapada en un máximo ó mínimo local. El método emplea una búsqueda aleatoria que no sólo acepta los cambios que mejoran la función objetivo, sino que también acepta cambios que la empeoren. Estos últimos se aceptan con un cierta probabilidad que depende de un parámetro de control llamado temperatura $T$, que disminuye en cada iteración según un conjunto de reglas.

Para resolver un problema de optimización con la estrategia *SA*, es necesario identificar lo siguiente: (*i*) **Parámetros Específicos**: *espacio de soluciones, función objetivo, vecindad de cada solución y solución inicial*; (*ii*) **Parámetros Generales**: *temperatura inicial* ($T_0$)*, regla de disminución de la temperatura, número número de iteraciones para cada valor de* $T$*,* ($N(T)$) y *condición de parada.* Describimos a continuación la adaptación de estos parámetros a nuestro problema.

### 3.1.1 Parámetros Específicos

*Espacio de soluciones.* El espacio de soluciones $S$ está formado por los subconjuntos de vértices de $P$, donde se colocan los $k$-módems. Representamos $S$ como $S = \{S_1, S_2, \ldots, S_m\}$, donde $S_i = v_0^i v_1^i \ldots v_{n-1}^i$ con $i = 1, \ldots, m$. De esta manera cada candidato $S_i$ está representado por una cadena de longitud $n$, donde $v_j^i$ con $j \in \{0, \ldots, n-1\}$ representa al vértice $v_j \in P$ y su valor es 0 ó 1. Si $v_j^i = 1$ entonces hay un $k$-módem en dicho vértice, siendo $v_j^i = 0$ en caso contrario.

*Función Objetivo.* La función objetivo $f : S \to \mathbb{N}$ asigna a cada elemento de $S$ el cardinal del correspondiente conjunto de $k$-módems.

*Vecindad de cada solución.* Según *SA*, para cada posible solución $S_i \in S$, se obtiene un vecino $S_{i+1} \in S$, que será el elemento a analizar en la siguiente iteración. Aquí, dado $S_i = v_0^i v_1^i \dots v_{n-1}^i$, generamos un número aleatorio natural $t \in [0, n-1]$ y procedemos de la siguiente manera: (a) si $v_t^i = 1$ ponemos $v_t^{i+1} = 0$, aceptando la nueva solución si es válida y rechazándola en caso contrario; (b) si $v_t^i = 0$ ponemos $v_t^{i+1} = 1$, aceptando esta nueva solución con un cierta probabilidad, pues en este caso empeoramos la solución.

*Solución Inicial.* Un conjunto de $k$-módems situados en los vértices de $P$ es nuestra solución inicial $S_0$. Se ha tomado la solución obtenida por el algoritmo genético tras aplicar el operador de *cruce* como se explica en la sección 3.2.

### 3.1.2 Parámetros Genéricos

Basándonos en los estudios realizados para un problema de iluminación con características similares a las que tratamos en este artículo, hemos considerado los siguientes parámetros para nuestro algoritmo:

- *Temperatura Inicial* ($T_0 = \frac{n}{4}$). Valor dependiente del número de vértices del polígono.
- *Regla de disminución de la temperatura.* Se considera la regla definida por $T_{k+1} = \frac{T_0}{e^k}$, (*very fast simulated annealing (VFSA) decrease*).
- *Número de iteraciones para cada temperatura*, $N(T_k) = T_k$. Esta elección del número de iteraciones garantiza que el número de iteraciones es mayor a temperaturas altas, que es solución se encuentra más alejada del óptimo.
- *Condición de terminación.* En teoría, el proceso de búsqueda se debe detener cuando se llegue a la congelación, es decir, cuando $T_k = 0$. No obstante, es posible detener el proceso para una temperatura $T_f$ mayor que 0, sin que ello afecte a la calidad de la solución. Por ejemplo se puede parar la búsqueda cuando no se obtenga ya ninguna mejora en la solución. Esta falta de mejora se puede definir de varias maneras, pero una solución útil es la siguiente: detener el proceso cuando no se encuentre ninguna mejora de la solución en la última serie consecutiva de las temperaturas y además el porcentaje de aceptación de nuevas soluciones se sitúa por debajo de un valor (pequeño) $\varepsilon\%$. En este sentido, la condición de terminación elegida en nuestro algoritmo consiste en detener la búsqueda cuando $T_f \leq 0,005$, ó cuando durante las últimas $l = 3000$ series consecutivas de temperaturas no se ha obtenido ninguna mejora en la solución y el porcentaje de aceptación de nuevas soluciones se sitúa por debajo del $\varepsilon = 2.0\%$.

## 3.2 Algoritmo Genético

Los *Algoritmos Genéticos (AG)* son técnicas que simulan los procesos de evolución biológica en la naturaleza (véase por ejemplo [6]). Para resolver un problema de optimización con *AG* es necesario identificar los siguientes parámetros: una representación de las posibles soluciones llamadas *individuos*, para el problema (*codificación*); una *población inicial*; una *función objetivo* adecuada; operadores genéticos (*selección*, *cruce* y *mutación*) y diversos valores de los parámetros utilizados por el algoritmo genético (por ejemplo, el tamaño de la población, la probabilidad de los operadores genéticos, la evaluación de la población y la condición de finalización). La adaptación de todos estos parámetros a nuestro problema se explica detalladamente en [1]. Un individuo $I$ está representado por una cadena $I = g_0 g_1 \dots g_{n-1}$, donde cada $g_i$ representa al vértice $v_i$ de $P$ y su valor puede ser 0 ó 1. Si $g_i = 1$, tenemos situado un $k$-módems en $v_i$, siendo $g_i = 0$ en caso contrario. El tamaño de la población que consideramos es el número de vértices cóncavos, (es decir con un ángulo interior superior a $\pi$) de $P$, y si $R = \{u_0, u_1, \dots, u_{r-1}\}$ es dicho conjunto de vértices cóncavos, para generar la población inicial aplicamos el siguiente proceso: $\forall i \in \{0, \dots, r-1\}$, si colocando un $k$-módem en cada vértice de $R \backslash \{u_i\}$ cubrimos todo el polígono, admitimos $R \backslash \{u_i\}$ como individuo de la población, tomando todos los vértices de $R$ como individuo en caso contrario. La función objetivo se define por $f(I) = \sum_{j=0}^{n-1} g_j$ y para los operadores de selección y cruce usamos el método de selección por torneo y una variante de cruce en un punto con probabilidad $p_c = 80\%$, respectivamente. Tras aplicar el operador de cruce

a dos individuos aplicamos con una probabilidad $p_{sa} = 10\%$ la heurística $SA$ al individuo obtenido para obtener un refinamiento de la solución. El proceso de mutación que se aplica en el algoritmo es el cambio con una probabilidad $p_m = 5\%$ de cada dígito binario de 0 a 1 ó viceversa. La evaluación de la población se obtiene tomando el menor de los valores obtenidos por la función objetivo $f$ en cada uno de los individuos, finalizando el algoritmo cuando dicha función no mejora en 500 generaciones.

# 4 Resultados Experimentales

La implementación de nuestro algoritmo se hizo en C/C++ (usando MS Visual Studio 2005) con CGAL 3.2.1 [2], realizando las pruebas en un PC Intel(R) Core (TM) 2 CPU 6400 a 2.66 Ghz. y 1 GB de RAM. Hemos realizado un significativo número de experimentos utilizando para ello dos generadores aleatorios de polígonos: para polígonos generales se uso la función de CGAL *random_polygon_2* y para polígonos ortogonales el generador desarrollado por Joseph O'Rourke (comunicación personal 2002). En las subsecciones 4.1 y 4.2 se dan los resultados para $k = 2$ y $k = 4$ respectivamente, tanto para polígonos generales como para polígonos ortogonales. En cada caso se proporciona el número mínimo de $k$-módems obtenido en función del número de vértices de los polígonos, así como los tiempos de respuesta y el número de iteraciones realizado por el algoritmo. Además se proporciona las funciones lineales obtenidas al realizar un ajuste por mínimos cuadrados, relacionando el número de vértices de los polígonos con el resultado en $k$-módems proporcionado por el algoritmo.

## 4.1 Resultados para 2-módems

En la siguientes tablas presentamos los datos obtenidos al aplicar nuestro algoritmo a conjuntos de 40 polígonos con 30, 50, 70, 100, 110, 130, 150 y 200 vértices cada uno de ellos, tanto para polígonos generales (Tabla 1) como ortogonales (Tabla 2). Para cada conjunto de polígonos presentamos el número promedio de 2-módems que proporciona como solución el algoritmo, así como el tiempo medio de respuesta y el número de iteraciones.

- **Resultados para polígonos generales** Los resultados que se presentan a continuación se han obtenido aplicando un operador de selección por torneo y cruce en un punto con probabilidad de cruce $pc = 0.8$, probabilidad de aplicar $SA$ $psa = 0.1$ y probabilidad de ejecutar el operador de mutación $pm = 0.05$.

| ***Vértices*** | 2-***módems*** | ***Tiempo/Runtime(seg.)*** | ***Iteraciones*** |
|---|---|---|---|
| 30 | 1.9 | 7.775 | 532.725 |
| 50 | 2.675 | 37.35 | 541.575 |
| 70 | 3.45 | 114.55 | 577.3 |
| 100 | 4.55 | 343.5 | 613.2 |
| 110 | 4.875 | 448.825 | 653.1 |
| 130 | 5.725 | 723.575 | 629.7 |
| 150 | 6.65 | 1091.5 | 613.7 |
| 200 | 8.35 | 2644.3 | 702.75 |

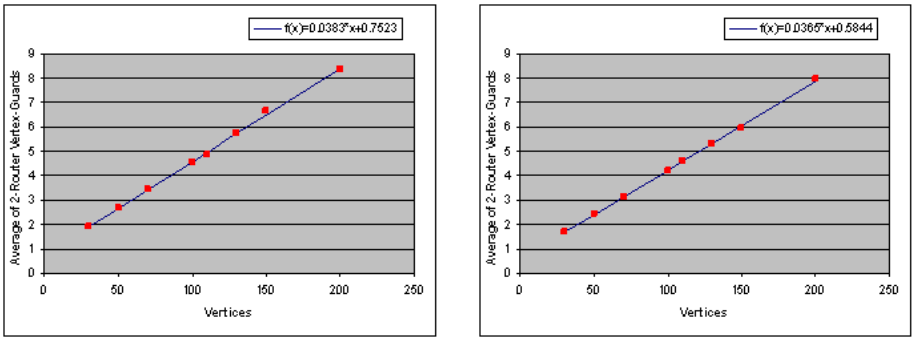
Tabla 1: Resultado para $k = 2$ en polígonos generales

  Usando el método de mínimos cuadrados, la función que ajusta el número de 2-módems con relación al número $n$ de vértices del polígono es $f(n) = 0.0383n + 0.7523 \approx \frac{n}{26.10} + 0.7523 \approx \frac{n}{26.10}$, con un factor de correlación de 0.9988, (ver figura 4(a))

- **Resultados para polígonos ortogonales** Con las mismas probabilidades que en el punto anterior la tabla de resultados para polígonos ortogonales con $k = 2$ es la Tabla 2 y la recta de ajuste correspondiente es $f(n) = 0.0365n + 0.5844 \approx \frac{n}{27.39} + 0.5844 \approx \frac{n}{27.39}$, con un factor de correlación de 0.9988, (ver figura 4(b))

| *Vértices* | 2-*módems* | *Tiempo/Runtime(seg.)* | *Iteraciones* |
|---|---|---|---|
| 30 | 1.675 | 6.1 | 523.075 |
| 50 | 2.45 | 32.25 | 559.275 |
| 70 | 3.15 | 101.375 | 547.65 |
| 100 | 4.2 | 318.175 | 618.6 |
| 110 | 4.6 | 409.675 | 585.375 |
| 130 | 5.325 | 708.175 | 604.6 |
| 150 | 5.95 | 1091.9 | 692.7 |
| 200 | 7.95 | 2541.3 | 656.45 |

Tabla 2: Resultado para $k = 2$ en polígonos ortogonales




(a) (b)

Figura 4: Ajuste lineal $k = 2$: (a) polígonos generales; (b) polígonos ortogonales

Por tanto podemos concluir que, en media y de forma aproximada, el número de 2-módems vértice necesarios para cubrir un polígono general es $\lceil \frac{n}{26.10} \rceil$ y $\lceil \frac{n}{27.39} \rceil$ si el polígono es ortogonal.

## 4.2 Resultados para 4-módems

Si $k = 4$, los resultados experimentales tanto para polígonos generales como ortogonales son los siguientes:

- **Resultados para polígonos generales**

| *Vértices* | 4-*módems* | *Tiempo/Runtime(seg.)* | *Iteraciones* |
|---|---|---|---|
| 30 | 1.075 | 4.475 | 538.65 |
| 50 | 1.65 | 30.975 | 517.675 |
| 70 | 2.0 | 114.975 | 539.625 |
| 100 | 2.65 | 417.45 | 582.175 |
| 110 | 2.825 | 550.4 | 543.825 |
| 130 | 3.05 | 1014.9 | 603.4 |
| 150 | 3.575 | 1527.0 | 572.1 |
| 200 | 4.375 | 4207.4 | 613.3 |

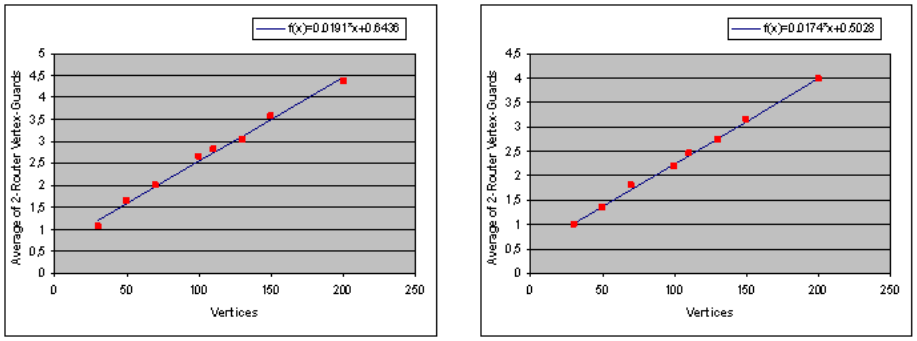
Tabla 3: Resultado para $k = 4$ en polígonos generales

La recta de ajuste para los datos de la siguiente tabla es $f(n) = 0.0191n + 0.6436 \approx \frac{n}{52.35} + 0.6436 \approx \frac{n}{52.35}$, (ver figura 5(a))

- **Resultados para polígonos ortogonales** Igualmente la recta de ajuste para los datos de la Tabla 4 es $f(n) = 0.0174n + 0.5028 \approx \frac{n}{57.47} + 0.5028 \approx \frac{n}{57.47}$, (ver figura 5(b)).

| ***Vértices*** | ***4-módems*** | ***Tiempo/Runtime(seg.)*** | ***Iteraciones*** |
|---|---|---|---|
| 30 | 1.0 | 2.725 | 517.6 |
| 50 | 1.35 | 22.825 | 555.5 |
| 70 | 1.8 | 87.6 | 536.15 |
| 100 | 2.225 | 359.15 | 594.2 |
| 110 | 2.45 | 481.475 | 603.625 |
| 130 | 2.75 | 827.475 | 567.7 |
| 150 | 3.15 | 1535.0 | 625.775 |
| 200 | 3.975 | 4088.894 | 649.325 |

Tabla 4: Resultado para $k = 4$ en polígonos ortogonales




(a) (b)

Figura 5: Ajuste lineal $k = 4$: (a) polígonos generales; (b) polígonos ortogonales

Estos resultados permiten asegurar que, de forma aproximada, el número de 4-módems necesarios para cubrir un polígono es $\lceil \frac{n}{52.35} \rceil$ en polígonos generales y $\lceil \frac{n}{57.47} \rceil$ en polígonos ortogonales

# Referencias

[1] A.L. Bajuelos, S. Canales, G. Hernández y A. M. Martins, Optimizing the Minimum Vertex Guard Set on Simple Polygons via a Genetic Algorithm, *WSEAS Transactions in Information Science and Applications*, Vol 5, Issue 11, pp. 1584-1596, 2008.

[2] CGAL, Computational Geometry Algorithms Library, *http://www.cgal.org*.

[3] D. Christ, M. Hoffmann, Y. Okamoto y T. Uno, Improved Bounds for Wireless Localization. *Proc. 11th Scandinavian Workshop on Algorithm Theory*, pp. 77–89, 2008.

[4] D. Eppstein, M.T. Goodrich y N. Sitchinava, Guard Placement for Efficient Point-in-Polygon Proofs. *Proc. 23rd Symposium on Computational Geometry*, pp. 27–36, 2007.

[5] O. Aichholzer, R. Fabila-Monroy, D. Flores-Peñaloza, T. Hackl, C. Huemer, J. Urrutia y B. Vogtenhuber, Modem Illumination of Monotone Polygons. *Proc.25th European Workshop on Computational Geometry*, pp. 167-170, Belgium, 2009.

[6] Reeves, C.R: Genetic Algorithms. *Handbook of Metaheuristics*, F. Glover e G.A. Kochenberger (eds). Kluwer, Boston, 55–82, (2003)

[7] J. Urrutia, Art Gallery and Illumination Problems. In: J.R. Sack, J. Urrutia (eds.) *Handbook of Computational Geometry*, pp. 973–1027, Elsevier Science Publishers B.V., 2000.

[8] Y. Wang, C. Hu, Y. Tseng, Efficient Placement and Dispatch of Sensors in a Wireless Sensor Network. *IEEE Transactions on Mobile Computing*, 7:262–274, 2008.

# Min-energy Broadcast in Fixed-trajectory Mobile *Ad-hoc* Networks

J.M. Díaz-Báñez * R. Fabila-Monroy † D. Flores-Peñaloza †
M. A. Heredia † J. Urrutia †

**Abstract**

This paper concerns about mobile *ad-hoc* wireless networks, but with the added restriction that each radio station has a rectilinear trajectory. We focus on the problem of computing an optimal range assignment for the stations, which allows to perform a broadcast operation from a source station, while the overall energy deployed is minimized. An $O(n^3 \log n)$-time algorithm for this problem is presented, under the assumption that all stations have equally sized transmission ranges. However, we prove that the general version of the problem is NP-hard, and it is not approximable within a sub-logarithmic factor (unless P=NP). We then consider a special case of the general problem (also NP-hard), and present an approximation algorithm whose approximation factor is $(\ln n + 1)m$, where $m$ is the size of the optimum solution.

## 1 Introduction

In recent years, optimization problems in ad-hoc wireless networks have attracted significant attention due to their potential applications in civil and military domains [5, 11]. Typically, the radio stations in such a network have a limited energy resource (battery for example), and consequently, energy efficiency is an important design consideration for these networks [2, 4]. On the other hand, the rapidly expanding technology of cellular communications, wireless LAN, and satellite servicesadds movement to the stations, making necessary to extend the concept of *ad-hoc* wireless networks to *mobile ad-hoc wireless networks*.

A *broadcast communication* is a task initiated by a source station which has to disseminate a message to all stations in the network [10]. In this paper, we focus on source-initiated broadcasting of messages in mobile ad-hoc wireless networks, yet we restrict the movement of the stations to rectilinear trajectories on the plane and constant velocity. This approach can be applied in settings like satellite networks [12], where the trajectories of the satellites are known beforehand, and the message sending is highly restricted by the positions and transmission ranges of the satellites.

The system model is as follows: Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of moving $n$ points on the plane, our radio stations, moving independently, continuously, and at constant speed on a straight line each one. For a mobile station $s_i \in S$, the *transmission range* of $s_i$, $C_{s_i}$ is a circle of radius $r \geq 0$ centered at $s_i$. A station $s_j$ can receive a transmission from $s_i$ at time $t$ if and only if $s_j \in C_{s_i}$ at time $t$. A *transmission range assignment* for $S$ is a function $R : S \to \mathbb{R}^+$ such that the station $s_i$ has assigned a range size of $R(s_i)$.

The following conditions about message transmission are assumed throughout this paper: A message transmission can be completed in an instant of time. If $s_i$ receives or generates a message at time $t$, then it will pass the message to every station with whom it can communicate at any time $t' \geq t$.

*Universidad de Sevilla, Spain,dbanez@us.es
†Universidad Nacional Autónoma de México, México. ruy@ciencias.unam.mx, dflores@matem.unam.mx, marco@ciencias.unam.mx, urrutia@matem.unam.mx

Given a source station $s \in S$, and a message $M$ generated in $s$ at time $t_0$, we are interested in completing a broadcast operation of $M$ to the rest of the stations. This paper studies the following problems:

**connectivity problem:** Given a transmission range assignment $R$ for $S$, decide if $M$ arrives to every station in $S$.

**min-equal-range problem:** Find the minimum value $r$ needed to perform the broadcast, supposing that each station has a range radius of $r$.

**min-sum problem:** Find a transmission range assignment $R$, such that the broadcast is performed and the sum of the squares of the range radii in $R$ is minimized.

**min-sum-binary problem:** Compute the minimum number of stations needed to perform the broadcast, assuming that each station can only transmit with range radii either 1 or 0.

The power (energy) needed to correctly transmit data from a station $s$ to a station $t$ depends on the term $d(s,t)^\alpha$, where $d(s,t)$ is the Euclidean distance between $s$ and $t$, and $\alpha \geq 1$ is the *distance-power gradient* (see [9]). In an ideal environment $\alpha = 2$.

This paper is organized as follows: Section 2 presents an algorithm for solving the connectivity problem. Section 3 proposes an $O(n^3 \log n)$-time algorithm to solve the min-equal-range problem. In Section 4 we prove that min-sum is an NP-hard problem, and discuss the impossibility of finding an algorithm whose approximation ratio is a sub-logarithmic factor from an optimum solution, unless P=NP. Next, in Section 5 we move to min-sum-binary problem, showing that min-sum-binary is also NP-hard, and then propose an approximation algorithm that achieves an approximation factor of $(\ln n + 1)m$, where $m$ is the size of the optimum solution. Finally, we present our conclusions in Section 6.

## 2 Connectivity

To solve the connectivity problem we propose an algorithm based on Dijkstra's algorithm, which, as a side result, also computes the first time at which each station receives the message.

Given a transmission range assignment $R$, since two different stations $s_i$ and $s_j$ move along different lines, $s_j$ can lie in $C_{s_i}$ only during one time interval, at which $s_i$ can send a message to $s_j$. The *transmission interval* of $s_j$ from $s_i$, $I_{s_i}(s_j)$, is the time interval $[t_a, t_b]$, where $t_a \in \mathbb{R}^+$ is the first instant of time in which $s_j$ lies in $C_{s_i}$, and $t_b \in \mathbb{R}^+$ is the instant of time in which $s_j$ leaves $C_{s_i}$.

We suppose that the trajectory of each $s_i \in S$ is given to us in a way that computing the *transmission interval* of two stations can be done efficiently, consequently this operation is assumed to be computed in constant time.

The *connectivity graph* $G_R$ generated by a set of mobile stations $S$ and a transmission range assignment $R$, is the directed graph having $S$ as vertex set, and there is an arc from $s_i$ to $s_j$ in $G_R$ if and only if $I_{s_i}(s_j)$ is a non empty interval. This arc is labeled with the interval $I_{s_i}(s_j)$. See the left side of Fig. 1 for an example.

If $t_i$ is the first time in which the station $s_i$ receives the message $M$, then $s_i$ can pass the message to a station $s_j$ if $t_i \leq t_b$, where $[t_a, t_b] = I_{s_i}(s_j)$. This concept can be expressed in $G_R$ in the following way: assign the value $t_i$ to the vertex $s_i$; consider the arc from $s_i$ to $s_j$, labeled $[t_a, t_b]$, and assign the time $t_j$ to $s_j$ (the time at which $s_j$ first receives $M$ from $s_i$), where $t_j = t_i$ if $t_i \in [t_a, t_b]$ or $t_j = t_a$ if $t_i \leq t_a$.

The following algorithm, which we call *IS_CONNECTED*, solves the connectivity problem: Construct the connectivity graph $G_R$ of $S$; then assign the time $t_0$ (time at which the source generates the message $M$) to the vertex $s$, the value of $\infty$ to the other vertices and run a modified Dijkstra's

algorithm [3] from $s$, sorting the vertices in the priority queue (of the Dijkstra's algorithm) by the time at which they receive $M$.

Typical Dijkstra's algorithm assures that the vertex going out of the priority queue has assigned the minimum distance to the source. Thus, *IS_CONNECTED* assures that each vertex going out of the queue has assigned the minimum time at which it receives the message $M$. Therefore, if the tree obtained from *IS_CONNECTED* is a spanning tree of $G_R$, then the broadcast from $s$ will succeed (see right side of Fig. 1). As a side result, we obtain the first time at which each vertex receives the message. The correctness and complexity of *IS_CONNECTED* follow from the correctness and complexity of Dijkstra's algorithm. As $G_R$ could be a complete graph, then the total running time of *IS_CONNECTED* is $O(n^2)$ and we arrive to the following result::

**Theorem 2.1.** *The connectivity problem can be solved in $O(n^2)$ time.*

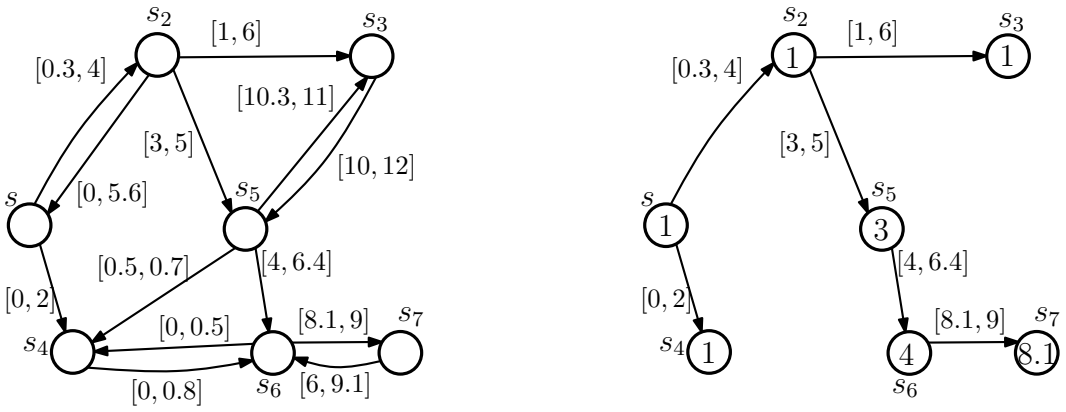



Figure 1: The graph $G_R$ of a mobile network and the spanning tree obtained by the algorithm ($t_0 = 1$).

# 3 Equally sized ranges

In this section we describe an $O(n^3 \log n)$ algorithm to solve the min-equal-range problem.

As the radius of $C_{s_i}$ is equal to the radius of $C_{s_j}$, then $I_{s_i}(s_j) = I_{s_j}(s_i)$, so we will use $I_{s_i}(s_j)$ and $I_{s_j}(s_i)$ interchangeably. This fact will transform the connectivity graph into an undirected graph.

For any given $r \geq 0$, we call $G_r$ the connectivity graph of $S$ obtained by assigning to each station the transmission radius $r$, and $T_r$ the tree obtained from running the algorithm *IS_CONNECTED* on $G_r$. The problem is then reduced to finding the minimum radius $r_{MIN}$ in which $T_{r_{MIN}}$ is a spanning tree of $G_{r_{MIN}}$.

The key idea of our algorithm is to calculate a discrete set of possible values for $r_{MIN}$, and then search over those values. We focus on all radii $r$ such that $T_r$ and $T_{r-\epsilon}$ could be different, for all small $\epsilon > 0$, implying that $T_{r-\epsilon}$ might not be a spanning tree of $G_{r-\epsilon}$. We call these radii *critical radii*.

We say that $r$ is a critical radius for $S$ if by assigning $r$ to all stations in $S$, one of the following cases arises:

a) Two stations in $S$, $s_i$ and $s_j$, have only one time $t$ of connection ($I_{s_i}(s_j) = [t,t]$). See Fig. 2 a).

b) Three different stations in $S$, $s_i$, $s_j$ and $s_k$, have the property of $I_{s_i}(s_j) \cap I_{s_i}(s_k) = [t,t]$. See Fig. 2 b).

A radius of type a) corresponds to the addition of an edge in $G_r$ that is not present in $G_{r'}$, with $r' < r$. The type b) corresponds to radii where an edge of $T_r$ is possibly not present in $T_{r'}$, with $r' < r$. The set of all critical radii of $S$ is denoted by $CR(S)$.

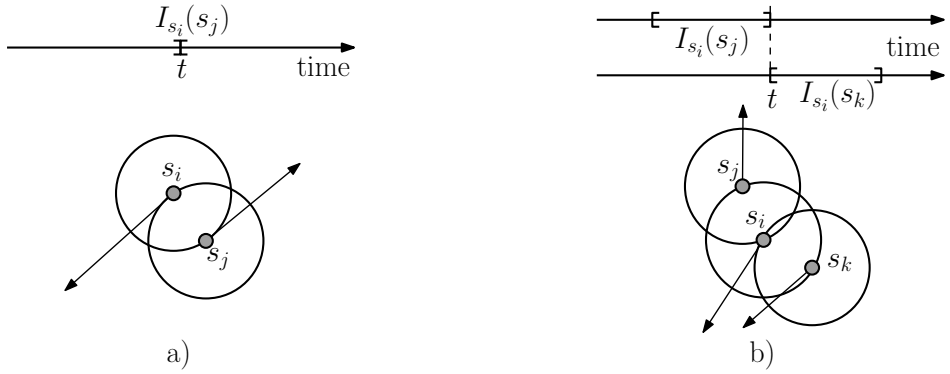



Figure 2: Example of the two cases of critical radius.

Given two different stations, $s_i$ and $s_j$, let $f_{s_i,s_j}(t)$ be the squared Euclidean distance between $s_i$ and $s_j$ at time $t$. As $s_i$ and $s_j$ move along lines, $f_{s_i,s_j}$ is a quadratic polynomial in $t$. Observe that $f_{s_i,s_j} = f_{s_j,s_i}$.

Consider the arrangement of $n-1$ functions involving $s_i$ ($\{f_{s_i,s_j} \mid i \neq j\}$). Any two of these functions intersect at most twice, then the arrangement contains $O(n^2)$ intersections. Each of the $O(n^2)$ intersections gives us a (squared) radius that corresponds to the type b) of critical radii, and each of the $n-1$ function minima gives us a (squared) radius that corresponds to the type a) of critical radii. Since we have $n$ different arrangements, then the size of $CR(S)$ is $O(n^3)$. Assuming that we can: obtain the minima of one of these functions and calculate the intersection of two of these functions in constant time; then we can compute $CR(S)$ in $O(n^3)$ time.

The algorithm *MIN_RADIUS*, for solve the min-equal-range problem, can be defined as follows: Compute the $CR(S)$ set ($O(n^3)$ time), sort the elements in $CR(S)$ ($O(n^3 \log n)$), and look for the minumim radius $r_{MIN}$ in which $T_{r_{MIN}}$ is a spanning tree of $G_{r_{MIN}}$, by using binary search and applying the *IS_CONNECTED* algorithm ($O(n^2)$) at each step. The total running time is then $O(n^3 \log n)$.

In summary, we have proven the following result:

**Theorem 3.1.** *The min-equal-range problem can be solved in $O(n^3 \log n)$ time.*

# 4 The general min-sum problem

This section focus on showing that the general version of min-sum problem is intractable (unless P=NP), by reducing the well-known weighted-set-cover problem [1, 3] to the min-sum problem.

An instance of the weighted-set-cover problem consists of a finite set $A$; a family $B$ of subsets of $A$, such that every element of $A$ belongs to at least one subset in $B$; and a weight function $w : B \to \mathbb{R}^+$. We say that a subset $c \in B$ *covers* the elements of $A$ with weight $w(c)$. The problem is to find a minimum-weight *cover* $\mathcal{C} \subseteq B$ whose members cover all of $A$, where the weight of $\mathcal{C}$ is $\sum_{c \in \mathcal{C}} w(c)$.

It is known that the decision version of weighted-set-cover problem is NP-complete and the optimization (minimization) version is NP-hard [1, 3]. We can prove the NP-hardness of the min-sum problem by reducing the weighted-set-cover problem to the min-sum problem.

Take an instance of the weighted-set-cover problem: $A = \{a_1, a_2, \ldots, a_k\}$ (the covered set), $B = \{b_1, b_2, \ldots, b_l\}$ (the covering set) and the weight function $w : B \to \mathbb{R}^+$. The transformation to an instance of the min-sum problem is as follows (see Figure 3):

Take a set of stations $SB = \{sb_1, sb_2, \ldots, sb_l\}$, collinear in a horizontal line $\mathcal{L}$, at distance $\delta$ (large enough) from each other, and moving leftwards with the same speed. We assign each station $sb_i$ of $SB$ to the element $b_i$ in $B$. The source $s$, that generates the message $M$ at time $t_0$, moves rightwards in a line parallel to $\mathcal{L}$ whose distance to $\mathcal{L}$ is 1, in a way that after a certain time, say $t_1 > t_0$, $s$ could have transmitted the message $M$ to each station of $SB$ if its transmission radius were equal to 1.

Now let $SA = \{sa_1, sa_2, \ldots, sa_k\}$ be a set of static points in the plane far away from the trajectories of $SB$ and $s$, that we call the *meeting points*. Each element $a_i$ in $A$ is assigned to the point $sa_i$ of $SA$.

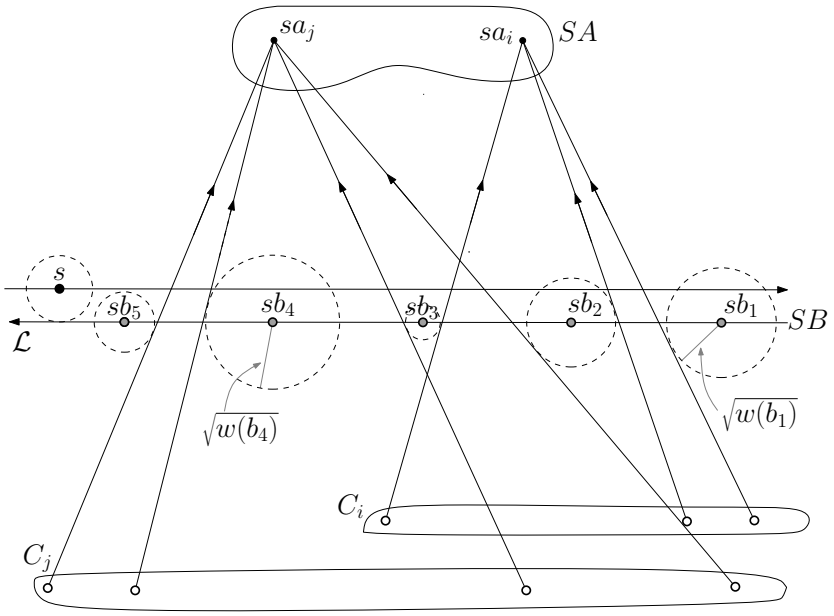



Figure 3: Reduction of weighted-set-cover problem to min-sum problem.

Finally we create the *cover relation* by adding station sets $C = C_1 \cup C_2 \cup \ldots \cup C_k$ in the following way: Take an element $a_i$ in $A$ and take $B_i$ as the set of elements in $B$ that covers $a_i$. For each $b_l$ in $B_i$ we create a new station $c_{i,l}$ in the set $C_i$, in a way that: $c_{i,l}$ passes at distance $\sqrt{w(b_l)}$ from $sb_l$ at time $t_{i,l} > t_1$, passes over $sa_i$ at time $t_{a_i} > t_{i,l}$, and at any time the minimum distance between $sb_l$ and $c_{i,l}$ is $\sqrt{w(b_l)}$. Intuitively, every station $sb_l$ that *covers* $sa_i$ sends a station to $sa_i$, and all stations in $C_i$ arrive to $sa_i$ at the same time ($t_{a_i}$). Notice that we can take suitable times and distances so all the events occur independent from each other.

In some sense we only permit that each element $sb_l \in SB$ have two choices, transmit with radius $\sqrt{w(b_l)}$ or 0. On the other hand, all the stations in $C_i$ can receive $M$ at time $t_{a_i}$ if one of them has it and transmit with radius 0.

Note that the above transformation can be done in polynomial time. A range assignment $R$ that minimizes the sum of the squared radii and allows a broadcast from $s$, is one that: assign radius 1 to $s$; takes $\mathcal{D} \subseteq SB$ such that $\sum_{sb_l \in \mathcal{D}} R(sb_l)^2$ is minimized, where $sb_l \in \mathcal{D}$ has assigned a radius $R(sb_l) = \sqrt{w(b_l)}$; leaves the stations of $SB \setminus \mathcal{D}$ with radius 0; and assures that at least one of the elements in each $C_i$ have the message $M$. But this assignment maps to a subset of $B$ of minimum weight that cover $A$, solving then the weighted-set-cover problem. Since weighted-set-cover problem can only be approximated to within a $\ln|A| + 1$ factor (unless P=NP), no polynomial time approximation algorithm for min-sum problem achieves a smaller approximation factor. The following result can be then established:

**Theorem 4.1.** *The min-sum problem is NP-hard and, unless P=NP, it is not approximable within a sub-logarithmic factor.*

# 5 The min-sum-binary problem

In this section we first prove that the min-sum-binary problem also belongs to the NP-hard complexity class, and then we describe an approximation approach based on the well-known greedy algorithm for the weighted-set-cover problem [1, 3].

## 5.1 NP-hardness

A similar proof to that of the NP-hardness of min-sum, shows that the min-sum-binary problem is NP-hard. By taking an instance of the weighted-set-cover problem, but with all the weights of the elements in $B$ set to 1, we obtain an instance of the classic set-cover problem [7], In the set-cover problem we need to find a minimum-size subset of elements of $B$ whose members cover all of $A$. The set-cover problem is also NP-hard and approximable within a $\ln|A|+1$ factor (unless P=NP) [7, 3, 6].

The same construction used in the previous section can be used to reduce the set-cover problem to the min-sum-binary problem, by assigning radius 1 or 0 to the elements of $SB$.

## 5.2 Approximation algorithm

As we cannot obtain a polynomial time approximation algorithm (for the min-sum-binary problem) that achieves a sub-logarithmic approximation ratio, then we propose an approach based on a greedy algorithm for the weighted-set-cover problem [1, 3] whose approximation ratio is $\ln|A|+1$.

Roughly speaking, at each step of the greedy algorithm, the subset that covers the most elements with the lowest weight is chosen. Specifically, for every subset the number of elements it covers (not yet covered) is divided by its weight. The subset with the highest ratio is then chosen.

In order for $s_i$ to be able to transmit the message $M$ to another station $s_j$, then $s_i$ must have received $M$ before the transmission interval between them ends. Then for each station $s_i$ we consider the times $t_{i,1} \leq t_{i,2} \leq \cdots \leq t_{i,k_i}$ at which $s_i$ loses connectivity with a station (with $k_i \leq n-1$, as we only consider those stations that are ever within transmission range of $s_i$). We denote by $s_{i,j}$, the set of stations that lose connectivity with $s_i$ at or after time $t_{i,j}$. By definition, $s_{i,j+1} \subset s_{i,j}$.

We construct a instance of the weighted-set-cover problem. The set $A = S$ of all stations is the set to be covered. As the family $B$ of subsets used to cover $S$, we use the subsets $s_{i,j}$. We define the weight function $w : B \to \mathbb{R}^+$ such that $w(s_{i,j})$ is the minimum number of stations, including $s_i$, that must be turned on (have transmission range equal to 1) in order for $s_i$ to receive the message at or before time $t_{i,j}$.

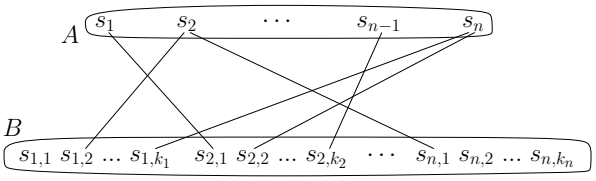



Figure 4: Transforming min-sum-binary problem to weighted-set-cover problem (edges represent the cover relation).

Using the greedy algorithm for weighted set cover at each step we choose the subset $s_{i,j}$ with the best ratio between new stations covered and $w(s_{i,j})$.

Denote by $\mathcal{D}'$ the set of stations (and times) $s_{i,j}$ chosen by the greedy algorithm. It is known that the greedy solution is at most at a $\ln|A|+1$ factor of the optimal solution of the weighted set cover problem. In particular it is at most at a $\ln|A|+1$ factor from any other solution.

Consider an optimal solution $\mathcal{D}_\mathcal{O}$ for the min-sum-binary problem. Let $s_i \in \mathcal{D}_\mathcal{O}$. Assume $s_i$ receives the message for the first time at time $t$. Replace $s_i$ with $s_{i,j}$, where $t_{i,j}$ is the first moment in time after $t$ at which $s_i$ loses connectivity with another station. Doing this for every element $s_i \in \mathcal{D}_\mathcal{O}$, we obtain a covering set ${\mathcal{D}_\mathcal{O}}'$ for our weighted instance of set cover. Therefore:

$$(\ln|A|+1)\left(\sum_{s_{i,j}\in{\mathcal{D}_\mathcal{O}}'} w(s_{i,j})\right) \geq \sum_{s_{i,j}\in\mathcal{D}'} w(s_{i,j}) \tag{1}$$

From $\mathcal{D}'$ we obtain a solution for the min-sum-binary problem. For every $s_{i,j}$: we turn on the corresponding $s_i$; then we turn on the other $w(s_{i,j}) - 1$ stations that enable $s_i$ to receive the message at or before time $t_{i,j}$. Denote by $\mathcal{D}$ the set of stations turned on in this manner.

By construction of $\mathcal{D}$, we have that $|\mathcal{D}| \leq \sum_{s_{i,j} \in \mathcal{D}'} w(s_{i,j})$. Also for every $s_{i,j} \in \mathcal{D_O}'$, its cost cannot be greater than $|\mathcal{D_O}|$. Therefore $\sum_{s_{i,j} \in \mathcal{D_O}'} w(s_{i,j}) \leq |\mathcal{D_O}||\mathcal{D_O}|$ and together with (1), we have:

$$((\ln |A| + 1)|\mathcal{D_O}|)\,|\mathcal{D_O}| \geq |\mathcal{D}| \tag{2}$$

As $|A| = n$, our greedy algorithm thus yields an approximation ratio of $(\ln n + 1)m$, where $m$ is the size of an optimal solution for the min-sum-binary problem.

We now describe how to obtain the weights $w(s_{i,j})$. For the time being, assume that all stations are turned on. For any moment in time $t$ let $G_t$ be the graph with $S$ as vertex set and two stations adjacent if they are currently within transmission range of each other.

Let $t_0$ be the moment in time at which the source receives the message for the first time. Let $t_1 \leq t_2 \leq ... \leq t_p$ be the moments in time at which communication is gained or lost between any two stations.

Consider the sequence of graphs $G_{t_0}, G_{t_1}, \ldots, G_{t_p}$. Two consecutive graphs in the sequence differ by an edge, that is either removed or added. Also since two stations meet each other at most once, the sequence has at most $O(n^2)$ terms.

The minimum number of stations that need to be turned on for $s_i$ to receive the message at time $t_l$ or before is none other than the minimum number of times that the message has been transmitted to reach $s_i$ before or at time $t_l$.

Starting from the source in $G_{t_0}$ we apply Dijkstra's algorithm and compute the distance of every station to the source. This distance (plus one) is the number of hops that the message must have made to reach a given station. For each station we keep track of: the number of hops, the path of stations and the time at which this was achieved.

We move from graph to graph in sequence. When we move to the next graph $G_{t_l}$ and a new edge is added, a station may be able to receive the message in fewer hops. Suppose that the added edge was $\{s_i, s_j\}$ and that, say $s_i$ received the message in $h$ hops, where as $s_j$ received the message in more than $h + 1$ hops. $s_j$ can now receive the message from $s_i$ in fewer hops. Also any station currently adjacent to $s_j$ might receive the message in fewer hops and so on. We compute this new values (keeping the old ones for further reference) by applying again Dijkstra's algorithm on the new graph $G_{t_l}$, with $s_i$ as the starting vertex; with the exception that we relax only the edges that leave vertices that now receive the message in fewer hops. The distance of a station to $s_i$ plus the number of hops used to reach $s_i$ is the number of hops that can now be used to reach the station. This number may or may not be less than the current number of hops. If it is less, we add a new entry with the number of hops, time at which this was achieved and the path used. If linked lists are used to represent paths, this can be done in constant time per station. If the next graph comes from the deletion of an edge, we delete this edge and move on to the next graph.

Once we have visited all graphs in the sequence, every vertex contains a decreasing list of number of hops, time and path. Using this list we can calculate the costs of each $s_{i,j}$ at a linear cost per station, thus $O(n^2)$ in total.

Finally since we have an instance of the weighted-set-cover problem with a set of $n$ elements and a family of $O(n^2)$ subsets, the greedy algorithm can be implemented to run in time $O(n^4)$. Thus the approximation algorithm runs in $O(n^4)$ time in total. We end this discussion by stating the following theorem:

**Theorem 5.1.** *Given a set of n mobile stations, a* $(\ln n + 1)m$*-approximation of the min-sum-binary problem can be computed in* $O(n^4)$ *time, where m is the size of the optimum solution.*

# 6 Conclusions

In this paper, we worked on the problem of energy-efficient source-initiated broadcasting in a mobile wireless network, where every station moves at constant velocity along a linear trajectory. First, we showed that the problem of minimizing the sum of the squares of the range radii for a broadcast operation is polynomial, when all stations have the same radius, and it is NP-hard in the general setting. We then presented a polynomial-time approximation algorithm for a special case of the min-sum problem for which the stations transmit with radius either 1 or 0. Our approach performs approximation within a factor $(\ln n + 1)m$ of the optimal, where $m$ is the size of the optimum solution.

It is worth mention that we never use the fact that the stations move at the same velocity, consequently, our results follows even when the speeds of the stations are different, as long as we forbid infinite speeds. On the other hand, the complexity of the problem does not change if we assume algebraic motions instead of rectilinear motions, as long as intersections and distances calculation between two trajectories can be computed efficiently.

The proof of the NP-hardness of the min-sum problem can be easily modified to proof the NP-hardness of any problem whose goal is to minimize the sum of a power of the range radii, which in terms of energy, makes these problems intractable (unless P=NP) no matter which *distance-power gradient* $\alpha \geq 1$ we choose to optimize, not only for $\alpha = 2$ (see [9]). This fact contrast with the static case of the problem, which is solvable in polynomial time for $\alpha = 1$ (see [2]).

Finally, we must note that the *IS_CONNECTED* algorithm can be used to obtain a simple heuristic method for the min-sum problem, by applying a tabu search strategy [8].

# References

[1] V. Chvatal. *A greedy heuristic for the set covering problem*. Math. Oper. Res., Vol. 4, 233–235, 1979.

[2] A. Clementi, P. Crescenzi, P. Penna, G. Rossi, P. Vocca. *On the Complexity of Computing Minimum Energy Consumption Broadcast Subgraphs*. In Proc. of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS-01), LNCS 2010, 121–132, 2001.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest. *Introduction to Algorithms*. MIT Press, McGraw-Hill, second edition, 2001.

[4] G. K. Das, S. Das, S. C. Nandy. *Range assignment for energy efficient broadcasting in linear radio networks*. Theoretical Computer Science, Vol. 352, 332–341, 2006.

[5] D. Estrin, R. Govindan, J. Heidemann, S. Kumar. *Next century challenges: scalable coordination in sensor networks*. Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, 263 – 270, 1999.

[6] U. Feige. *A Threshold of ln n for Approximating Set Cover*. Journal of the ACM (JACM), Vol. 45 n.4, 634 – 652, 1998.

[7] M. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, 1979.

[8] F. Glover, M. Laguna. *Tabu Search*. Kluwer Academic, 1997.

[9] K. Pahlavan, A. Levesque. *Wireless information networks*. Wiley-Interscience, 1995.

[10] D. Peleg. *Time-Efficient Broadcasting in Radio Networks: A Review*. In Proc. of the 4th Int. Conf. on Distributed Computing and Internet Technology (ICDCIT), 2007.

[11] J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, S. Roundy. *PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking*. Computer, Vol. 33, 42–48, IEEE Comp. Soc. Press, 2000.

[12] M.G.C. Resende, P.M. Pardalos. *Handbook of Optimization in Telecommunications*. Springer, 2006.

# On symmetric realizations of the simplicial complex of 3-crossing-free sets of diagonals of the octagon*

Jürgen Bokowski† Vincent Pilaud‡

**Abstract**

Motivated by the question of the polytopal realizability of the simplicial complex $\Gamma_{n,k}$ of $(k+1)$-crossing-free sets of diagonals of the convex $n$-gon, we study the first non-obvious case, namely the simplicial complex $\Gamma_{8,2}$ of 3-crossing-free sets of diagonals of the convex octagon. We give a complete description of the space of symmetric realizations of $\Gamma_{8,2}$, that is, of the polytopes $P$ whose boundary complex is isomorphic to $\Gamma_{8,2}$, and such that the natural action of the dihedral group on $\Gamma_{8,2}$ defines an action on $P$ by isometry.

## 1 Introduction

Let $k$ and $n$ denote two integers such that $k \geq 1$ and $n \geq 2k+1$. A set of $\ell$ mutually crossing diagonals of the convex $n$-gon is called an *$\ell$-crossing* (see Fig. 2a). We consider the abstract simplicial complex $\Gamma_{n,k}$ of $(k+1)$-crossing-free sets of diagonals of the convex $n$-gon.

When $k = 1$, the complex $\Gamma_{n,1}$ is the simplicial complex of non-crossing sets of diagonals, and has one maximal face for each *triangulation* of the convex $n$-gon. It is well-known that this simplicial complex is realized by the $(n-3)$-dimensional *associahedron* (or, more precisely, by an $n$-fold cone over the polar of the associahedron). Various realizations of the associahedron have been developed: see [14, 9, 1] and the references therein. In this article, for reasons of symmetry, we are particularly interested in the construction of the associahedron as the *secondary polytope* of the convex regular $n$-gon (see [1] for the details of the construction and Fig. 1 for an example).

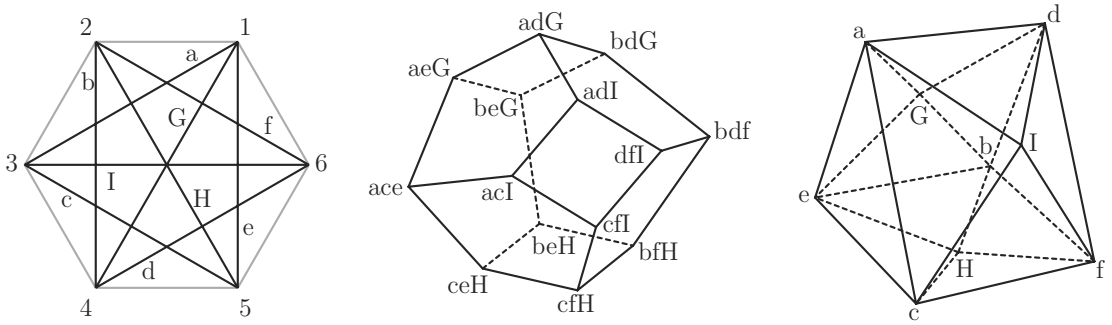



Figure 1: The 3-dimensional associahedron (vertices are labeled by triangulations of the hexagon) and its polar (vertices are labeled by internal diagonals of the hexagon).

The maximal faces of $\Gamma_{n,k}$ are called *$k$-triangulations* (see Fig. 2c). They have recently been studied under different aspects in the literature: as complexes of star polygons [16, 4], as diagonal-free fillings

*This research was initiated when the first author was invited by the Département d'Informatique of the École Normale Supérieure de Paris. Research of the second author was also partially funded by grant MTM2008-04699-C03-02 of the Spanish Ministry of Education and Science.

†Technische Universität, Darmstadt, Germany – juergen@bokowski.de

‡Département d'Informatique, École Normale Supérieure, Paris, France – vincent.pilaud@ens.fr

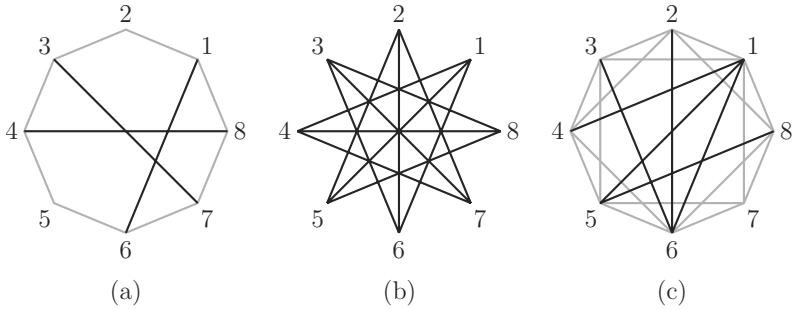



Figure 2: (a) A 3-crossing; (b) the 2-relevant diagonals of the octagon; (c) a 2-triangulation.

of certain polyominoes [12], as line arrangements in the hyperbolic plane [7], as particular cyclic split systems [6, 8]. It turns out that the complex $\Gamma_{n,k}$ is a topological sphere [11, 5] (or more precisely, a $kn$-fold cone over a sphere). But it remains an open question to know whether $\Gamma_{n,k}$ is *polytopal*, that is, whether it is the boundary complex of a polytope. So far, we only know that it holds for $k = 1$ (associahedron) and $n = 2k + 1, 2k + 2$ or $2k + 3$ (cyclic polytopes).

Let $\mathbb{D}_n$ denote the *dihedral group* of isometries of the convex regular $n$-gon. There is a natural action of $\mathbb{D}_n$ on the complex $\Gamma_{n,k}$ defined by $\rho E = \{\rho(e) \,|\, e \in E\}$. When $k = 1$, the associahedron obtained as the secondary polytope of the convex regular $n$-gon is */symmetric under* $\mathbb{D}_n$, meaning that $\mathbb{D}_n$ acts on its vertices by isometry. For general $n$ and $k$, we are also interested in realizing $\Gamma_{n,k}$ as a polytope symmetric under $\mathbb{D}_n$, not only because of the nice aspect of the result, but also because it is actually much easier to restrict to symmetric realizations.

In this paper, we only tackle the problem of the first unknown case: when $n = 8$ and $k = 2$. We give a complete description of the space of symmetric realizations of the simplicial complex $\Gamma_{8,2}$. To obtain this description, we first realize $\Gamma_{8,2}$ as a "symmetric oriented matroid polytope", and then as a "symmetric polytope under the dihedral group".

# 2 Preliminaries

## 2.1 General properties and particular examples

Let us remind some relevant properties of the simplicial complex $\Gamma_{n,k}$:

1. Observe first that a diagonal $uv$ can be involved in a $(k + 1)$-crossing only if there remains at least $k$ vertices on each side, *i.e.*, if $v - u > k$ and $n - v + u > k$. We call such a diagonal *$k$-relevant* and denote $R_{n,k}$ the set of all $k$-relevant diagonals of the $n$-gon (see Fig. 2b). Among the $\binom{n}{2}$ diagonals of the $n$-gon, $kn$ are $k$-irrelevant while the other $\binom{n}{2} - nk$ are $k$-relevant. Let $\Delta_{n,k}$ be the simplicial complex of $(k + 1)$-crossing-free subsets of $R_{n,k}$. The simpicial complex $\Gamma_{n,k}$ is just a $kn$-fold cone over the complex $\Delta_{n,k}$, and we will only study the latter.

2. A set of less than $k$ diagonals never contains a $(k + 1)$-crossing. Thus, the complex $\Delta_{n,k}$ is *$k$-neighborly*: it contains all the sets of at most $k$ diagonals.

3. The dihedral group $\mathbb{D}_n$ of isometries of the convex regular $n$-gon naturally acts on the simplicial complex $\Delta_{n,k}$ by $\rho E = \{\rho(e) \,|\, e \in E\}$. This action is obviously compatible with inclusion.

4. Any $k$-triangulation of the convex $n$-gon contains exactly $k(n - 2k - 1)$ $k$-relevant diagonals [3, 15, 7, 16]. Thus, the simplicial complex $\Delta_{n,k}$ is *pure* of dimension $k(n - 2k - 1) - 1$.

5. Any $k$-relevant diagonal of a $k$-triangulation of the convex $n$-gon can be flipped in a unique way to obtain a new $k$-triangulation of the convex $n$-gon [15, 16]. Consequently, every face of codimension 1 of $\Delta_{n,k}$ is contained in exactly two facets.

6. The simplicial complex $\Delta_{n,k}$ is a *vertex decomposable piece-wise linear sphere* [11, 5].

Being a topological sphere is a necessary, but not sufficient condition for being polytopal and it remains open to know if $\Delta_{n,k}$ can be realized as the boundary complex of a simplicial polytope of dimension $k(n-2k-1)$. As mentioned previously, when $k=1$, it is known that $\Delta_{n,1}$ is the boundary complex of the dual of the associahedron of dimension $n-3$. Appart from this case, we only know that $\Delta_{n,k}$ is polytopal in the following cases:

1. $n=2k+1$: the complete graph is the unique $k$-triangulation of the convex $(2k+1)$-gon (since $R_{2k+1,k}$ is empty). Thus, $\Delta_{2k+1,k}$ is realized by a single point.
2. $n=2k+2$: the $k+1$ long diagonals are the only $k$-relevant diagonals of the $(2k+2)$-gon (see Fig. 3a). Any proper subset of $R_{2k+2,k}$ is $(k+1)$-crossing-free and $\Delta_{2k+2,k}$ is the boundary complex of the $k$-simplex.
3. $n=2k+3$: the $k$-relevant diagonals of the $(2k+3)$-gon form a cycle $C$ of length $2k+3$ (more precisely, a star polygon with $2k+3$ edges; see Fig. 3b). The $k$-triangulations are precisely the sets of $2k$ $k$-relevant diagonals of the $(2k+3)$-gon that satisfy Gale's evenness criterion (see Theorem 0.7 in [17] and Fig. 3c & 3d). Thus, the simplicial complex $\Delta_{2k+3,k}$ is the boundary complex of the *cyclic polytope* of dimension $2k$ with $2k+3$ vertices.

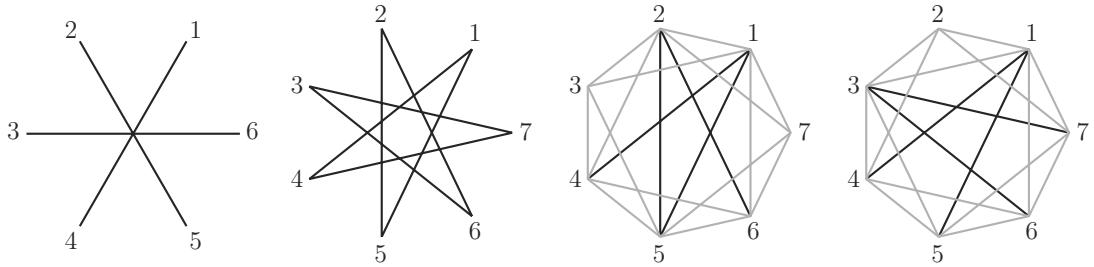



Figure 3: (a) The 2-relevant diagonals of the hexagon; (b) the 2-relevant diagonals of the heptagon; (c & d) the two prototypes of 2-triangulations of the heptagon (the other 2-triangulations of the heptagon are obtained by symmetry).

## 2.2 Symmetric realizations

**Symmetric polytope.** Let $\Delta$ be an abstract simplicial complex. We say that a polytope $P \subset \mathbb{R}^d$ is a *realization* of $\Delta$ if its boundary complex $\partial(P) = \{\text{proper faces of } P\}$ is isomorphic to $\Delta$, *i.e.*, if there is a bijection $\phi : \Delta \to \partial(P)$ that respects inclusion: $S \subset T \Leftrightarrow \phi(S) \subset \phi(T)$. Assume now that a group $G$ acts on $\Delta$ by $G \times \Delta \to \Delta : (g,x) \mapsto gx$. Then $G$ also acts on the boundary complex of $P$ by $G \times \partial(P) \to \partial(P) : (g,y) \mapsto gy = \phi(g\phi^{-1}(y))$. We say that $P$ is a *symmetric realization* (under $G$) if this action is isometric, *i.e.*, if for any $g \in G$, the application $\partial(P) \to \partial(P) : y \mapsto gy$ is an isometry of $P$.

For example, the dual polytope of the $(n-3)$-dimensional associahedron obtained as the secondary polytope of the convex regular $n$-gon [1] is a symmetric realization of $\Delta_{n,1}$ under the dihedral group $\mathbb{D}_n$. One can for instance visualize the action of $\mathbb{D}_6$ on both the associahedron and its dual in Fig. 1 (the reflection of the hexagon with respect to the horizontal axis corresponds to a reflection of the associahedron; the reflexion of the hexagon with respect to the vertical axis is a rotation of the associahedron; *etc.*).

As another family of examples, observe that there exists symmetric realizations of $\Delta_{2k+1,k}, \Delta_{2k+2,k}$, and $\Delta_{2k+3,k}$. Indeed, a single point (when $n=2k+1$) and a regular $k$-simplex (when $n=2k+2$) are clearly symmetric under any permutation of their vertices. When $n=2k+3$, the $2k$-dimensional cyclic polytope with $2k+3$ vertices, embedded on the Caratheodory curve, is symmetric under the dihedral group $\mathbb{D}_{2k+3}$ (see [13]).

**Symmetric oriented matroid.** Let $P \subset \mathbb{R}^d$ be a symmetric realization (under $G$) of $\Delta$. Let $V \subset \mathbb{R}^d$ denote the vertex set of $P$. For any vertex $v$, we denote $\vec{v} = (v, 1)$ the vector of homogeneous coordinates of $v$. For any $v_0, \ldots, v_d \in V$, we denote $\sigma(v_0, \ldots, v_d)$ the orientation of the simplex spanned by $v_0, \ldots, v_d$, *i.e.*, the sign (+1, −1, or 0) of the determinant of the matrix $(\vec{v_i})_{0 \le i \le d}$. The application $\sigma : V^{d+1} \to \{1, 0, -1\}$ is called the *oriented matroid* associated to $P$ and satisfies the following relations (see [2] for more details):

1. *alternating relations*: for any $v_0, \ldots, v_d \in V$ and any permutation $\pi$ of $\{0, \ldots, d\}$ of signature $\varepsilon$,
$$\sigma(v_{\pi(0)}, \ldots, v_{\pi(d)}) = \varepsilon\sigma(v_0, \ldots, v_d).$$
2. *Grassmann-Plucker relations*: for any $v_0, \ldots, v_{d-2}, w_1, w_2, w_3, w_4 \in V$, the set
$$\begin{Bmatrix} \sigma(v_0, \ldots, v_{d-1}, w_1, w_2).\sigma(v_0, \ldots, v_{d-1}, w_3, w_4), \\ -\sigma(v_0, \ldots, v_{d-1}, w_1, w_3).\sigma(v_0, \ldots, v_{d-1}, w_2, w_4), \\ \sigma(v_0, \ldots, v_{d-1}, w_1, w_4).\sigma(v_0, \ldots, v_{d-1}, w_2, w_3) \end{Bmatrix}$$
either contains $\{-1, 1\}$ or is contained in $\{0\}$.
3. *necessary simplex orientations*: if $v_0, \ldots, v_d \in V$ are such that both $\{v_0, \ldots, v_{d-2}, v_{d-1}\}$ and $\{v_0, \ldots, v_{d-2}, v_d\}$ are facets of $P$, then for any $w \in V \setminus \{v_0, \ldots, v_d\}$,
$$\sigma(v_0, \ldots, v_{d-2}, v_{d-1}, v_d) = \sigma(v_0, \ldots, v_{d-2}, v_{d-1}, w) = \sigma(v_0, \ldots, v_{d-2}, w, v_d).$$
4. *symmetry*: there exists a morphism $\tau : G \to \{\pm 1\}$ such that for any $v_0, \ldots, v_d \in V$, and any $g \in G$,
$$\sigma(gv_0, \ldots, gv_d) = \tau(g)\sigma(v_0, \ldots, v_d).$$

We want to observe in particular that if $v_0, \ldots, v_d \in V$, $g \in G$ and $\pi$ is the permutation of $\{0, \ldots, d\}$ defined by $g$ (*i.e.*, $gv_i = v_{\pi(i)}$ for all $0 \le i \le d$) with signature $\varepsilon$, then

$$\sigma(v_0, \ldots, v_d) = \tau(g)\sigma(gv_0, \ldots, gv_d) = \tau(g)\sigma(v_{\pi(0)}, \ldots, v_{\pi(d)}) = \varepsilon\tau(g)\sigma(v_0, \ldots, v_d).$$

If $\varepsilon = -\tau(g)$, this implies that $\sigma(v_0, \ldots, v_d) = 0$: we say that $\sigma(v_0, \ldots, v_d)$ is a *necessary zero*.

Any application $\sigma$ that associates to the vertices of $\Delta$ a sign in $\{1, 0, -1\}$ and satisfies the four properties above will be called a *symmetric oriented matroid* realizing $\Delta$.

# 3 Symmetric realizations of $\Delta_{8,2}$

In this section, we restrict our attention to the question of the realization of the simplicial complex $\Delta_{8,2}$. The polytope we want to construct would be a 6-dimensional polytope, with 12 vertices (the 2-relevant diagonals of the octagon), 66 edges (since $\Delta_{8,2}$ is 2-neighborly), 192 2-dimensional faces, 306 3-dimensional faces, 252 ridges, and 84 facets (the 2-triangulations of the octagon).

It is convenient to label the diagonals of $R_{8,2}$ with letters: we will denote a = 14, b = 25, c = 36, d = 47, e = 58, f = 16, g = 27, h = 38, I = 15, J = 26, K = 37, and L = 48 (observe that the four capital letters denote the four long diagonals; see Fig. 2b). For example, the 2-triangulation of Fig. 2c is the set of diagonals acefIJ.

In order to find a symmetric realization of $\Delta_{8,2}$, we first enumerate all possible symmetric oriented matroids realizing it (Subsection 3.1), and then use this information to study the symmetric polytopes realizing it (Subsection 3.2).

## 3.1 Symmetric oriented matroids

We are looking for an oriented matroid realizing $\Delta_{8,2}$, that is, for an application

$$\sigma : \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}, \mathrm{f}, \mathrm{g}, \mathrm{h}, \mathrm{I}, \mathrm{J}, \mathrm{K}, \mathrm{L}\}^7 \to \{1, 0, -1\}$$

which satisfies the four properties mentioned before. In other words, we have to choose $12^7$ compatible signs. But up to alternating relations and to symmetry, we only have 62 orbits. The enumeration of all possible solutions has been done by computer (using the functional language HASKELL [10, 2]), passing through the three following steps:

1. The necessary simplex orientations given by $\Delta_{8,2}$ already provide the signs of 25 orbits.
2. There are two orbits that are necessary zeros. Applying Grassmann-Plucker relations, these two zeros provide the signs for 11 new orbits.
3. Finally, in order to complete the 24 remaining signs, we have to start "guessing" signs. The method is to pick one orbit whose sign remains unknown, and to try the three possibilities $1, 0$, and $-1$. Applying Grassmann-Plucker relations, we may obtain:
   (a) either a contradiction, and we eliminate this choice of sign for this orbit;
   (b) or a complete oriented matroid realizing $\Delta_{8,2}$;
   (c) or a certain number of additional signs, but not all, and we have to iterate the guessing process until being in situation (a) or (b).

   Naturally, a good choice for the orbit we want to guess should provide as much information as possible, *i.e.*, should be involved in many Grassmann-Plucker relations for which only two signs remain unknown.

This computation provides the complete list of symmetric oriented matroids realizing our complex:

**Proposition 3.1.** *There are exactly* 15 *symmetric oriented matroids realizing* $\Delta_{8,2}$.

To present them, we only give the sign of one representant for each of the 62 orbits. First, all 15 solutions have the following 59 common signs:

| | | | | |
|---|---|---|---|---|
| $\sigma(\mathrm{abcdefg}) = 0$ | $\sigma(\mathrm{abcdefI}) = -1$ | $\sigma(\mathrm{abcdefJ}) = 1$ | $\sigma(\mathrm{abcdefK}) = -1$ | $\sigma(\mathrm{abcdegI}) = 1$ |
| $\sigma(\mathrm{abcdegJ}) = -1$ | $\sigma(\mathrm{abcdeIK}) = -1$ | $\sigma(\mathrm{abcdeIL}) = 1$ | $\sigma(\mathrm{abcdeJK}) = 1$ | $\sigma(\mathrm{abcdfgI}) = -1$ |
| $\sigma(\mathrm{abcdfgJ}) = 1$ | $\sigma(\mathrm{abcdfgL}) = 1$ | $\sigma(\mathrm{abcdfIJ}) = 1$ | $\sigma(\mathrm{abcdfIK}) = 1$ | $\sigma(\mathrm{abcdfIL}) = -1$ |
| $\sigma(\mathrm{abcdfJK}) = -1$ | $\sigma(\mathrm{abcdfJL}) = 1$ | $\sigma(\mathrm{abcdfKL}) = -1$ | $\sigma(\mathrm{abcdIJK}) = -1$ | $\sigma(\mathrm{abcdIJL}) = 1$ |
| $\sigma(\mathrm{abcdIKL}) = -1$ | $\sigma(\mathrm{abcefgI}) = 0$ | $\sigma(\mathrm{abcefgK}) = 0$ | $\sigma(\mathrm{abcefIJ}) = -1$ | $\sigma(\mathrm{abcefIK}) = 1$ |
| $\sigma(\mathrm{abcefIL}) = 1$ | $\sigma(\mathrm{abcefJK}) = -1$ | $\sigma(\mathrm{abcefJL}) = -1$ | $\sigma(\mathrm{abcefKL}) = 1$ | $\sigma(\mathrm{abcegIJ}) = 1$ |
| $\sigma(\mathrm{abcegIK}) = -1$ | $\sigma(\mathrm{abcegIL}) = -1$ | $\sigma(\mathrm{abcegKL}) = -1$ | $\sigma(\mathrm{abceIJL}) = -1$ | $\sigma(\mathrm{abceIKL}) = 1$ |
| $\sigma(\mathrm{abceJKL}) = -1$ | $\sigma(\mathrm{abcfIJK}) = -1$ | $\sigma(\mathrm{abcfIKL}) = -1$ | $\sigma(\mathrm{abcIJKL}) = 1$ | $\sigma(\mathrm{abdegIJ}) = 1$ |
| $\sigma(\mathrm{abdegIK}) = 1$ | $\sigma(\mathrm{abdegIL}) = 1$ | $\sigma(\mathrm{abdegJK}) = -1$ | $\sigma(\mathrm{abdeIJK}) = 1$ | $\sigma(\mathrm{abdeIJL}) = -1$ |
| $\sigma(\mathrm{abdfIJL}) = 1$ | $\sigma(\mathrm{abdfIKL}) = 1$ | $\sigma(\mathrm{abdfJKL}) = -1$ | $\sigma(\mathrm{abdgIJK}) = -1$ | $\sigma(\mathrm{abdgIJL}) = -1$ |
| $\sigma(\mathrm{abdgJKL}) = 1$ | $\sigma(\mathrm{abdIJKL}) = -1$ | $\sigma(\mathrm{abefIJK}) = -1$ | $\sigma(\mathrm{abefIJL}) = -1$ | $\sigma(\mathrm{abefJKL}) = 1$ |
| $\sigma(\mathrm{abeIJKL}) = 1$ | $\sigma(\mathrm{acegIJK}) = -1$ | $\sigma(\mathrm{aceIJKL}) = -1$ | $\sigma(\mathrm{acfIJKL}) = 1$ | |

The three remaining orbits are the orbits of abcdeIJ, abceIJK and abdfIJK. The following table summarizes the possible signs for these three orbits (only 15 of the 27 possibilities are admissible):

| | $\sigma_A$ | $\sigma_B$ | $\sigma_C$ | $\sigma_D$ | $\sigma_E$ | $\sigma_F$ | $\sigma_G$ | $\sigma_H$ | $\sigma_I$ | $\sigma_J$ | $\sigma_K$ | $\sigma_L$ | $\sigma_M$ | $\sigma_N$ | $\sigma_O$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma(\mathrm{abcdeIJ})$ | $-1$ | $-1$ | $-1$ | $0$ | $0$ | $0$ | $1$ | $1$ | $1$ | $1$ | $1$ | $1$ | $1$ | $1$ | $1$ |
| $\sigma(\mathrm{abceIJK})$ | $1$ | $1$ | $1$ | $1$ | $1$ | $1$ | $-1$ | $-1$ | $-1$ | $0$ | $0$ | $0$ | $1$ | $1$ | $1$ |
| $\sigma(\mathrm{abdfIJK})$ | $-1$ | $0$ | $1$ | $-1$ | $0$ | $1$ | $-1$ | $0$ | $1$ | $-1$ | $0$ | $1$ | $-1$ | $0$ | $1$ |

## 3.2 Symmetric polytopes

We are now ready to study the symmetric polytopal realizations of $\Delta_{8,2}$. Assume that a polytope $P \subset \mathbb{R}^6$ is a polytope realizing $\Delta_{8,2}$ and symmetric under the action of $\mathbb{D}_8$. Let $\mathrm{a}, \dots, \mathrm{L}$ denote its vertices (labeled by the corresponding 2-relevant diagonals of the octagon), let $\vec{\mathrm{a}} = (\mathrm{a}, 1), \dots, \vec{\mathrm{L}} = (\mathrm{L}, 1)$ be the corresponding vectors of homogeneous coordinates, and let $M$ denote the matrix whose columns are $\vec{\mathrm{a}}, \dots, \vec{\mathrm{L}}$. From Subsection 3.1, we know that the submatrix $N = (\vec{\mathrm{a}}, \vec{\mathrm{b}}, \vec{\mathrm{c}}, \vec{\mathrm{I}}, \vec{\mathrm{J}}, \vec{\mathrm{K}}, \vec{\mathrm{L}})$ is invertible, and we denote $\tilde{M} = N^{-1}M$. This matrix can be written:

$$\tilde{M} = \begin{bmatrix} 1 & 0 & 0 & d_0 & e_0 & f_0 & g_0 & h_0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_1 & e_1 & f_1 & g_1 & h_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_2 & e_2 & f_2 & g_2 & h_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d_3 & e_3 & f_3 & g_3 & h_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & d_4 & e_4 & f_4 & g_4 & h_4 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & d_5 & e_5 & f_5 & g_5 & h_5 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & d_6 & e_6 & f_6 & g_6 & h_6 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} I_3 & T & 0_{3\times 4} \\ 0_{4\times 3} & B & I_4 \end{bmatrix},$$

where $T \in \mathbb{R}^{3\times 5}$ and $B \in \mathbb{R}^{4\times 5}$ denote respectively the "top" and "bottom" unknown submatrices.

The determinants of the submatrices of size 7 of $\tilde{M}$ are symmetric under the action of the dihedral group. We use these symmetries to get informations on $T$ and $B$:

**Lemma 3.2.** *The matrix $T$ equals*

$$\begin{bmatrix} -1 & 1+\sqrt{2} & -2-\sqrt{2} & 2+\sqrt{2} & -1-\sqrt{2} \\ -1-\sqrt{2} & 2+2\sqrt{2} & -3-2\sqrt{2} & 2+2\sqrt{2} & -1-\sqrt{2} \\ -1-\sqrt{2} & 2+\sqrt{2} & -2-\sqrt{2} & 1+\sqrt{2} & -1 \end{bmatrix}$$

**Lemma 3.3.** *There exists $u \in (-1-2\sqrt{2}, -1-3\sqrt{2}/2)$ such that $B$ equals*

$$\begin{bmatrix} 1+\sqrt{2}/2 & u & -(1+\sqrt{2})(1+u) & (1+\sqrt{2})(1+u)+1 & -\sqrt{2}/2-u \\ -\sqrt{2}/2-u & (1+\sqrt{2})(1+u)+1 & -(1+\sqrt{2})(1+u) & u & 1+\sqrt{2}/2 \\ 1+\sqrt{2}/2 & -2-2\sqrt{2}-u & (1+\sqrt{2})(3+u)+2 & -(1+\sqrt{2})(3+u)-1 & 2+3\sqrt{2}/2+u \\ 2+3\sqrt{2}/2+u & -(1+\sqrt{2})(3+u)-1 & (1+\sqrt{2})(3+u)+2 & -2-2\sqrt{2}-u & 1+\sqrt{2}/2 \end{bmatrix}$$

To illustrate the use of the symmetry, we prove Lemma 3.2 (Lemma 3.3 can be proved similarily):

*Proof of Lemma 3.2.* For any $\alpha_0, \dots, \alpha_6 \in \{\mathrm{a}, \dots, \mathrm{L}\}$, we denote $[\alpha_0\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6]$ the determinant of the submatrix of $\tilde{M}$ whose columns correspond to the diagonals $\alpha_0, \dots, \alpha_6$.

From the symmetry, we have

$$[\mathrm{abcIJKL}] = -[\mathrm{bcdIJKL}] = -[\mathrm{abhIJKL}] \quad \Longrightarrow \quad 1 = -d_0 = -h_2;$$

$$[\mathrm{abdIJKL}] = -[\mathrm{abgIJKL}] = -[\mathrm{acdIJKL}] = -[\mathrm{achIJKL}] = -[\mathrm{bceIJKL}] = [\mathrm{bchIJKL}]$$
$$\Longrightarrow \quad d_2 = -g_2 = d_1 = h_1 = -e_0 = h_0; \quad (\alpha)$$
$$[\mathrm{abeIJKL}] = -[\mathrm{abfIJKL}] = -[\mathrm{bcfIJKL}] = [\mathrm{bcgIJKL}] \Longrightarrow \quad e_2 = -f_2 = -f_0 = g_0; \quad (\beta)$$
$$\text{and} \quad [\mathrm{aceIJKL}] = [\mathrm{acgIJKL}] \quad \Longrightarrow \quad -e_1 = -g_1. \quad (-\gamma)$$

Thus, we can already write the matrix $T$ as follows:

$$T = \begin{bmatrix} -1 & -\alpha & -\beta & \beta & \alpha \\ \alpha & \gamma & \delta & \gamma & \alpha \\ \alpha & \beta & -\beta & -\alpha & -1 \end{bmatrix}.$$

Furthermore,

$$[\mathrm{bdhIJKL}] = -[\mathrm{aceIJKL}] \quad \Longrightarrow \quad \alpha^2 - 1 = \gamma;$$

$$\text{and} \quad [\text{adhIJKL}] = -[\text{abeIJKL}] \quad \Longrightarrow \quad -\alpha^2 - \alpha = -\beta.$$

Thus, the matrix $T$ can be written:

$$T = \begin{pmatrix} -1 & -\alpha & -\alpha^2-\alpha & \alpha^2+\alpha & \alpha \\ \alpha & \alpha^2-1 & \delta & \alpha^2-1 & \alpha \\ \alpha & \alpha^2+\alpha & -\alpha^2-\alpha & -\alpha & -1 \end{pmatrix}.$$

Finally,

$$[\text{defIJKL}] = -[\text{abcIJKL}] \quad \Longrightarrow \quad -\alpha^4-2\alpha^3-2\alpha^2-\alpha+\delta\alpha = -1 \quad \Longrightarrow \quad \delta = \frac{1}{\alpha}(\alpha^4+2\alpha^3+2\alpha^2+\alpha-1)$$

$$\text{and} \quad [\text{dehIJKL}] = -[\text{abeIJKL}] \quad \Longrightarrow \quad \alpha^3+2\alpha^2-1 = -\alpha^2-\alpha \Longrightarrow \quad \alpha \in \{-1, -1+\sqrt{2}, -1-\sqrt{2}\}.$$

According to Subsection 3.1, for any symmetric matroid solution, we have $\sigma(\text{adgIJKL}) \neq 0$ and $\sigma(\text{abdIJKL}) < 0$. Thus, $\alpha \neq -1$ and $\alpha < 0$. Consequently, $\alpha = -1-\sqrt{2}$, which completes the proof of the lemma. □

These lemmas imply that only three of the 15 oriented matroids realizing $\Delta_{8,2}$ are realizable by a polytope with symmetric determinants: if $-1-2\sqrt{2} < u < -2-\sqrt{2}$ we obtain the oriented matroid $\sigma_G$, if $u = -2-\sqrt{2}$ we obtain $\sigma_K$, and if $-2-\sqrt{2} < u < -1-3\sqrt{2}/2$ we obtain $\sigma_O$.

In order to complete our understanding of the space of symmetric realizations of $\Delta_{8,2}$, it only remains to study the possible values of the matrix $N$. To determine $N$, we again use symmetry, but this time on the length of the edges of $P$.

For example, we know that the vertices I, J, K, and L span a 3-dimensional simplex with $|\text{IJ}| = |\text{JK}| = |\text{KL}| = |\text{IL}|$ and $|\text{IK}| = |\text{JL}|$ (where $|\text{IJ}|$ denotes the euclidean distance from I to J). Thus, since neither an orthogonal transformation, nor an homothecy destruct the symmetry, we can assume that $N$ is of the form:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_2 & y_2 & 0 & 0 & 0 & 0 & 0 \\ x_3 & y_3 & z_3 & 0 & 0 & 0 & 0 \\ x_4 & y_4 & z_4 & 1 & -1 & 1 & -1 \\ x_5 & y_5 & z_5 & v & 0 & -v & 0 \\ x_6 & y_6 & z_6 & 0 & v & 0 & -v \end{bmatrix},$$

with $x_1 > 0$, $y_2 > 0$, $z_3 > 0$, and $v > 0$.

Using the remaining equations given by the symmetries, we obtain the following constraints:

**Lemma 3.4.** *The coefficients of the matrix $N$ satisfy*

$$x_1 = \sqrt{\frac{(2+\sqrt{2})(2y_3+z_3\sqrt{2})(y_3+z_3)z_3}{y_3-z_3}}, \quad x_2 = \frac{y_3(2+\sqrt{2})(y_3+z_3)}{\sqrt{z_3^2-y_3^2}},$$

$$y_2 = \sqrt{z_3^2-y_3^2}, \quad x_3 = -(2+\sqrt{2})(y_3+z_3\frac{\sqrt{2}}{2}), \quad x_4 = y_4 = z_4 = 0,$$

$$x_5 = -x_6 = y_5 = y_6 = -z_5 = z_6 = -\frac{1}{2}\sqrt{2}v(1+\sqrt{2}+u),$$

*with $-z_3 < y_3 < -\frac{z_3}{\sqrt{2}}$.*

Reciprocally, it is easy to check that under these conditions, the convex hull of the column vectors of $NM$ is a symmetric realization of $\Delta_{8,2}$. Thus, we obtain our main result:

**Proposition 3.5.** *The space of symmetric realizations of $\Delta_{8,2}$ has dimension 4 (up to orthogonal tranformations and homothecies of $\mathbb{R}^6$).*

# 4 Conclusion

This paper was motivated by the question of the realizability of $\Delta_{n,k}$ and solve the first non-trivial case $\Delta_{8,2}$. The method we used to find one realization was to completely describe the space of symmetric realizations of $\Delta_{8,2}$. Even if this study can not be directly generalized to any $n$ and $k$, we consider that it provides new evidence and motivation to the general investigation.

# References


[1] L. J. Billera, P. Filliman, and B. Sturmfels. Constructions and complexity of secondary polytopes. *Adv. Math.*, 83(2):155–179, 1990.

[2] J. Bokowski. *Computational Oriented Matroids*. Cambridge University Press, 2006.

[3] V. Capoyleas and J. Pach. A Turán-type theorem on chords of a convex polygon. *J. Combin. Theory Ser. B*, 56(1):9–15, 1992.

[4] A. Dress, S. Grünewald, J. Jonsson, and V. Moulton. The simplicial complex $\Delta_{n,k}$ of $k$-compatible line arrangements in the hyperbolic plane. Part 1: The structure of $\Delta_{n,k}$. *Preprint*, 2008.

[5] A. Dress, S. Grünewald, J. Jonsson, and V. Moulton. The simplicial complex $\Delta_{n,k}$ of $k$-compatible line arrangements in the hyperbolic plane. Part 2. *In preparation*, 2009.

[6] A. W. M. Dress, M. Klucznik, J. H. Koolen, and V. Moulton. $2kn - \binom{2k+1}{2}$: A note on extremal combinatorics of cyclic split systems. *Sém. Lothar. Combin.*, 47:Article B47b, 17 pp. (electronic), 2001/02.

[7] A. W. M. Dress, J. H. Koolen, and V. Moulton. On line arrangements in the hyperbolic plane. *European J. Combin.*, 23(5):549–557, 2002.

[8] A. W. M. Dress, J. H. Koolen, and V. Moulton. $4n - 10$. *Ann. Comb.*, 8(4):463–471, 2004.

[9] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, resultants, and multidimensional determinants*. Mathematics: Theory and Applications. Birkhäuser, 1994.

[10] G. Hutton. *Programming in Haskell*. Cambridge University Press, 2007.

[11] J. Jonsson. Generalized triangulations of the $n$-gon. *Unpublished manuscript*, 2003. An abstract was included in: "Topological and Geometric Combinatorics, April 6th - April 12th, 2003", Mathematisches Forschungsinstitut Oberwolfach, Report No. 16/2003, `http://www.mfo.de/programme/schedule/2003/15/Report16_2003.pdf`.

[12] J. Jonsson. Generalized triangulations and diagonal-free subsets of stack polyominoes. *J. Combin. Theory Ser. A*, 112(1):117–142, 2005.

[13] V. Kaibel and A. Waßmer. Automorphism groups of cyclic polytopes. In F. H. Lutz, *Triangulated Manifolds*. to appear.

[14] C. W. Lee. The associahedron and triangulations of the $n$-gon. *Europ. J. Combin.*, 10(6):551–560, 1989.

[15] T. Nakamigawa. A generalization of diagonal flips in a convex polygon. *Theoret. Comput. Sci.*, 235(2):271–282, 2000.

[16] V. Pilaud and F. Santos. Multitriangulations as complexes of star-polygons. *Discrete Comput. Geom.*, 41(2):284–317, 2009.

[17] G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, 1995.

# Recoloring Directed Graphs

Stefan Felsner* Clemens Huemer† Maria Saumell‡

**Abstract**

Let $G$ be a directed graph and $k$ a positive integer. We define the $k$-color graph of $G$ ($D_k(G)$ for short) as the directed graph having all $k$-colorings of $G$ as node set, and where two $k$-colorings $\beta$ and $\varphi$ are joined by a directed edge $\beta \to \varphi$ if $\varphi$ is obtained from $\beta$ by choosing a vertex $v$ and recoloring $v$ so that its color is different from the colors of all its out-neighbors. We investigate reachability questions in $D_k(G)$. In particular we want to know whether a fixed legal $k'$-coloring $\psi$ of $G$ with $k' \leq k$ is reachable in $D_k(G)$ from every possible initial $k$-coloring $\beta$. Interesting instances of this problem arise when $G$ is planar and the orientation is an arbitrary $\alpha$-orientation for fixed $\alpha$. Our main result is that reachability can be guaranteed if the orientation has maximal out-degree $\leq k-1$ and an accessible pseudo-sink.

## 1 Introduction

Given a $k$-colored directed graph $G$, we study the problem of recoloring the vertices of $G$, one at a time and maintaining a certain recoloring rule, in order to obtain a legal $k$-coloring of $G$. This kind of question has been investigated recently for the case of undirected graphs [1, 2, 3], where only legal $k$-colorings are considered and one wants to recolor vertices to reach any legal $k$-coloring from a given legal $k$-coloring of the graph.

A main object of interest are maximal bipartite planar graphs, or quadrangulations. It is well known that they admit a 2-orientation, that is, the edges of a quadrangulation $G$ can be directed so that each vertex (but two special ones) has out-degree 2 (precise definitions are given below). Such a 2-orientation is not unique, and it is possible to modify a given 2-orientation by inverting the direction of the edges of a directed cycle. This local modification defines a graph on all 2-orientations of $G$, and some work has been done to identify the properties of this graph. For example, it is known that it has the structure of a distributive lattice [4] and that its diameter is $O(n^2)$ [9], where $n$ is the number of vertices of $G$.

Here we consider an arbitrary fixed 2-orientation of a quadrangulation $G$, and look at the graph $D_3(G)$, which is defined on all 3-colorings of $G$. The edges of this graph correspond to local modifications obtained by recoloring a vertex: we allow to recolor a vertex if its new color is different from the colors of the two vertices it points to. We study reachability questions in $D_3(G)$. In particular, given any 3-coloring of the vertices of $G$, can a legal 3-coloring be reached? And how many recoloring steps are needed? We answer the first question in the positive -in fact we can even reach a legal two-coloring of $G$- and show an upper bound of $O(n^2)$ on the diameter of $D_3(G)$.

These 2-orientations are a particular instance of $\alpha$-orientations. We generalize our result to a special class of $\alpha$-orientations. This includes maximal planar graphs, or triangulations, which are known to admit a 3-orientation. We obtain that, given any 4-coloring of a 3-oriented triangulation, there is a sequence of $O(n^2)$ recoloring steps that transforms the given 4-coloring into a legal 4-coloring.

In the following we formally state the problem and some definions.

*Institut für Mathematik, Technische Universität Berlin, felsner@math.tu-berlin.de.

†Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, Barcelona, Spain, clemens.huemer@upc.edu.

‡Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain, maria.saumell@upc.edu.

Let $G$ be a directed graph and $k$ a positive integer. A *$k$-coloring* of $G$ is an arbitrary assignment of colors from the set $\{1, 2, \ldots, k\}$ to the vertices of $G$, that is, a mapping $\gamma : V \to \{1, 2, \ldots, k\}$. A *legal $k$-coloring* is a $k$-coloring with the property that the end-vertices of every edge receive different colors.

Given $G$ and $k$, we define the *$k$-color graph of $G$*, denoted by $D_k(G)$, as follows: its node set consists of all $k$-colorings of $G$, and two $k$-colorings $\beta$ and $\varphi$ are joined by a directed edge $\beta \to \varphi$ if they differ in color on precisely one vertex $v$ of $G$ and $\varphi(v) \neq \varphi(w)$ for all out-neighbors $w$ of $v$, in other words, if $\varphi$ is obtained from $\beta$ by choosing a vertex $v$ and recoloring $v$ so that its color is different from the colors of all its out-neighbors. A recoloring of a vertex in this way will be called a *valid recoloring step*.

If $\beta$ and $\varphi$ are two $k$-colorings of $G$, we may ask whether there is a path from $\beta$ to $\varphi$ in $D_k(G)$. In this case, we say that $\varphi$ is *reachable* from $\beta$.

An *$\alpha$-orientation* of a graph $G = (V, A)$ is an orientation of its edges such that the out-degree of every vertex $v \in V$ is given by a function $\alpha : V \to \mathbb{N}$ (see, for example, [7, 4, 8]). We will say that a vertex $v$ is *accessible* if $G$ contains a directed path from every other vertex to $v$. We deal with $\alpha$-orientations with an special vertex accessible from every other vertex and in which the out-degree of each vertex is bounded from above. More precisely, in Section 2 we look at the particular case of 2-orientations of planar quadrangulations (of which we describe the 3-color graph). In Section 3 we generalize our result and we highlight its consequences on 3-orientations of planar triangulations.

# 2 2-orientation and 3-colorings of planar quadrangulations

## 2.1 Definitions and results

A *plane quadrangulation* is a plane graph $Q = (V \cup \{s, t\}, E)$ such that $Q$ is a maximal bipartite graph with $|V| = n$, and $s, t$ are two non-consecutive vertices on the outer face. A *2-orientation* of $Q$ is an orientation of its edges such that every vertex different from $s$ and $t$ has out-degree two. Given a 2-orientation of $Q$, it is easy to see that $s$ and $t$ are sinks, i.e., vertices with out-degree zero. It has been proved that every quadrangulation admits a 2-orientation [7, 9].

Since a quadrangulation is a bipartite graph, its vertices can be divided into two groups, the white ones and the black ones, in such a way that every edge connects a white and a black vertex. Let us assume that $s$ and $t$ are black vertices. The following definition is useful for our purposes:

**Definition 2.1.** A *separating decomposition* is an orientation and coloring of the edges of $Q$ with colors red and blue satisfying that:

(1) The edges incident to $s$ are colored red and the edges incident to $t$ are colored blue.

(2) Every vertex $v \neq s, t$ is incident to an interval of red edges and an interval of blue edges. Among these edges, only one red edge and one blue edge are outgoing. If $v$ is white, the two outgoing edges are the first ones in clockwise order in the interval of edges of their color. If $v$ is black, the outgoing edges are the clockwise last in their color.

It can be shown that separating decompositions fulfill the additional property that the red edges form a directed tree rooted in $s$ ($T_r$ for short) and the blue edges form a tree rooted in $t$ ($T_b$ for short) [5]. See Figure 1 for an example. This ensures that there exist paths in $Q$ from every vertex $v \neq s, t$ to $s$ and $t$.

For each non-special vertex $v \neq s, t$ we denote the directed paths in $T_r$ and $T_b$ from $v$ to the corresponding sink by $P_r(v)$ and $P_b(v)$, respectively. Then one can prove that the paths $P_r(v)$ and $P_b(v)$ only share vertex $v$, so they split the quadrangulation into two regions [6]. We define $R_r(v)$ (respectively, $R_b(v)$) to be the region to the right of $P_r(v)$ (respectively, $P_b(v)$) and including both paths. A nice observation here is that, for each vertex of the path, all incoming edges belonging to a fixed region have the same color.

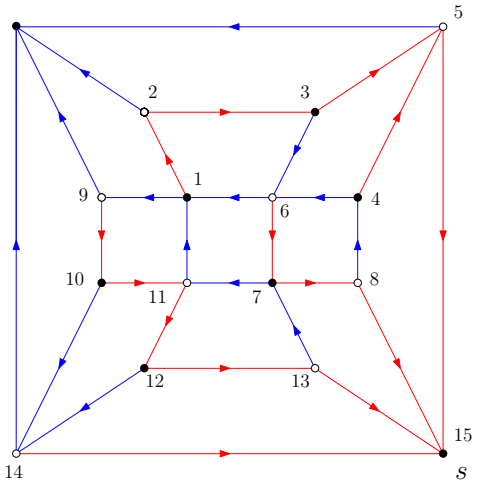



Figure 1: A quadrangulation with a separating decomposition.

Separating decompositions and 2-orientations are closely related. Indeed let $Q = (V \cup \{s,t\}, E)$ be a plane quadrangulation; then it has been proved that there is a bijection between separating decompositions and 2-orientations of $Q$ [5, 7]. To be precise, a separating decomposition yields a 2-orientation, while the edges of a 2-orientation can be colored so as to obtain a separating decomposition.

**Theorem 2.2.** *Let $G$ be a 2-orientation of a planar quadrangulation. Then, given any 3-coloring $\beta$ of $G$, it is always possible to reach a legal 2-coloring in $D_3(G)$.*

## 2.2 Proof of Theorem 2.2

Our proof is constructive, that is, we provide an algorithm to obtain the path from $\beta$ to a legal 2-coloring in $D_3(G)$, and we show its correctness.

A key ingredient is the following Lemma 2.3. The idea behind it is as follows. Suppose that $v$ is a non-special vertex such that $v$ and all its out-neighbors have different colors in $\beta$ ($v$ is a *rainbow* in $\beta$). Although the two outgoing edges from $v$ are legal, we might need to recolor $v$ to reach a legal coloring in $D_3(G)$, which is not possible in one single valid recoloring step. Lemma 2.3 shows that $v$ can nevertheless be recolored.

Suppose that $\varsigma, \tau$ are two different colors in the set $\{1,2,3\}$. The expression $\overline{\{\varsigma,\tau\}}$ will denote the color in $\{1,2,3\}$ different from $\varsigma$ and $\tau$.

**Lemma 2.3.** *Let $P = \{v_l, v_{l-1}, \ldots, v_2, s\}$ be a directed path in $G$ without any directed chord. Let $f(v_l)$ denote the color of the vertex pointed by $v_l$ different from $v_{l-1}$. If $v_l$ is colored $\varsigma$ and $f(v_l) = \tau$, there exists a sequence of valid recoloring steps involving only $v_l, v_{l-1}, \ldots, v_2, s$ such that, at the end, $v_l$ has color $\overline{\{\varsigma,\tau\}}$.*

*Proof.* We prove the statement by induction on the length $l$ of $P$. First observe that, in any situation, we can assume that $s$ has the most convenient color, as all edges incident to this vertex are incoming.

Let us look at the case $l = 2$, i. e., $P = \{v_2, s\}$. Suppose $s$ is colored $\varsigma$. Then, in a valid recoloring step, $v_2$ can be assigned color $\overline{\{\varsigma,\tau\}}$. Now assume that the lemma is valid for all paths of length at most $l-1$. If $v_{l-1}$ has color different from $\overline{\{\varsigma,\tau\}}$, then $v_l$ can be recolored with this color. Otherwise, by hypothesis of induction, $v_{l-1}$ can be assigned a color different from $\overline{\{\varsigma,\tau\}}$ by applying a sequence of valid recoloring steps that only involves vertices $v_{l-1}, v_{l-2}, \ldots, v_2, s$. Since $v_l$ does not point to any of the vertices $v_{l-2}, v_{l-3}, \ldots, v_2, s$, these recoloring steps do not modify the value of $f(v_l)$. Therefore, in the new coloring $v_l$ points to a vertex with color $\tau$ and a vertex with color different from $\overline{\{\varsigma,\tau\}}$, so it can be assigned color $\overline{\{\varsigma,\tau\}}$. □

Observe that the previous proof already provides an algorithm to recolor vertex $v_l$ in $O(l)$ valid recoloring steps. It consists of visiting the vertices of the path (starting from $v_l$) until a non-rainbow vertex is found, changing the color of this vertex and then recoloring backwards. Figure 2 shows an example of this process.

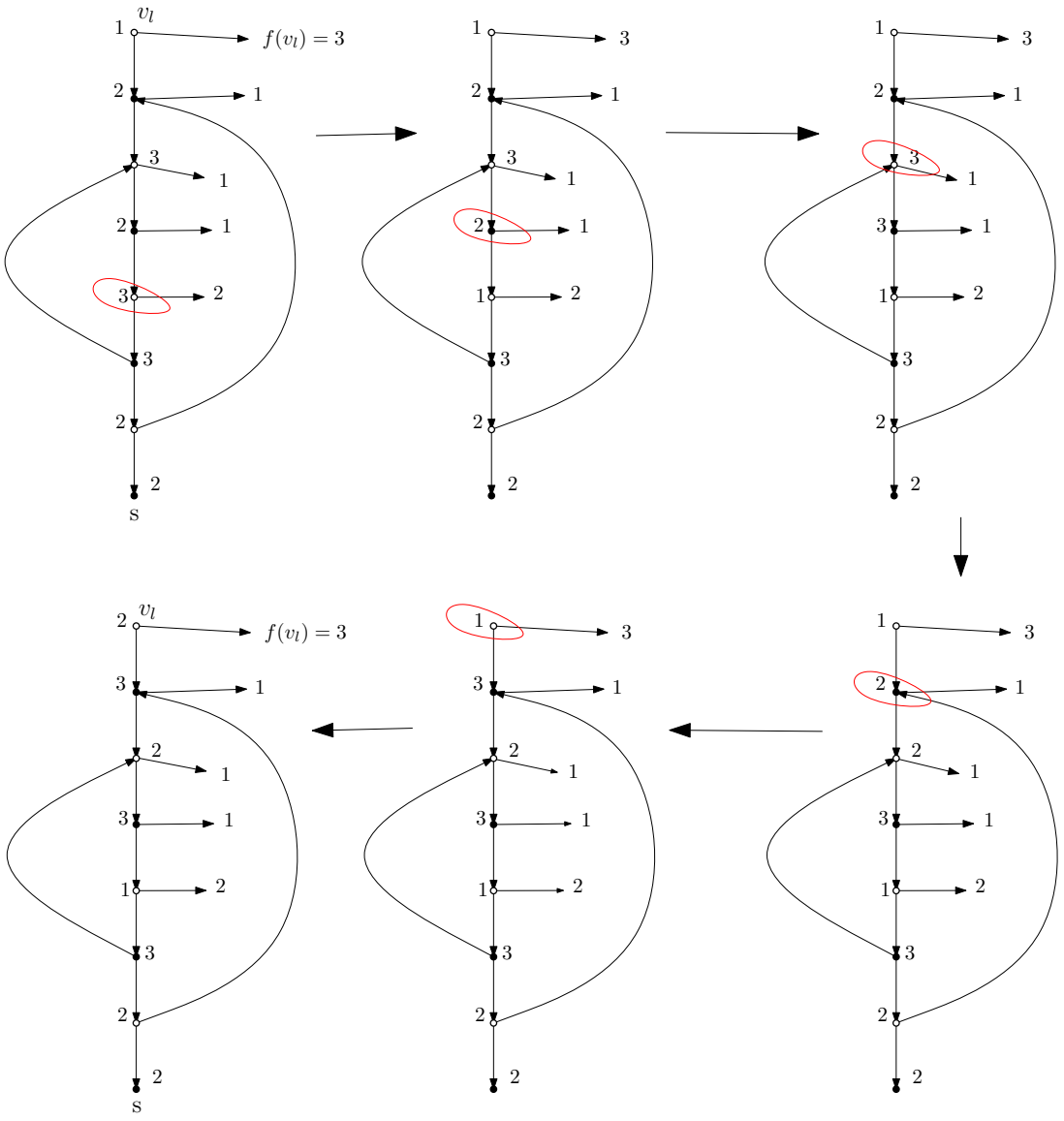



Figure 2: An example for Lemma 2.3. It is shown how $v_l$ can be recolored.

Let $\psi$ be the (legal) coloring of $G$ where the black vertices have color 1 and the white vertices have color 2. Algorithm 1 gives a path in $D_3(G)$ from the coloring $\beta$ to the coloring $\psi$.

Step 2 of Algorithm 1 involves postorder traversal in the red tree. In a binary tree, postorder traversal performs the following operations recursively at each node: (1) traverse the left subtree; (2) traverse the right subtree; (3) visit the root. This process can be generalized in the natural way to plane $m$-ary trees, as the notions of left and right are well-defined. The vertices in Figure 1 are labelled according to postorder in the red tree.

Let us introduce some notation. If $v \neq s, t$ is a vertex in $G$, then the red (respectively, blue) edge outgoing from $v$ points to a vertex that will be denoted by $v_r$ (respectively, $v_b$). If the path $P_r(v)$ is of the form $P_r(v) = \{v, v_{l-1}, \ldots, v_2, s\}$, we define $P'_r(v)$ as one of the paths from $v$ to $s$ on a subset of $\{v, v_{l-1}, \ldots, v_2, s\}$ which cannot be shortened by directed chords.

**Lemma 2.4.** *Given $\beta$, Algorithm 1 gives a sequence of $O(n^2)$ valid recoloring steps to reach $\psi$.*

**Algorithm 1** Reaching a legal 2-coloring

1: If necessary, recolor vertex $t$ with 1.
2: Sort all vertices different from $s$ and $t$ according to the postorder traversal sequence in the red tree. Let $L$ be the ordered list containing the vertices.
3: Let $v$ be the first element in $L$. Remove $v$ from $L$.
If $v$ is white and $v_r$ has color 2, apply the algorithm in Lemma 2.3 to recolor $v_r$ using the path $P'_r(v_r)$. Afterwards, assign color 2 to $v$.
If $v$ is black, apply (if necessary) the algorithm in Lemma 2.3 to make vertices $v_r$ and $v_b$ have color different from 1 by using the paths $P'_r(v_r)$ and $P'_r(v_b)$, respectively. Afterwards, assign color 1 to $v$.
Repeat until $L$ is empty.
4: If necessary, recolor vertex $s$ with 1.

*Proof.* Let us look at Step 3 of the algorithm. Let $v$ be the first element in $L$ at some point of the algorithm. If $v$ is a white vertex, the red edge outgoing from $v_b$ belongs to the region $R_b(v)$ (see Figure 3). This implies that $v_b$ has already been processed and, as will be shown in a few lines, has its final color, which, in this case, is 1. Our aim is to assign to $v$ its final color, namely, 2. If $v_r$ is not colored 2, we can do it in one single valid recoloring step. Otherwise we first change the color of $v_r$ by means of the algorithm in Lemma 2.3; we use the path $P_r(v_r) = \{v_r, v_{l-1}, \ldots, v_2, s\}$, possibly shortened by directed chords. Since this algorithm only recolors vertices in the path and all of them are after $v$ in $L$, we do not change the color of any vertex that has already been processed.

If $v$ is black, the red edge outgoing from $v_b$ belongs to $R_r(v)$ (see Figure 3). Since all red edges pointing to $v$ are in $R_b(v)$, $v_b$ is after $v$ if $T_r$ is traversed in postorder. So, in this case, neither $v_r$ nor $v_b$ have already been processed and we might change their color to be able to assign color 1 to $v$. Again, if we use the paths $P'_r(v_r)$ and $P'_r(v_b)$, we do not recolor any vertex that has already been treated.

To conclude, after an iteration of Step 3 the number of vertices that are assigned its final color and will not be recolored in future steps increases by one. Consequently, the algorithm is correct.

As for the number of valid recoloring steps performed by the algorithm, the assignment of the final color of each vertex $v$ costs $O(l)$ recolorings, where $l$ is the length of the path $P_r(v)$. Therefore, the total number of recoloring steps is $O(n^2)$. □

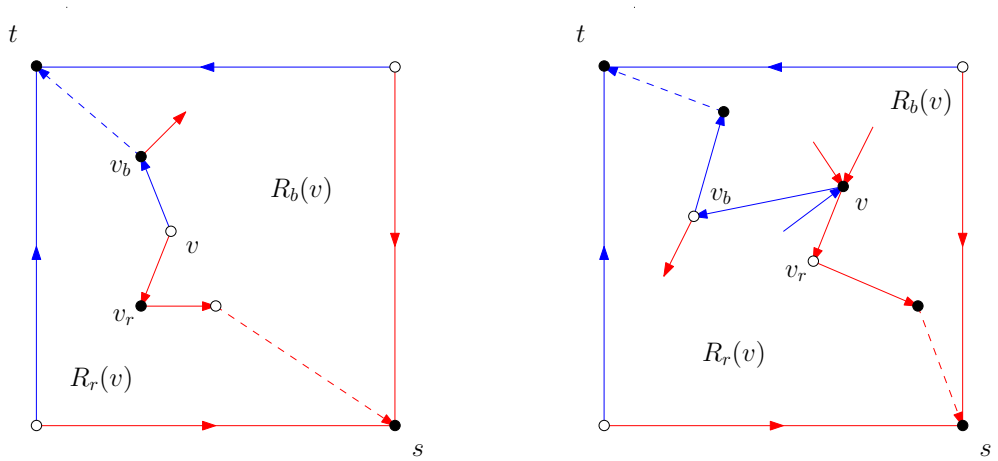



Figure 3: Postorder in the red tree of the vertices $v, v_r, v_b$.

## 3 A general result and an interesting application

In this section we generalize Theorem 2.2 to a special class of $\alpha$-orientations that includes 2-orientations of planar quadrangulations and 3-orientations of planar triangulations.

Let $G$ be a directed graph in which the maximum out-degree of the vertices is $k-1$, and let $\beta$ be a $k$-coloring of $G$. As in the case $k=3$, we will say that a vertex $v$ is a *rainbow in* $\beta$ if $v$ has out-degree $k-1$, and $v$ and all its out-neighbors have different colors in $\beta$. We will say that $v$ is a *pseudo-sink* if it is guaranteed to be a non-rainbow in every $k$-coloring of $G$. Finally, recall that $v$ is *accessible* if $G$ contains a directed path from every other vertex to $v$.

**Theorem 3.1.** *Let $G$ be a directed graph where the maximum out-degree of the vertices is $k-1$. Assume that $G$ has an accessible pseudo-sink $s$. Then, given any $k$-coloring $\beta$ of $G$ and any legal $k'$-coloring $\psi$ of $G$ with $k' \leq k$, $\psi$ is reachable from $\beta$ in $D_k(G)$.*

The method we propose to obtain a path from $\beta$ to $\psi$ in $D_k(G)$ is detailed in Algorithm 2. Here we also use an auxiliary lemma ensuring that each vertex in $G$ can be recolored modifying only the color of some particular vertices. We omit its proof because its very similar to the one in Lemma 2.3:

**Lemma 3.2.** *Let $P = \{v_l, v_{l-1}, \ldots, v_2, s\}$ be a directed path in $G$ without any directed chord. If $v_l$ is colored $\varsigma$, there exists a sequence of $O(l)$ valid recoloring steps involving only $v_l, v_{l-1}, \ldots, v_2, s$ such that, at the end, $v_l$ has a color different from $\varsigma$.*

In Algorithm 2 we talk about distances in the graph $G$. The *distance* between a vertex $v$ and a vertex $w$ in $G$ is the length (i.e., the number of edges) of the shortest directed path from $v$ to $w$ in $G$ ($d_G(v,w) = +\infty$ if there are no directed paths from $v$ to $w$).

**Algorithm 2** Reaching a legal $k'$-coloring

1: Sort all vertices by decreasing distance to $s$ (choose any order if there are ties). Let $L$ be the ordered list containing the vertices. For each $v \in L$, let $\delta_{\leq}(v)$ be the set of out-neighbors of $v$ that are after $v$ in $L$.
2: Let $v$ be the first element in $L$. Remove $v$ from $L$.
Apply (if necessary) the algorithm in Lemma 3.2 to make the vertices in $\delta_{\leq}(v)$ have color different from $\psi(v)$ by using their shortest paths in $G$ to $s$. Afterwards, assign color $\psi(v)$ to $v$.
Repeat until $L$ is empty.

**Lemma 3.3.** *Given $\beta$ and $\psi$, Algorithm 2 gives a sequence of $O(n^2)$ valid recoloring steps to reach $\psi$ from $\beta$.*

*Proof.* The vertices are processed in such a way that the recolorings necessary to assign the definite color to a vertex do not involve the vertices that have already been treated. Thus, after each iteration of Step 2, the number of vertices that are assigned its final color and will not be recolored increases.

The number of valid recoloring steps performed by the algorithm can be counted as in the proof of Lemma 2.4. □

An remarkable example of a family of directed graphs that fulfill the hypothesis of Theorem 3.1 are 3-orientations of planar triangulations.

A *plane triangulation* $T$ is a maximal plane graph with $n$ vertices and three special vertices $a_1, a_2, a_3$ in the outer face. A *3-orientation* of $T$ is an orientation of its edges such that every vertex different from $a_1, a_2, a_3$ has out-degree three. It can be proved that every triangulation admits a 3-orientation [7].

Furthermore, it is known that the interior edges of a triangulation can be colored with 3 colors with the property that the edges of each color form a directed tree that spans all the inner vertices and is rooted at one of the vertices in the outer face [10]. These are so-called *Schnyder woods*. An example of a triangulation with a Schnyder wood is shown in Figure 4.

An important analogy with separating decompositions and 2-orientions is the fact that, if $T$ is a plane triangulation with outer vertices $a_1, a_2, a_3$, then Schnyder woods and 3-orientations of $T$ are in bijection (see [5, 7]). Furthermore, the edges of a 3-orientation of $T$ can be colored to obtain a Schnyder wood.

**Corollary 3.4.** *Let $G$ be a 3-orientation of a planar triangulation. Then, given any 4-coloring $\beta$ of $G$, it is always possible to reach any legal 4-coloring in $D_4(G)$.*

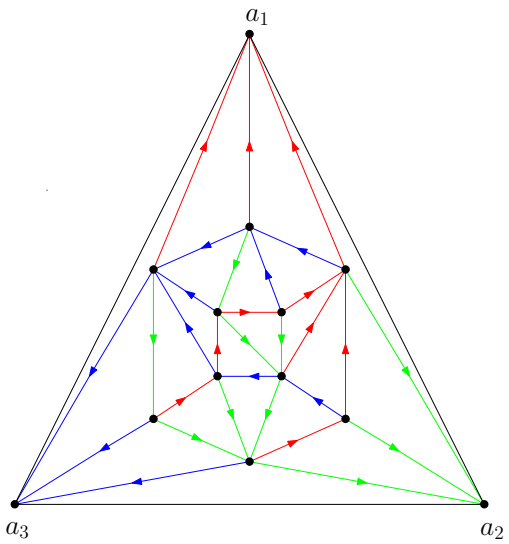



Figure 4: A triangulation with a Schnyder wood.

## Acknowledgements

This research has been carried out in the context of the I-Math Winter School DocCourse Combinatorics and Geometry 2009: Discrete and Computational Geometry, organized by Centre de Recerca Matemàtica. We thank the organizers of the course and the other participants for providing a stimulating working environment.

# References

[1] Paul Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. In Mathematical Foundations of Computer Science 2007 (MFCS 07). Lecture Notes in Computer Science, 4708:738–749, 2007.

[2] Luis Cereceda, Jan van den Heuvel and Matthew Johnson. Mixing 3-colourings in bipartite graphs. In Graph-Theoretic Concepts in Computer Science, Proceedings of the 33rd International Workshop on Graph Drawing (WG 2007). Lecture Notes in Computer Science, 4769:166-177, 2007.

[3] Luis Cereceda, Jan van den Heuvel and Matthew Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(5-6):913-919, 2008.

[4] Stefan Felsner. Lattice structures from planar graphs. *Electronic Journal of Combinatorics*, 11(1), Research paper R15, 24 pp., 2004.

[5] Stefan Felsner, Éric Fusy, Marc Noy and David Orden. Bijections for Baxter families and related objects. *Submitted*, arXiv:0803.1546, 2008.

[6] Stefan Felsner, Clemens Huemer, Sarah Kappes and David Orden. Binary labelings for plane quadrangulations and their relatives. *Submitted*, arXiv:math.CO/0612021, 2008.

[7] Hubert de Fraysseix and Patrice Ossona de Mendez. On topological aspects of orientations. *Discrete Mathematics*, 229:57–72, 2001.

[8] Patrice Ossona de Mendez. Orientations bipolaires. PhD thesis, École Des Hautes Études en Sciences Sociales, Paris, 1994.

[9] Atsuhiro Nakamoto and Mamoru Watanabe. Cycle reversals in oriented plane quadrangulations and orthogonal plane partitions. *Journal of Geometry*, 68:200-208, 2000.

[10] Walter Schnyder. Planar graphs and poset dimension. *Order*, 5(4):323-343, 1989.

# Sesión 2, 17:00–18:40

# Terrain approximation from grid data points

Narcís Coll * Marité Guerrieri * Maria-Cecilia Rivara †
J. Antoni Sellarès *

**Abstract**

We propose the LEPP-surface triangulation method that starts with a coarse initial triangulation of input grid terrain data, and incrementally adds data points that reduce the worst edge approximation error in the mesh. The method generalizes a previous LEPP-centroid method in two dimensions as follows: for the edge $e$, having highest error in the mesh, one or two points close to (one or two) terminal edges associated to $e$, are inserted in the mesh. The edge error is computed by adding the triangle approximation errors of the two triangles that share $e$, while each triangle error in $L_2$-norm is computed by using a curvature tensor (good approximation of the surface) at a representative point associated to both triangles. The method produces triangular approximations that capture well the relevant features of the terrain surface by naturally producing well-shaped triangles.

## 1 Introduction

Techniques for the efficient and timely management of huge terrain grid data produced by satellites have been intensively developed in the last ten years. Problems related with storage requirements, real-time rendering, data transmission, and visualization arise when dealing with these huge data sets. Data representation methods that minimize the data size by modeling well the important features of the terrain are highly desirable.

For visualization purposes, a fast and efficient large scale rendering with minimal memory and hardware requirements is always desirable, since the grid data set is usually too large to be displayed as a whole. Two main approaches have been mainly used to efficiently deal with the rendering task: the use of different levels of detail (LODs) depending on the view focus, and the reduction of the data size without loosing visual representation. View dependent LOD methods display the surfaces with adaptive resolution depending on the distance to the view focus [15, 6]. Data reduction methods basically consider two alternative strategies for the treatment of grid data, top-down approach [22, 3, 2, 12, 13, 9] and bottom-up approach [10, 8, 5]. The top-down algorithms start with a few triangles and incrementally add vertices until obtaining a triangulation that accomplishes requirements previously established. This kind of process is known as refinement process. The second strategy starts with a mesh of the whole data and deletes vertices or edges and sometimes moves vertices to a new better location until it achieves the desired requirements. These methods have been called decimation or simplification. The vertices added or moved in both strategies can belong to the initial data set or can be extrapolated from them. Different criteria to finish the process have been used: number of elements, faces or vertices, error estimation between the initial and final model, element quality, etc.

For numerical simulations, additional constraints need also to be taken into account. The triangles of the fitting triangulated surface must be reasonably good-shaped (with bounded smallest angle) which guarantees the convergence of the numerical method [23]. A triangulation of a terrain surface can be built either by using only the $xy$-coordinates of the input points, or the 3D information of the input points. In the latter case it is called a data-dependent triangulation [4]. When the fitting

*Institut d'Informàtica i Aplicacions - Universitat de Girona - Spain, {coll,mariteg,sellares}@ima.udg.edu. Authors partially supported by the Spanish Ministerio de Educación y Ciencia under grant TIN2007-67982-C02-02.

†Departament of Computer Science - Universidad de Chile - Chile, mcrivara@dcc.uchile.cl

surface is created from a triangulation in the $xy$ plane, usually the Delaunay triangulation is used since this maximizes the minimum angle. However, having well-shaped triangles in the $xy$ plane does not guarantee well-shaped triangles on the the fitting surface. Moreover, it is also known that when a data-dependent triangulation is used to fit a triangulated surface by minimizing the $L_2$ norm of the difference between the original and the approximated surfaces, long and skinny triangles can be obtained throughout the fitting process [16].

In this paper we center on the problem of fitting a data-dependent triangulation with a small number of triangles to a set of grid data points so that the triangles are well-shaped and the $L_2$ error remains low. Note that the decimation strategy centers on the elimination of biggest possible number of input data. On the contrary, our goal is to obtain an accurate approximation by progressively adding the smallest possible number of input data points.

## 2 Our approach

A common method of modeling a terrain is to use a regular grid, where a set of sampled points representing measures of elevation are stored at regularly spaced $x$ and $y$ coordinates. Then, the input of our problem is a finite set $V$ of $n$ grid data points sampled from an unknown terrain surface $F$. Given a fixed size data $\overline{n} \ll n$, we want to construct an interpolating triangulated surface (piecewise linear surface defined by a set of triangles) $\overline{F}$ whose vertices belong to a subset $\overline{V} \subset V$ of size $\overline{n}$. The process selects relevant points of $V$ in such a way that: 1) the points in $\overline{V}$ ensure that $\overline{F}$ fits well the original surface $F$; 2) the triangles of $\overline{F}$, when possible, are well-shaped. The proposed method estimates the $L_2$ error of the triangles by using the curvature tensor that locally approximates the surface at each point of the model. The algorithm gives more importance to the $L_2$ metric but when it is possible the aspect-ratio of the elements is taken also into account, achieving in this way an approximation of the terrain with good quality elements. This is the main difference of our algorithm with previous works in this area [10, 9, 5, 2].

In the following sections we present the concepts required by our refinement method: the error associated to each edge based on the $L_2$ norm, the point insertion mechanism used to improve the fitting of the approximated surface to the original surface or to obtain triangles with good aspect-ratio, and the two LEPPs strategy in order to determine the data points, surface-midpoints or surface-centroids, to be selected and inserted into the triangulation $\overline{F}$.

## 3 Edge error

Let $p$ be a point of the surface $F$. Using the frame coordinates with origin at $p$ and axes determined by two orthogonal tangent directions and the surface normal at $p$, locally, the surface $F$ can be approximated by a surface patch of the form $(u, v, f(u, v))$ being $f(u, v)$ a quadratic function. In the canonical $\mathbb{R}^3$ frame coordinates the quadratic form defined by the hessian of $f$ coincides with the curvature tensor.

Given an input vertex $v$, let $T_v$ be the curvature tensor at $v$. Then, each vertex $v$ defines a local interpolation $L_2$ error for a triangle $t$ defined by three points $v_1, v_2, v_3$ on $F$. Nadler in [14] showed that this error can be computed by:

$$E_2(t, v) = \frac{|t|}{180}\left((d_1 + d_2 + d_3)^2 + d_1^2 + d_2^2 + d_3^2\right) ,$$

where $|t|$ is the area of the projection of $t$ onto the tangent plane at $v$ and

$$d_i = \frac{1}{2}(v_{i+1} - v_i)^T T_v (v_{i+1} - v_i) .$$

For a given edge $e$ adjacent to triangles $t_e$ and $t'_e$ we consider the grid data point $c_e$ whose projection onto the $xy$ plane is the closest data point to the centroid of the quadrilateral determined by union of

the projections onto the $xy$ plane of $t_e$ and $t'_e$ among the projections of input points in $V$. Then, we take $E_2(e) = E_2(t_e, c_e) + E_2(t'_e, c_e)$ as the interpolation error of the edge $e$.

We approximate the curvature tensor $T_{c_e}$ needed to compute $E_2(e)$ by a quadratic patch fitted with the least-squares technique. The size of the neighborhood of $c_e$ is taken proportionally to the length of the shortest edge of triangles $t_e$ and $t'_e$.

# 4 Point insertion

When a new point $p$ is selected for insertion in the triangulation, a simple valid mesh is obtained by joining $p$ with vertices of the triangle whose $xy$-projection contains the $xy$-projection of $p$. Then, a sequence of edge flips are used either for error improvement or for triangle quality improvement as follows:

(i) if the error of one of the two configurations is far bigger than the error associated to the other configuration, the configuration of less error is selected.

(ii) if both configurations have approximately the same error, the configuration of better angles on the terrain surface is selected.

Notice that triangles with small angles can be obtained even in regions with low curvature by only applying this strategy, and vertices need to be well fitted by the surface patch defined by the local curvature tensor in order to properly apply the error criterion. Consequently, adjustment and planarity measures are needed for an accurate point insertion. Following sections are dedicated to define these measures and to specify how the edge flipping operator works.

## 4.1 Adjustment and planarity measures

For each triangle $t$ of area $A(t)$ adjacent to an edge $e$ let

- $\overline{A}(t)$ be the area of the triangle determined by the projection of vertices of $t$ onto the surface patch defined by the local curvature tensor at $c_e$,
- $\overline{\overline{A}}(t)$ be the area of the projection of $t$ onto the tangent plane at $c_e$.

Note that $A(t) \cong \overline{A}(t)$ when $t$ is well fitted by the local surface patch, $\overline{\overline{A}}(t) \leq A(t)$ and $A(t) \cong \overline{\overline{A}}(t)$ when $t$ is within a region with low curvature. We define the adjustment measure of $t$ by

$$Ad(t, e) = \frac{|A(t) - \overline{A}(t)|}{A(t)},$$

and the planarity measure by

$$Pl(t, e) = \frac{A(t) - \overline{\overline{A}}(t)}{A(t)}.$$

Consequently, the adjustment measure of $e$ is defined by

$$Ad(e) = \max\{Ad(t_e, e), Ad(t'_e, e)\},$$

and the planarity measure by

$$Pl(e) = \max\{Pl(t_e, e), Pl(t'_e, e)\}.$$

## 4.2 Edge flipping operator

We say that an edge $e = v_1v_3$ is *2D-flippable* if the $xy$-projections of its two adjacent triangles $t_e = \triangle v_1v_2v_3$, $t'_e = \triangle v_3v_4v_1$ form a convex quadrilateral.

Now we specify when a 2D-flippable edge $e$ is replaced by the edge $e' = v_2v_4$. First we compute the adjustment measures of $e$ and $e'$. If at least one of these measures is considered unacceptable, i.e.:

$$\max\{Ad(e), Ad(e')\} > \epsilon_1 \,,$$

edge $e$ is replaced by edge $e'$ only when

$$Ad(e) > Ad(e') \,.$$

Otherwise, we compute the planarity measures and the edge errors of $e$ and $e'$. When the edges are within a low curved region, i.e.:

$$\max\{Pl(e), Pl(e')\} < \epsilon_2 \,,$$

or when the edge errors are similar, i.e.:

$$\frac{|E_2(e) - E_2(e')|}{\max\{E_2(e), E_2(e')\}} < \epsilon_3 \,,$$

edge $e$ is replaced by edge $e'$ only when $e'$ improves the triangle quality, i.e.:

$$\angle v_1v_2v_3 + \angle v_3v_4v_1 > \angle v_2v_3v_4 + \angle v_4v_1v_2 \,.$$

On the contrary, $e$ is replaced by edge $e'$ if $e'$ improves the edge error, i.e. when

$$E_2(e) > E_2(e') \,.$$

By varying the thresholds $\epsilon_1$, $\epsilon_2$ and $\epsilon_3$ we can prioritize the adjustment, planarity and $L_2$ error measures. For all the results in this paper, we use $\epsilon_1 = 1.0$, $\epsilon_2 = 0.2$ and $\epsilon_3 = 0.2$

## 4.3 Two LEPPs strategy

The LEPP (Longest Edge Propagation Path) midpoint algorithm was designed to improve the smallest angles in a Delaunay triangulation. Given a triangle $t_0$, LEPP$(t_0) = \{t_0, t_1 ... t_{n-1}, t_n\}$ is a sequence of neighbor triangles in the mesh such that $t_i$ is neighbor of $t_{i-1}$ by the longest edge of $t_{i-1}$. Then, either $t_{n-1}, t_n$ share its longest edge, or the longest edge of $t_n$ is a boundary/constrained terminal edge. The longest edge of $t_n$ is said the *terminal* edge. The algorithm, introduced in a rather intuitive basis as a generalization of previous longest edge algorithms in [18, 17] and studied in [20, 21], can be simply described as follows: iteratively, each bad triangle $t_{bad}$ with smallest angle less than a certain bound in the current triangulation is eliminated by finding LEPP$(t_{bad})$ and associated terminal edge $l$. If $l$ is not a constrained edges, then the midpoint of $l$ is Delaunay inserted into the mesh. Otherwise a constrained point insertion criterion is used. The process is repeated until $t_{bad}$ is destroyed in the mesh, and the algorithm finishes when the minimum angle in the mesh is greater than or equal to the angle bound. In order to improve the performance of the previous LEPP midpoint algorithm, in [19] a new LEPP-centroid algorithm for quality triangulation is introduced. The centroid of the quadrilateral formed by the union of two triangles which share a non-constrained terminal $l$ is inserted into the mesh instead of the midpoint of $l$.

Our approach is based on destroying edges with higher error by inserting one or two points close to (one or two) associated terminal edges. In this way we create well-shaped triangles, like the LEPP Delaunay methods do. The three possible cases for each processing bad edge $e$ with adjacent triangles $t_e$ and $t'_e$ are as follows:

1. Edge $e$ is terminal (see Figure 1(a)).

2. Edge $e$ is the longest edge for $t_e$ but not for $t'_e$ (o viceversa). In this case we use the LEPP of triangle $t'_e$ for finding a terminal edge $e'$ (see Figure 1(b)).

3. Edge $e$ is not the longest edge for both $t_e$ and $t'_e$. In this case we use the LEPPs of triangles $t_e$ and $t'_e$ for finding two terminal edge $e'_1$ and $e'_2$. Figure 1(c) shows the two LEPPs used for finding $e'_1$ and $e'_2$ (in order to make clear how the method works, only interior triangles and insertion of midpoints are considered).

For the three cases, we insert a point inside the terminal quadrilateral defined by the associated terminal edges $e$, $e'$, $e'_1$, $e'_2$ as described in next section.

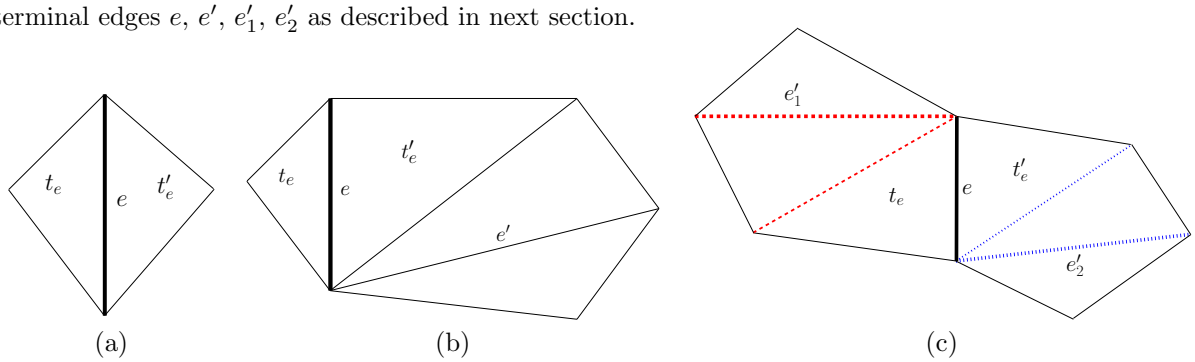



Figure 1: (a) Special case where $e$ is terminal. (b) Special case where $e$ is the longest edge of $t_e$ but not of $t'_e$ (or viceversa). (c) The two LEPP for edge $e$.

### 4.4 Surface-midpoint or surface-centroid selection

In this section we determine the point to be inserted for each reached terminal edge $e$. We denote by $\pi_e$ the vertical plane containing $e$ and by $F_e$ the intersection between the surface $F$ and $\pi_e$.

The *surface-midpoint* $sm_e$ is the point of $F_e$ equidistant from the endpoints $v_i$ and $v_j$ of $e$. The point $sm_e$ can be computed as the intersection between $F$ and the line contained in $\pi_e$, orthogonal to $e$ and passing through the point $(v_i + v_j)/2$. Since in the common case this computation can not be done exactly, the point $sm_e$ can be approximated by a bipartition algorithm. Given four points $p_1$, $p_2$, $p_3$, $p_4$ in a plane, the centroid of the quadrilateral $p_1p_2p_3p_4$ is defined by $(p_1 + p_2 + p_3 + p_4)/4$ or equivalently by $((p_1 + p_3)/2 + (p_2 + p_4)/2)/2$. We define the *surface-centroid* $sc_e$ of an edge based on the last expression. Let $e'$ be the edge determined by two vertices of the triangles adjacent to $e$ not contained in $e$. The point $sc_e$ is defined as the surface midpoint of the edge determined by $sm_e$ and $sm_{e'}$.

Our method inserts the surface-centroid when the angle criterion can be applied to $e$ (the edge is said to be 3D-flippable), and the surface-midpoint otherwise. After the point insertion the link edges of the point are tested to be flipped with the edge-flip operator in a similar way as in Lawson [11] algorithm.

## 5 LEPP-surface refinement algorithm

The process starts with a triangulation $\overline{F}$ of a subset of representative points of the real terrain. Then, new points of the real terrain are progressively added and triangulated using our algorithm until a predetermined number of vertices is inserted. The set of representative points is composed by a reduced subset of boundary points and some singular interior points which represent peaks, valleys, passes, etc. The algorithm iterates on the list of edges $E$ sorted in decreasing order by its approximation error. For each processed edge $e_f$ the two LEPPs strategy is applied and one or two terminal edges are obtained. If a terminal edge $e$ is 3D-flippable, we select the point $p$ as the surface centroid $sc_e$, otherwise we select $p$ as the surface midpoint $sm_e$. Next the closest input vertex $v$ to $p$ in $V$ is determined and inserted to $\overline{F}$. The determination of $v$ has a low computational cost because

of the subjacent grid data input. As each vertex insertion changes the triangulation, the list of edges $E$ have to be updated.

# 6 Results

In this section we present some results obtained with the application of our algorithm, implemented in C++ language and using Qt libraries, to real terrain data (Cervino, Kilimanjaro, Lake Como, Lake Maggiore, and Dolomiti). The refined terrains obtained have been compared with the real terrain data using Mesh1.3 [1], a freely available and widely used software that calculates the $L_2$ error between two surfaces. The $L_2$ error is expressed as a percentage over the length of the bounding-box diagonal of the model. We also have compared our results with simplified terrains obtained by QSlim [7]. For all the executions the following values $\epsilon_1 = 1.0$, $\epsilon_2 = 0.2$ and $\epsilon_3 = 0.2$ for the thresholds established in Section 4.2 were used.

Table 1 shows the quantity of elements of the original model and the resulting quantity applying the LEPP-surface and the QSlim algorithms. Table 2 presents a comparison of the approximation of the refined method and the simplified method with more and less the same quantity of faces as it was specified in Table 1. Table 3 summarizes the results of a statistical study of the minimum angle of the models when a refinement or a simplification process is applied. Figure 2 allows to observe graphically these comparison results.

| Model | Initial model | LEPP-surface triangulation | | QSlim triangulation | |
|---|---|---|---|---|---|
| | Vertices | Triangles | Vertices | Triangles | Vertices |
| Cervino | 9801 | 949 | 505 | 948 | 508 |
| Kilimanjaro | 9801 | 940 | 505 | 939 | 502 |

Table 1: Number of triangles and vertices in the original model and in the triangulation obtained with the LEPP-surface and the QSlim methods.

| Model | LEPP-surface method | | | QSlim method | | |
|---|---|---|---|---|---|---|
| | Min. Error | Max. Error | Mean Error | Min. Error | Max. Error | Mean Error |
| Cervino | 0 | 1.93 | 0.2 | 0 | 1.40 | 0.13 |
| Kilimanjaro | 0 | 0.74 | 0.1 | 0 | 0.84 | 0.07 |

Table 2: $L_2$ norm errors computed by using Mesh1.3 software. Minim, maxim, and mean errors are presented for triangulations of the same size obtained with LEPP-surface method and QSlim method.

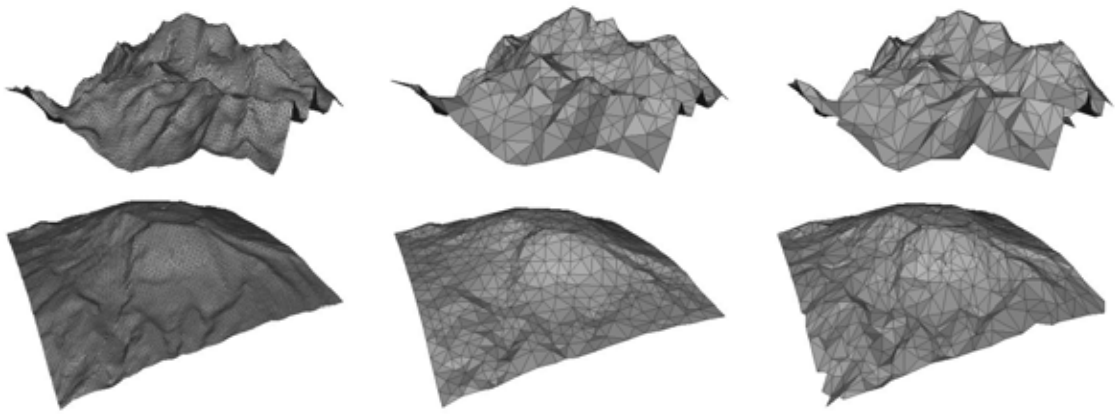


Figure 2: Cervino mountain in first line and Kilimanjaro mountain in second line. From left to right, original model, refinement and simplification.

| Model | LEPP-surface method | | | | | | QSlim method | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Angles | | | | Mean angle | 1st Quartile | Angles | | | | Mean angle | 1st Quartile |
| | $< 15°$ | $< 20°$ | $< 25°$ | $< 30°$ | | | $< 15°$ | $< 20°$ | $< 25°$ | $< 30°$ | | |
| Cervino | 6 | 22 | 60 | 127 | 40.31 | 34.55 | 45 | 128 | 245 | 402 | 32.90 | 24.68 |
| Kilimanjaro | 0 | 0 | 9 | 31 | 42.50 | 38.45 | 50 | 122 | 241 | 431 | 32.23 | 24.74 |

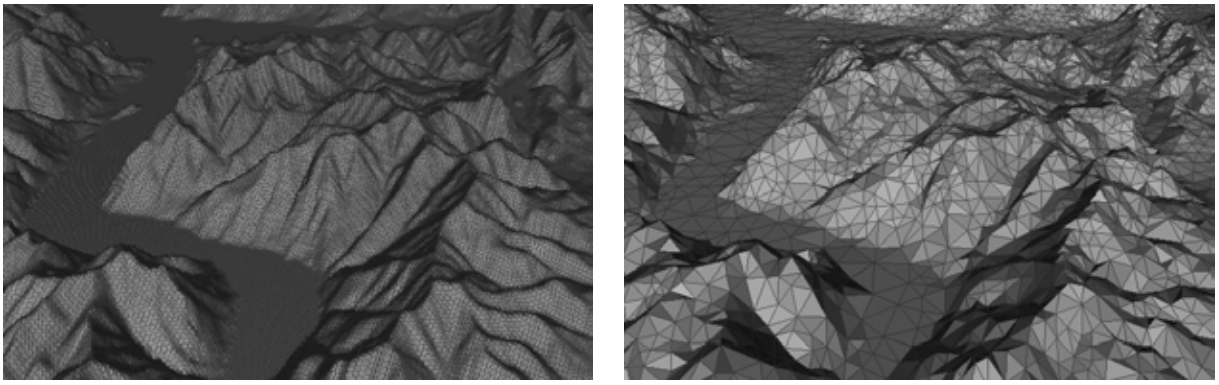
Table 3: Distribution of the minimum angles obtained with LEPP-surface method and QSlim method.

These results show that the refinement process achieves the main goals proposed: 1) the approximation of a terrain surface considering only original grid data; 2) with triangles of good aspect ratio. The error obtained with the refinement process is comparable with that obtained with a simplification process, as it is shown in Table 2 and Figure 2. The statistical study of Table 3 shows that the mean minimum angle, the quantity of triangles with angles less than 15°, 20°, 25° and 30°, and the first quartile obtained with the refinement are better than with the simplification.

Table 4 presents results obtained applying LEPP-surface method to several models. Here, error approximation and statistical data of angles are shown for each model. The algorithm takes about 9 seconds in generating this models in an Intel Pentium 3.00 GHz with 1.00 GB of RAM. Finally, Figure 3 shows a detail both of the original data and of the LEPP-surface approximation for the Lake Como with 5% of the original data set points with a total running time of about 60 seconds.

| Model | Vertices original model | LEPP-surface method | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Triangles | Vertices | Angles | | | | Mean angle | 1st Quartile | Max. error | Mean error |
| | | | | $< 15°$ | $< 20°$ | $< 25°$ | $< 30°$ | | | | |
| Lake Como | 810000 | 15985 | 8105 | 38 | 290 | 915 | 2078 | 40.40 | 34.74 | 0.57 | 0.001 |
| Dolomiti | 810000 | 15974 | 8105 | 31 | 277 | 842 | 1993 | 40.39 | 34.67 | 0.9 | 0.002 |
| Lake Maggiore | 810000 | 15957 | 8105 | 44 | 259 | 765 | 1750 | 40.76 | 35.70 | 0.64 | 0.0009 |

Table 4: Distribution of the minimum angles and error approximation obtained with LEPP-surface method for different models.



(a) (b)

Figure 3: Detail of the Lake Como. In (a) the original model and in (b) the refined model (5%).

# References

[1] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, *Mesh: Measuring errors between surfaces using the Hausdorff distance*, Proceedings of the IEEE International Conference in Multimedia and Expo, 2002, pp. 705–708.

[2] M. Bertram, J.C. Barnes, B. Hamann, K.I. Joy, H. Pottmann, and D. Wushour, *Piecewise optimal triangulation for the approximation of scattered data in the plane*, Computer Aided Geometric Design **17** (2000), no. 8, 767–787.

[3] L. DeFloriani, P. Magillo, and E. Puppo, *Variant: A system for terrain modeling at variable resolution*, GeoInformatica **4** (2000), no. 3, 287–315.

[4] N. Dyn, D. Levin, and S. Rippa, *Data dependent triangulations for piecewise linear interpolation*, IMA Journal of Numerical Analysis **10** (1990), no. 1, 137–154.

[5] P.J. Frey and H. Borouchaki, *Geometric surface mesh optimization*, Computing and Visualization in Science **1** (1998), no. 3, 133–121.

[6] M. Garland, *Multiresolution modeling: Survey & future opportunities*, Eurographics State of The Art Report, 1999.

[7] ———, *Quadric-based polygonal surface simplification*, Phd thesis, Carnegie Mellon University, 1999.

[8] M. Garland and P. Heckbert, *Surface simplification using quadric error metrics*, SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer graphics and Interactive Techniques, 1997, pp. 209–216.

[9] M. Garland and P-S. Heckbert, *Fast polygonal approximation of terrains and height fields*, Technical report CMU-CS-95-181, Carnegie Mellon University, 1995.

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Mesh optimization*, Computer Graphics **27** (1993), no. Annual Conference Series, 19–26.

[11] C.-L. Lawson, *Software for $\mathcal{C}^1$ surface interpolation*, Mathematical Software III (John R.Rice, editor) (1977), 161–194.

[12] P. Lindstrom and V. Pascucci, *Visualization of large terrains made easy*, Proceedings of the of IEEE Visualization 2001, 2001, pp. 363–370, and 574.

[13] ———, *Terrain simplification simplified: a general framework for view-dependent out-of-core visualization*, IEEE Transactions on Visualization and Computer Graphics **8** (2002), no. 3, 239–254.

[14] E. Nadler, *Piecewise linear best $L_2$ approximation on triangulations*, Approximation Theory V (C. K. Chui et al., ed.), 1986, pp. 499–502.

[15] R. Pajarola and E. Gobbetti, *Survey on semi-regular multiresolution models for interactive terrain rendering*, The Visual Computer **23** (2007), no. 8, 585–605.

[16] S. Rippa, *Long and thin triangles can be good for linear interpolation*, SIAM Journal on Numerical Analysis **29** (1992), no. 1, 257–270.

[17] M.-C. Rivara, *New mathematical tools and techniques for the refinement and / or improvement of unstructured triangulations*, Proceedings of the 5th International Meshing Roundtable, Pittsburgh, 1996, pp. 77–86.

[18] ———, *New longest-edge algorithms for the refinement and/or improvement of unstructured triangulation*, International Journal for Numerical Methods in Engineering **40** (1997), 3313–3324.

[19] M.-C. Rivara and C. Calderon, *LEPP terminal centroid method for quality triangulation: A study on a new algorithm*, Geometric Modeling and Processing, LNCS 4975, 2008, pp. 215–230.

[20] M.-C. Rivara, N. Hitschfeld, and R-B. Simpson, *Terminal edges Delaunay (small angle based) algorithm for the quality triangulation problem*, Computer-Aided Design **33** (2001), 263–277.

[21] M.-C. Rivara and M. Palma, *New LEPP algorithms for quality polygon and volume triangulation: Implementation issues and practical behavior*, 1997 Trends unstructured mesh generation (S. Saigal S. A. Cannan, ed.), vol. 220, ASME Applied Mechanics Division, 1997, pp. 1–8.

[22] S. Röttger, W. Heidrich, P. Slussallek, and H.-P. Seidel, *Real-time generation of continuous levels of detail for height fields*, Proceedings of 1998 International Conference in Central Europe on Computer Graphics and Viaualization, 1998, pp. 315–322.

[23] J-R. Shewchuk, *What is a good linear element? interpolation, conditioning, and quality measures*, Proceedings of the 11th International Meshing Roundtable, 2002, pp. 115–126.

# Una aproximación computacional al pegado de formas simples a lo largo de singularidades

Antonio Hurtado García * Javier Finat Codes†

**Resumen**

En este trabajo se presenta una aproximación constructiva al problema del pegado en presencia de singularidades para la descripción de objetos sólidos complejos en ausencia de información previa sobre las ecuaciones de las superficies que acotan dichos objetos. Para ello, se sigue una estrategia basada en la identificación de puntos críticos de una función de Morse adaptándola al caso singular discreto. En la Teoría de Morse estratificada, en lugar de pegar células como en la Teoría de Morse para el caso liso, se pegan datos normales y tangenciales de Morse; dichos datos se expresan mediante conos construidos sobre las singularidades.

## 1 Introducción

Existe un número creciente de repositorios de objetos digitales $3D$ que son accesibles en la red. Ello ha motivado una atención creciente a problemas de Reconocimiento de objetos $3D$ desde finales del siglo XX . Los objetos $Y$ que se consideran en este trabajo son sólidos en el espacio cartesiano $\mathbb{R}^3$. Para simplificar supondremos que el borde $\partial Y = X$ del sólido es una superficie compacta conexa orientada de género topológico arbitrario $g$; el género topológico se puede interpretar como el número de agujeros ó de asas de la superficie $X$ que acota el sólido $Y$. En la práctica, sólo se dispone de información digital sobre cada objeto $Y$ ó su borde $X$ que está descrita en términos de nubes de puntos $3D$ (procedentes de dispositivos de rango, como los escáneres $3D$), en términos de mallas (habitualmente triangulares) asociadas a estas nubes de puntos ó como modelos lisos a trozos (representaciones del borde para PS-modelos); actualmente, no se conoce ningún procedimiento óptimo general para el mallado de objetos $3D$ con topología arbitraria. Para facilitar el pegado de las PL- y PS-estructuras asociadas a nubes de puntos es necesario diseñar una estrategia para modelos de propagación que sea compatible con la presencia de singularidades en el borde de las regiones.

Los objetos del mundo real no son variedades lisas y los modelos asociados a dichos objetos presentan aristas y puntos singulares; por ello, inicialmente se trata de variedades analíticas que suponemos definidas por polinomios. Para resolver el problema del pegado, hay que restringir el tipo de "patologías" que pueden presentarse. Para empezar sólo se consideran singularidades sencillas aisladas (el vértice de un cono, p.e.) ó singularidades genéricas correspondientes a puntos múltiples ordinarios, es decir, singularidades que aparecen al realizar una proyección general desde un punto; la justificación de esta restricción procede de los objetos reales y del punto de vista del observador que se modela como una proyección genérica. Como en las aristas ó puntos singulares no está definido el plano tangente, es necesario reemplazar el espacio tangente en un punto liso por el *cono tangente* en un punto singular. Asimismo, es necesario adaptar los métodos descritos al caso digital discreto (ver [7] para un primer survey).

El "control" del comportamiento en el entorno del punto singular requiere identificar los datos tangenciales y normales, y un control de su variación en el entorno de puntos singulares. Existen diferentes criterios de control que intuitivamente afectan a condiciones de incidencia relativas a "elementos" (líneas ó planos) secantes y tangentes, y sus duales. Así, p.e., en el entorno de un punto de inflexión

*antonio.hurtado.garcia@gmail.com
†jfinat@agt.uva.es

para una curva plana hay una alternancia entre una región cóncava y otra convexa que se traduce en la anulación de la segunda derivada; esta condición se puede expresar en términos de un cambio de orientación del vector normal ó bien en términos de la posición relativa de la superficie con respecto a las cuerdas que aproximan las rectas tangentes desde el punto de vista discreto. La expresión en términos de datos normales es más natural desde el punto de vista computacional, pero la envolvente de las normales (cáustica ó superficie canal) *siempre* presenta singularidades; en el caso más simple de una elipse, la cáustica tiene forma de envoltura de caramelo con dos nodos y cuatro cúspides. Habitualmente, se elude este problema imponiendo la condición de "simplicidad" (sin auto-intersecciones) para las poligonales ó politopos resultantes y suprimiendo las componentes que quedan fuera; esta solución es inconsistente e impide obtener la forma original a partir de la envolvente.

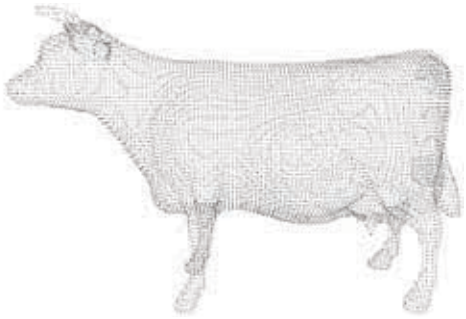


Figura 1: Nube de puntos procedente de escaneo

Una de las áreas de aplicación más importantes del problema del pegado en $3D$ es la clasificación de objetos sólidos, que incluye técnicas de hallazgo de contenido, etiquetado, indexación y clasificación. Este área aún sigue presentando un gran número de problemas abiertos en Matemáticas y en el diseño e implementación de los algoritmos correspondientes. El problema del Reconocimiento de objetos digitales $3D$ desde finales del siglo XX se ha abordado según diferentes enfoques que, en buena parte, extienden estrategias similares para el Reconocimiento de objetos en vistas digitales $2D$; ambas participan del mismo esquema: extracción de hechos, reconocimiento a bajo nivel, etiquetado, indexación, pegado y clasificación. Las estrategias de extracción de hechos utilizan diferentes técnicas. En este trabajo se supone realizada la extracción de hechos y la identificación (al menos visual) de las patologías concernientes a los cambios en propiedades geométricas (concavidad ó convexidad), limitándonos a algunos aspectos relacionados con la estimación de curvaturas (total y media) de la superficie que acota el objeto en relación con el problema del pegado.

Una vez extraídos hechos significativos es necesario "pegarlos" en objetos más complicados. El "pegado" requiere conocer las características de las componentes, las relaciones entre componentes (enfoque estructural) y el papel que juega cada componente (enfoque funcional). Por ello, la aproximación semántica al problema del Reconocimiento de Objetos $3D$ distingue tres fases ([1]) etiquetadas como geométrica (extracción de hechos), estructural (identificación de hechos como componentes y de las relaciones entre componentes) y funcional (papel desempeñado por las componentes dentro de un objeto global).

En el marco topológico más general, las componentes son células $k$-dimensionales, es decir, conjuntos topológicamente equivalentes a espacios cartesianos y el pegado se resuelve mediante procedimientos de adjunción de células. Una forma efectiva de representar dicha adjunción en el marco diferencial (Teoría de Morse) está dada por modelos de propagación controlados por el campo gradiente; en este caso, el agrupamiento de curvas de máxima pendiente en cilindroides (topológicamente equivalentes a cilindros con sus meridianos y paralelos) proporciona el soporte para el pegado de células. Las células se visualizan como subconjuntos de discos y el pegado se realiza a través de cilindroides (cilindros curvilíneos); una estimación de superficies esféricas y cilindros con aplicaciones a la Arquitectura (procedentes de escaneos $3D$ de bóvedas) se puede ver en [4]. En presencia de singularidades, las curvas integrales del campo gradiente (antes meridianos de un cilindroide) pueden colapsar a un punto ó presentar una arista cuspidal a lo largo de una curva, p.e.. Por ello, hay que estratificar previamente la variedad (atendiendo a las singularidades) en términos de datos normales y tangenciales, y reemplazar la descomposición celular estándar en Teoría de Morse por una descomposición en conos construidos

sobre pares de células. Los detalles técnicos presentan cierta complejidad. Este trabajo está enfocado a mostrar cómo reemplazar los meridianos y paralelos (líneas de máxima y mínima variación del campo gradiente) del caso liso por la noción diferencial correspondiente al caso discreto que son las "bandas superficiales" asociadas a las líneas de curvaturas principales. Las bandas superficiales permite extender el procedimiento de pegado al caso de objetos digitales singulares, al menos para las singularidades más sencillas. Estas tareas y su implementación computacional han sido llevadas a cabo en [6] con una especial atención al caso de conos, pues proporcionan las piezas básicas para abordar el "pegado" de datos tangenciales y normales para el caso singular.

El resto del trabajo está organizado como sigue: La sección 2 está enfocada a la discretización del caso liso. En la sección siguiente se describe la "apariencia" de las superficies basada en la detección de regiones cóncavas y convexas. La información métrica ligada a una versión discreta de las curvaturas se aborda en la sección cuatro, con una estimación de las curvaturas principales.

## 2 Marco conceptual: algunas restricciones para el Reconocimiento

Para superficies compactas conexas sin borde en $\mathbb{R}^3$, el reconocimiento topológico se realiza en términos de la Teoría de Morse para el caso liso ó bien de la Teoría de Morse Estratificada para el caso no-liso.Aún no existe una teoría similar para el caso discreto estratificado. La Teoría de Morse permite reconstruir variedades lisas con una topología complicada a partir de la adjunción de células en puntos críticos y la propagación del campo gradiente a lo largo de líneas de máxima pendiente. Para una hipersuperficie $X$ definida localmente por $f(\underline{x}) = 0$, la clave para el pegado de células radica en el *control de la variación del espacio tangente* mediante la identificación de puntos críticos del campo gradiente $grad(f)$ que representa el vector normal en el punto $\underline{x} \in X$. En el caso discreto no se tiene ninguna información sobre ninguna ecuación local $f$ de la hipersuperficie $X$; por ello, la propagación sólo puede tener lugar sobre la PL- ó la PS-estructura asociada a la superficie $X$ que acota el objeto $Y$. La propagación se define habitualmente en términos de operadores diferenciales; las PL-estructuras no verifican la condición de "suavidad" a lo largo de las aristas.

Para objetos digitales, el diseño e implementación del Reconocimiento es "activo", es decir, se basa en una interacción que debe combinar el barrido, la extracción y el tratamiento de la información; es preciso diseñar e implementar el tipo de interacción con las nubes de puntos $3D$ que permita extraer y analizar la información. La recuperación de la forma original del objeto (como lugar de puntos) se realiza a partir de datos tangenciales ó datos normales; en general, las envolventes de ambos datos tienen singularidades. Desde un punto de vista local, los datos tangenciales son intrínsecos y, por tanto, controlan el comportamiento en un punto de la superficie. Los datos normales son extrínsecos, es decir, dependen de la inmersión y están ligados al comportamiento local de *todas* las curvas que pasan por el punto. Desde el punto de vista diferencial, las curvaturas total y media en cada punto de la superficie proporcionan una medida de la variación de los datos tangenciales y normales, respectivamente. Por ello, es necesario diseñar e implementar algoritmos que permitan estimar dicha variación a partir de datos extraídos en modelos digitales. El comportamiento local de una superficie en $\mathbb{R}^3$ en el entorno de un punto no es rotacionalmente simétrico (depende de las curvaturas en el caso liso ó del tipo de singularidad), el operador de Laplace sobre una superficie arbitraria tiene un formalismo matemático más complicado que el operador de Laplace para el plano y los procesos de propagación responden a diferentes tipos de flujo (elíptico, hiperbólico ó parabólico) vinculados a una segmentación de la superficie por el tipo de curvatura. Debido a limitaciones de espacio, es imposible abordar los problemas anteriores. Por ello, a pesar del carácter incompleto y no-intrínseco de la aproximación realizada, en este trabajo se adopta un enfoque clásico ligado a la descripción de las curvaturas principales, relegando la elección del operador diferencial y los correspondientes fenómenos de propagación para una publicación posterior.

En relación con la *fase temprana del procesamiento*, algunos problemas a destacar son la significación de la nube (nivel de muestreo de la información), del punto de vista del observador (solapamiento visual de regiones), de los umbrales permitidos (propagación en las direcciones normales para detección temprana de singularidades); asimismo y a diferencia del caso de vistas $2D$ hay que tener presente que la información digital está dispersa, es decir, la mayor parte de los vóxeles adyacentes a uno dado están vacíos, es decir, no hay información sobre la función de intensidad definida, por lo que el diseño

de filtros es bastante más complicado. Es necesario construir una malla triangular y evaluar la viabilidad de un proceso de rellenado en términos de la variación de curvaturas, en ausencia de información analítica sobre la superficie.

Una vez construida una malla triangular, es posible desarrollar la *fase de análisis*, teniendo presente que las conclusiones dependen fuertemente de factores externos como el carácter incompleto de la información, la resolución ó las irregularidades en la distribución de la nube de puntos. El análisis está orientado a proporcionar información sobre *Reconocimiento a bajo nivel*. En ausencia de información sobre las ecuaciones de la superficie ó de funciones definidas sobre $X$, el Reconocimiento de objetos sólidos requiere identificar y conectar componentes que puedan ser asimiladas a modelos geométricos "significativos"; la significación afecta en este caso a la evaluación de características cualitativas (como la concavidad ó convexidad) ó cuantitativas (como las curvaturas y su variación) de las regiones. Para fijar ideas, llamamos *componente* a una unión maximal conexa de regiones con características comunes relativa a propiedades geométricas (concavidad ó convexidad) ó diferenciales (curvatura total del mismo tipo). Llamamos *segmentación* de $Y$ (resp. $X$) a una descomposición en unión disjunta de componentes $Y_i$ (resp. $X_i$) que pueden ser subconjuntos (segmentación topológica) ó subvariedades (segmentación como variedad). El *Reconocimiento de objetos* llevado a cabo en este trabajo desarrolla un enfoque constructivo para algunas propiedades de tipo diferencial a trozos que sean computacionalmente implementables. El análisis de la apariencia geométrica (concavidad vs. convexidad) y la Teoría de Morse proporcionan el marco conceptual para el enfoque constructivo y se abordan en las dos secciones siguientes.

# 3 Geometría de las Apariencias: Estimación de la concavidad

Las "apariencias" corresponden a los aspectos externos ligados a la *percepción visual*; por ello, se trata de una aproximación extrínseca, es decir, que depende de la inmersión del objeto en el espacio y de la localización relativa del observador con respecto al objeto. Así, por ejemplo, desde el punto de vista de un observador interno al objeto, las concavidades representan las regiones que están curvadas hacia dentro, (intuitivamente están "rehundidas"); en términos diferenciales el paso de una región convexa a una cóncava (ó viceversa) se traduce en el cambio de signo de la curvatura seccional de una curva contenida en la superficie; intuitivamente, el objeto planar ó volumétrico pasa a estar "del otro lado" del espacio tangente, comportamiento que está controlado por la geometría de los elementos normales (rectas/planos para el caso liso, conos para el caso singular).

El objetivo de esta sección es mostrar procedimientos para estimar la concavidad/convexidad local en el entorno de un punto $\mathbf{P}$ de una superficie $X$ a partir de la PL-información asociada a las nubes de puntos . La concavidad ó la convexidad de regiones de una superficie son *propiedades intrínsecas*, es decir, no dependen del sistema de coordenadas elegido. En el caso diferenciable, las condiciones de concavidad ó convexidad se expresan en términos de la posición relativa de una curva plana $C$ con respecto a (una estimación) su línea tangente $t_{\mathbf{P}}C$ ó de una superficie $\mathcal{X}$ con respecto a (una estimación de) el plano tangente $T_{\mathbf{P}}X$. Para la mayor parte de los objetos reales, sólo se dispone de datos discretos aislados; en nuestro caso, sólo se tienen nubes de puntos a diferentes resoluciones y con una distribución irregular. En particular, no se dispone de ecuaciones locales y el coste computacional de calcularlas (usando diferentes tipos de splines, p.e.) puede ser elevado; por ello, hay que aproximar los elementos geométricos (rectas/planos) tangentes mediante elementos discretos (segmentos/triángulos) secantes y calcular los cambios de posición del objeto con respecto a los elementos discretos próximos.

*Estrategias de acotación* La Geometría Computacional proporciona una batería de herramientas para formalizar la percepción visual. Siguiendo una estrategia de refinamientos sucesivos (coarse-to-fine), el tratamiento de la información para PL-modelos se aborda a partir de 1) Acotación tosca, 2) Envolvente Convexa y 3) Envolvente visual

*Acotaciones recursivas* La acotación tosca descrita en el primer apartado se puede refinar mediante la utilización recursiva de quadtrees/octrees que subdividen la caja seleccionada inicialmente suprimiendo las células rectangulares vacías. Los pasos a dar son los siguientes: 1)Realizar una subdivisión recursiva de la célula 2D/3D por un sistema de ejes/planos rectangulares,2)Etiquetar las células como vacías(E) ó llenas(F) según que haya información "significativa" , 3)Para cada orientación rectangular,

generar un mapa de barras 2D/3D que es un subdiagrama del quadtree/octree correspondiente a las células llenas y 4)Intersecar las orientaciones rectangulares correspondientes a los diferentes mapas de barras.

De este modo, se obtiene un *mapa 2D/3D de super-píxels/-vóxels* que facilita la gestión de la información. Este procedimiento es un primer paso para la generación de representaciones 3D a partir de varias cámaras. Proporciona una simplificación de escenas reales de utilidad para interacción dinámica en entornos complejos. Una implementación eficiente es el primer paso para el video 3D.

*Envolvente Convexa y tratamiento de la concavidad* El cálculo de la envolvente convexa permite acotar el rango de búsqueda de propiedades significativas para el objeto. En este trabajo, para el cálculo de la envolvente convexa se ha utilizado CGAL 3.3.1. No obstante, la envolvente convexa suprime todas las regiones cóncavas, lo cual no es coherente con las apariencias del objeto que se manifiestan como oclusiones parciales, p.e. Además, para objetos volumétricos complejos, la envolvente convexa suprime los "agujeros" por lo que no se ajusta a la percepción visual humana. La detección automática de la convexidad en $3D$ y, por consiguiente de la concavidad, está recibiendo una atención creciente ([2] ). Habitualmente, se define una región cóncava como la que no es convexa,; esta definición es de poca ayuda para objetos generados a partir de información discreta. La definición de la *concavidad* como la diferencia entre el volumen de la envolvente convexa y el "verdadero" volumen del objeto es de poca utilidad, pues supone conocido el "verdadero" volumen del objeto, algo que habitualmente se desconoce. Una estimación del volumen del objeto puede calcularse usando herramientas topológicas asociadas al volumen de todas las "franjas" (rebanadas sólidas) no vacías comprendidas entre dos isolíneas; es conveniente seleccionar varias direcciones con respecto a las cuales calcular los mapas de bandas respectivos y sus correspondientes volúmenes acotados por franjas sólidas).

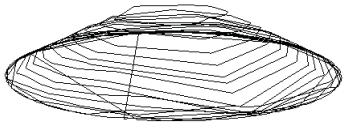


Figura 2: Isolíneas calculadas sobre un cono

Una región cóncava está separada por dos regiones convexas. En relación con esta cuestión, se han implementado los procedimientos siguientes:

1. *Inicialización:* Seleccionar una región del espacio 3D con candidatos a zonas cóncavas teniendo en cuenta discontinuidades (es decir, "agujeros" que no corresponden a agujeros del objeto) de las mallas debidas a oclusiones parciales
2. *Construcción de triángulo secante:* Cada terna de puntos no-alineados $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ genera el triángulo $conv[\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2]$ secante que es el soporte del plano secante $s_{012} =< \mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2 >$.
3. *Construcción de Direcciones en el espacio de planos secantes:* Dos ternas de puntos no-alineados $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ y $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ adyacentes (no necesariamente coplanarias) determinan una dirección $\ell_{0123} = (1-t)s_{012} + ts_{123}$ ó línea dentro del espacio de planos secantes.
4. *Identificación de planos bitangentes:* Un estimador para planos bitangentes viene dado por las intersecciones no-vacías de pares de familias disjuntas uniparamétricas de direcciones no vacías dentro del conjunto de líneas construidas sobre el espacio analítico que parametriza los planos secantes.

Obviamente, el punto más delicado es el último. Para obtener una implementación robusta, es necesario identificar "franjas sólidas" que jueguen el papel de sistemas coordenados rectangulares adaptados a la superficie del objeto. La distribución espacial de los datos es muy irregular y no cabe esperar puntos que estén a la "misma altura" para generar de forma automática curvas de nivel para un valor fijo de altura; por ello, las curvas de nivel lisas se reemplazan por "poligonales de nivel aproximado" en las que se enlazan puntos digitales situados en una franja modulo un umbral de tolerancia para la función altura aproximada. Llamamos *banda superficial* a la unión de triángulos comprendida entre

dos "poligonales de nivel aproximado". Llamamos *franja sólida* a la porción del espacio limitada por dos bandas superficiales. La construcción de las bandas superficiales permite estimar la concavidad ó la convexidad local en relación con la posición relativa de los elementos triangulares de superficie con respecto a los planos cordales (como estimación del plano tangente). Asimismo, la evaluación de la continuidad en la adyacencia entre componentes permite caracterizar y localizar las patologías relativas a discontinuidades para el soporte (malla ó superficie) ó singularidades para las aplicaciones definidas sobre dicho soporte.

# 4 Geometría Diferencial: Estimación de curvaturas

La noción más fina relativa a la forma de un objeto corresponde a la(s) curvatura(s). La curvatura proporciona invariantes intrínsecos pues caracteriza de forma única a la superficie en cada punto. Por tanto, la caracterización en términos de la curvatura permite comparar y "alinear" regiones para estimar si dos objetos son comparables ó no. De ahí el interés en estimar las curvaturas; el cálculo de las curvaturas está basado en propiedades métricas de la superficie. En el caso discreto, como sólo se dispone de una nube de puntos que presentan una distribución irregular, el mapa de curvaturas presenta habitualmente un elevado nivel de ruido que es necesario corregir. Sería deseable contar con procedimientos de regularización, pero el diseño e implementación para superficies en $\mathbb{R}^3$ no es elemental. El diseño del procedimiento de regularización debe equilibrar la información procedente de la curvatura total (con soporte en cada punto), con la curvatura media (con soporte en un un pequeño entorno del punto). El primer paso consiste en mostrar un procedimiento para estimar la curvatura a partir de datos discretos; para ello, se adopta un enfoque clásico:

La forma más intuitiva (Meusnier) de tratar la curvatura consiste en seleccionar el vector normal a una superficie $M$ en $\underline{p} \in M$ y considerar el haz uniparamétrico de planos $H_\alpha$ que contienen al vector normal $\mathbf{n}_{\underline{p}}$; los valores extremos para las curvaturas de las curvas planas $H_\alpha \cap M$ son las curvaturas principales $\kappa_1$, $\kappa_2$ de $M$ en el punto $\underline{p}$. Este procedimiento es muy intuitivo pero no es intrínseco. K.F.Gauss desarrolló un procedimiento intrínseco que consiste en introducir dos formas cuadráticas que miden la aproximación del plano tangente con respecto a la superficie (primera forma fundamental I) y la variación de dicho plano tangente (segunda forma fundamental II). Los valores propios correspondientes al polinomio característico de II son las curvaturas principales. La forma local de una superficie $M$ en un punto $\underline{p}$ está unívocamente caracterizada por la curvatura media $\kappa_m = \frac{1}{2}(\kappa_1 + \kappa_2)$ y la curvatura total $\kappa_t = \kappa_1\kappa_2$. La *estimación de las curvaturas principales de la superficie* $X$ se realiza mediante un procedimiento de interpolación adaptativa con parámetros seleccionables por el usuario a partir de las formas canónicas de Monge para los jets de orden bajo. Para ello, se elige un punto arbitrario $\underline{p} \in X$ con una referencia formada por dos vectores en el plano tangente y otro vector unitario en la dirección normal. En este caso, la superficie se puede representar como una función vectorial $z(x, y) = J_{B,d}(x, y) + h.d.t.$ donde

$$J_{B,d}(x,y) = \sum_{j=0}^{j=d} \sum_{i=j}^{i=0} \frac{B_{j-i,i} x^{j-i} y^i}{i!(j-i)!} \ ,$$

siendo $d$ el grado del polinomio de $J_{B,d}$ que se calcula como el desarrollo de Taylor ($d$-jet) de la función $z$. Fuera de los puntos umbílicos, donde las curvaturas principales son idénticas, las direcciones principales $\mathbf{d}_1$ y $\mathbf{d}_2$ están bien definidas, y junto con el vector normal $\mathbf{n}$, definen un sistema ortonormal directo. Si $\mathbf{v}_1$ es un vector unitario de dirección $\mathbf{d}_1$, existe un único vector unitario orientado $\mathbf{v}_2$ tal que $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{n})$ es directo; el otro posible eje orientado sería $(-\mathbf{v}_1, -\mathbf{v}_2, \mathbf{n})$. Estos sistemas de coordenadas reciben el nombre de sistema de coordenadas de Monge (ver figura 3) La *forma canónica de Monge* de la superficie $X$ esta dada por

$$z(x,y) = \frac{1}{2}(\kappa_1 x^2 + \kappa_2 y^2) + \frac{1}{6}(b_0 x^3 + 3b_1 x^2 y + 3b_2 x y^2 + b_3 y^3) + \frac{1}{24}(c_0 x^4 + 4c_1 x^3 y + 6c_2 x^2 y^2 + 4c_3 x y^3 + c_4 y^4) + h.d.t.$$

siendo $\kappa_{i_1}$ y $\kappa_2$ las curvaturas principales, $b_0$ y $b_3$ las derivadas direccionales de $\kappa_1$ y $\kappa_2$ a lo largo

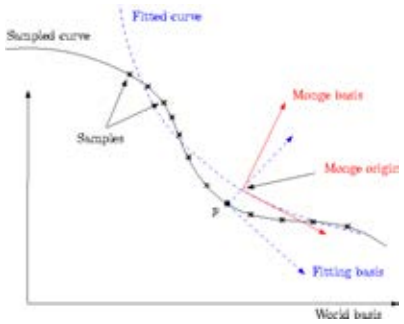



Figura 3: Sistema de coordenadas de Monge

de sus líneas de curvatura respectivamente, y $b_1$ y $b_2$ las derivadas direccionales de $\kappa_1$ y $\kappa_2$ a lo largo de sus otras líneas de curvatura. El algoritmo de cálculo tiene 4 pasos:

1. Análisis de componentes principales (PCA) sobre $P^+$. Con este análisis, conseguimos los 3 autovectores ortonormales, y los autovalores asociados. La base consiste en esos 3 vectores. El vector asociado al menor autovalor es el último vector de la base
2. Se realiza un *cambio de coordenadas* del sistema original al sistema centrado en el punto $\underline{p}$ donde se hace la estimación. Después se reordena para interpolar (o aproximar) al *d-jet* de la superficie $X$ en este sistema de coordenadas. Este polinomio de 2 variables reduce el problema a operaciones lineales de álgebra.
3. A partir del *d-jet*, se obtiene la *base de Monge* $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{n})$
4. Obtener los coeficientes de Monge: $\kappa_i$, $b_i$, $c_i$

Una vez estimadas las curvaturas principales $\kappa_1$ y $\kappa_2$, se calculan las curvaturas media $\kappa_m$ y total $\kappa_t$. El procedimiento de regularización para el mapa de curvaturas comentado más arriba afecta a los puntos de la nube que son los vecinos más próximos al punto sobre el que se pivota. La búsqueda de estos puntos próximos se realiza sobre los vértices de los triángulos contenidos en las bandas superficiales que pasan por el punto pivote; en particular, la estimación de las curvaturas principales se realiza sobre las bandas superficiales para las cuales la variación del campo gradiente es extremal (máxima ó mínima). Los resultados obtenidos ponen de manifiesto una "estabilidad local" de los valores correspondientes a las curvaturas media y total a lo largo de las franjas horizontales y verticales, tal y como cabía esperar. Actualmente, se trabaja en la comparación de los resultados obtenidos por este método con otros métodos descritos en la literatura

# 5 Elementos de Teoría de Morse Digital para el pegado

La Teoría de Morse es un área de la Topología Diferencial que trata de recuperar la forma de los objetos a partir de la información sobre los puntos críticos correspondientes a funcionales definidos sobre una variedad lisa $M$. Los funcionales más frecuentemente utilizados son de tipo potencial (la función altura, p.e.), una distancia (habitualmente la euclídea, p.e.) y un funcional de energía (que se puede expresa en términos de la curvatura, p.e.), según un orden de complejidad creciente. La interrelación entre todas ellas es uno de los tópicos más interesantes de la Topología en la segunda mitad del siglo XX. La utilización de la función altura se traduce en la recuperación de la topología de la variedad a partir del solapamiento de las curvas de nivel $f^{-1}(r)$ donde $r \in \mathbb{R}$ es un valor regular de la función potencial altura $f : M \to \mathbb{R}$; esta técnica es de uso frecuente en Imágenes Biomédicas (Tomografía ó Resonancia Magnética, p.e.). Los cambios en la topología de las curvas de nivel determinan la forma de la variedad; en particular, las curvas de nivel correspondientes a valores regulares comprendidas entre dos valores críticos son topológicamente equivalentes. Esta condición de trivialidad topológica permite construir

"cilindroides" que conectan las células que se insertan para cada punto crítico y que proporcionan, por consiguiente la clave para el "pegado".

En el caso discreto, en lugar de una variedad lisa $M$ se tiene una PL-superficie $X$ (en el mejor de los casos una PS-superficie); para calcular invariantes locales, se ha reemplazado más arriba la noción de curvas de nivel por la noción de "bandas superficiales" (cadenas conexas de triángulos comprendidas entre curvas de nivel). La trivialidad topológica para las curvas de nivel comprendidas entre dos valores críticos se traduce en una equivalencia topológica entre las bandas superficiales que permite construir franjas sólidas con forma de cilindroide ó de conoide para facilitar el "pegado" de objetos con una topología compleja. Las contribuciones más importantes en el terreno de la Topología Computacional relacionada con la Teoría de Morse digital conciernen al diseño e implementación de algoritmos para algunas relaciones entre propiedades locales y globales. Estas contribuciones incluyen: 1) Obtención de franjas (diseño e implementación del análogo discreto a las isolíneas) 2)Transversalidad entre franjas (mapas de isolíneas horizontales y verticales) y 3) Cálculo del número de componentes conexas correspondientes a las franjas.

# 6 Conclusiones

En este trabajo se han presentado modelos y algoritmos para el pegado de datos a lo largo de "bordes"a partir de información discreta soportada por nubes densas de puntos procedentes de escaneos y mallas triangulares superpuestas. Para integrar y gestionar las herramientas desarrolladas, se ha diseñado e implementado la plataforma software RAS (Recognition of Appearance and Shape, [6]). Esta plataforma incluye funcionalidades para la estimación de datos geométricos ligados a la Geometría de las Apariencias (estimación de la concavidad), la Geometría Diferencial de Superficies (estimación de curvaturas principales) y evaluación de las patologías (caracterización y localización de discontinuidades). Asimismo, se han desarrollado módulos que mejoran algunas funcionalidades más propias de la Informática Gráfica, orientadas hacia la mejora de la gestión de la información contenida en las mallas, dependiendo de la distribución en la nube de puntos y los mapas de normales unitarios asociados; estas mejoras conciernen a la *Detección de Patologías* en relación con la identificación de discontinuidades en las mallas y evaluación en términos de características topológicas (conexividad) ó de defectos en los procedimientos de adquisición. Por último, se ha iniciado una extensión de la Teoría de Morse Digital al caso estratificado con el análisis de las piezas básicas (conos sobre datos tangenciales y normales) para los modelos discretos ligados a nubes densas de puntos procedentes de escaneos de objetos $3D$.

# Referencias

[1] AIM & SHAPE: http://www.aimatshape.net/

[2] M.Attene, M.Moratar, M.Spagnuolo and B.Falcidieno: "Hierarchical Convex Approximation of 3D Shapes for Fast Region Selection", Technical Report, Genova, 23 Maggio 2008.

[3] http://www.cgal.org/ Computational Geometry Algorithms Library V3.3.1

[4] J.Finat, A.Hurtado, et al: "A systemic approach for 3D Recognition of simple primitives in discrete models", *The Intl. Arch of the Photogrammetry, remote sensing and spatial information sciences*, Sabry-El-Hakim and F.Remondino (eds), ISSN 1682-1777, 2009

[5] M.Hilaga, Y.Shinagawa, T.Kohmura, and T.L. Kunii: "Topology matching for fully automatic similarity estimation of $3D$ shapes". In *Proc. 28th annual conf. on Computer graphics and interactive techniques*, 203-212, ACM Press, 2001.

[6] A.Hurtado García: "Reconocimiento de la Apariencia en la Forma", Proyecto Fin de Carrera Ingeniería Informática, Universidad de Valladolid, Febrero 2009.

[7] R.C Veltkamp: "Shape matching: Similarity measure and algorithms", *Proceedings Shape Modeling Intl*, 188-197, 2001.

# Solving Reverse $k$-Nearest Queries on Road Networks with the GPU *

Yago Diez † Marta Fort ‡ J. Antoni Sellarès §

**Abstract**

We present a GPU-based approach for computing discretized distance functions on road networks. As applications, we provide algorithms for computing discrete Order-$k$ Nearest Neighbor diagrams and for approximately solving (Bichromatic) Reverse $k$-Nearest Neighbor queries on road networks. Finally, we present experimental results obtained with the implementation of our algorithms that demonstrate the effectiveness and efficiency of our approach.

## 1 Introduction

A Road Network Database allows to efficiently store and query objects in road networks. Static objects are represented by points of interest located on the network (hotels or gas stations) and moving objects are represented as points running along the network (cars or pedestrians). The network distance between objects on the road network (car and gas station), defined as the cost of the shortest path between them, depends on the connectivity and weights of the underlying network.

Computing shortest paths, and consequently network distances, is a fundamental combinatorial optimization problem with important applications in various domains, like Geographic Information Systems, Location-Based Services, Navigation Systems, Mobile Computing Systems, Data Clustering, etc. The computation of network distances often arises as a subroutine in the solution of many proximity problems related the determination of the influence of an object on other objects.

### 1.1 Preliminaries

We model a road network as an undirected weighted graph $N(V, E, W)$. The set $V$, $n = |V|$, is a set of vertices representing road intersections (with degree above 2), terminal points (with degree 1) and shape points (with degree 2). Shape points are neither intersection nor terminal points that are interpolated to create sequences of arbitrarily small linear parts of the road network. The spatial position of each vertex in $V$ with respect to a reference coordinate system is also given. The set $E$, $m = |E|$, is a set of edges representing road segments, each connecting two vertices. $W : E \to \mathbb{R}^+$ associates each edge $e$ with a positive weight $w(e)$, that may represent, for example, the Euclidean length of $e$, denoted $|e|$, or the time, toll, energy consumption, etc., required to travel between the two endpoints of $e$. A portion $\overline{e}$ of and edge $e$ connecting to points of $e$ is called subedge. We define the weight $w(\overline{e})$ of the subedge $\overline{e}$ as $w(\overline{e}) = (|\overline{e}|/|e|)w(e)$. In this paper we consider static road networks with a fixed weight for each edge. The set of all the points of the network $N(V, E, W)$, vertices and points on edges, is denoted by $\mathcal{N}$. Although in the worst case a graph could be dense ($m = O(n^2)$), in general, graphs modelling networks have constant degree and consequently are sparse ($m = \Theta(n)$). We want to note that for simplicity we have modelled a road network as an undirected graph, but that our methods can be easily applied to road networks modelled as a directed graph, in which one-way roads or roads where the weight is different for the two directions are allowed.

*Marta Fort and J. Antoni Sellarès are partially supported by the Spanish MEC grant TIN2007-67982-C02-02.

†Institut d'Informàtica i Aplicacios, Universitat de Girona, Spain, ydiez@ima.udg.edu

‡Institut d'Informàtica i Aplicacios, Universitat de Girona, Spain, mfort@ima.udg.edu

§Institut d'Informàtica i Aplicacios, Universitat de Girona, Spain, sellares@ima.udg.edu

A path $\pi(p,q)$ between two points $p$ and $q$, $p$ located on edge $(v_0,v_1)$ and $q$ located on edge $(v_l,v_{l+1})$, is a sequence $[p,v_1,\cdots,v_l,q]$, where $v_1,\cdots,v_l \in V$ and $(v_{i-1},v_i) \in E$, $1 < i \leq l$. The cost of the path $\pi(p,q)$ is defined by:

$$\|\pi(p,q)\| = w(\overline{pv_1}) + \sum_{2<i\leq l} w(\overline{v_{i-1}v_i}) + w(\overline{v_lq})\,.$$

The path of least cost connecting $p$ and $q$ is called shortest path between $p$ and $q$. The cost of the shortest path is called the network distance between $p$ and $q$, denoted $d(p,q)$. In the following, for simplicity, we use the term distance to refer to the network distance. The distance function defined by a site $p$ on $\mathcal{N}$ is a function $d_p$ such that for any point $q \in \mathcal{N}$, $d_p(q)$ is the distance $d(p,q)$ between $p$ and $q$.

The Order-$k$ Nearest Neighbor diagram of a set of points of interest, called sites, on a network is the partition of the network into $k$-th Nearest regions, where the region associated with a site is the set of points on the network for which the site ranks number $k$ in the ascending order of the sites by network distance. A Reverse $k$-Nearest Neighbor query returns all sites that have a given query point among their $k$-nearest neighbors. In the bichromatic case there are two distinct site types, and a Bichromatic Reverse $k$-Nearest Neighbor query returns all sites of one type that have a given query site of the other type among their $k$-nearest neighbors.

## 1.2 Graphics Hardware

The graphics pipeline is divided into several stages. The input is a list of 3D geometric primitives expressed as vertices defining points, lines, etc. with attributes associated such as color or texture coordinates, among others. The output is an image in the frame buffer, a collection of several hardware buffers corresponding to 2D grids whose cells are called pixels. A pixel can store a depth value in the depth buffer and up to four values in the color buffer, namely the color channels R, G, B, and alpha. In the first stage of the pipeline, per-vertex operations take place, each input vertex is transformed from 3D coordinates to window coordinates. Next stage is rasterization, when it finishes we obtain a fragment, with its associated attributes, for each pixel location covered by a geometric primitive. Fragment attributes are obtained from the attributes associated to the vertices by linear interpolation. The third stage, the fragment stage, computes the color for each pixel in the frame buffer, according to the results of a series of per-fragment operations and tests, such as the depth test. This output can be transferred to the CPU or stored in a texture, a 2D array of pixels, and then used in additional computations. The programmable parts of the graphics pipeline are the vertex, geometric and fragment shaders which are used to change the vertex attributes, create new vertices or change fragment attributes. Fragment shaders can read in and modify texture contents in a per-pixel fashion, so that the same instruction call is executed for all pixels.

## 1.3 Related Work

In [2], a multi-pass GPU-based algorithm to efficiently compute approximated Order-$k$ Nearest Neighbor diagrams for a set of points in the Euclidean plane is presented. Algorithms for solving (Bichromatic) Reverse $k$-Nearest Neighbor queries in large graphs are presented in [6]. The majority of works on queries present solutions that mainly focus on spatial data structures (R-trees, etc) and branch and bound pruning techniques that use location and connectivity information to guide the search.

## 1.4 Our Contribution

By working toward practical solutions, we present an algorithm that takes advantage of the capabilities of the GPU to compute distance functions on road networks. The discretization of the network motivated by the use of the GPU makes solutions approximate by necessity. However this is reasonable since, in practice, the underlying network is often known only approximately. Moreover, in real-world settings, for example it is useful to know not only the nearest gas station, but also other stations that are "close enough" to that nearest one. As applications, we provide algorithms to compute discrete Order-$k$ Nearest Neighbor diagrams and for approximately solving (Bichromatic) Reverse $k$-Nearest Neighbor queries. Finally, we present experimental results that demonstrate the effectiveness and efficiency of the proposed approach.

# 2 Distance Function Computation

We compute the distance from a given site $p$ on $\mathcal{N}$ to all vertices of $V$ by using the classical Dijkstra's algorithm [1]. Dijkstra's algorithm computes the shortest path from site $p$ to all vertices of $V$ by incrementally growing a tree of shortest paths from $p$ out to the most distant vertices. Dijkstra's algorithm for sparse graphs, like road networks, has $O(n \log n)$ asymptotic time complexity for an undirected weighted graph with $n$ vertices [5]. In order to reduce computation times several speed-up techniques have been developed during the last years [3, 5]. .

Now we face the most general problem of computing the distance from a given site $p$ to any point on $\mathcal{N}$. Since the continuous nature of the problem makes it difficult to efficiently compute this distance, we propose an alternative GPU solution based on a discretization (approximation) process that allows us to explicitly store the distance function on a compact way.

Along the paper several time complexity analysis are presented using the following notation. We denote by $\rho$ the time needed to render and color a fragment, which is much smaller than that needed to read a pixel, denoted by $t$, or the time needed to transfer information from the CPU to the GPU, denoted by $T$. The time needed to render a rectangle covering $H \times W$ pixels is denoted by the constant $HW\rho$, and finally we denote $O(\mathcal{N}\rho)$ the time needed to render the edges defining the network $\mathcal{N}$.

## 2.1 Road Network Parameterization

Let $\mathcal{R}$ be a rectangular grid of the $xy$-plane of size $W \times H$. Consider a mapping $\tau$ that maps each edge $e$ of the road network $\mathcal{N}$ to a row segment of $\mathcal{R}$ represented by the centers of its cells (see Fig. 1a). The mapping optimally packs the mapped edges into $\mathcal{R}$ so that: a) the mappings of two different edges of $\mathcal{N}$ do not overlap in $\mathcal{R}$; b) the mapping of all edges of $\mathcal{N}$ covers "almost" all grid cells of $\mathcal{R}$, it is to say, the number of row cells covered by $\tau(e)$ "approximately" equals $WH|e|/\sum_{e' \in E}|e'|$. This way we obtain a discrete parametrization of the road network $\mathcal{N}$ meaning that each cell in the grid map represents a unique location on the road network. Notice that the converse is not true, a vertex of degree $d$ is mapped to $d$ different positions of $\mathcal{R}$. Finally, we denote $\tau(e, v)$ the mapping of vertex $v$ when considering it as part of edge $e$.

## 2.2 Distance Function Discretization

To compute a discretization of the distance function $d_p$, defined by site $p$ on $\mathcal{N}$, the region $\mathcal{R}$ is represented by using the depth buffer. Consequently, $\mathcal{R}$ is thought as a grid of size $W \times H$ where distance values, normalized into the interval $[0, 1]$, are stored (see Fig. 1b). Discrete distances are computed by using Dijkstra's algorithm and the GPU.

We compute the discretized distance function $d_p$ during Dijsktra's algorithm. Initially all the points in $\mathcal{R}$ are assumed to be at distance one (the maximal depth value) from the source $p$. During Dijkstra's algorithm we compute the distances from $p$ to the vertices $v$ of $\mathcal{N}$ and to the other points of $\mathcal{N}$ simultaneously. When the distance from $p$ to a vertex $v$ is updated by Dijkstra's algorithm, the distances to the edges incident to $v$ are also recalculated and updated by using graphics hardware.

For every edge $e = vw$ incident to $v$ we render $\tau(e)$ on $\mathcal{R}$ in order to recompute and update the distances to the points of $e$. When segment $\tau(e)$ is rendered we use texture coordinates to associate distance 0 to the endpoint $\tau(e, v)$ and the distance from $v$ to $w$ to $\tau(e, w)$. The current distance from $p$ to $v$, $\overline{d_p(v)}$ and a constant to normalize distance values into $[0, 1]$ are sent to the fragment shader. OpenGL rasterizes the segment subdividing it into fragments conforming the discretized edge points. Then, the fragment shader computes the current distance from $p$ to each fragment by adding to $\overline{d_p(v)}$ the distance from the current point to vertex $v$ which is obtained in the corresponding texture coordinate. This distance is normalized and stored as the fragment depth.

In order to correctly compute a distance function, in the initialization process of Dijkstra's algorithm we initialize the depth buffer to the maximal depth value (1). When a new fragment is processed, the depth buffer is updated if and only if the depth value of the current fragment is smaller than the

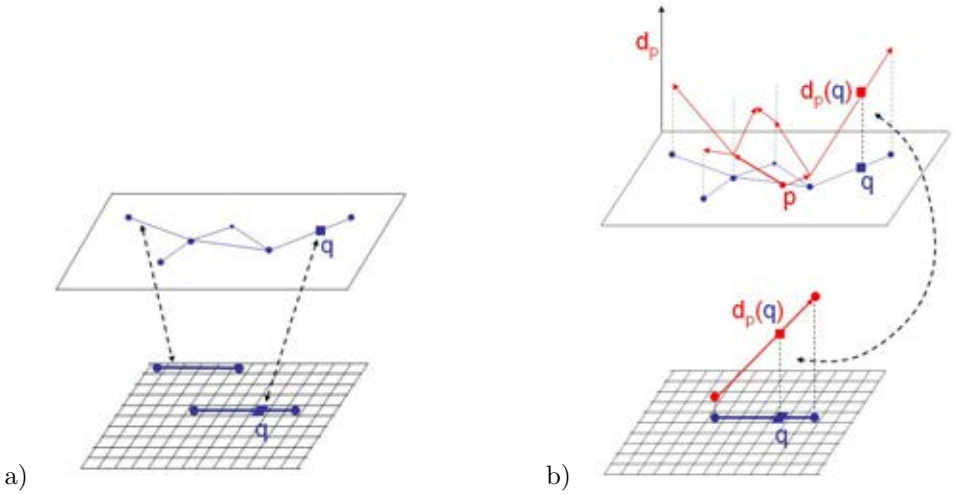



Figure 1: a) Discrete Road Network Parameterization; b) Discrete distance function.

current value in the depth buffer. At the end of the process the value stored in the depth buffer is the minimum depth (distance) defined by all the processed fragments. Consequently the discrete representation of the distance function is obtained during Dijkstra's algorithm computation.

The time needed to compute and store a distance function in the GPU is $O(\mathcal{N}_\tau\ \rho)$, where we denote by $\mathcal{N}_\tau$ the time needed in Dijkstra's algorithm to render the image of the edges of $\mathcal{N}$ on $\mathcal{R}$. Notice that edges are rendered when the distance to one of its vertices is updated, thus each edge is generally rendered more than once. Consequently, the time needed to obtain and store in the GPU a collection of $s$ distance functions is $O(s\mathcal{N}_\tau\ \rho)$.

# 3 Applications

Let $P = \{p_1, \cdots, p_s\}$ be a set of $s$ static sites lying on the road network $\mathcal{N}$ and $k \in \{1, \cdots, s\}$. From now on, we assume that: a) each site in $P$ has an identifying index $i = 1 \ldots s$ and an associated color; b) the discrete distance functions of all the sites have already been computed and are stored in the GPU by using textures; c) each texture stores as much distance functions as possible according to the discretization size and the maximal texture size which depends on the graphics card characteristics. The assumption that all the distance functions are stored in the GPU implies that the maximal number of sites that can be processed depends on the GPU memory amount. However, it notably speeds up the processes and reduces the CPU memory storage.

## 3.1 Order-$k$ Nearest Neighbor Diagrams

For any point $q \in \mathcal{N}$, let $n_k(q)$ denote the $k$-th nearest site to $q$ in $P$, i.e. the site that ranks number $k$ in the ordering of the sites by increasing distance from point $q$. The $k$-th Nearest Neighbor region $N_k(p)$ of a site $p \in P$, is the set of points $q$ of $\mathcal{N}$ for which $p$ is the $k$-th nearest site:

$$N_k(p) = \{q \in \mathcal{N} \,|\, n_k(q) = p\}\,.$$

The region $N_k(p)$ is a, possibly unconnected, collection of edges and subedges of $\mathcal{N}$.

The Order-$k$ Nearest Neighbor diagram of $P$, denoted $\mathcal{N}_k(P)$, is the partition of $\mathcal{N}$ into $k$-th Nearest Neighbor regions:

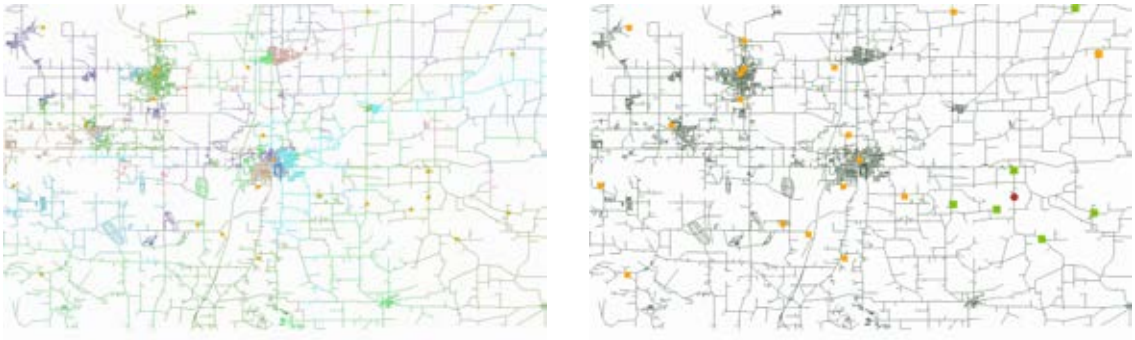
$$\mathcal{N}_k(P) = \{N_k(p) \,|\, p \in P\}\,.$$

The Order-$k$ Nearest Neighbor diagram coincides with the $k$-lower level of the arrangement defined by the distance functions of the sites in $P$.

### 3.1.1 GPU-based Solution

The Order-$k$ Nearest Neighbor diagram can be computed by using the "depth peeling" technique [2]. At each pass all the distance functions are rendered using their corresponding color by defining a rectangle covering $\mathcal{R}$ and associating to its vertices the texture coordinates of the current distance function. By using the depth test and initializing the depth buffer to one, the minimal depth value is stored in the depth buffer. In the first pass the representation of $\mathcal{N}_1(P)$ on $\mathcal{R}$ is obtained in the color buffer. The depth buffer contains the distance from every point of $\mathcal{N}$ to its nearest site. When the first pass finishes, the depth buffer is transferred to a texture. In the second pass all the distance functions are again rendered. In the fragment shader the distance function that is being rendered is compared with the distances obtained in the previous pass. Only the fragments with distance bigger than the one obtained in the previous pass, the others are discarded (by setting them to a depth value bigger than one). Therefore, the distance values obtained in the second pass correspond to the distances to the second nearest site and thus in the color buffer we obtain a representation of $\mathcal{N}_2(P)$ on $\mathcal{R}$. After this process has been repeated $k$ times the representation of $\mathcal{N}_k(P)$ on $\mathcal{R}$ is obtained in the color buffer and the depth buffer stores the distance to the $k$-nearest site.

The representation of $\mathcal{N}_k(P)$ on $\mathcal{N}$ can be obtained by transferring the resulting color information in a texture and rendering $\mathcal{N}$. When edge $e = (v, w)$ is rendered we associate the texture coordinates corresponding to $\tau(e, v)$ and $\tau(e, w)$ to its endpoints so that it uses the color information stored in the texture to color it. Figure 2a shows an Order-7 Nearest Neighbor diagram of 25 sites.



a) b)

Figure 2: a) Order-7 Nearest Neighbor diagram; b) Reverse 7-Nearest Neighbor query example.

Computing an Order-$k$ Nearest Neighbor diagram requires rendering the $s$ distance functions $k$ times during the peeling process and, consequently , $O(s\ k\ HW\ \rho)$ time is needed. In order to solve the other problems the Order-$i$ Nearest Neighbor diagrams, $i \leq k$, are stored in the CPU in float arrays, it provides extra $O(k\ HWt)$ time. Consequently, the total time complexity is $O(s\ k\ HW\ \rho + k\ HWt)$.

## 3.2 Reverse $k$-Nearest Neighbor Queries

Given a set $P$ of sites and a query point $q$, $P$ and $q$ located on the network, a Reverse $k$-Nearest Neighbor (R-$k$-NN) query retrieves the subset $RNN_k(P, q)$ of sites of $P$ that have $q$ in the set of their $k$-nearest neighbors:

$$RNN_k(P, q) = \{p \in P \,|\, q \in NN_k(P, p)\}\,.$$

For example, a R-$k$-NN query may be issued to find the three stores that are most affected, because of its geographical proximity, by opening a new store at some specific location.

### 3.2.1 GPU-based Solution

A Reverse $k$-Nearest Neighbor query, $RNN_k(P, q)$, is solved by computing the distance function defined by point $q$ and the distance of each point of $\mathcal{N}$ to its $(k+1)$-nearest site. Notice that the distance to the $(k+1)$-nearest site is the value stored in the depth buffer when the Order-$(k+1)$ Nearest Neighbor diagram is obtained. For each site $p_i \in P$ we compare the distance to its $(k+1)$-nearest site, denoted $dn_{k+1}(P, p_i)$, with the distance from $p_i$ to $q$, $d_q(p_i)$. Site $p_i$ is contained in $RNN_k(P, q)$ if and only if $d_q(p_i) \leq dn_{k+1}(P, p_i)$. In Fig. 2b, the solution to a Reverse 7-Nearest Neighbor for a query point, represented by a circle, is presented in a color gradation according to the ordering of the sites by network distance.

$k$-Reverse Nearest Neighbor queries can be answered in $O(\mathcal{N}_\tau\ \rho + HWt + s)$ assuming that we have already computed and stored the corresponding Nearest Network diagram and its distances in the CPU. With this assumptions we only have to compute the distance function $d_q$, store it in the CPU and compare the distance values of the $s$ sites.

## 3.3 Bichromatic Reverse $k$-Nearest Neighbor Queries

In some applications, the points in $P$ belong to two different categories $B$ and $R$. These two categories may be thought of as being colored blue and red and are usually named sites and points respectively. A Bichromatic Reverse $k$-Nearest Neighbor (BR-$k$-NN) query for a point $r \in R$ seeks to determine the set $BRNN_k(B, R, r)$ of sites $b \in B$ for which $r$ is a $k$-Nearest Neighbor in $R$:

$$BRNN_k(B, R, r) = \{b \in B \,|\, r \in NN_k(R, b)\}\,.$$

For example, given several choices for the location of a new restaurant, a BR-$k$-NN query may be used to evaluate the benefit of the new restaurant in terms of the customers that it may attract from rival restaurants based on the customers living in the five nearest buildings.

### 3.3.1 GPU-based Solution

In order to solve a Bichromatic $k$-Reverse Nearest Neighbor query, $BRNN_k(B, R, r)$, we first compute the $i$-Nearest Neighbor diagrams of $R$: $\mathcal{N}_1(R), .., \mathcal{N}_k(R)$. Next for each $b \in B$ we determine whether $r \in NN_k(R, b)$. This is done by checking if $b$ is contained in a $i$-nearest neighbor region of $r$, $N_i(r)$. If $b \in N_i(r)$ with $i \leq k$ then $b \in BRNN_k(R, R, r)$.

The time complexity of a Bichromatic Reverse $k$-Nearest Neighbor query is $O(k\ |B|)$, where $|B|$ denotes the number of blue sites. In this case we assume that we have already computed the $i$-Nearest Network diagrams of the red sites and consequently we provide the time needed to check for each blue site whether it is contained in the desired region.

# 4 Experimental Evaluation

In this section we present experiments that discuss on the main characteristics of the algorithms presented as well as provide details on their degree of efficiency. We have implemented the proposed methods using C++, OpenGL and Cg. All the experiments have been carried out on a Intel(R) Core(TM)2 Quad CPU at 2.83GHz with 3.25GB of RAM and a NVIDIA GeForce GTX 280 graphics board. We have used the road network of Nevada with 280,942 nodes and 338,205 edges. In this section we present experiments that show the degree of efficiency of our algorithms. We have worked with TIGER/lines® files that can be found at [4]. In our case, CPU time does not provide meaningful information as most of our calculations are performed in the GPU. Consequently the time presented throughout this section stands for real (calendar) time measurements.

Any graphics board has some specific memory limitations. Our NVIDIA GeForce GTX 280 graphics board has 1G of memory and can work with at most 32 textures simultaneously. Storing the discretized distance functions in the GPU produces that the number of sites that can be fitted in the GPU depends

on the GPU characteristics and the discretization size ($W \times H$). These two limitations (memory limitation and maximal number of simultaneously storable textures) make an efficient management of the GPU mandatory. We have worked with big textures (of size $4096 \times 4096$) and stored as many discretized distance functions as possible in each of them. For example, for the map of Nevada with $2048 \times 1024$ resolution, we are able to store up to 256 distance functions.

## 4.1 Distance Function Computation

In this section we present details on some of the fundamental characteristics of our algorithms. First we provide an experimental study on the precision achieved and then we discuuss on the algorithm that computes distance functions. In order to highlight the characteristics of the other algorithms, we consider distance function computation as a necessary preprocessing step and only present its computation times in this section.

**Error Analysis**

We can distinguish between two different types of algorithm-related errors: a) Discretization error that depends on the discretization size i.e. the biggest the grid size the smallest the error, and b) Floating point representation errors, which are related to the depth buffer and depth texture precision. A 16-bit precision is sufficient to store the normalized distances which take values in the interval $[0, 1]$.

In the following experiment we aim at quantifying how much does the discretization error contribute to the total error. We have worked with different resolutions for the map of Nevada and calculated the distance between two randomly placed points in the road network using two different methods: a) Software method: We calculate the distance between the two points using Dijkstra's algorithm with the first point as source; b) Hardware method: We calculate the distance function of the first point and then look up the position of the second point.

The first method suffers only from possible errors in data and of the representation errors related to any algorithm. The second method includes also errors related to the discretization produced by our algorithm and to the precision limitations of GPUs. We compared the distance obtained with each algorithm and obtained a measure of the discretization and GPU precision errors which are only present in the second method. Table 1 presents the name of the maps, the number of nodes in each map, the parametrization resolution, the maximum theoretical error related to the discretization and the mean distance difference observed in 1000 repetitions of the experiment (mean observed error). The maximum theoretical error is calculated as half the length in meters of the side of one of the squares of the discretization grid. Errors are given in meters. The table shows how the use of finer discretization grids grants less theoretical and practical error, it also shows how the precision of our algorithm is high.

Table 1: Parameterization precision.

| Disc. size | Max. err. (m) | Mean err. (m) |
|---|---|---|
| $1024 \times 1024$ | 75.77 | 37.24 |
| $2048 \times 1024$ | 30.64 | 19.27 |
| $2048 \times 2048$ | 13.98 | 7.08 |

Table 2: Order-$k$ Nearest Network diagrams.

| $s$ | $k = 1$ | $k = 10$ | $k = 25$ | $k = 50$ | $k = 100$ |
|---|---|---|---|---|---|
| 50 | 0.04 | 0.43 | 0.91 | - | - |
| 100 | 0.05 | 0.53 | 1.18 | 2.35 | - |
| 150 | 0.15 | 1.56 | 3.77 | 7.52 | 15.03 |

**Distance Function Computation Times**

In the following we exemplify the amount of time needed to complete the parametrization computation and the process that discretizes distance functions and runs Dijkstra's algorithm: For the map of Nevada ($n = 280942$) with a resolution of $2048 \times 1024$, the computation of the parametrization takes 0.11 seconds and the computation of Dijkstra's algorithm plus 150 distance functions takes 39.64 seconds. This shows how the parallelization capabilities of graphic cards produce low computational times when computing distance functions, even for maps with a large number of nodes at high resolution. For the rest of experiments we use the map of Nevada with a 2048x1024 resolution.

## 4.2 Applications

**Nearest Network Diagrams**

In Table 2 we present the time needed to compute Order-$k$ Nearest Network Diagrams from already computed distance functions.

**Reverse Neighbor queries**

We used 150 randomly distributed sites in the map of Nevada and performed random $k$-Reverse Nearest Neighbor queries. We executed these queries for several values of $k$ and obtained that, once the times needed for distance function and Nearest Neighbor diagram computations had been subtracted, the time needed to answer the query was a constant 0.55 seconds. $k$-Reverse Nearest Neighbor queries are not affected by $k$ as for each query we only need to compute and read one distance field and can access the desired Order-$i$ Nearest Network diagrams directly.

**Bichromatic Reverse Neighbor queries**

In order to illustrate the way Bichromatic reverse neighbor queries, first of all we present the time needed for the computation of the necessary Order-$k$ Nearest Network diagrams from already computed distance functions.

Table 3: Bichromatic reverse neighbor queries.

| $k$ | $\|B\| = 150, \|R\| = 150$ | $\|B\| = 50, \|R\| = 150$ | $\|B\| = 150, \|R\| = 50$ |
|---|---|---|---|
| 10 | 1.58 | 1.59 | 0.45 |
| 20 | 3.14 | 3.14 | 0.87 |
| 30 | 4,60 | 4.60 | 1.19 |
| 40 | 6.26 | 6.26 | 1.72 |
| 50 | 7.71 | 7.70 | 2.03 |

Table 3 presents running times for sets of red and blue points of varying cardinality. We observe that the times presented in the first two columns are almost the same while the time in the third column is much lower. Thus, the time is determined mainly by the number of red points. This happens because for each blue point we only have to decide if it belongs to the Order-$i$ Nearest Network diagrams. Concerning the times needed to solve the Reverse Bichromatic problem once the necessary Order-$k$ Nearest Network diagrams are all around $3.5 \times 10^{-4}$ seconds. Consequently, the time needed to solve the problem is governed by the time needed to compute Order-$k$ Nearest Network diagrams.

# References

[1] E. Dijkstra, "A note on two problems in connection with graphs," *Numeriche Mathematik*, vol. 1, pp. 269–271, 1959.

[2] I. Fischer and C. Gotsman, "Fast approximation of high-order Voronoi diagrams and distance transforms on the GPU," *Journal of Graphics Tools*, vol. 11, no. 4, pp. 39–60, 2006.

[3] P. Sanders and D. Schultes, "Engineering fast route planning algorithms," in *WEA*, pp. 23–36, 2007.

[4] U.S. Census Bureau, Topologically Integrated Geographic Encoding and Referencing System, TIGER®, TIGER/Line® and TIGER®-Related Products. *http://www.census.gov/geo/www/tiger/.*

[5] D. Wagner and T. Willhalm, "Speed-up techniques for shortest-path computations," in *STACS*, pp. 23–36, 2007.

[6] M. L. Yiu, D. Papadias, N. Mamoulis, and Y. Tao, "Reverse nearest neighbors in large graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 4, pp. 540–553, 2006.

# Cálculo de la triangulación de Delaunay en la GPU *

Jon Valdés Furriel **

**Resumen**

En este trabajo presentamos una técnica simple y rápida para obtener triangulaciones Delaunay de nubes de puntos 2D, aprovechando las capacidades de las tarjetas gráficas actuales. El método se basa en el cálculo gráfico del diagrama de Voronoi y la posterior extracción de la triangulación a partir de éste.

## 1 Introducción

Las triangulaciones de Delaunay son una herramienta muy útil para generar mallas irregulares a partir de un conjunto de puntos. Existen buenos algoritmos para su cálculo que han sido implementados con éxito (véase, por ejemplo, la librería CGAL, `www.cgal.org`). Sin embargo, en general, las implementaciones que se emplean no utilizan la potencia y capacidad de las tarjetas gráficas. En el desarrollo de una aplicación informática para modelizar la evolución de bosques (*Vorest* [1]) se calculan diagramas de Voronoi de forma gráfica como medio de solventar el problema del cálculo de diagramas de Voronoi generalizados con pesos tanto aditivos como multiplicativos. Por esta razón, se decidió en el desarollo del proyecto emplear esa misma herramienta gráfica para calcular las mallas que dan soporte a los modelos digitales de los terrenos.

En este trabajo presentamos una técnica simple y rápida para obtener triangulaciones Delaunay-correctas de nubes de puntos 2D, aprovechando las capacidades de las tarjetas gráficas actuales. Esta técnica da buenos resultados en la generación de mallas irregulares de terrenos obtenidas, como es frecuente, de nubes de puntos densas y homogéneas.

El método consiste en el cálculo gráfico del diagrama de Voronoi para los puntos y la posterior extracción de la triangulación a partir de este, aprovechando que la triangulación de Delaunay es el grafo dual del diagrama de Voronoi. Existen sin embargo varios puntos donde el proceso no resulta trivial, para los que veremos a continuación las soluciones adoptadas. Se indican así mismo los puntos débiles del método y las líneas de trabajo futuras.

## 2 Construcción gráfica del diagrama de Voronoi

La técnica clásica (presentada en [3]) para la generación de diagramas de Voronoi de una nube de puntos de forma gráfica, permite generar una imagen en la que cada pixel contiene un color que representa al punto de la nube más cercano a ese pixel. Para ello se genera en tres dimensiones un cono por cada punto, y se dibuja en vista ortográfica superior, utilizando el algoritmo de Z-buffer para determinar qué cono resulta el más cercano a la cámara en cada pixel.

Esta técnica sin embargo, resulta problemática en el hardware actual, ya que es por lo general incapaz de dibujar superficies curvas como las de un cono. Es por esto que la técnica aproxima los conos utilizando triángulos, con la consiguiente pérdida de precisión en la superficie, y los problemas de rendimiento provocados por tener que aumentar la cantidad de polígonos por cada cono para reducir

*Parcialmente subvencionado por el proyecto Consolider Ingenio 2010 i-MATH C3-0159

**juanval@gmail.com

los errores en el Voronoi. La técnica aquí presentada, sin embargo, permite dibujar superficies que se comportan como conos perfectos en el Z-buffer, y que a pesar de todo presentan un rendimiento excelente.

Nuestra técnica consiste en dibujar un cuadrado en el lugar de cada cono y calcular, en cada pixel cubierto por el cuadrado, la profundidad exacta que tendría un cono perfecto. Para esto, utilizando un *fragment shader* en la tarjeta gráfica, se calcula en cada pixel la distancia desde ese pixel hasta el punto de la nube al que corresponde, y se asigna ese valor como profundidad Z del pixel. A continuación, al igual que en [3], el Z-buffer se encarga de seleccionar la superficie cuyo punto central sea el más cercano a ese pixel.

En la Figura 1, podemos ver arriba el método clásico y la imagen que genera, con imperfecciones en las aristas del diagrama de Voronoi. Debajo vemos las superficies que aparecen al calcular la profundidad de la superficie del cono en cada pixel, y cómo los diagramas de Voronoi generados no presentan más imperfecciones que las propias de la discretización.

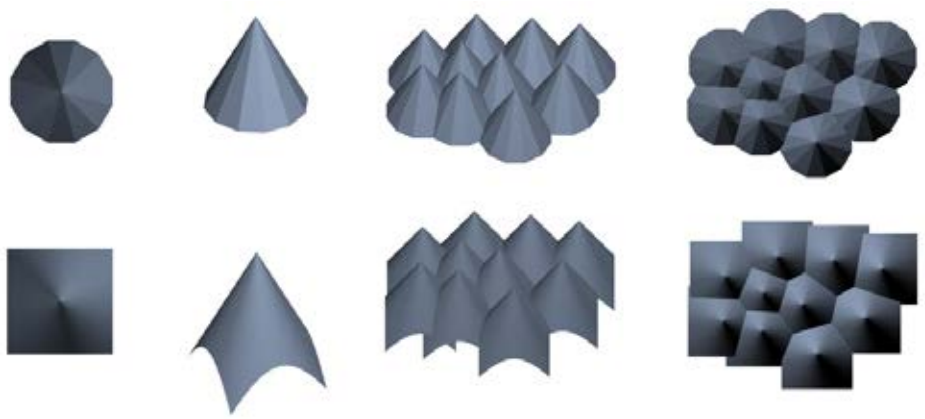


Figura 1: Arriba, el método clásico, con los conos triangulados, genera imperfecciones en el diagrama de Voronoi. Abajo, nuestro método genera superficies cónicas exactas en cada pixel, y el diagrama de Voronoi aparece libre de imperfecciones.

Por otro lado, las mayores texturas con las que puede trabajar una tarjeta gráfica actual son generalmente de 8192 pixels de lado, y esto puede ser insuficiente para representar el diagrama de Voronoi de una nube suficientemente grande. En este caso necesitaremos partir la imagen del diagrama en varias partes para conseguir suficiente resolución por cada celda.

Para esto, generamos varias imágenes contiguas en el espacio de los puntos, y a continuación las tratamos como secciones de una única imagen general.

# 3 Extracción de la triangulación

A continuación extraemos la imagen de la GPU, y realizamos el análisis del diagrama de Voronoi en la CPU.

A este efecto, en [2] se presenta una técnica para la extracción de la triangulación de Delaunay en 3D a partir de un diagrama de Voronoi 3D discretizado. Nuestra técnica es esencialmente la misma, trasladada a 2D.

Inicialmente localizamos en la imagen todos aquellos pixeles que tengan más de 1 color distinto al suyo propio entre sus 8-vecinos. A continuación, agrupamos estos pixeles seleccionados buscando las componentes 8-conexas entre ellos. Cada una de estas componentes conexas será un vértice del diagrama de Voronoi.

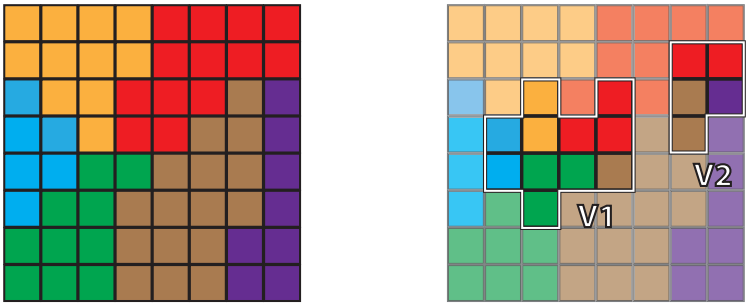



Figura 2: A la izquierda, imagen ampliada de un vértice del diagrama de Voronoi gráfico. A la derecha, extracción de las componentes conexas de los puntos vértice.

Finalmente, sabiendo que el diagrama de Voronoi es el grafo dual de la triangulación de Delaunay, cada uno de estos vértices generará una cara en la triangulación.

En la Figura 2 podemos ver un ejemplo de un diagrama de Voronoi gráfico en el cual se buscan las componentes 8-conexas de pixeles vértice, y se encuentran 2 vértices del diagrama de Voronoi: el vértice V1 aparece donde confluyen las caras roja, amarilla, marrón, verde y azul; y V2 aparece en la unión de las caras roja, marrón y morada. Esto supone que los puntos correspondientes a los colores de V1 formarán una cara en la triangulación de Delaunay, y los de V2 formarán otra.

Aquí aparece un problema evidente con los vértices del diagrama de Voronoi con más de 3 zonas incidentes, que suponen una cara del Delaunay de más de 3 lados. Esto sucederá siempre que existan más de 3 puntos situados en posición concéntrica. En ese caso, deberemos buscar por otro método una triangulación para estos puntos, pero su disposición obligatoriamente convexa permite simplificar considerablemente este proceso.

# 4 Resultados

Las triangulaciones obtenidas con este método, siempre que se cumplan las condiciones de densidad y resolución mínimas, son Delaunay-correctas en todos los casos probados.

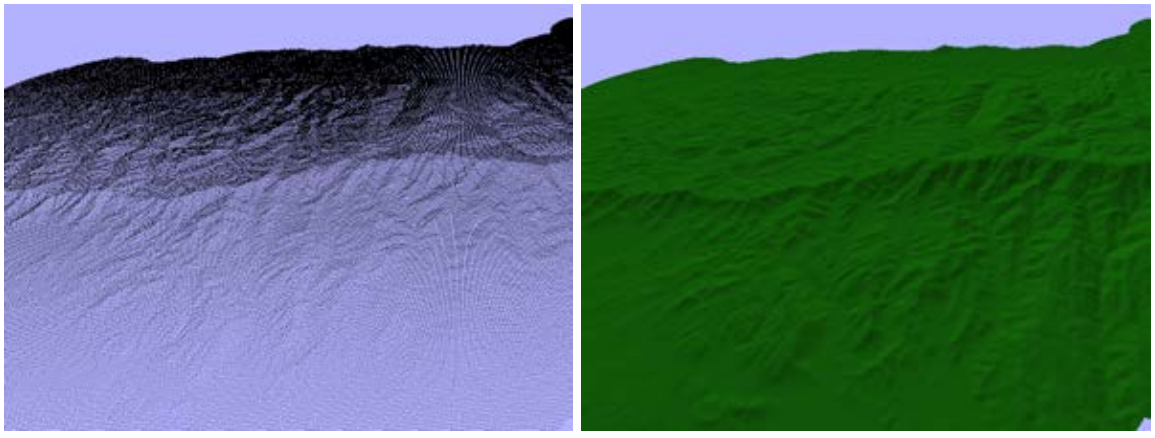


Figura 3: Nube de puntos de un terreno de la sierra de Filabres, y su correspondiente triangulación con normales e iluminación aplicadas.

En cuanto a rendimiento, en la máquina de pruebas (un Core 2 Duo a 2.4 GHz con 4GB RAM y una tarjeta gráfica GeForce 8600M GT), los tiempos encontrados para la nube de la Figura 3, de unos 260.000 puntos, son los siguientes:

1. Generación y extracción del diagrama de Voronoi y sus vértices: 3.93 segundos
2. Extracción de la triangulación a partir de los vértices del diagrama de Voronoi: 0.72 segundos

En total, 4.65 segundos para generar una triangulación con unos 530.000 triángulos.

Es importante tener en cuenta que éste es el caso óptimo para este algoritmo, con los puntos en disposición regular y uniforme. En otro caso sería necesario aumentar la resolución de las imágenes y el tamaño de los conos, lo que degradaría el rendimiento.

Existen así mismo varias oportunidades de optimización que no se han aprovechado, como la utilización de GPGPU para todo el proceso. Esto eliminaría la transmisión del diagrama de Voronoi entre la GPU y la CPU, y aceleraría también el proceso de las imágenes. En la implementación actual, se consumen aproximadamente 1.6 segundos por cada imagen en estos dos pasos.

## 4.1 Limitaciones del algoritmo

Este método presenta ciertas limitaciones para casos generales, tal y como veremos a continuación.

En primer lugar, la discretización del diagrama de Voronoi da lugar a problemas siempre que la resolución de la imagen no sea suficiente para mostrar con detalle la estructura de éste.

En nuestro caso, simplemente detectamos si dos vértices resultan demasiado cercanos en la imagen usando una técnica de bucketing, y solucionamos esto aumentando la resolución del diagrama de Voronoi mientras no desaparezca el problema. En caso de tener una nube de densidad no uniforme, este método presenta el problema de renderizar zonas a resolución innecesariamente alta. Métodos más sofisticados podrían reducir las zonas calculadas a alta resolución, mejorando el rendimiento notablemente.

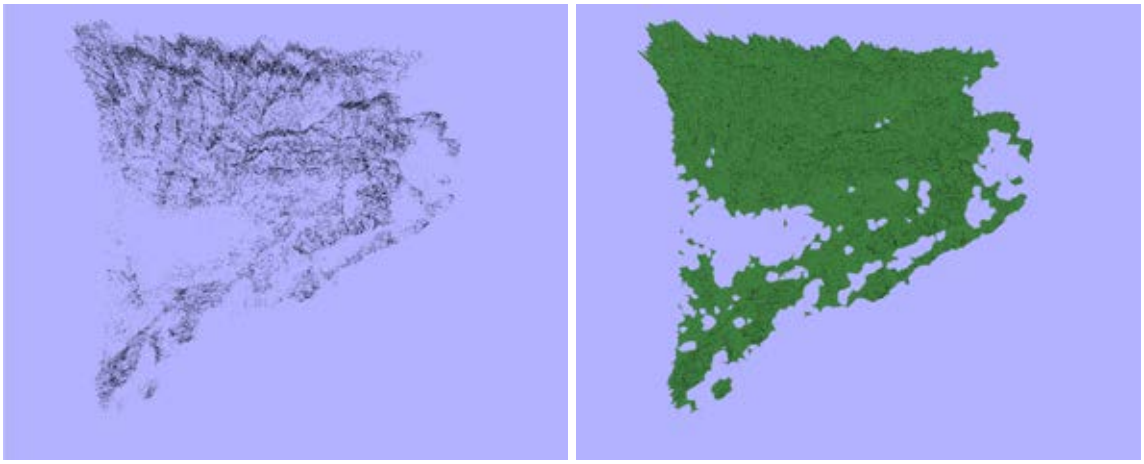


Figura 4: Aquí vemos una nube con zonas poco densas, que provocan el fallo en la triangulación.

Por otro lado, en la práctica resulta inviable asegurar que la triangulación encontrada es la triangulación de Delaunay completa de los puntos, ya que para esto los cuadrados utilizados deberían tener tamaño infinito. Esto supone que la triangulación encontrada siempre será Delaunay-correcta en el ambiente local definido por el tamaño de los cuadrados. Así mismo, la adyacencia (y la triangulación) entre puntos a mayor distancia que esta, no será encontrada por el algoritmo. Podemos ver un ejemplo de esto en la Figura 4

En la Figura 5 se aprecia un ejemplo de un diagrama de Voronoi que presenta ambos problemas, causados por una nube de puntos con zonas de densidades muy dispares. Mientras en las zonas densas es imposible generar una triangulación correcta, en las poco densas no es posible siquiera generar triangulación alguna.

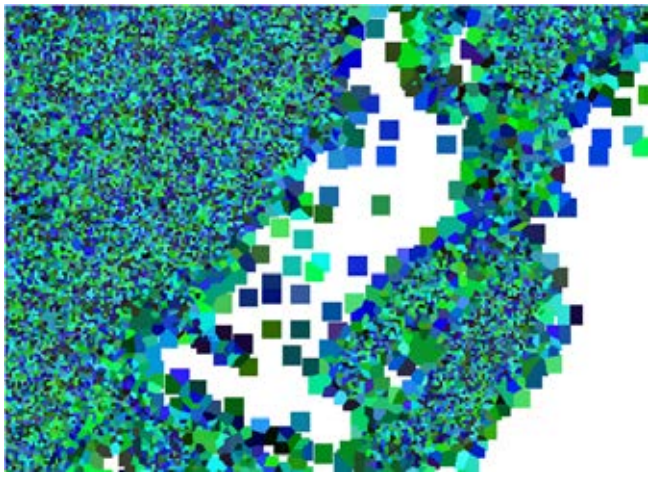


Figura 5: Las zonas de muy alta o baja densidad resultan problemáticas en el diagrama de Voronoi gráfico.

## 5 Conclusión y trabajo futuro

Este método presenta excelentes cualidades para mallas de densidad uniforme, consiguiendo un rendimiento comparable a otros métodos de triangulación. Esto es así a pesar de no haber utilizado el propio hardware gráfico para el análisis de las imágenes, lo que podría haber aumentado el rendimiento considerablemente.

Por el contrario, trabajando con mallas de densidad variable resulta necesario aumentar la resolución para que la celda más pequeña del diagrama de Voronoi tenga un número suficiente de pixeles, así como aumentar el tamaño de los cuadrados para que sean capaces de contener las mayores regiones del Voronoi. Esto provocará irremediablemente la pérdida de rendimiento en el proceso.

## 6 Agradecimientos

Este trabajo forma parte del proyecto *GeoVys* (Consolider Ingenio 2010 i-MATH C3-0159) desarrollado en colaboración entre las universidades Politécnica de Madrid y de Valladolid y dirigido por Manuel Abellanas.

## Referencias

[1] Begoña Abellanas, Manuel Abellanas, Carlos Vilas, *VOREST: Modelización de bosques mediante diagramas de Voronoi* , Actas de los XII Encuentros de Geometría Computacional, Universidad de Valladolid, junio 2007. 249-256.

[2] Boltcheva, D.; Bechmann, D.; Thery, S.; 2007 *Discrete Delaunay: Boundary extraction from voxel objects.* 3-D Digital Imaging and Modeling, 2007. 3DIM '07. Sixth International Conference on 21-23 Page(s):209 - 216

[3] Hoff,III, Kenneth E. and Keyser,, John and Lin,, Ming and Manocha,, Dinesh and Culver,, Tim. 1999 *Fast computation of generalized Voronoi diagrams using graphics hardware.* In SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, New York, 277–286. Computer Graphics Proceedings, Annual Conference Series, ACM.

[4] Rong,, Guodong and Tan,, Tiow-Seng and Cao,, Thanh-Tung and Stephanus, 2008 *Computing two-dimensional Delaunay triangulation using graphics hardware* I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games, Pages: 89–97, ACM.

# V-Proportion: A Method Based on the Voronoi Diagram to Study Spatial Relations Between Neuronal Mosaics

Oscar M. Mozos* Jose A. Bolea† Eduardo Fernandez‡ Peter K. Ahnelt§

**Abstract**

The study of the spatial relations between neural populations has shown its importance to investigate possible constrains or connectivities between different cell types. In this paper we present the application of the Voronoi diagram to detect possible spatial relations between cells.

## 1 Introduction

The study of the spatial relations between neural populations has shown its importance to investigate possible constrains or connectivities between different cell types. A positive spatial dependencies between two different populations of neurons may be an indicator of some connection patterns between them [1, 5, 4, 2]. The spatial information is also helpful to study dependencies during development [10].

Different methods to study spatial relations between different cell populations have been proposed in the past. The work in [14] extends the density recovery profile method [17] to study interdependences between different S-cone cells. In [7], the author uses the $K$-function [16] to analyze the spatial relation between *on* and *off* cells in the eye retina. The same cells are studied in [8] by fitting a bivariate pairwise interaction model in the data. Spatial relations between beta cells are studied in [20] using K-nearest neighbor histograms. This data was also analyzed in [19] using the $J$-function.

This paper introduces a method to study spatial relations between two cell populations. The key idea is to calculate the Voronoi diagram of one of the populations, and then study the spatial distribution of cells in the second population inside the polygons of the diagram. Using polygonal areas around neural cells is a realistic approximation, since neurons present different irregularities in their structure [11]. In this work we extend preliminary ideas and results presented in [2].

## 2 The Voronoi Diagram

The Voronoi diagram [3] is one of the most useful geometrical constructions to study point patterns because it provides all the information needed to study the proximity relations in a set of points.

We define a Voronoi diagram as follows. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points in the two dimensional Euclidean plane. We refer to these points as *reference points*. Now we assign every other point in the plane to its nearest reference point $p_i$. This process creates a tessellation of the plane into (sometimes unbounded) convex polytopes, also called Voronoi polygons. Let $V(p_i)$ be the Voronoi polygon corresponding to reference point $p_i$, then all the points inside this polygon are at least as close to $p_i$ as to any other reference point $p_j$

$$V(p_i) = \{p \mid d(p_i, x) \leq d(p_j, x),\ \forall p_i \neq p_j\},$$

*Department of Computer Science and System Engineering, University of Zaragoza, Spain. ommozos@unizar.es
†Bioengineering Institute, Miguel Hernandez University, Spain
‡Bioengineering Institute, Miguel Hernandez University, Spain
§Department of Physiology, Medical University of Vienna, Austria

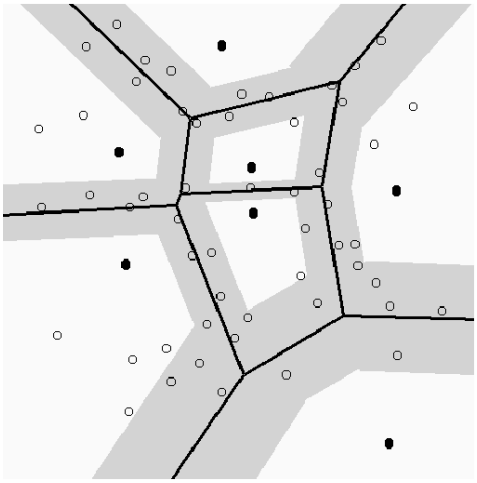


Figure 1: Example of a Voronoi diagram (black lines) for a set of reference points (black circles). Bands around the edges of the Voronoi diagram are shown in gray. In this particular case $\delta = 0.29$, indicating the 29% of the distance between a Voronoi edge and its closest reference point.

with $d(x, y)$ representing the Euclidean distance function.

The edges of the Voronoi regions form the Voronoi diagram $V(P)$ of a the reference points set. Note that a point on the edges of the Voronoi diagram has two nearest neighbors and each vertex has at least three. An example of a Voronoi Diagram is shown in Figure 1.

## 3 The V-Proportion Measurement

Let's suppose we want to study the possible spatial interaction between two cell populations denoted as $P$ and $Q$. In principle, we want to study whether the cells in the $Q$ population have a tendency to be far from the cells in $P$. In this case, the average density of $q$-cells will be higher near the edges of the Voronoi diagram than in the surrounding of the $p$-cells (Voronoi polygons). An example is given in Figure 1, where $p$-cells are shown as black circles (reference points), and $q$-cells as white circles.

To quantify this far concept we construct bands around the edges of the Voronoi diagram with a specified width (gray areas in Figure 1). If both populations, $P$ and $Q$ are spatially independent, then the average density of $q$-cells should be the same in any region inside the Voronoi polygons. If there exists an inhibition between the $P$ and $Q$ populations, then the density of $q$-cells should be higher in the band zone. On the contrary, if there exists an attraction, then the density of $q$-cells should be higher close to $p$-points and far from the edges. A way to quantified these spatial relations can be obtaining by looking at the relative proportion between $q$-cells inside the bands $n_b$ and the total number of $q$-cells inside the Voronoi polygons $n_{vp}$

$$\text{V-proportion} = \frac{n_b}{n_{vp}}. \tag{1}$$

If the measured V-proportion is higher than expected, then there should be more points inside the bands, suggesting a negative correlation. If the measured V-proportion is lesser than expected, then a positive correlation can be assumed.

An important issue in this method is the determination of the band widths. To determine this value, we have to take into account that distances between the reference points $p_i$ and the edges of their respective Voronoi polygon $V(p_i)$ are usually different for each reference point. As a result, polygons with different sizes and shapes are obtained. This effect mainly appears in neuron populations, and it is due to dendritic tree irregularities [11, 2]. To solve this problem a non-fixed value for the width in each band is used. This value is represented by the parameter $\delta$, $0 < \delta < 1$, and indicates the proportion of the distance between each of the edges of $V(p_i)$ and the reference point $p_i$ . As an example, a value

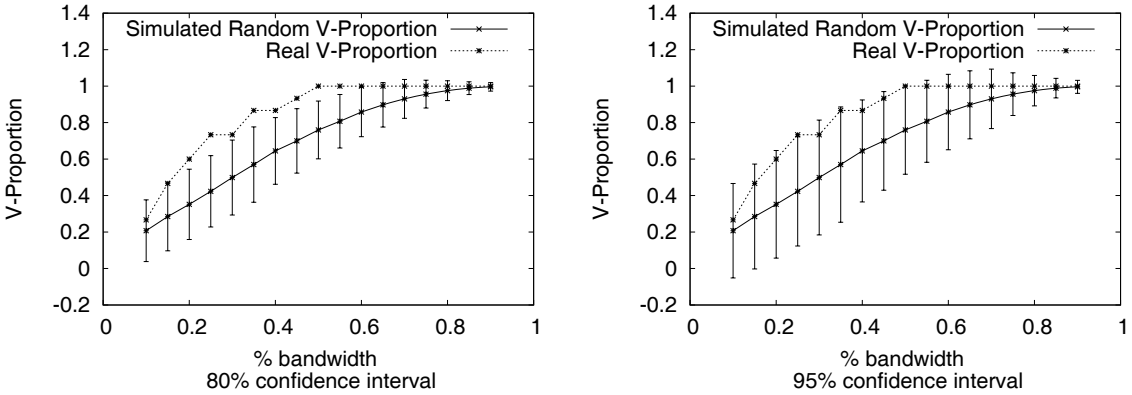



Figure 2: The plots shows the V-proportion for the populations of Figure 1 in comparison with the simulated ones. The V-proportion is calculated for different band widths. Errors bars on the left plot are at the 80% confidence interval, while on the right plot are at the 95% confidence interval.

of $\delta = 0.29$ means that the width of the band for one edge must be 29% of the distance between that edge and the reference point (Figure 1).

It remains to explain how to quantify the expected values to be compared with Expression 1. To test the significance of the V-proportion, we use a Monte Carlo test procedure. This involves generating a set of two random and independent patterns, each with the same number of points as the two empirical populations $P$ and $Q$, and in a study area identical to that of these patterns. We repeat the generation $N$ times for each $\delta$, and then the mean and the standard deviation of the V-proportion for each simulation are calculated. The resulting plot is compared with the V-proportion obtained with the two real $P$ and $Q$ populations. Whenever the real V-proportion raises above the random equivalent simulation, then we can assume a negative interaction between patterns, that is, $P$ inhibits $Q$. If the real V-proportion is below the simulated one, then a clustering process occurs in which $P$ cells attract $Q$ cells. If the real V-proportion is close to the simulated one, we can not assume any spatial interaction between the two populations.

Finally, an important problem when working with point populations is the edge effect. Typically, the observation of the two point patterns is restricted to a regular sampling window, while usually the patterns extend beyond the window. Furthermore, the Voronoi polygons on the boundary of the area are open because they have no neighboring points in those directions. This edge effect affects the sense of bands in the Voronoi polygons adjacent to the edges. In our case, we solve this problem by removing the Voronoi polygons intersecting the edges of the sampling area (open polygons). Similar approaches have been applied in other works for studying cell mosaics [13, 9]. Although this heuristic reduces the total number of points used for the study, we think the results represent the spatial relations in a more reliable manner.

An example of this process is shown in Figure 2. Here the V-proportion for the populations in Figure 1 is calculated, where $P$ cells correspond to black circles and $Q$ cells to white ones. We generated 100 simulations for the two populations. The simulations points were distributed randomly in the space. As both plots in Figure 2 indicate, the V-proportion of the two populations is always above the simulated values. This indicates that the $Q$ population tends to be inhibited by the $P$ population. In the left plot of Figure 2, we can see that the V-proportion values fall most of the time above the 80% confidence interval. That means that the negative interaction between cells is significant at the 0.20 level. However, this difference is no more significant at the 0.05 level (95% confidence interval) as shown in the right plot in Figure 2.

# 4 Results

This section presents experimental results that validate our approach. We first show experiments in simulated populations that follow different spatial distributions. We then study the spatial relations

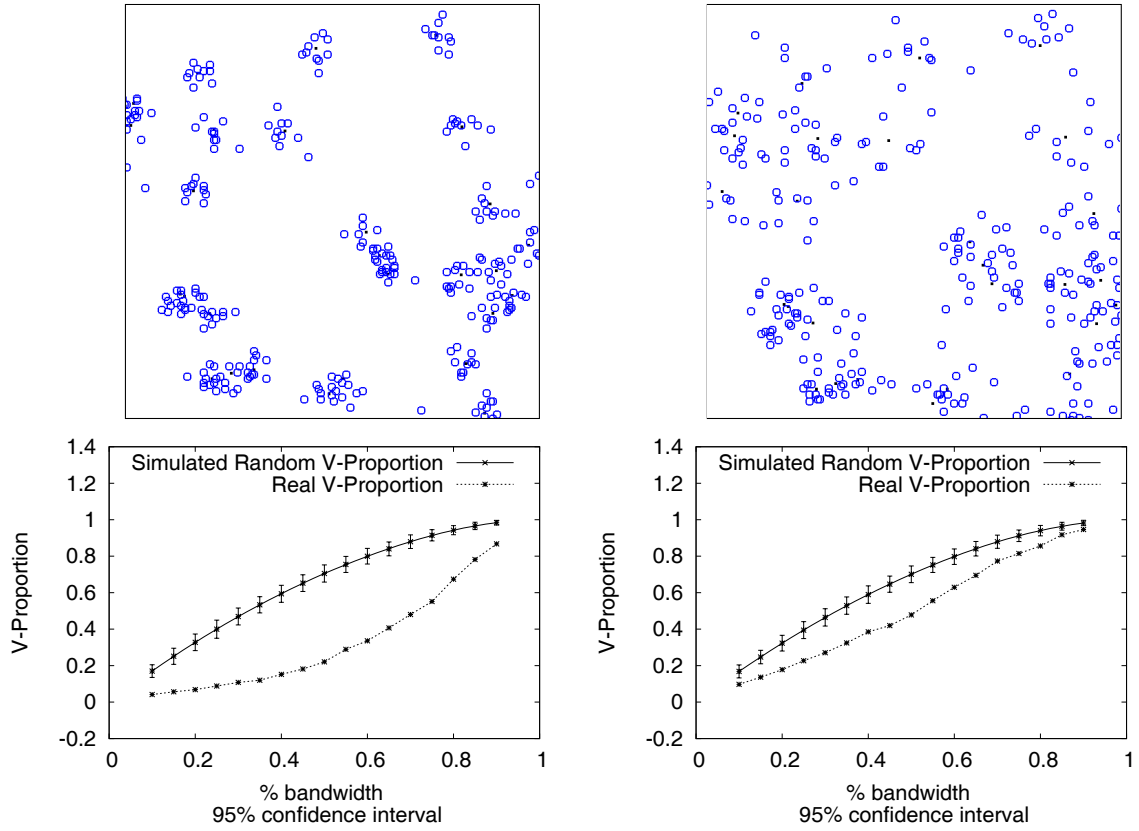



Figure 3: Two Poisson clusters (upper images) with different values of $\sigma^2$ in pixel units (left $\sigma^2 = 25$, right $\sigma^2 = 100$). The area of the original image was $300 \times 300$ pixels. $P$ points are shown in black, whereas $Q$ points are depicted in blue circles. Both clustering behaviors are clearly detected.

between real cell populations that have been previously studied in the literature.

## 4.1 Simulated Populations

In the following experiments we generated pairs of point patterns following a mathematical probability distribution, and then we calculated their V-proportion. As explained in Section 3, we use two set of populations. The $P$ population will be the reference one, and the $Q$ population will be the one to study its spatial distribution.

We first study clusters behaviors of $Q$ points with respect $P$ points To generate the both patterns, we simulate the Poisson cluster process presented in [15, 6] as follows. Each parent point $p$ is generated following a bidimensional Poisson process. For each $p$-point, we generate a set of $q$ offspring points. The positions of the $q$-point relative to their parents are independent and identically distributed according to a bidimensional normal distribution. Details on the simulations can be found in [18]. We generate different pairs of $P$ and $Q$ populations with 50 and 300 points respectively. The points are contained in an image area of $300 \times 300$ pixels. For each pair of populations, we change the variance $\sigma^2$ of the offsprings $q$-points. Example simulations are shown in Figure 3 together with their V-proportions. In these plots the interval confidence represented by the error bars is of 95%. As we can see the clusters are detected by the V-proportion plots. A way to determine the level of clustering could be calculating the area under the curve between the random simulations and the real patterns.

In a second simulation we generate a $Q$ population in which each point can not be closer to a $P$ point less than a certain radius (80 pixels). Details on the simulation are found in [12]. Results are shown in Figure 4. The plot clearly indicates the negative correlation between the populations at the 95% confidence interval. In a third simulation, we generate independent Poisson processes for both $P$ and $Q$ populations. Part of the populations and V-proportions are shown in Figure 5. The plot

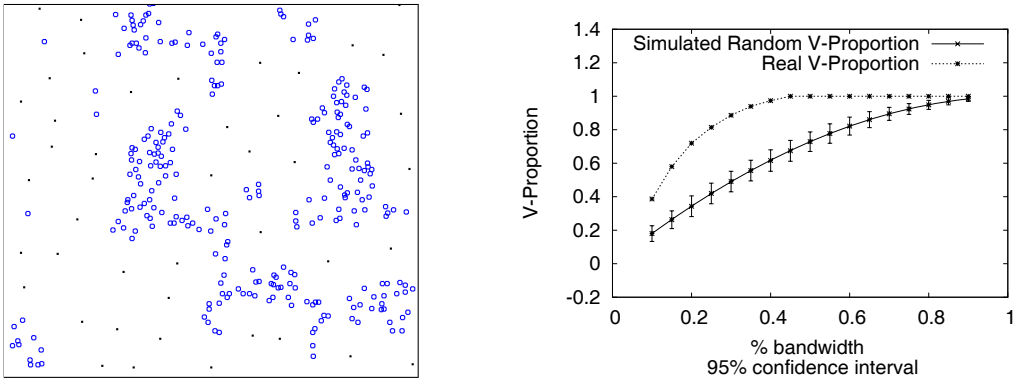



Figure 4: The left image depicts the populations $P$ and $Q$ generated following a repulsion model [12]. The right plot clearly indicates a negative spatial relation between both populations.

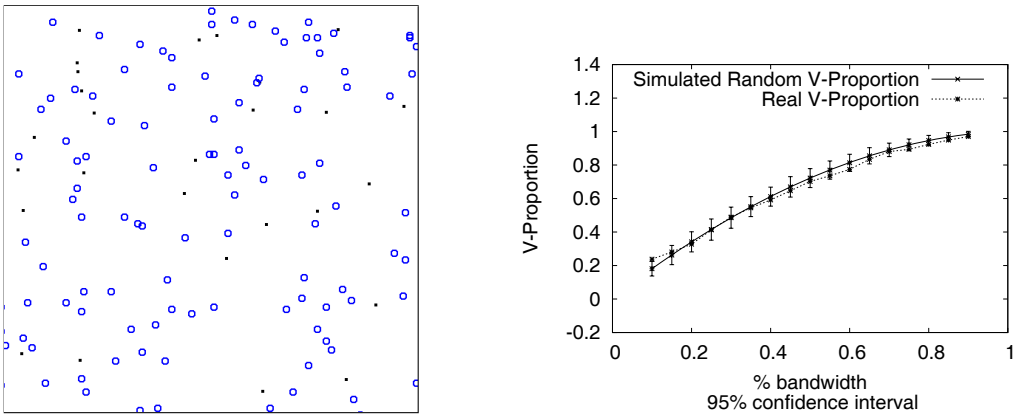



Figure 5: The left image depicts the populations $P$ and $Q$ generated following two independents Poisson point processes. The right plot does not indicate any spatial relation between both populations.

correctly indicates that we can not describe any positive or negative spatial relation between the two populations.

In summary, the previous experiments clearly show that the V-proportion value is able to detect cluster and repulsion behaviors, as well as the lack of them.

## 4.2 V-proportion in real cell populations

In this section we apply the V-proportion measurement to real populations of cells. In addition, we compare the results with previous works analyzing the same patterns.

### 4.2.1 Beta cells in the cat retina

The left image in Figure 6 depicts a pattern of beta-type ganglion cells in the retina of a cat [20]. Beta cells are associated with the resolution of fine detail in the cat's visual system. They can be classified as *on* or *off*, depending on the branching level of their dendritic tree in the inner plexiform layer. Analysis of the spatial pattern provides information on the cat's visual discrimination. In particular, independence of the *on*- and *off*-components would strengthen the assumption that there are two separate channels for *brightness* and *darkness*. More details can be found in [20].

In [20] this pattern is investigated using histograms of nearest-neighbor distances (ignoring edge effects). To test independence of the *on* and *off* patterns, a random translation of the *off*-component was superimposed on the *on*-component, and the resulting nearest-neighbor histogram was compared

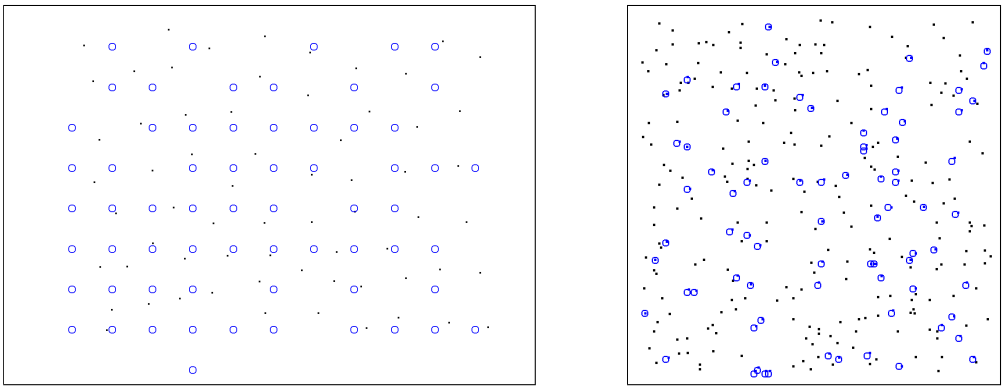


Figure 6: the left image shows the superimposed patterns of *on* and *off* cells in the cat retina. In the right image we can see the cell nuclei of the hamster's kidney.

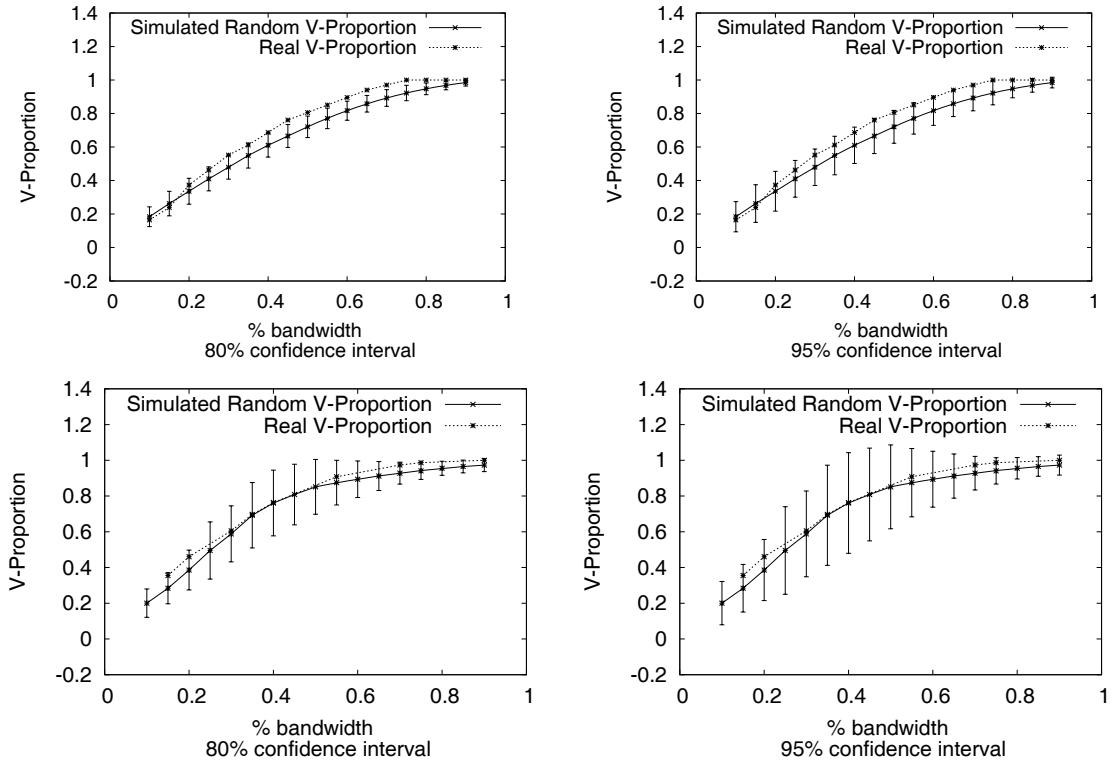



Figure 7: The top plots depicts the V-Proportion for the superimposed patterns of *on* and *off* cells in the cat retina (left image in Figure 6). The repulsion between populations is slightly significant at the 80% confidence level. However, this negative relation is not significant at the 95% confidence level. The bottom plots present the V-proportion for the nuclei of a hamster (right image in Figure 6). Both plots do not indicate neither positive or negative spatial relations between the two cell populations.

with the original one by a sign reversal test. The authors in [20] concluded that both types of beta cells form a regular lattice, which are superimposed independently. This data was also analyzed in [19] using the $J$-function with a result greater than 1, which confirms conclusions of [20]. Our results using the V-proportion are shown in the top plots of Figure 7. Using an confidence interval of 95% we cannot assume a significant negative spatial relation between *on* and *off* cells. However, at a confidence interval of 80% we can see a slightly negative repulsion. This small negative effect was not indicated in previous works analyzing this pattern.

#### 4.2.2 Hamster tumor

The right image in Figure 6 shows the positions of cell nuclei in approximately $0.25mm$ square histological sections of tissue from laboratory-induced metastasising lymphoma in the kidney of a hamster. Two types of cells are distinguished: 77 pyknotic nuclei corresponding to dying cells, and 226 nuclei arrested in metaphase. In [6] this pattern was studied and the random labeling hypothesis using the $K$-function was accepted. Results in [19] agreed on the random hypothesis using the $J$-function. In our analysis, we support the independence between both populations at the 80% and 95% confidence level, as shown in the bottom plots in Figure 7.

## 5 Conclusion

This paper introduced a method to evaluate spatial relations between different neural cells. Our approach is based on the Voronoi diagram and considers the relation between the density of points inside Voronoi polygons and bands around it edges. The method has been tested in simulated populations as well as in real ones. Results demonstrate that our measurement can be used to detect different spatial relations between populations, such as repulsions, attractions or none of them.

## References

[1] P.K. Ahnelt and H. Kolb, Horizontal cells and cone photoreceptors in primate retina: a Golgi-light microscopic study of spectral connectivity, *The Journal of Comparative Neurology*, vol. 343, pp. 387–405, May 1994.

[2] P.K. Ahnelt, E. Fernandez, O. Martinez, J.A. Bolea, and A. Kueber-Heiss, Irregular S-cone mosaics in felid retinas. Spatial interaction with axonless horizontal revealed by cross-correlation, *Journal of the Optical Society of America A*, vol. 17(3), pp. 580–588, March 2000.

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, third edition, 2008.

[4] T.L. Chan, and U. Gruenert, Horizontal cell connections with short wavelength-sensitive cones in the retina: a comparison between New World and Old World primates, *The Journal of Comparative Neurology*, vol. 393(2), pp. 196–209, April 1998.

[5] D.M. Dacey, B.B. Lee, D.K. Stafford, J. Pokorny, and V.C. Smith, Horizontal cells of the primate retina: cone specificity without spectral opponency, *Science*, vol. 271(5249), pp. 656–659, February 1996.

[6] P. Diggle, *Statistical analysis of spatial point patterns*, Academic Press 1983.

[7] P.J. Diggle, Displaced amacrine cells in the retina of a rabbit: analysis of a bivariate spatial point pattern, *Journal of Neuroscience Methods*, vol. 18(1-2), pp. 115–125, October 1986.

[8] P.J. Diggle, S.J. Eglen, and J.B. Troy, *Case Studies in Spatial Point Process Modelling*, ch. Modelling the bivariate spatial distribution of amacrine cells, pp. 215–233, Springer, 2006.

[9] C. Duyckaerts, and G. Godefroy, Voronoi tessellation to study the numerical density and the spatial distribution of neurones, *Journal of Chemical Neuroanatomy*, vol. 20(1), pp. 83–92, October 2000.

[10] S.J. Eglen, Development of regular cellular spacing in the retina: theoretical models, *Mathematical Medicine and Biology*, vol. 23(2), pp. 79–99, June 2006.

[11] E. Fernandez, J. A. Bolea, G. Ortega, and E. Louis, Are neurons multifractals?, *Journal of Neuroscience Methods*, vol. 89(2), pp. 151–157, July 1999.

[12] E. Fernandez and N. Cuenca and J. De Juan, A compiled BASIC program for analysis of spatial point patterns: application to retinal studies, *Journal of Neuroscience Methods*, vol. 50(1), pp. 1–15, October 1993.

[13] L. Galli-Resta, E. Novelli, Z. Kryger, G.H. Jacobs, and B.E. Reese, Modelling the mosaic organization of rod and cone photoreceptors with a minimal-spacing rule, *The European Journal of Neuroscience*, vol. 11(4), pp. 1461–1469, April 1999.

[14] N. Kouyama, and D.W. Marshak, The topographical relationship between two neuronal mosaics in the short wavelength-sensitive system of the primate retina, *Visual Neuroscience*, vol. 14(1), pp. 159–167, January 1997.

[15] J. Neyman and E.L. Scott, A statistical approach to problems of cosmology, *Proceedings of the Royal Society of London. Series B (Methodological)*, vol. 20(1), pp. 1–43, 1958.

[16] B.D. Ripley, The Second-Order Analysis of Stationary Point Processes, *Journal of Applied Probability*, vol. 13(2), pp. 255–266, June 1976.

[17] R.W. Rodieck, The density recovery profile: a method for the analysis of points in the plane applicable to retinal studies, *Visual Neuroscience*, vol. 6(2), pp. 95–111, February 1991.

[18] S. M. Ross, *Simulation*, Academic Press, 1997.

[19] M.N.M. van Lieshout, and A.J. Baddeley, Indices of Dependence Between Types in Multivariate Point Patterns, *Scandinavian Journal of Statistics*, vol. 26, pp. 511–532, 1999.

[20] H. Wassle, B. B. Boycott, and R.-B. Illing, Morphology and mosaic of on- and off-beta cells in the cat retina and some functional considerations, *Proceedings of the Royal Society of London. Series B (Biological Sciences)*, vol. 212(1187), May 1981.

# Martes, 30 de junio

# Sesión 3, 10:00–11:00

# Maximum Sets of $(k, \overline{k})-$Connected Digital Surfaces *

J. C. Ciria[†], E. Domínguez[†], A. R. Francés[†] and A. Quintero [‡]

**Abstract**

For each adjacency pair $(k, \bar{k}) \neq (6, 6)$ with $k, \bar{k} \in \{6, 18, 26\}$, we find a regular $(k, \bar{k})$-connected digital space $E^{U}_{k,\bar{k}}$ whose set $S_{k,\bar{k}}$ of digital surfaces (i.e., digital objects whose continuous analogue is a combinatorial surface) is the largest in that class of spaces. In particular, several families of $(k, \bar{k})$-surfaces reported in literature within the graph-theoretical approach to Digital Topology are shown to be contained in $S_{k,\bar{k}}$.

## 1 Introduction

In the graph–theoretical approach to Digital Topology, the search for a definition of digital surfaces as subsets of voxels has been carried out since the 80's. Their applications range from visualization to image segmentation [10, 13] and graphics [14]. Despite these applications, it is not yet well established a general notion that naturally extends to higher dimensions (see [5] for a proposal). Instead, since the first definition of surface, proposed by Morgenthaler [15] for the discrete space $\mathbb{Z}^3$ provided with the usual adjacency pairs $(26, 6)$ and $(6, 26)$, each new proposal [11, 4, 9] have either increased the number of surfaces or extended the definition to other adjacencies, but still leave out some objects considered as surfaces for practical purposes [13].

In this paper we find, for each adjacency pair $(k, \overline{k})$, $k, \overline{k} \in \{6, 18, 26\}$ and $(k, \overline{k}) \neq (6, 6)$, a set of digital surfaces larger than those quoted above, that in addition are the largest possible in a certain sense. We obtain these sets of surfaces within the framework for Digital Topology we proposed in [3]. In this framework the voxels of an image are represented by the $3-$cells of a polyhedral complex $K$ and, following Kovalevsky [12], the connectivity paradoxes of the graph–theroetical approach are avoided by assigning to the foreground of the image a subset of the lower dimensional cells of $K$. In this setting, a digital space is defined as a pair $(K, f)$ where $f$ is a function that performs this assignment to each image according to the same rule. Going further than Kovalevsky, from this set we associate an Euclidean polyhedron, called *continuous analogue*, to the image. This way, not only connectivity, but also other topological notions can be naturally defined on digital images. In particular, a digital surface is any object $S$ whose continuous analogue is a (combinatorial) surface; that is, if $S$ "looks like" a surface.

This definition of digital surface has two main advantages: it readily generalizes to the notion of digital manifold in higher dimensions; and, furthermore, digital counterparts of properties of topological surfaces can be shown for these digital surfaces from the corresponding continuous results (see [3, 2] for a digital Jordan–Brouwer and Index theorems for digital manifolds). Moreover, it is also closely related to the graph–based definitions quoted above. More precisely, for each one of those families of surfaces defined on $\mathbb{Z}^3$ it is found [1, 3, 6] a particular digital space $(R^3, f_{k,\overline{k}})$, resembling the $(k, \overline{k})-$connectedness, whose set of digital surfaces coincides or contains the corresponding family, where the complex $R^3$ is the standard decomposition of the Euclidean space $\mathbb{R}^3$ by unit cubes, which is canonically identified with $\mathbb{Z}^3$.

*This work has been partially supported by the project MTM2007–65726 MICINN

[†]Dpto. de Informática e Ingeniería de Sistemas Facultad de Ciencias, Universidad de Zaragoza, E-50009 – Zaragoza

[‡]Dpto. de Geometría y Topología, Facultad de Matemáticas, Universidad de Sevilla, Apto. 1160, E-41080 – Sevilla

From these contributions we reach our claim showing, in our main result, that, for each adjacency pair $(k,\overline{k})$, there exists a "universal" space $(R^3, f^U_{k,\overline{k}})$ whose set of digital surfaces is the largest within the class of spaces resembling the $(k,\overline{k})-$connectedness. This result, which extends a simpler one in [7] for the $(26,6)-$adjacency, will allow us in a future work to characterize these surfaces just in terms of voxels, without using the lower dimensional cells of the complex $R^3$, now needed to built their continuous analogues.

## 2 A framework for Digital Topology

This section summarizes the framework for Digital Topology we proposed in [3]. In this approach a *digital space* is a pair $(K, f)$, where $K$ is a polyhedral complex, representing the spatial layout of voxels, and $f$ is a *lighting function* from which we associate to each digital image an Euclidean polyhedron, called its continuous analogue, that intends to be a continuous interpretation of the image.

In this paper we will only consider spaces of the form $(R^3, f)$, where the complex $R^3$ is determined by the collection of unit cubes in the Euclidean space $\mathbb{R}^3$ centered at points of integer coordintes. Each 3-cell in $R^3$ is representing a voxel, and so the digital object displayed in an image is a subset of the set $\mathrm{cell}_3(R^3)$ of 3-cells in $R^3$; while the other lower dimensional cells in $R^3$ (actually, $k$-cubes, $0 \leq k < 3$) are used to describe how the voxels could be linked to each other. Notice that each $k$-cell $\sigma \in R^3$ can be associated to its center $c(\sigma)$. In particular, if $\dim \sigma = 3$ then $c(\sigma) \in \mathbb{Z}^3$, so that every digital object $O$ in $R^3$ can be naturally identified with a subset of the discrete space $\mathbb{Z}^3$. Henceforth we shall use this identification without further comment.

As it is usual, given two cells $\gamma, \sigma \in R^3$ we write $\gamma \leq \sigma$ if $\gamma$ is a face of $\sigma$, and $\gamma < \sigma$ if in addition $\gamma \neq \sigma$. The interior of a cell $\sigma$ is the set $\mathring{\sigma} = \sigma - \partial\sigma$, where $\partial\sigma = \cup\{\gamma\,;\, \gamma < \sigma\}$ stands for the boundary of $\sigma$. We refer to [16] for further notions on polyhedral topology.

A lighting function is just a "face membership rule" as defined by Kovalevsky [12] that assigns to each digital object $O$ a subset of cells $f_O \subseteq R^3$ of all dimensions. Given a lighting function $f$, the *continuous analogue* of $O$ is the underlying polyhedron $|\mathcal{A}^f_O|$ of the order complex $\mathcal{A}^f_O$, called its *simplicial analogue*, whose $k$-simplexes are $\langle c(\alpha_0), c(\alpha_1), \ldots, c(\alpha_k)\rangle$, where $\alpha_0 < \alpha_1 < \cdots < \alpha_k$ are cells in $f_O$. We often drop the "$f$" from the notation and write $\mathcal{A}_{R^3}$ instead $\mathcal{A}_{\mathrm{cell}_3(R^3)}$.

Continuous analogues intend to be a "continuous interpretation" of digital objects. For example, we say that an object $O$ is *connected* in a digital space $(R^3, f)$ if $|\mathcal{A}_O|$ is a connected polyhedron. However, to avoid continuous analogues which are contrary to our usual topological intuition, we require to our lighting functions five properies [2, 3]. We need some more notation to introduce them.

Given a cell $\alpha \in R^3$ and a digital object $O \subseteq \mathrm{cell}_3(R^3)$ the *star of $\alpha$ in $O$* is the set $\mathrm{st}_3(\alpha; O) = \{\sigma \in O\,;\, \alpha \leq \sigma\}$ of 3-cells (voxels) in $O$ having $\alpha$ as a face. Similarly, the *extended star of $\alpha$ in $O$* is the set $\mathrm{st}^*_3(\alpha; O) = \{\sigma \in O\,;\, \alpha \cap \sigma \neq \emptyset\}$. Finally, the *support* of $O$ is the set $\mathrm{supp}(O)$ of cells of $R^3$ (not necessarily voxels) that are the intersection of 3-cells in $O$; that is, $\alpha \in \mathrm{supp}(O)$ if and only if $\alpha = \cap\{\sigma\,;\, \sigma \in \mathrm{st}_3(\alpha; O)\}$. To ease the writing, we use the following notation: $\mathrm{st}_3(\alpha; R^3) = \mathrm{st}_3(\alpha; \mathrm{cell}_3(R^3))$ and $\mathrm{st}^*_3(\alpha; R^3) = \mathrm{st}^*_3(\alpha; \mathrm{cell}_3(R^3))$. Finally, we write $\mathcal{P}(A)$ for the family of all subsets of a given set $A$.

A *lighting function* on $R^3$ is a map $f : \mathcal{P}(\mathrm{cell}_3(R^3)) \times R^3 \to \{0, 1\}$ satisfying the following five axioms for all $O \in \mathcal{P}(\mathrm{cell}_3(R^3))$ and $\alpha \in R^3$:

(1) *object axiom*: if $\alpha \in O$ then $f(O, \alpha) = 1$;

(2) *support axiom*: if $\alpha \notin \mathrm{supp}(O)$ then $f(O, \alpha) = 0$;

(3) *weak monotone axiom*: $f(O, \alpha) \leq f(\mathrm{cell}_3(R^3), \alpha)$;

(4) *weak local axiom*: $f(O, \alpha) = f(\mathrm{st}^*_3(\alpha; O), \alpha)$; and,

(5) *complement axiom*: if $O' \subseteq O \subseteq \mathrm{cell}_3(R^3)$ and $\alpha \in R^3$ are such that $\mathrm{st}_3(\alpha; O) = \mathrm{st}_3(\alpha; O')$, $f(O', \alpha) = 0$ and $f(O, \alpha) = 1$, then the set $\alpha(O', O) = \cup\{\mathring{\omega}\,;\, \omega < \alpha, f(O', \omega) = 0,\ f(O, \omega) = 1\} \subseteq \partial\alpha$ is non-empty and connected.

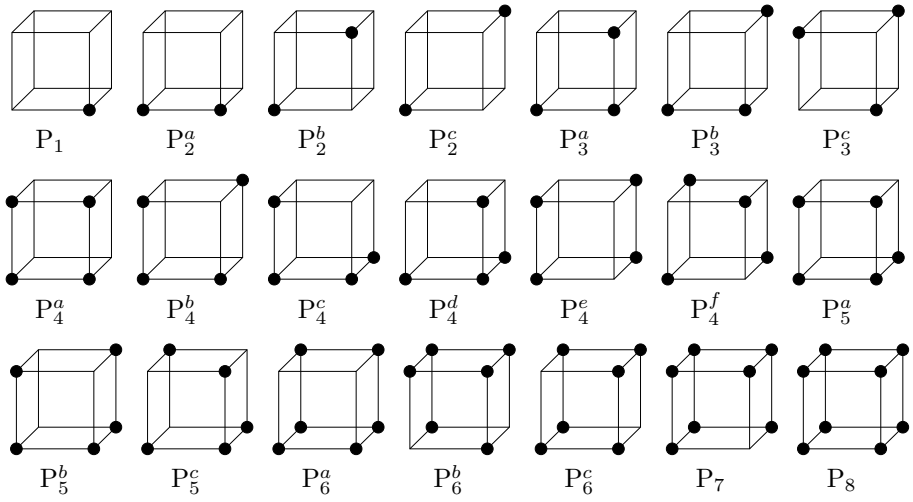



Figure 1: Canonical patterns around a vertex. The empty pattern $\mathrm{P}_0$ is not shown.

If $f(O, \alpha) = 1$ we say that $f$ *lights* the cell $\alpha$ for the object $O$, otherwise $f$ vanishes on $\alpha$ for $O$. We assume that a lighting function assigns to each object $O$ the set $f_O = \{\alpha \in R^3 \,;\, f(O, \alpha) = 1\}$ of cells it lights for $O$.

**Example 2.1.** The following are lighting functions on $R^3$: (a) $f_{\max}(O, \alpha) = 1$ if and only if $\alpha \in \mathrm{supp}(O)$; (b) $g(O, \alpha) = 1$ if and only if $\mathrm{st}_3(\alpha; R^3) \subseteq O$.

A digital space $(R^3, f)$ is said to be *homogeneous* if for any spatial motion $\varphi : \mathbb{R}^3 \to \mathbb{R}^3$ preserving $\mathbb{Z}^3$ the equality $f(\varphi(O), \varphi(\alpha)) = f(O, \alpha)$ holds for all cells $\alpha \in R^3$ and objects $O \subseteq \mathrm{cell}_3(R^3)$; see [7]. According to Axiom 4, a minimal family of objects, called *(canonical) patterns*, suffices to determine the lighting function $f$ of a homogeneous space. Actually, $f(O, \alpha) = f(\mathrm{P}(O, \alpha), \alpha_k)$ for any object $O$ and any $k$-cell $\alpha \in R^3$, $0 \le k < 3$, where $\alpha_k$ is a fixed but arbitrary $k$-cell and the object $\mathrm{P}(O, \alpha) \subseteq \mathrm{st}_3^*(\alpha_k; R^3)$, called the *pattern of* $O$ *in* $\alpha$, is the unique canonical pattern for which there exists a spatial motion $\mathbb{R}^3 \to \mathbb{R}^3$ preserving $\mathbb{Z}^3$ that carries $\mathrm{st}_3^*(\alpha; O)$ to $\mathrm{P}(O, \alpha)$; see Fig. 1.

Finally, we say that a homogeneous digital space $(R^3, f)$ is *Euclidean* if its continuous analogue is $\mathbb{R}^3$; that is, if $f(\mathrm{cell}_3(R^3), \alpha) = 1$ for each cell $\alpha \in R^3$.

# 3 $(k, \bar{k})$-connected digital spaces

In this section we assume that $(R^3, f)$ is Euclidean. Our aim is to find some necessary conditions on the lighting function $f$ so that the space $(R^3, f)$ provides us with the $(k, \bar{k})$-connectedness, as usually defined on $\mathbb{Z}^3$ by mean of adjacencies, $k, \bar{k} \in \{6, 18, 26\}$. For this, we first recall that a digital object $O$ in $(R^3, f)$ is said to be connected if its continuous analogue $|\mathcal{A}_O|$ is connected. And, similarly, its complement $\mathrm{cell}_3(R^3) - O$ of $O$ is declared to be connected if $|\mathcal{A}_{R^3}| - |\mathcal{A}_O|$ is connected. Moreover, we call $C \subseteq \mathrm{cell}_3(R^3)$ a component of $O$ ($\mathrm{cell}_3(R^3) - O$) if it consists of all voxels $\sigma$ whose centers $c(\sigma)$ belong to a component of $|\mathcal{A}_O|$ ($|\mathcal{A}_{R^3}| - |\mathcal{A}_O|$, respectively). These notions of connectedness are characterized by the following notions of adjacency

**Definition 3.1.** Two cells $\sigma, \tau \in O$ are $\emptyset$-*adjacent* iff $f(O, \alpha) = 1$ for some common face $\alpha \le \sigma \cap \tau$. Moreover, $\sigma, \tau \in \mathrm{cell}_3(R^3) - O$ are $O$-*adjacent* iff $f(\mathrm{cell}_3(R^3), \alpha) = 1$ and $f(O, \alpha) = 0$ for some $\alpha \le \sigma \cap \tau$.

More precisely, for $X \in \{\emptyset, O\}$, the notion of $X$-adjacency directly leads to the notions of $X$-path, $X$-connectedness and $X$-component. Then, it can be proved that $C \subseteq O$ is component of $O$ if and only it is an $\emptyset$-component, while $C \subseteq \mathrm{cell}_3(R^3) - O$ is a component of the complement $\mathrm{cell}_3(R^3) - O$ if and only if it is a $O$-component. See Section 4 in [2] for a detailed proof of this fact in a much more general context.

By using this characterization, one may get intuitively convinced that the lighting functions $f_{\max}$ and $g$ in Example 2.1 describe the $(26,6)$- and $(6,26)$-adjacencies usually defined on $\mathbb{Z}^3$. Actually the digital spaces $(R^3, f_{\max})$ and $(R^3, g)$ are $(26,6)$- and $(6,26)$-connected, respect., in the sense of the following

**Definition 3.2.** Given an adjacency pair $(k,\overline{k})$ in $\mathbb{Z}^3$ we say that the digital space $(R^3, f)$ is $(k,\overline{k})$-*connected* or, simply, a $(k,\overline{k})$-*space* if the two following properties hold for any digital object $O \subseteq \mathrm{cell}_3(R^3)$:

1. $C$ is a $\emptyset$-component of $O$ if and only if it is a $k$-component of $O$; and,
2. $C$ is an $O$-component of the complement $\mathrm{cell}_3(R^3) - O$ if and only if it is a $\overline{k}$-component.

The following propositions result from definitions 3.2 and that of Euclidean space, together with the conditions satisfied by lighting functions:

**Proposition 3.3.** *Let* $\mathrm{P}(O,\alpha)$ *the pattern of a digital object* $O \subseteq \mathrm{cell}_3(R^3)$ *in a vertex* $\alpha \in R^3$. *The following properties hold:*

1. *If* $\mathrm{P}(O,\alpha) = \mathrm{P}_8$ *then* $f(O,\alpha) = 1$.
2. $f(O,\alpha) = 0$ *whenever* $\mathrm{P}(O,\alpha) \in \{\mathrm{P}_0, \mathrm{P}_1, \mathrm{P}_2^a, \mathrm{P}_3^a, \mathrm{P}_4^a\}$.
3. *If* $\mathrm{P}(O,\alpha) = \mathrm{P}_2^c$, *then* $f(O,\alpha) = 1$ *iff* $k = 26$.
4. *For* $\overline{k} = 6$, $f(O,\alpha) = 1$ *whenever* $\mathrm{P}(O,\alpha) \in X = \{\mathrm{P}_3^c, \mathrm{P}_4^b, \mathrm{P}_4^e, \mathrm{P}_4^f, \mathrm{P}_5^b, \mathrm{P}_6^b\}$.
5. *If* $\mathrm{P}(O,\alpha) = \mathrm{P}_6^c$, *then* $f(O,\alpha) = 1$ *iff* $\overline{k} \in \{6, 18\}$.

**Proposition 3.4.** *Let* $O \subseteq \mathrm{cell}_3(R^3)$ *be a digital object and* $\beta = \langle \alpha_1, \alpha_2 \rangle \in R^3$ *be a* 1*-cell such that* $f(O,\alpha_1) = f(O,\alpha_2)$. *If* $\mathrm{st}_3(\beta; O) = \mathrm{st}_3(\beta; R^3)$ *then* $f(O,\beta) = 1$. *Moreover, also* $f(O,\beta) = 1$ *in case* $\mathrm{st}_3(\beta; O) = \{\sigma, \tau\}$, *with* $\beta = \sigma \cap \tau$, *and* $(R^3, f)$ *is a* $(k,6)$*-space.*

**Proposition 3.5.** *Let* $O \subseteq \mathrm{cell}_3(R^3)$ *be a digital object and* $\gamma \in \mathrm{supp}(O)$ *a* 2*-cell. Then* $\gamma$ *or some of its faces are lighted for* $O$.

# 4 $(k,\overline{k})$-surfaces

Similarly to our previous definition of connectedness on a digital space $(R^3, f)$, we use continuous analogues to introduce a notion of digital surface. Namely, an object $S$ is a *digital surface* if $|\mathcal{A}_S|$ is a (combinatorial) surface without boundary; that is, if for each vertex $v \in \mathcal{A}_S$ its link $\mathrm{lk}(v; \mathcal{A}_S) = \{A \in \mathcal{A}_S\,;\, v, A < B \in \mathcal{A}_S \text{ and } v \notin A\}$ is a 1-sphere. If $(R^3, f)$ is an Euclidean $(k,\overline{k})$-space we call $S$ a $(k,\overline{k})$-*surface*.

Along this section $S$ will stand for an arbitrary $(k,\overline{k})$-surface $S$, where $k, \overline{k} \in \{6, 18, 26\}$ and $(k,\overline{k}) \neq (6,6)$. Our goal is to compute its continuous analogue; actually, we will determine the value $f(S,\delta)$ for almost each cell $\delta \in R^3$. A simple although crucial tool for this task is the following

**Remark 4.1.** By the definition of continuous analogues, each vertex in $\mathcal{A}_S$ is the center $c(\gamma)$ of a cell $\gamma \in R^3$ lighted for the surface $S$. Moreover, the cycle $\mathrm{lk}(c(\gamma); \mathcal{A}_S)$ is the order complex determined by the set of cells $X_S^\gamma = \{\delta \in R^3\,;\, \delta < \gamma \text{ or } \gamma < \delta \text{ and } f(S,\delta) = 1\}$, which is contained in $Y_S^\gamma = \{\delta \in R^3\,;\, \delta < \gamma \text{ or } \gamma < \delta \text{ and } \delta \in \mathrm{supp}(S)\}$ by Axiom 2.

If $\delta$ is a 3-cell, Axioms 1 and 2 in the definition of lighting functions imply that $f(S,\delta) = 1$ if and only if $\delta \in S$. The following result holds for 1-cells:

**Theorem 4.2.** *Given a* 1*-cell* $\beta \in R^3$, *let* $\alpha_1, \alpha_2 < \beta$ *be its vertices and* $\gamma_j \in R^3$ *be the four* 2*-cells such that* $\beta < \gamma_j$, $1 \leq i \leq 4$. *Then* $f(S,\beta) = 1$ *if and only if one of the two following properties hold:*

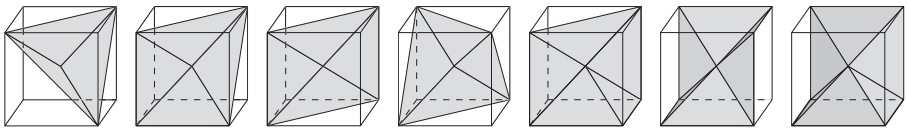



Figure 2: Continuous analogue of a surface $S$ around a vertex $\alpha \in R^3$.

1. $\mathrm{st}_3(\beta; S) = \{\sigma, \tau\}$, *with* $\beta = \sigma \cap \tau$ *and, moreover,* $f(S, \alpha_i) = 1$, $i = 1, 2$, *and* $f(S, \gamma_j) = 0$, $1 \leq j \leq 4$.

2. $\mathrm{st}_3(\beta; S) = \mathrm{st}_3(\beta; R^3)$, $f(S, \alpha_i) = 0$ *and* $f(S, \gamma_j) = 1$, $i = 1, 2$, $1 \leq j \leq 4$.

The proof of Theorem 4.2 needs the results in Sect. 3 as well as to know the lighting of some vertices of $R^3$ for the surface. However, the "only if" part is a consequence of the following proposition, which in turn results from remark 4.1

**Proposition 4.3.** *Let* $\beta \in R^3$ *be a* 1*-cell. Then* $f(S, \beta) = 0$ *if one of the following conditions hold:*

1. $\mathrm{st}_3(\beta; S)$ *consists of exactly three elements.*

2. $\mathrm{st}_3(\beta; S) = \mathrm{st}_3(\beta; R^3)$ *and* $f(S, \alpha) = 1$ *for some vertex* $\alpha < \beta$.

**Remark 4.4.** If a vertex $\alpha \in R^3$ is lighted for the surface $S$ the proof of the "only if" part of Theorem 4.2 allow us to be more precise than in Remark 4.1. Indeed, the set of cells $X_S^\alpha$, determining the link $\mathrm{lk}(c(\alpha); \mathcal{A}_S)$, is contained in the disjoint union $\mathrm{st}_3(\alpha; S) \cup Z_S^\alpha$, where $Z_S^\alpha = \{\delta > \alpha\,;\, \mathrm{st}_3(\delta; S) = \{\sigma, \tau\}, \delta = \sigma \cap \tau\}$.

Using this remark we are able to describe locally the continuous analogue of a surface $S$ around any vertex $\alpha$ which is lighted for it. Before, and in addition to those found in Proposition 3.3, we next identify some more patterns of a surface $S$ around a vertex $\alpha$ for which $f(S, \alpha) = 0$.

**Proposition 4.5.** *Let* $\mathrm{P}(S, \alpha)$ *be the pattern of a surface around a vertex* $\alpha \in R^3$. *If* $\mathrm{P}(S, \alpha) \in \{\mathrm{P}_2^c, \mathrm{P}_3^b, \mathrm{P}_4^c, \mathrm{P}_4^d, \mathrm{P}_5^a, \mathrm{P}_5^c, \mathrm{P}_6^b, \mathrm{P}_7\}$ *then* $f(S, \alpha) = 0$.

**Remark 4.6.** Given a $(k, \bar{k})$-surface $S$ and a vertex $\alpha \in R^3$ such that $f(S, \alpha) = 1$ we know that $\mathrm{P}(S, \alpha) \in \{\mathrm{P}_3^c, \mathrm{P}_4^b, \mathrm{P}_4^e, \mathrm{P}_4^f, \mathrm{P}_5^b, \mathrm{P}_6^a, \mathrm{P}_6^c\}$ from Propositions 3.3 and 4.5. For each of these patterns we can determine the lighting of every cell in $Z_S^\alpha$ by using the result that, as $S$ is a digital surface, $\mathrm{lk}(c(\alpha); \mathcal{A}_S)$ is a 1-sphere. That is, we completely know the continuous analogue of the surface inside the unit cube $C_\alpha \subset \mathbb{R}^3$ whose vertices are the centers of the eight voxels containing $\alpha$; see Fig. 2.

Indeed, if $\mathrm{P}(S, \alpha) \in \{\mathrm{P}_3^c, \mathrm{P}_4^b, \mathrm{P}_4^e, \mathrm{P}_5^b, \mathrm{P}_6^c\}$ each set $Z_\sigma$, $\sigma \in \mathrm{st}_3(\alpha; S)$, consists of two cells. Therefore, all of them are lighted for the surface (i.e., $Z_S^\alpha \subset X_S^\alpha$).

For $\mathrm{P}_6^a$, let $\sigma, \tau$ be the two voxels in $\mathrm{st}_3(\alpha; S)$ which are 6-adjacent to three other voxels in the star of $\alpha$ in $S$. One readily checks that for each voxel $\rho \in A = \mathrm{st}_3(\alpha; S) - \{\sigma, \tau\}$ the set $Z_\rho$ has two elements and then it is contained in $X_S^\alpha$; moreover, $X_S^\alpha = \mathrm{st}_3(\alpha; S) \cup (\cup_{\rho \in A} Z_\rho)$ since this set determines a cycle in $\mathrm{lk}(c(\alpha); \mathcal{A}_S)$. In other words, $\gamma = \sigma \cap \tau$ is the only 2-cell in $\mathrm{supp}(\mathrm{st}_3(\alpha; S))$ such that $f(S, \gamma) = 0$; moreover, also $f(S, \beta_i) = 0$ for the two edges $\alpha < \beta_1, \beta_2 < \gamma$.

Finally, at this point we can only determine the continuous analogue for $\mathrm{P}_4^f$ up to symmetries. We know that only two of the three 1-cells in each of the sets $Z_{\sigma_i}$, where $Z_\sigma = \{\delta < \sigma;\, \mathrm{st}_3(\delta; S) = \{\sigma, \tau\},\, \delta = \sigma \cap \tau\}$ and $\mathrm{st}_3(\alpha; S) = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, can be lighted for the surface. However the selection of two cells in $Z_{\sigma_1}$ determines what cells in $Z_{\sigma_2}, Z_{\sigma_3}, Z_{\sigma_4}$ must be lighted in order that $\mathrm{lk}(c(\alpha); \mathcal{A}_S)$ is a cycle.

To complete our analysis of the lighting on vertices and edges of $R^3$ for $S$ we use some separation properties of $(k, \bar{k})$-surfaces. Firstly, notice that, from our definition of $(k, \bar{k})$-space and the characterization of connectedness on a Euclidean space $(R^3, f)$ in Sect. 3, the $\bar{k}$-components of $\mathrm{cell}_3(R^3) - S$

are characterizing the connected components of $\mathbb{R}^3 - |\mathcal{A}_S| = |\mathcal{A}_{R^3}| - |\mathcal{A}_S|$. Therefore, we obtain the following separation result as a corollary of the well-known Jordan-Brouwer Theorem; see Sect. 5 in [3].

**Theorem 4.7.** *Let $S$ be a $k$-connected $(k,\overline{k})$-surface. Then $S$ separates its complement $\mathrm{cell}_3(R^3) - S$ into two $\overline{k}$-components.*

Next proposition shows that the $\overline{k}$-componentes of $\mathrm{cell}_3(R^3) - S$ can be locally determined by using the notion of relative ball from polyhedral topology. Recall that given a closed topological surface $M \subset \mathbb{R}^3$ in the Euclidean space, a *relative ball* $(B^3, B^2)$ in $(\mathbb{R}^3, M)$ is a pair of topological balls $B^2 \subset B^3$ such that $B^2 \cap \partial B^3 = \partial B^2$ and $B^2 = B^3 \cap M$. The key property of relative balls, which is also a consequence of the Jordan-Brouwer Theorem, states that the difference $B^3 - B^2$ has exactly two connected components, each contained in a distinct component of $\mathbb{R}^3 - M$; see [2] for details.

**Proposition 4.8.** *Two 26-adjacent voxels $\sigma, \tau \notin S$ belong to the same $\overline{k}$-component of $\mathrm{cell}_3(R^3) - S$ if and only if $f(S, \sigma \cap \tau) = 0$.*

**Corollary 4.9.** *Assume $\overline{k} \in \{18, 26\}$. Then $f(S,\delta) = 0$ for each cell $\delta \in R^3$ satisfying one of the following conditions:*

1. *$\delta$ is an edge such that $\mathrm{st}_3(\delta; S) = \{\sigma, \tau\}$ with $\delta = \sigma \cap \tau$.*
2. *$\delta$ is a vertex and $\mathrm{P}(S,\delta) \in \{\mathrm{P}_3^c, \mathrm{P}_4^b, \mathrm{P}_4^e, \mathrm{P}_4^f, \mathrm{P}_5^b\}$.*

**Theorem 4.10.** *Let $S$ be a $(k,\overline{k})$-surface and $\alpha \in R^3$ be a vertex such that $\mathrm{P}(S,\alpha) \notin \{\mathrm{P}_6^a, \mathrm{P}_8\}$. Then $\mathrm{P}(S,\alpha) \notin \mathbb{FP}_{k,\overline{k}}$ and, moreover, $f(S,\alpha) = 1$ if and only if $\mathrm{P}(S,\alpha) \in \mathbb{P}_{\overline{k}}$. The sets $\mathbb{FP}_{k,\overline{k}}$ and $\mathbb{P}_{\overline{k}}$, whose elements are respectively called $(k,\overline{k})$-*forbidden patterns *and $\overline{k}$-*plates*, are defined as follows: $\mathbb{FP}_{6,26}$, $\mathbb{FP}_{18,26}$, $\mathbb{FP}_{6,18}$ and $\mathbb{FP}_{18,18}$ are the empty set, $\mathbb{FP}_{26,26} = \mathbb{FP}_{26,18} = \{\mathrm{P}_2^c\}$, $\mathbb{FP}_{18,6} = \{\mathrm{P}_5^c, \mathrm{P}_6^b\}$ and $\mathbb{FP}_{26,6} = \{\mathrm{P}_2^c, \mathrm{P}_5^c, \mathrm{P}_6^b\}$; $\mathbb{P}_6 = \{\mathrm{P}_3^c, \mathrm{P}_4^b, \mathrm{P}_4^e, \mathrm{P}_4^f, \mathrm{P}_5^b, \mathrm{P}_6^c\}$, $\mathbb{P}_{18} = \{\mathrm{P}_6^c\}$ and $\mathbb{P}_{26} = \emptyset$.*

Notice that in the theorem above we have excluded $\mathrm{P}_8$ from our analysis. This pattern is usually considered as a small blob and should not be part of a surface. However $\mathrm{P}_8$ can appear in surfaces of rare Euclidean $(k,\overline{k})$-spaces, $\overline{k} \in \{18, 26\}$, that could be considered anomalous. For this reason, from now on we will only consider regular spaces according to the next definition.

**Definition 4.11.** An Euclidean $(k,\overline{k})$-space is said to be *regular* if $\mathrm{P}(S,\alpha) \neq \mathrm{P}_8$ for any surface $S$ and any vertex $\alpha \in R^3$.

**Remark 4.12.** We have also eluded $\mathrm{P}_6^a$ in Theorem 4.10. Actually, a surface $S$ containing this pattern can have two different but homeomorphic continuous analogues, depending on the value of $f(S,\alpha)$ for any vertex $\alpha \in R^3$ such that $\mathrm{P}(S,\alpha) = \mathrm{P}_6^a$ (recall that we are only considering homogeneous digital spaces). If $f(S,\alpha) = 1$, we have already described locally the continuous analogue of $S$ around $\alpha$ in Remark 4.6. For describing the intersection $|\mathcal{A}_S| \cap C_\alpha$, where $C_\alpha \subset \mathbb{R}^3$ is the unit cube with vertices at the centers of the voxels in $\mathrm{st}_3(\alpha; R^3)$, in case $f(S,\alpha) = 0$, let us consider the two edges $\beta_i = \langle \alpha, \alpha_i \rangle$ of $R^3$ such that $\mathrm{st}_3(\beta_i; S) = \mathrm{st}_3(\beta_i; R^3)$, $i = 1, 2$. If $(R^3, f)$ is a regular space Theorem 4.10 yields that $f(S,\alpha_i) = 0$ for $i = 1, 2$ and, then, $f(S,\beta_i) = 1$ by Theorem 4.2; moreover, any 2-cell containing $\alpha$ is lighted for $S$. So that, the continuous analogue of $S$ around $\alpha$ is the union of the two unit squares whose vertices are the centers of the voxels in $\mathrm{st}_3(\alpha; S)$.

We finish this section showing that certain patterns around an edge are forbidden in a $(k,\overline{k})$-surface when $k, \overline{k} \in \{18, 26\}$.

**Proposition 4.13.** *Let $S$ be a digital surface in a Euclidean $(k,\overline{k})$-space $(R^3, f)$, $k, \overline{k} \in \{18, 26\}$. If $\beta \in R^3$ is an edge such that $\mathrm{st}_3(\beta; S) = \{\sigma, \tau\}$, with $\beta = \sigma \cap \tau$, then $\mathrm{st}_3^*(\beta; S)$ is 6-connected.*

# 5 Universal $(k,\overline{k})$-spaces

In this section we reach our goal: for each adjacency pair $(k,\overline{k}) \neq (6,6)$, $k, \overline{k} \in \{6, 18, 26\}$, we define a lighting function on the complex $R^3$ which gives us a regular $(k,\overline{k})$-space $E_{k,\overline{k}}^U$, whose set of digital surfaces is the largest within that class of digital spaces:

**Theorem 5.1.** *Any digital surface $S$ in a regular $(k,\overline{k})$-space $(R^3,f)$ is also a digital surface in the space $E^U_{k,\bar{k}}=(R^3,f^U_{k,\bar{k}})$, which is called the* universal $(k,\overline{k})$-space.

In the definition of $E^U_{k,\bar{k}}$ we use the set of forbidden patterns introduced in Theorem 4.10 and the set of $\overline{k}$-plates, that are the elementary "bricks" from which $(k,\overline{k})$-surfaces are built. Indeed, the lighting function $f^U_{k,\bar{k}}$ is defined as follows. Given a digital object $O\subseteq \mathrm{cell}_3(R^3)$ and a cell $\delta\in R^3$, $f^U_{k,\bar{k}}(O,\delta)=1$ if and only one of the following conditions hold:

1. $\dim\delta\geq 2$ and $\delta\in\mathrm{supp}(O)$
2. $\dim\delta=0$ and $\mathrm{P}(O,\delta)\in\mathbb{P}_{\overline{k}}\cup\mathbb{FP}_{k,\overline{k}}\cup\{\mathrm{P}_8\}$
3. $\dim\delta=1$ and $\mathrm{st}_3(\delta;O)=\mathrm{st}_3(\delta;R^3)$ (*a square plate*), or
4. $\dim\delta=1$ and $\mathrm{st}_3(\delta;O)=\{\sigma,\tau\}$, with $\delta=\sigma\cap\tau$, and one of the next further conditions also holds:
   (a) for $\overline{k}=6$, and $k\neq 6$, $f_{k,6}(O,\alpha_1)=f_{k,6}(O,\alpha_2)$, where $\alpha_1,\alpha_2$ are the two vertices of the 1-cell $\delta$; or
   (b) if $\mathrm{st}^*_3(\delta;O)$ is not 6-connected, for $k,\overline{k}\in\{18,26\}$.

Notice that if $k=6$, and $\overline{k}\in\{18,26\}$, none of the conditions 4(a) and 4(b) hold. So, $f^U_{6,\overline{k}}(O,\delta)=1$ for a 1-cell $\delta\in R^3$ if and only if $\mathrm{st}_3(\delta;O)=\mathrm{st}_3(\delta;R^3)$.

It is not difficult, but a tedious task, to check that each $f^U_{k,\bar{k}}$ is a lighting function, and to prove that $E^U_{k,\bar{k}}$ is actually a homogeneous $(k,\overline{k})$-space. To show that all of them are regular spaces, assume that $O$ is a digital objet and $\alpha\in R^3$ is a vertex such that $\mathrm{P}(O,\alpha)=\mathrm{P}_8$; that is, $\mathrm{st}_3(\alpha;O)=\mathrm{st}_3(\alpha;R^3)$. Then, it is readily checked from the definition that $f^U_{k,\bar{k}}(O,\delta)=1$ for each cell $\delta\geq\alpha$. Therefore, the unit cube $C_\alpha\subset\mathbb{R}^3$ centered at $\alpha$, and whose vertices are the centers of all voxels in $\mathrm{st}_3(\alpha;R^3)$, is contained in $|\mathcal{A}_O|$; thus no surface in $E^U_{k,\bar{k}}$ can contain the pattern $\mathrm{P}_8$. This also proves that $E^U_{k,\bar{k}}$ is Euclidean since $f(\mathrm{cell}_3(R^3),\delta)=1$ for every cell $\delta\in R^3$.

The proof of theorem 5.1 relies on the following partial results:

In Remark 4.12 we suggested that the continuous analogue of a given surface in two different $(k,\overline{k})$-spaces may differ only if it contains the pattern $\mathrm{P}^a_6$. Next results make more precise this in relation with the universal $(k,\overline{k})$-space.

**Proposition 5.2.** *Let $S$ be a surface in a regular $(k,\overline{k})$-space $(R^3,f)$. If $\delta\in R^3$ is a cell such that $\mathrm{P}(S,\alpha)\neq\mathrm{P}^a_6$ for all vertices $\alpha\leq\delta$, then $f(S,\delta)=f^U_{k,\bar{k}}(S,\delta)$.*

In other words, if $S$ is a surface in a $(k,\bar{k})$-space $(R^3,f)$, $|\mathcal{A}^f_S|$ and $|\mathcal{A}^{f^U_{k,\bar{k}}}_S|$ may differ only in the neighbourhood of vertices $c(\alpha)$ such that $\alpha$ is a vertex in $R^3$ and $P(S,\alpha)=D^a_6$. By using the following proposition, which in turn results from remarks 4.6 and 4.12, it can be proved that the polyhedra $|\mathcal{A}^f_S|$ and $|\mathcal{A}^{f^U_{k,\bar{k}}}_S|$ are *pl*-homeomorphic. Thus, the continuous analogues of $S$ in both digital spaces $(R^3,f)$ and $E^U_{k,\bar{k}}$ are combinatorial surfaces.

**Proposition 5.3.** *Let $S$ be a surface in a regular $(k,\overline{k})$-space $(R^3,f)$. Given a vertex $\alpha\in R^3$, let $C_\alpha\subset\mathbb{R}^3$ be the unit cube centered at $\alpha$. If $\mathrm{P}(S,\alpha)=\mathrm{P}^a_6$ the two following properties hold:*

1. *$D^\alpha_f=|\mathcal{A}^f_S|\cap C_\alpha$ and $D^\alpha_U=|\mathcal{A}^{f^U_{k,\bar{k}}}_S|\cap C_\alpha$ are both 2-balls with common border; that is, $\partial D^\alpha_f=\partial D^\alpha_U$.*
2. *There exists a pl-homeomorphism $\varphi_\alpha: D^\alpha_f\to D^\alpha_U$ which extends the identity in the border; moreover, if $f(S,\alpha)=0$ then $\varphi_\alpha=\mathrm{id}$.*

From theorem 5.1, together with the results in [1], [3], [6] and [9], it follows that our digital surfaces extend those families of surfaces proposed up to now within the graph-theoretical approach to Digital Topology. More precisely:

- Every Kong-Roscoe's $(k, \bar{k})$-surface is a surface in $E^U_{k,\bar{k}}$
- Every *simplicity 6-surface* is a surface in $E^U_{6,26}$
- Every *strong 26-surface* is a *simplicity 26-surface*
- Every *simplicity 26-surface* is a surface in $E^U_{26,6}$

# Referencias

[1] R. Ayala, E. Domínguez, A.R. Francés, A. Quintero. Digital lighting functions. *Lecture Notes in Computer Science.* **1347**, 139–150 (1997).

[2] R. Ayala, E. Domínguez, A.R. Francés, A. Quintero. A digital index theorem. *Int. J. Pattern Recog. Art. Intell.* **15**(7), 1–22 (2001).

[3] R. Ayala, E. Domínguez, A.R. Francés, A. Quintero. Weak lighting functions and strong 26−surfaces. *Theoretical Computer Science.* **283**, 29–66 (2002).

[4] G. Bertrand, R. Malgouyres. Some topological properties of surfaces in $\mathbb{Z}^3$. *Jour. of Mathematical Imaging and Vision* **11**, 207–221 (1999).

[5] V.E. Brimkov, R. Klette. Curves, hypersurfaces, and good pairs of adjacency relations. *Lecture Notes in Computer Science.* **3322**, 276–290 (2004).

[6] J.C. Ciria, E. Domínguez, A.R. Francés. Separation theorems for simplicity 26−surfaces. *Lecture Notes in Computer Science.* **2301**, 45–56 (2002).

[7] J.C. Ciria, A. De Miguel, E. Domínguez, A.R. Francés, A. Quintero. A maximum set of $(26, 6)$−connected digital surfaces. *Lecture Notes in Computer Science.* **3322**, 291–306 (2004).

[8] J.C. Ciria, A. De Miguel, E. Domínguez, A.R. Francés, A. Quintero. Local characterization of a maximum set of digital $(26, 6)$−surfaces. *Image and Vision Computing.* **25**, 1685–1697 (2007).

[9] M. Couprie, G. Bertrand. Simplicity surfaces: a new definition of surfaces in $\mathbb{Z}^3$. *SPIE Vision Geometry V.* **3454**, 40–51 (1998).

[10] A. Esnard, J.-O. Lachaud, A. Vialard, Discrete deformable boundaries for 3D image segmentation. Tech. Report No. 1270-02, LaBRI, Univerity of Bordeaux 1, 2002.

[11] T.Y. Kong, A.W. Roscoe. Continuous analogs fo axiomatized digital surfaces. *Comput. Vision Graph. Image Process.* **29**, 60–86 (1985).

[12] V.A. Kovalevesky. Finite topology as applied to image analysis. *Comput. Vis. Graph. Imag. Process.* **46**, 141–161 (1989).

[13] G. Malandain, G. Bertrand, N. Ayache. Topological segmentation of discrete surfaces. *Int. Jour. of Computer Vision* **10:2**, 183-197 (1993).

[14] P.Y. Chatelier, R. Malgouyres. A low-complexity discrete radiosity method. *Computers & Graphics* **30**, 37–45 (2006).

[15] D.G. Morgenthaler, A. Rosenfeld. Surfaces in three–dimensional digital images. *Inform. Control.* **51**, 227-247 (1981).

[16] C.P. Rourke, B.J. Sanderson. *Introduction to piecewise–linear topology.* Ergebnisse der Math. **69**, Springer 1972.

# On Crossings in Geometric Proximity Graphs

Bernardo M. Ábrego* Ruy Fabila-Monroy† Silvia Fernández-Merchant*
David Flores-Peñaloza† Ferran Hurtado‡ Vera Sacristán‡ Maria Saumell‡

**Abstract**

We study the number of crossings among edges of some higher order proximity graphs of the family of the Delaunay graph. That is, given a set $P$ of $n$ points in the Euclidean plane, we give lower and upper bounds on the minimum and the maximum number of crossings that these geometric graphs defined on $P$ have.

## 1 Introduction

Let $P$ be a set of $n$ points in the plane in general position (no three are collinear). A *geometric graph on* $P$ is a graph with vertex set $P$ and such that its edges are drawn as straight-line segments. When two edges share an interior point we say that they give rise to a *crossing*.

The number of crossings is a parameter that has been attracting extensive studies in the context of combinatorial graphs. Given a graph $G$, the *crossing number of* $G$, denoted by $cr(G)$, is the minimum number of crossings in any drawing of $G$, i.e., in any non-degenerate representation of the graph in the plane. The *rectilinear crossing number of* $G$, denoted by $\overline{cr}(G)$, is the smallest number of crossings in any drawing of $G$ in which the edges are represented by straight-line segments.

There are several classes of graphs for which the number of crossings has been computed or approximated. There are also several results sensitive to the size of the graph -particulary the famous crossing lemma [5, 12, 6]- and to the exclusion of some configurations [6, 13, 15, 19, 8].

The most famous problems in this setting are to obtain the exact values of the crossing number and the rectilinear crossing number of the complete graph $K_n$ and the complete bipartite graph $K_{m,n}$. Recently some of these problems have received a great amount of attention and a continous chain of improvements has led to the bounds shown in Table 1.

| | $K_n$ | $K_{m,n}$ |
|---|---|---|
| $cr \geq$ | $0.8594\,\frac{1}{4}\lfloor\frac{n}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n-2}{2}\rfloor\lfloor\frac{n-3}{2}\rfloor$ (for $n\to\infty$) [11] | $0.8594\,\frac{m}{m-1}\lfloor\frac{m-1}{2}\rfloor\lfloor\frac{m}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n}{2}\rfloor$ (for $n\to\infty$) [11] |
| $cr \leq$ | $\frac{1}{4}\lfloor\frac{n}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n-2}{2}\rfloor\lfloor\frac{n-3}{2}\rfloor$ [17] | $\lfloor\frac{m-1}{2}\rfloor\lfloor\frac{m}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n}{2}\rfloor$ [20] |
| $\overline{cr} \geq$ | $0.379972\binom{n}{4}+\Theta(n^3)$ [4] | $0.8594\,\frac{m}{m-1}\lfloor\frac{m-1}{2}\rfloor\lfloor\frac{m}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n}{2}\rfloor$ (for $n\to\infty$) [11] |
| $\overline{cr} \leq$ | $0.380488\binom{n}{4}+\Theta(n^3)$ [2] | $\lfloor\frac{m-1}{2}\rfloor\lfloor\frac{m}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n}{2}\rfloor$ [20] |

Table 1: Current bounds on the crossing number and the rectilinear crossing number of $K_n$ and $K_{m,n}$.

*Department of Mathematics, California State University, Northridge, CA, {bernardo.abrego,silvia.fernandez}@csun.edu.

†Instituto de Matemáticas, Universidad Nacional Autónoma de México, ruy@ciencias.unam.mx, dflores@math.unam.mx.

‡Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain, {ferran.hurtado,vera.sacristan,maria.saumell}@upc.edu. Partially supported by projects MEC MTM2006-01267 and DURSI GenCat 2005SGR00692.

In this paper the graphs under study are some higher order proximity graphs of the family of the Delaunay graph. In a proximity graph of a set of points in the plane two points are connected by a straight-line segment if some proximity rule is satisfied. There are several rules that have been proposed in the literature leading to different proximity graphs (see the survey [10]); here we focus on the $k$-nearest neighbour graph, the $k$-relative neighbourhood graph, the $k$-Gabriel graph and the $k$-Delaunay graph.

If $P$ is a set of points in the plane, each of the specified proximity graphs is a geometric graph on $P$ that has some number of crossings that will be denoted by $\boxtimes(\ )$. For example, consider the $k$-nearest neighbour graph of $P$ ($k$-$\mathsf{NNG}(P)$ for short). We define the *lowest crossing number of* $k$-$\mathsf{NNG}(n)$ and, respectively, the *worst crossing number of* $k$-$\mathsf{NNG}(n)$ as

$$\begin{aligned} \mathsf{lcr}(k\text{-}\mathsf{NNG}(n)) &= \min_{|P|=n} \boxtimes(k\text{-}\mathsf{NNG}(P)), \\ \mathsf{wcr}(k\text{-}\mathsf{NNG}(n)) &= \max_{|P|=n} \boxtimes(k\text{-}\mathsf{NNG}(P)). \end{aligned}$$

We define analogous parameters for all the mentioned proximity graphs and give upper and lower bounds for all cases. We are only interested in $k > 1$ ($k > 2$ for $k$-$\mathsf{NNG}(P)$), because otherwise we recover the simplest versions of the graphs, which are known to be planar.

Observe that there is a subtle difference between the lowest crossing number and the rectilinear crossing number, as the former does not deal with purely combinatorial graphs, but with geometric ones. In fact, the lowest crossing number of a proximity graph equals the rectilinear crossing number of the drawings of all graphs that can be represented as this proximity graph.

An important part of our research has been devoted to look at particular cases in which $k$ is small (see Sections 2 and 3). This decision is well-grounded, as it has been explained in the literature (for example, in [1] and [9]) that these are the most interesting cases from an applications point of view. In Section 4 we look at the number of crossings for large values of $k$. We point out that our results hold if, besides not containing three collinear points, $P$ satisfies two more non-degeneracy assumptions: no four points are concyclic and, for each $p \in P$, the set of its $k$ nearest points in $P$ is well-defined.

Full proofs and examples can be found in [3].

## 2 Bounds on $\mathsf{lcr}(k\text{-}\mathsf{NNG}(n))$ for small values of $k$

We define the *$k$-nearest neighbour graph of $P$,* denoted by $k$-$\mathsf{NNG}(P)$, as the geometric graph on $P$ in which two points $p_i, p_j$ are joined by an edge if either $p_j$ is one of the $k$ nearest neighbours in $P$ of $p_i$ or viceversa.

In this section we provide bounds on the lowest crossing number of the $k$-nearest neighbour graph for small values of $k$. Because of the hierarchical relation satisfied by the graphs we investigate (see Section 4), the lower bounds also hold for the lowest crossing number of the other proximity graphs if we shift the value of $k$ one unit down.

The results we have obtained are summarized in Table 2. Refer to [3] for a more detailed analysis. Due to space limitations, here we only explain the case $k = 10$.

The lower bound for $\mathsf{lcr}(10\text{-}\mathsf{NNG}(n))$ follows from the following result, proved in [14]:

**Theorem 2.1.** *The crossing number of any graph $G$ with $v(G) \geq 3$ vertices and $e(G)$ edges satisfies*

$$cr(G) \geq \frac{7}{3}e(G) - \frac{25}{3}(v(G) - 2).$$

Indeed we know that the number of edges of the 10-nearest neighbour graph of $P$ is greater than or equal to $5n$, as each vertex has degree at least 10. So we have

$$\boxtimes(10\text{-}\mathsf{NNG}(P)) \geq cr(10\text{-}\mathsf{NNG}(P)) \geq \frac{7}{3}e(10\text{-}\mathsf{NNG}(P)) - \frac{25}{3}\left(v(10\text{-}\mathsf{NNG}(P)) - 2\right) = \frac{10n}{3} + \frac{50}{3}.$$

| $k$ | $\mathsf{lcr}(k\text{-}\mathsf{NNG}(n))$ |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0, for $n \geq 14$ |
| 5 | 0, for $n \geq 44$ |
| 6 | $\leq 66$, for $n \geq 84$ |
| 7 | $n/2 + \Theta(1)$ |
| 8 | $n + \Theta(1)$ |
| 9 | $13n/6 + 50/3 \leq \mathsf{lcr} \leq 31n/13 + \Theta(1)$ |
| 10 | $10n/3 + 50/3 \leq \mathsf{lcr} \leq 4n + \Theta(1)$ |

Table 2: $\mathsf{lcr}(k\text{-}\mathsf{NNG}(n))$ for the first values of $k$.

As for the upper bound, consider the configuration in Figure 1 (a precise description of the point coordinates can be found in [3]). An easy assignment of crossings to points shows that each point that is not in the boundary of the set or in the next layer causes four crossings. The points in the boundary and the next layer add $\Theta(\sqrt{n})$ crossings. Observe that the point positions can be slightly perturbed without modifying the ten nearest neighbours of each point. Thus, if we arrange the points in circular strips and each strip contains exactly the minimum number of points ensuring that the adjacencies in the 10-nearest neighbour graph do not change, the number of points in the boundary of the set becomes constant. Consequently, the 10-nearest neighbour graph of the resulting set of points has $4n + \Theta(1)$ crossings.

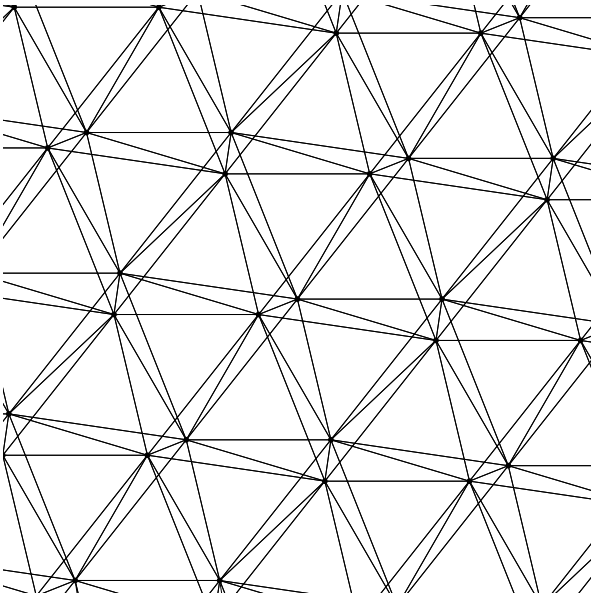


Figure 1: $\text{10-}\mathsf{NNG}(P)$ with $4n + \Theta(\sqrt{n})$ crossings.

## 3 1-Delaunay graphs

The *k-Delaunay graph* of $P$, which was introduced in [1] and will be denoted by $k\text{-}\mathsf{DG}(P)$, is the geometric graph with vertex set $P$, and an edge between points $p_i$ and $p_j$ if there exists a disk through $p_i$ and $p_j$ with at most $k$ points of $P$ in its interior.

We have already remarked that, from the viewpoint of applications, the most significant $k$-Delaunay graphs are those where the value of $k$ is very small. In this section we carry out a detailed analysis of the 1-Delaunay graph.

As in [1], we have studied the general case but we have also devoted some attention to the situation where all points are in convex position. Our contributions are presented in Table 3. As all the proofs for the general case are quite long, we only describe here the value of $\mathsf{lcr}$ in the convex case. Specifically, we prove that, if $P$ is in convex position, $\mathsf{lcr}(\text{1-}\mathsf{DG}(n)) = 6n - 3\lfloor \frac{n}{2} \rfloor - 19$.

| | general case | convex case |
|---|---|---|
| $\mathsf{lcr}$ | $n-4$ | $6n - 3\lfloor \frac{n}{2} \rfloor - 19$ |
| $\mathsf{wcr}$ | $n^2 + \Theta(n) \leq \mathsf{wcr} \leq 4n^2 + \Theta(n)$ | $\frac{n^2}{2} + \Theta(n) \leq \mathsf{wcr} \leq \frac{7n^2}{8} + \Theta(n)$ |

Table 3: 1-Delaunay graphs.

Let $e_b = e(\text{0-}\mathsf{DG}(P)) = 2n - 3$ and $e_r = e(\text{1-}\mathsf{DG}(P)) - e(\text{0-}\mathsf{DG}(P))$. In [1] it is shown that $e_r \geq \lceil \frac{3n}{2} \rceil - 5 = 2n - \lfloor \frac{n}{2} \rfloor - 5$. We make use of this result to prove the lower bound for $\mathsf{lcr}(\text{1-}\mathsf{DG}(n))$ :

**Theorem 3.1.** *For every set $P$ in convex position, $\boxtimes(\text{1-}\mathsf{DG}(P)) \geq 6n - 3\lfloor \frac{n}{2} \rfloor - 19$.*

*Proof.* Note that all edges forming an ear in $P$ are in $\text{1-}\mathsf{DG}(P)$, and that the number of crossings between two edges of this type is $n$. Let $H$ be the graph obtained from $\text{1-}\mathsf{DG}(P)$ by removing these edges and the ones in the convex hull of $P$. Since the size of $\text{1-}\mathsf{DG}(P)$ is $e_b + e_r$, $H$ contains at least $2n - \lfloor \frac{n}{2} \rfloor - 8$ edges. Each of them induces two crossings with the ear edges.

Let $H_p$ be a maximal planar subgraph of $H$. It is easy to see that $H_p$ contains at most $n-5$ edges. Thus there are at least $n - \lfloor \frac{n}{2} \rfloor - 3$ edges in $H$ but not in $H_p$, each of which induces at least one crossing with an edge of $H_p$.

Adding everything up, we can conclude that $\text{1-}\mathsf{DG}(P)$ has no less than $6n - 3\lfloor \frac{n}{2} \rfloor - 19$ crossings. □

The point set in Figure 2, which is carefully described in [3], shows that this bound can be attained and, consequently, is best possible.

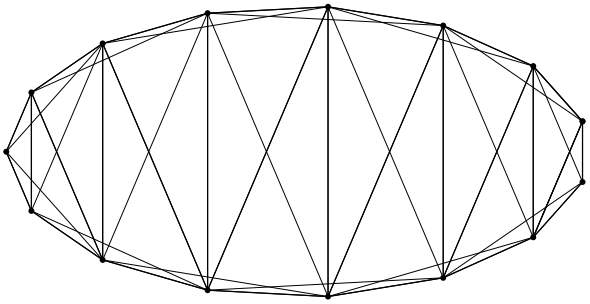


Figure 2: $\text{1-}\mathsf{DG}(P)$ with $6n - 3\lfloor \frac{n}{2} \rfloor - 19$ crossings.

# 4 General bounds

In this section we are interested in the asymptotic behavior when $k$ is large of the number of crossings in the graphs under study. First we define the two remaining graphs.

For every pair of points $p_i, p_j \in P$, let $\text{O-}\mathsf{LENS}(p_i, p_j) = \{x \in \mathbb{R}^2 : |p_i x| < |p_i p_j| \text{ and } |p_j x| < |p_i p_j|\}$. The *$k$-relative neighbourhood graph* of $P$, denoted by $k\text{-}\mathsf{RNG}(P)$, is the geometric graph on $P$ where

two vertices $p_i, p_j$ are adjacent if $\mathsf{O\text{-}LENS}(p_i, p_j)$ contains at most $k$ points in $P$. The graph $k\text{-}\mathsf{RNG}(P)$ was introduced in [7].

Now let $\mathsf{C\text{-}DISC}(p_i, p_j)$ be the closed disc centered at the midpoint of the segment $\overline{p_i p_j}$ with both $p_i$ and $p_j$ on its boundary. The *$k$-Gabriel graph* of $P$, denoted by $k\text{-}\mathsf{GG}(P)$, is the geometric graph with vertex set $P$, and an edge between points $p_i$ and $p_j$ if $\mathsf{C\text{-}DISC}(p_i, p_j)$ contains at most $k$ points in $P$ different from $p_i, p_j$. The first definition of the *$k$-Gabriel graph*, which is slightly different from ours, was given in [18].

We have derived bounds for both the lowest crossing number and the worst crossing number of all graphs (see Table 4). In some cases, we have used the fact that there exists a relation of containment among the different classes of proximity graphs we have presented:

$$(k+1)\text{-}\mathsf{NNG}(P) \subseteq k\text{-}\mathsf{RNG}(P) \subseteq k\text{-}\mathsf{GG}(P) \subseteq k\text{-}\mathsf{DG}(P).$$

| | $k$-NNG$(n)$ | $k$-RNG$(n)$ | $k$-GG$(n)$ | $k$-DG$(n)$ |
|---|---|---|---|---|
| lcr $\geq$ | $\frac{128}{31827}k^3n$ | $\frac{128}{31827}k^3n$ | $\frac{128}{31827}k^3n$ | $\frac{128}{31827}k^3n$ |
| lcr $\leq$ | $\frac{2}{27\pi^2}k^3n$ | $\frac{128}{27\pi^2}k^3n$ | $\frac{128}{27\pi^2}k^3n$ | $\frac{128}{27\pi^2}k^3n$ |
| wcr $\geq$ | $\frac{1}{3}k^3n$ | $\frac{1}{3}k^3n$ | $\frac{1}{4}k^2n^2$ | $\frac{1}{2}k^2n^2$ |
| wcr $\leq$ | $k^3n$ | $9k^3n$ | $\frac{9}{2}k^2n^2$ | $\frac{9}{2}k^2n^2$ |

Table 4: General bounds.

The lower bounds for the lowest crossing numbers follow from an improved version of the crossing lemma proved in [14]. As for the upper bounds, we use a construction given in [16] of a graph with fixed size and few crossings, and show that it can be seen as a higher order proximity graph.

In contrast, our approach to bound from above the worst crossing numbers relies strongly on the geometric properties of each particular graph. As a representative example, we sketch the proof of the upper bound for $\mathsf{wcr}(k\text{-}\mathsf{RNG}(n))$.

**Lemma 4.1.** *In any angular sector with apex $p \in P$ and amplitude $\alpha \leq \pi/3$ there are at most $k+3$ points in $P$ that are connected to $p$ in the graph $k$-$\mathsf{RNG}(P)$.*

**Theorem 4.2.** *For every point set $P$, $\boxtimes(k\text{-}\mathsf{RNG}(P)) \leq 9k^3n + o(k^3n)$.*

*Proof.* We use a charging scheme that assigns every crossing in $k\text{-}\mathsf{RNG}(P)$ to each of the two involved edges $e$ satisfying that at least one of the endpoints of the other edge is contained in the lens associated to $e$.

Each crossing defines a quadrilateral that has at least one obtuse angle at some vertex $p_i$. If the diagonal opposite to this obtuse angle is the edge $p_jp_k$, $p_i$ is contained in the lens associated to $p_jp_k$, so the crossing is assigned to (at least) the edge $p_jp_k$.

Now we bound the maximum number of crossings that may be assigned to an edge $e$. The lens associated to $e$ contains at most $k$ points in $P$. By Lemma 4.1, each of these points is adjacent to no more than $3k+9$ other points in $P$ such that the edge that connects them crosses $e$. Consequently, at most $3k^2+9k$ crossings may be assigned to $e$.

Since each vertex in $P$ has degree at most $6k+18$ (see Lemma 4.1), the number of edges of $k\text{-}\mathsf{RNG}(P)$ is bounded by $3kn + o(kn)$, which yields the theorem. □

# References

[1] M. Abellanas, P. Bose, J. García, F. Hurtado, C. M. Nicolás and P. A. Ramos. On structural and graph theoretic properties of higher order Delaunay graphs. *Internat. J. Comput. Geom. Appl.*, to appear.

[2] B. M. Ábrego, M. Cetina, S. Fernández-Merchant, J. Leaños and G. Salazar. 3-symmetric and 3-decomposable drawings of $K_n$. *Submitted*, arXiv:0805.0016, 2008.

[3] B. M. Ábrego, R. Fabila-Monroy, S. Fernández-Merchant, D. Flores-Peñaloza, F. Hurtado, V. Sacristán and M. Saumell. On the number of crossings of some higher order proximity graphs. *Manuscript*, 2008.

[4] B. M. Ábrego, S. Fernández-Merchant, J. Leaños and G. Salazar. A central approach to bound the number of crossings in a generalized configuration. *Electron. Notes Discrete Math.*, 30:273–278, 2008.

[5] M. Ajtai, V. Chvátal, M. M. Newborn and E. Szemerédi. Crossing-free subgraphs. *Ann. Discrete Math.*, 12:9–12, 1982.

[6] P. Brass, W. Moser and J. Pach. Graph drawings and geometric graphs. Chapter in *Research Problems in Discrete Geometry*, pp. 373–416. Springer, 2005.

[7] M. S. Chang, C. Y. Tang and R. C. T. Lee. Solving the Euclidean bottleneck matching problem by $k$-relative neighborhood graphs. *Algorithmica*, 8:177–194, 1992.

[8] V. Dujmović, K. Kawarabayashi, B. Mohar and D. R. Wood. Improved upper bounds on the crossing number. *Proc. SoCG '08*, 375–384, 2008.

[9] J. Gudmundsson, M. Hammar and M. van Kreveld. Higher order Delaunay triangulations. *Comput. Geom.*, 23:85–98, 2002.

[10] J. W. Jaromczyk and G. T. Toussaint. Relative neighborhood graphs and their relatives. *Proc. IEEE'92*, 80(9):1502–1517, 1992.

[11] E. de Klerk, D. V. Pasechnik and A. Schrijver. Reduction of symmetric semidefinite programs using the regular $*$-representation. *Math. Program.*, Ser. B, 109:613–624, 2007.

[12] F. T. Leighton. Complexity Issues in VLSI. *MIT Press*, 1983.

[13] J. Pach, ed. Towards a Theory of Geometric Graphs, Contemporary Mathematics, 342. *Amer. Math. Soc.*, 2004.

[14] J. Pach, R. Radoičić, G. Tardos and G. Tóth. Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete Comput. Geom.*, 36:527–552, 2006.

[15] J. Pach, J. Spencer and G. Tóth. New bounds on crossing numbers. *Discrete Comput. Geom.*, 24:623–644, 2000.

[16] J. Pach and G. Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997.

[17] R. B. Richter and C. Thomassen. Relations between crossing numbers of complete and complete bipartite graphs. *Amer. Math. Monthly*, 104(2):131–137, 1997.

[18] T.-H. Su and R.-Ch. Chang. The K-Gabriel graphs and their applications. *Proc. SIGAL'90*, 66–75, 1990.

[19] D. R. Wood and J. A. Telle. Planar decompositions and the crossing number of graphs with an excluded minor. *New York J. Math.*, 13:117–146, 2007.

[20] K. Zarankiewicz. On a problem of P. Turán concerning graphs. *Fund. Math.*, 41:137–145, 1954.

# Augmenting the edge connectivity of planar straight line graphs to three

Csaba D. Tóth* Pavel Valtr†

**Abstract**

We characterize the 2-connected and 2-edge connected planar straight line graphs, respectively, that have embedding preserving augmentations to 3-connected and 3-edge connected planar straight line graphs. If such an augmentation is possible, then it can be done using at most $n-2$ new edges in the worst case. These bounds are best possible.

## 1 Introduction

Connectivity augmentation is a classical problem in graph theory with application in network design. Given a graph $G(V, E)$, with vertex set $V$ and edge set $E$, and a constant $k \in \mathbb{N}$, find a minimum set $F$ of *new edges* such that $G'(V, E \cup F)$ is $k$-connected (respectively, $k$-edge connected). For $k = 2$, Eswaran and Tarjan [2] and Plesník [16] showed independently that both edge- and vertex-connectivity augmentation problems can be solved in linear time. Watanabe and Nakamura [21] proved that the edge-connectivity augmentation problem can be solved in polynomial time for any $k \in \mathbb{N}$. The runtime was later improved (using the edge-splitting technique of Lovász [11] and Mader [12]) by Frank [4] and by Nagamochi and Ibaraki [14]. Jackson and Jordán [7] proved that the vertex-connectivity augmentation problem can be solved in polynomial time for any $k \in \mathbb{N}$. For related problems, refer to surveys by Nagamochi and Ibaraki [15], and by Kortsarz and Nutov [10].

Sometimes it is desirable that the augmented graph $G'$ preserves certain properties of the input graph $G$. An augmentation is *planarity preserving* if both $G$ and $G'$ are planar (i.e., do not contain $K_5$ or $K_{3,3}$ as a minor). A given planar graph cannot be augmented to $k$-connected planar graph for an arbitrary large $k \in \mathbb{N}$. In particular, planarity preserving augmentation problems do not make make sense for $k > 5$, since every planar graph has a vertex of degree at most 5. Kant and Bodlaender [9] showed that the planarity preserving vertex-connectivity augmentation problem is NP-complete already for $k = 2$. Fiala and Mutzel [3] gave a $\frac{5}{3}$-approximation algorithm in $O(n^3)$ time for $k = 2$, and Kant and Bodlaender [9] gave a $\frac{5}{4}$-approximation in $O(n^3)$ time for $k = 3$. Rutter and Wolff [18] recently showed that the planarity preserving edge-connectivity augmentation problem is also NP-complete. Linear time algorithms for the planarity preserving versions are known for the case that $k = 2$ and the input $G$ is an outerplanar graph [8, 13]; and for the version of the problem where both the input $G$ and the output $G'$ are required to be outerplanar [5].

Abellanas *et al.* [1] extended the augmentation problem to planar straight line graphs (PSLG, for short), which is a planar graph together with a straight line embedding. An augmentation of a PSLG is *embedding preserving* if every edge of the input graph $G$ is mapped to the same line segment in the embedding of both $G$ and $G'$. The fixed embedding of the edges of the input graph severely limit the possible positions of the new edges. For example, planar straight line path cannot always be augmented to a planar straight line cycle. In fact, it may require $\lfloor n/2 \rfloor$ new edges to augment a planar straight line path to a 2-connected or 2-edge connected PSLG [1].

*Department of Mathematics, University of Calgary, AB, Canada, Email: cdtoth@ucalgary.ca Research done while visiting Tufts University. Supported in part by NSERC grant RGPIN 35586.

†Department of Applied Mathematics, Charles University, Prague, Czech Republic, Email: valtr@kam.mff.cuni.cz

A few combinatorial bounds are known on the maximum number of new edges needed for the embedding preserving augmentation of PSLGs. It is easy to see that every PSLG with $n \geq 2$ vertices and $c \geq 2$ connected components has an embedding preserving augmentation to a connected PSLG by adding at most $c-1 \leq n-1$ new edges. Every connected PSLG $G$ with $n \geq 3$ vertices and $b \geq 2$ distinct 2-connected blocks has an embedding preserving augmentation to a 2-connected PSLG by adding at most $b-1 \leq n-2$ new edges [1]. Every connected PSLG with $n \geq 3$ vertices has an embedding preserving augmentation to a 2-edge connected PSLG by adding at most $\lfloor (2n-2)/3 \rfloor$ new edges [19]. These bounds are tight in the worst case [1].

## 2 Which graphs can be augmentated?

In this section we study 3-connected and 3-edge connected triangulations, and then specify which PSLGs have embedding preserving augmentation to such straight line triangulations.

For a set $V$ of points in the plane (e.g., a vertex set of a PSLG), let $\mathrm{ch}(V)$ denote the convex hull of $V$. For a PSLG $G$ we use the shorthand notation $\mathrm{ch}(G) := \mathrm{ch}(V(G))$. A *chord* of $\mathrm{ch}(G)$ is an edge that connects two vertices in $\mathrm{ch}(G)$, which are not adjacent along $\mathrm{ch}(G)$. A *triangulation* of a point set $V$ in the plane is a connected PSLG on the vertex set $V$ such that all bounded faces are triangles and the unbounded face is the complement of $\mathrm{ch}(V)$. If $V$ is a set of $n$ points, $k$ of which in the interior of $\mathrm{ch}(V)$, then the triangulation of $V$ consists of $2n+k-2$ triangular faces. Every PSLG, with no three collinear vertices, can be augmented to a triangulation.

**Proposition 2.1.** *Let $T$ be a triangulation on $n \geq 4$ vertices, no three of which are collinear. $T$ is 3-connected if and only if no edge of $T$ is a chord of* $\mathrm{ch}(V)$. *$T$ is 3-edge connected if and only if no triangular face of $T$ has two edges along* $\mathrm{ch}(V)$.

**Proof.** If an edge $e \in E(T)$ is a chord of $\mathrm{ch}(V)$, then the removal of the two endpoints of $e$ disconnects $T$. Assume that $T$ can be disconnected by removing two vertices $u, v \in V$. Then there is a closed curve $\gamma$ that separates the components of $G \setminus \{u, v\}$ such that $\gamma$ passes through $u$ and $v$ but does not cross any edge of $T$. So $\gamma$ traverses exactly two faces, which are incident to both $u$ and $v$. The two faces cannot be adjacent, otherwise there would be no vertices in the interior of $\gamma$. If a triangular face is incident to both $u$ and $v$, it is adjacent to edge $uv$. Hence, at least one of the faces is the outer face, and the other face is a triangle not adjacent to the outer face. So $u$ and $v$ are two nonconsecutive vertices along $\mathrm{ch}(G)$, and they are adjacent in $T$. That is, $uv$ is a chord of $T$.

If a triangular face of $T$ has two edges along $\mathrm{ch}(V)$, then the removal of these two edges disconnects $T$. Assume now that $T$ can be disconnected by removing two edges $e, f \in E(T)$. Then there is a closed curve $\gamma$ that separates the components of $G \setminus \{e.f\}$ such that $\gamma$ crosses $e$ and $f$ but no other edges (and does not pass through any vertex). So $\gamma$ traverses exactly two faces. There faces share two edges, $e$ and $f$. Since no two triangles share two edges, one of the faces is the outer face, and the other face is a triangle. Since any two edges of a triangle are adjacent, $e$ and $f$ are adjacent along $\mathrm{ch}(T)$. □

**Proposition 2.2.** *A PSLG $G$ with $n \geq 4$ vertices has an embedding preserving augmentation to a 3-connected PSLG if and only if the vertices are not in convex position or no edge of $G$ is a chord of* $\mathrm{ch}(G)$.

**Proof.** Let $G'$ be an embedding preserving augmentation of $G$. Since additional edges can only increase the connectivity, we may assume that $G'$ is a triangulation with $E(G) \subseteq E(G')$. If $V(G)$ is in convex position, then any triangulation of $V(G)$ contains a vertex of degree two, so $G'$ cannot be 3-connected. If $G$ contains a chord of $\mathrm{ch}(G)$, then so does $G'$, which implies that $G'$ cannot be 3-connected.

Assume that $G$ is a PSLG such that $V(G)$ is not in convex position and $E(G)$ contains no chord of $\mathrm{ch}(G)$. We augment $G$ to a triangulation $T$ such that none of the new edges is a chord of $\mathrm{ch}(G)$. By Proposition 2.1, $T$ will be 3-connected. First augment $G$ with all edges along the convex hull $\mathrm{ch}(G)$; then for every vertex in the interior of $\mathrm{ch}(G)$, greedily add new incident edges as long as they do not cross previous edges. Denote the resulting graph by $G'$. Note that no edge of $G'$ is a chord of $\mathrm{ch}(G)$,

and every bounded face of $G'$ is convex. We show that every bounded face of $G'$ is a triangle, and so $G'$ is the required triangulation. Assume, to the contrary, that $G'$ has a convex face $F$ with at least 4 vertices. All vertices of $F$ are on the convex hull of $G$, otherwise a vertex of $F$ in the interior of $\mathrm{ch}(G)$ could be connected to all other vertices of $F$ (thereby triangulating $F$). However, the edges of $F$ are not chords, so all edges of $F$ lie along $\mathrm{ch}(G)$. Therefore, all vertices of $G'$ are incident to $F$, that is, $V(G)$ is in convex position, contradicting our assumption. □

**Proposition 2.3.** *A* PSLG $G$ *with* $n \geq 4$ *vertices has an embedding preserving augmentation to a 3-edge connected* PSLG *if and only if* $G$ *has no edge* $e$ *such that* $e$ *is a chord of* $\mathrm{ch}(G)$ *and all vertices of* $G$ *on one side of* $e$ *are incident to* $\mathrm{ch}(G)$.

**Proof.** Let $G'$ be an embedding preserving augmentation of $G$. Since additional edges can only increase the edge-connectivity, we may assume that $G'$ is a triangulation with $E(G) \subseteq E(G')$. If $V(G)$ is in convex position, then any triangulation of $V(G)$ contains a vertex of degree two, so $G'$ cannot be 3-edge connected. If $G$ contains a chord $uv$ of $\mathrm{ch}(G)$ such that all vertices on one side of $uv$ are incident to $\mathrm{ch}(G)$, then let $V_1$ be the set of nodes on this side of $uv$ together with the endpoints $u$ and $v$. Then the subgraph of $T$ induced by $V_1$ is a triangulation $T_1$ on at least three vertices. If $|V_1| = 3$, then the single vertex in $V_1 \setminus \{u, v\}$ has degree two in $T'$. If $|V_1| \geq 4$, then note that every triangulation on at least 4 vertices contains two nonadjacent vertices of degree two in (since the dual graph of the triangulation has at least two leafs). Hence, there is a vertex in $V_1 \setminus \{u, v\}$ of degree two in $T'$. The degree of this node is 2 in $T$ as well, since it cannot be connected to the vertices on the opposite side of $uv$.

Assume that $G$ is a PSLG such that $V(G)$ is not in convex position, and $E(G)$ contains no chord of $\mathrm{ch}(G)$ such that the vertices on one side are all incident to $\mathrm{ch}(G)$. We augment $G$ to a triangulation $T$ such that no triangle has two edges along $\mathrm{ch}(G)$. By Proposition 2.1, $T$ will be 3-edge connected. First augment $G$ with all edges along the convex hull $\mathrm{ch}(G)$; then for every vertex in the interior of $\mathrm{ch}(G)$, greedily add new incident edges as long as they do not cross previous edges. Denote the resulting graph by $G'$. Note that no edge of $G'$ is a chord of $\mathrm{ch}(G)$, and every bounded face of $G'$ is convex. If every bounded face of $G'$ is a triangle, then $G'$ is the required triangulation. Consider a convex face $F$ of $G'$ with at least 4 vertices. All vertices of $F$ are on the convex hull of $G$, otherwise a vertex of $F$ in the interior of $\mathrm{ch}(G)$ could be connected to all other vertices of $F$ (thereby triangulating $F$). At least two edges of $F$ are chords of $\mathrm{ch}(G)$, otherwise $V(G)$ would be in convex position (if no edge of $F$ is a chord) or would have a chord with all vertices on one side incident to $\mathrm{ch}(G)$ (if exactly one edge of $F$ is a chord). Let $e$ and $f$ be two edges of $F$ that are chords of $G$. Note that $G$ must have vertices in the interior of $\mathrm{ch}(G)$ on the opposite side of $e$ and $f$ each. Triangulate $F$ with edges that separate $e$ and $f$. The new edges are chords such that there are vertices in the interior of $\mathrm{ch}(G)$ on both sides of the chord. Therefore the resulting triangulation is 3-edge connected. □

We call a PSLG with at least 4 vertices *3-augmentable* (respectively, *3-edge augmentable*), if it satisfies the necessary and sufficient condition in Propositions 2.2 (resp., Proposition 2.3) and thus has an embedding preserving augmentation to a 3-connected (resp., 3-edge connected) triangulation.

# 3 Connectivity augmentation from two to three

A *2-cut* (resp., *2-bridge*) in a graph is a pair of vertices (resp., edges) whose removal disconnects the graph. We show that the number of 2-cuts (reps., number of 2-bridges) can be decreased in a 3-augmentable (resp., 3-edge augmentable) PSLG by adding a single new edge.

**Proposition 3.1.** *Let* $G$ *be a 2-connected 3-augmentable* PSLG*, and let* $\{u, v\} \subset V(G)$ *be 2-cut in* $G$ *such that* $G - \{u, v\}$ *has* $k \geq 2$ *components. Then* $G$ *has an embedding preserving augmentation to a 3-augmentable* PSLG $G'$ *with one new edge such that* $G' - \{u, v\}$ *has* $k - 1$ *components.*

**Proof.** Since $G$ is 3-augmentable, it has an augmentation to a 3-connected triangulation $T$. Since $T$ is 3-connected, it contains an edge $e$ between two distinct components of $G - \{u, v\}$. Augment $G$ with this edge $e$. Clearly, $G' - \{u, v\}$ has one fewer components than $G - \{u, v\}$ □

**Proposition 3.2.** *Let $G$ be a 2-edge connected 3-edge augmentable* PSLG*, and let $\{e, f\} \subset E(G)$ be a 2-bridge such that $G - \{e, f\}$ is disconnected. Then $G$ has an embedding preserving augmentation to a 3-edge augmentable* PSLG *$G'$ with one new edge $\hat{e}$ such that $G' - \{e, f\}$ is connected.*

**Proof.** Since $G$ is 3-edge augmentable, it has an augmentation to a 3-edge connected triangulation $T$. Since $T$ is 3-edge connected, it contains an edge $\hat{e}$ between the two components of $G-\{e, f\}$. Augment $G$ with this edge $\hat{e}$. Clearly, $G' - \{e, f\}$ has one fewer components than $G - \{e, f\}$ □

The new edge $\hat{e}$ of $G'$ in Proposition 3.2 cannot be part of any 2-bridge: If $\{\hat{e}, f\}$ were a 2-bridge of $G'$ for some $f \in E(G)$, then $f$ would be a bridge of $G$. So there are strictly fewer 2-bridges in $G'$ than in $G$.

**Increasing edge-connectivity from two to three.** Note that the 2-bridges of a graph form "cliques" in the following sense. If $\{e_1, e_2\} \subset E(G)$ and $\{e_2, e_3\} \subset E(G)$ are both 2-bridges in graph $G$, then $\{e_1, e_3\}$ is also a 2-bridge in $G$ (since $e_1$ and $e_3$ are both bridges in $G - \{e_2\}$). A *clique of 2-bridges* is a set of edges of $G$, any two of which is a 2-bridge of $G$. We have seen that cliques of 2-bridges are pairwise disjoint. If a clique of 2-bridges contains $k \geq 2$ edges, then the simultaneous removal of all $k$ edges splits $G$ into exactly $k$ components.

By repeatedly applying Proposition 3.2, we have the following result:

**Proposition 3.3.** *Let $G$ be a 2-edge connected 3-edge augmentable* PSLG*, and assume that it has $m \geq 1$ maximal cliques of 2-brides which contain $k_1, k_2, \ldots, k_m \geq 2$ edges. Then $G$ has an embedding preserving augmentation to a 3-edge connected* PSLG *with at most $(\sum_{i=1}^{m} k_i) - m$ new edges.*

**Theorem 3.4.** *Every 2-edge connected 3-edge augmentable* PSLG *with $n \geq 4$ vertices has an embedding preserving augmentation to a 3-edge connected* PSLG *by adding at most $n - 2$ new edges. This bound is best possible.*

**Proof.** Let $G$ be a 2-edge connected 3-edge augmentable PSLG with $n \geq 4$ vertices. Assume that $G$ has $m \geq 1$ maximal cliques of 2-brides which contain $k_1, k_2, \ldots, k_m \geq 2$ edges. Since the simultaneous removal of all 2-bridges splits $G$ into $\sum_{i=1}^{m} k_i$ nonempty components, we have $\sum_{i=1}^{m} k_i \leq n$, hence $(\sum_{i=1}^{m} k_i) - m \leq n - 1$. By Proposition 3.3, our proof is complete if $(\sum_{i=1}^{m} k_i) - m \leq n - 2$ new edges.

Assume that $(\sum_{i=1}^{m} k_i) - m = n - 1$. Then $G$ has one clique of 2-bridges, and the removal of all 2-bridges splits $G$ into $n$ singletons. It follows that $G$ is a Hamiltonian cycle. Since the vertices of $G$ are not in convex position, there is an edge $e$ of the convex hull $\mathrm{ch}(G)$ which is not an edge of $G$. Let $G' = G + \{e\}$. The endpoints of $e$ partition the Hamiltonian cycle $G$ into two paths $P_1$ and $P_2$. Therefore $G'$ has two cliques of 2-bridges: the edges of $P_1$ and the edges of $P_2$. Since both endpoints of $e$ are incident to $\mathrm{ch}(G)$, all internal vertices of one of the path, say $P_1$, lie in the interior of $\mathrm{ch}(G)$.

Note that $G$ is a simple polygon, in which $e$ is an external diagonal. In the triangulation of the interior of the polygon, there is an edge $f$ that connects two internal vertices of paths $P_1$ and $P_2$. Let $G'' = G + \{e, f\}$. The four distinct endpoints of $e$ and $f$ partition the Hamiltonian path $G$ into four paths, each containing at least one edge. $G''$ has at most four cliques of 2-bridges, which correspond to these paths (the paths with tow or more edges). By Proposition 3.2, $G''$ has an embedding preserving augmentation to a 3-edge connected PSLG with at most $n - 4$ new edges. Together with $e$ and $f$, we can augment $G$ to a 3-edge connected PSLG with $n - 2$ new edges.

The following construction shows that this bound is best possible for every $n \geq 4$. Let $H$ be a straight line Hamiltonian cycle with $n-1$ vertices in convex position and one vertex lying in the interior of $\mathrm{ch}(H)$ (see Fig. 1). $H$ has a unique embedding preserving augmentation to a 3-edge connected PSLG: Connect the vertex in the interior of $\mathrm{ch}(H)$ to all $n - 3$ nonadjacent vertices, and connect its two neighbors by an $(n - 2)$nd edge. □

**Increasing connectivity from two to three.**

**Theorem 3.5.** *Every 2-connected 3-augmentable* PSLG *with $n \geq 4$ vertices has an embedding preserving augmentation to a 3-connected* PSLG *with at most $n - 2$ new edges. This bound is best possible.*

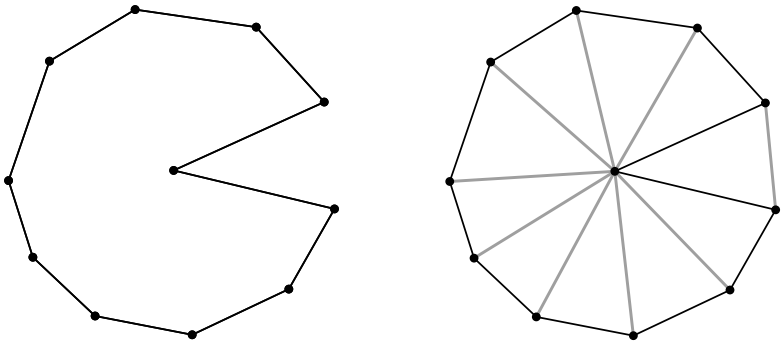


Figure 1: Left: A Hamiltonian cycle with $n-1$ vertices in convex position and one vertex lying in the interior of $\mathrm{ch}(H)$. Right: The only possible embedding preserving augmentation to a 2-connected (resp., 2-edge connected) PSLG.

**Proof.** The decomposition of a 2-connected graph into 3-connected components was described by Tutte [20] (see also [17]), and it can be computed in $O(n)$ time [6]. The number of 3-connected components is at most $n$, with equality if and only if $G$ is a cycle. By Proposition 3.1, as long as there is a 2-cut, we can decrease the number of 3-connected components by adding a new straight line edge between two distinct 3-connected components. So if $G$ is not a cycle, we can eliminate all 2-cuts by successively invoking Proposition 3.1 at most $n-2$ times.

If $G$ is a Hamiltonian cycle, then we can argue analogously to the proof of Theorem 3.4. We can add an edge $e$ on the convex hull, which splits $G$ into two paths, and another edge $f$ connecting two internal vertices, one from each of the two paths. The resulting graph has a $K_4$ minor, and so it has at most $n-3$ tri-connected components. Hence it can be augmented to a 3-connected PSLG by at most $n-4$ invocations of Proposition 3.1. That is, we can augment a $G$ to a 3-connected PSLG with at most $n-2$ new edges. The Hamiltonian cycle $H$ described in the proof of Theorem 3.4 (c.f., Fig. 1) shows that the upper bound $n-2$ is best possible. □

# 4 Planar topological graphs

We may consider embedding preserving augmentations of a PSLG such that the new edges may be embedded as arbitrary continuous arcs (not necessarily straight line segments). In this case, the input is a PSLG and the output is an *planar topological graph* (for short, PTG). If the new edges are not restricted to straight line segments, then the constraint in Proposition 2.1 and Proposition 2.2 no longer apply. Any PSLG with $n \geq 4$ vertices has an embedding preserving augmentation to a 3-connected and 3-edge connected PTG: Any PSLG can be triangulated with curved edges such that the outer face is a triangle. In fact, only the edges lying in the exterior of the convex hull $\mathrm{ch}(V)$ must have non-straight embeddings. Determining the minimum number of new edges that can augment any PSLG with $n \geq 4$ vertices to a 3-connected or 3-edge connected PTG remains an open problem.

**Conjecture 4.1.** Every 2-connected PSLG $G$ with $n \geq 4$ vertices can be augmented to a 3-connected topological graph by adding at most $\lceil \frac{4}{5}(n-2) \rceil$ new edges.

**Conjecture 4.2.** Every 2-edge connected PSLG $G$ with $n \geq 4$ vertices can be augmented to a 3-edge connected topological graph by adding at most $\frac{4}{5}n - O(n)$ new edges.

These bounds could not be improved. A lower bound construction consists of a (straight line) triangulation on $k$ vertices, which has $2k-4$ faces, and a path with two internal vertices inserted in every face. So $G$ has a total of $n = k + 2(2k-4) = 5k-8$ vertices, and we have $k = (n+8)/5$. In order to increase the degree of every vertex in the interior of the triangular faces to 3, we need at least $2(2k-4) = \frac{4}{5}(n-2)$ new edges. This is also a lower bound on the number of curved edges required for augmenting the connectivity or edge-connectivity to three.

## References


[1] M. Abellanas, A. García, F. Hurtado, J. Tejel, and J. Urrutia, Augmenting the connectivity of geometric graphs, *Comput. Geom. Theory Appl.* **40** (3) (2008), 220–230.

[2] K. P. Eswaran and R. E. Tarjan, Augmentation problems, *SIAM J. Comput.* **5** (4) (1976), 653-–665.

[3] S. Fialko and P. Mutzel, A new approximation algorithm for the planar augmentation problem in *Proc. 9th ACM-SIAM Sympos. on Discrete Algorithms*, ACM Press, 1998, pp. 260–269.

[4] A. Frank, Augmenting graphs to meet edge-connectivity requirements, *SIAM J. Discrete Math.* **5** (1) (1992), 22–53.

[5] A. García, F. Hurtado, M. Noy, and J. Tejel, Augmenting the connectivity of outerplanar graphs, *Algorithmica* (2007), to appear. `http://dx.doi.org/10.1007/s00453-008-9167-1`

[6] J. E. Hopcroft, R. E. Tarjan, "Dividing a graph into triconnected components," SIAM J. Comput., 1973, pp. 135-158. linear time algorithm for ear decomposition

[7] B. Jackson and T. Jordán, Independence free graphs and vertex connectivity augmentation, *J. Comb. Theory Ser. B* **94** (2005), 31-–77.

[8] G. Kant, Augmenting outerplanar graphs, *J. Algorithms* **21** (1996), 1—25.

[9] G. Kant and H. L. Bodlaender, Planar graph augmentation problems, in: *Proc. 2nd Workshop on Algorithms and Data Structures*, vol. 519 of LNCS, Springer, 1991, pp. 286–298.

[10] G. Kortsarz and Z. Nutov, Approximating minimum cost connectivity problems, Chap. 58 in *Handbook of Approximation Algorithms and Metaheuristics (T. F. Gonzalez, ed.)*, CRC Press, Boca Raton, 2007.

[11] L. Lovász, *Combinatorial Problems and Excercises*, North-Holland, Amsterdam, 1979.

[12] W. Mader, A reduction method for edge-connectivity in graphs, *Ann. Discr. Math.* **3** (1978), 145–164.

[13] H. Nagamochi and P. Eades, An edge-splitting algorithm in planar graphs, *J. Combin. Opt.* **7** (2) (2003), 137-159.

[14] H. Nagamochi and T. Ibaraki, Augmenting edge-connectivity over the entire range in $\tilde{O}(nm)$ time, *J. Algorithms* **30** (1999), 253-–301.

[15] H. Nagamochi and T. Ibaraki, Graph connectivity and its augmentation: applications of MA orderings, *Discrete Appl. Math.* **123** (2002), 447-–472.

[16] J. Plesník, Minimum block containing a given graph, *Arch. Math.* **27** (6) (1976), 668–672.

[17] R. B. Richter, Decomposing infinite 2-connected graphs into 3-connected components, *Electr. J. Comb.* **11** (1) (2004).

[18] I. Rutter and A. Wolff, Augmenting the connectivity of planar and geometric graphs, in *Proc. Conf. Topological Geom. Graph Theory (Paris, 2008), Electronic Notes in Discrete Mathematics*, 55–58.

[19] Cs. D. Tóth, Connectivity augmentation in plane straight line graphs, *Proc. Intl. Conf. on Topological and Geometric Graph Theory (Paris, 2008)*, pp. 51–54.

[20] W. T. Tutte, Connectivity in Graphs, University of Toronto Press, 1966.

[21] T. Watanabe and A. Nakamura, Edge-connectivity augmentation problems, *Comp. Sys. Sci.* **35** (1) (1987), 96–144.

[22] H. Whitney, "Non-separable and planar graphs," Tr ans. Amer. Math. Soc. 34, 1932, pp. 339–362.

## Sesión 4, 11:30–12:10

# Triangular Configurations and Strictly Upper-Triangular Matrices Lie Algebras

Manuel Ceballos, Juan Núñez * Ángel F. Tenorio†

**Abstract**

This paper shows how to associate the Lie algebra $\mathfrak{g}_n$ ($n \in \mathbb{N}$), of $n \times n$ strictly upper-triangular matrices, with a specific 2-dimensional triangular configuration. The properties of those configurations are analyzed here by using Graph Theory and, finally, applied to obtain some results about Representation Theory for nilpotent Lie algebras.

## 1 Introduction

One of the most important goals in Mathematics is to find new relations between their different fields. In this way, different techniques can be used to solve old and new problems as well as improving well-known theories and revealing new ones.

The research on Lie Theory is very extended due to its applications to Engineering, Physics and Applied Mathematics. Besides, we have to consider all the theoretical study about Lie algebras, because several aspects on this topic still have to be studied and established. For example, the classification of nilpotent and solvable Lie algebras is considered an open problem, although other types of Lie algebras (like semisimple and simple) were completely classified at the end of the 19th century. Other open questions and results about Lie algebras belong to the field of Representation and Structure Theory. With respect to these two fields, we are interesting, respectively, in minimal faithful representations by using special types of matrices (see [1, 3]) and in maximal abelian subalgebras (see [2, 6]).

Regarding Graph Theory, this branch of Mathematics is also running in a very high level in both theory and applications. Indeed, Graph Theory is nowadays used as an essential tool for dealing with problems of a lot of subjects due to its wide range of applications. In fact, one of our main goals is to establish a relation between Graph Theory and Lie Theory (more concretely, nilpotent Lie algebras) in a similar way as [5, 7]. To obtain this relation between graphs and nilpotent Lie algebras, we consider the Lie algebras $\mathfrak{g}_n$ and construct representations by graphs. The algebras $\mathfrak{g}_n$ are very important because every nilpotent Lie algebra is isomorphic to a subalgebra of some Lie algebra $\mathfrak{g}_n$ [12, Proposition 3.6.6].

The properties of triangular configurations can be used for solving the open problems previously mentioned, obtaining advances and improvements in the research about nilpotent Lie algebras. Let us note that Lie algebras $\mathfrak{g}_n$ and nilpotent Lie algebras, in general, are very important because they are usually applied as a tool for several topics in Physics. Indeed, Lie algebras and Lie groups are widely used in this science at present. For example, the study of symmetries [9, 10] is a very classical use of Lie Theory. Other topics applying Lie Theory are Einstein's spaces and General Relativity [8, 11].

In our own opinion, the tools introduced in this paper are very useful to give a step forward in the classification problem of nilpotent Lie algebras: by obtaining the classification of all the possible triangular configurations. In this sense, the resulting method for classifying triangular configurations could be easier than the original method for classifying Lie algebras directly.

*Departamento de Geometría y Topología. Facultad de Matemáticas. Universidad de Sevilla. Aptdo. 1160. 41080-Seville (Spain), mceballos@us.es, jnvaldes@us.es

†Dpto. de Economía, Métodos Cuantitativos e H.ª Económica. Escuela Politécnica Superior. Universidad Pablo de Olavide. Ctra. Utrera km. 1, 41013-Seville (Spain), aftenorio@upo.es

# 2 Preliminaries on Lie algebras

Some preliminary concepts on Lie algebras are recalled in this section, although the interested reader can consult [12] for a general overview. From here on, only finite-dimensional complex Lie algebras are considered.

## 2.1 Definitions and Notations

**Definition 2.1.** A Lie algebra $\mathfrak{g}$ is a vector space endowed with a second inner bilinear composition law $[\cdot,\cdot]$, called bracket product and satisfying the following two conditions:

1. $[X,X]=0,\ \forall X\in\mathfrak{g}$;
2. $[[X,Y],Z]+[[Y,Z],X]+[[Z,X],Y]=0,\ \forall X,Y,Z\in\mathfrak{g}$.

Condition 2 is usually named Jacobi's identity and is denoted by $J(X,Y,Z)=0$.

**Definition 2.2.** Let $\mathfrak{g}$ be a Lie algebra with a basis $\mathcal{B}=\{e_1,\ldots,e_n\}$. The law of $\mathfrak{g}$ with respect to this basis is completely determined by the structure constants or Maurer-Cartan constants:

$$[e_i,e_j]=\sum c_{i,j}^h e_h,\quad \text{where } 1\leq i,j\leq n.$$

**Definition 2.3.** Given a Lie algebra $\mathfrak{g}$, its *lower central series* is:

$$\mathcal{C}^1(\mathfrak{g})=\mathfrak{g},\ \mathcal{C}^2(\mathfrak{g})=[\mathfrak{g},\mathfrak{g}],\ \ldots,\ \mathcal{C}^k(\mathfrak{g})=[\mathcal{C}^{k-1}(\mathfrak{g}),\mathfrak{g}],\ \ldots$$

So $\mathfrak{g}$ is called *nilpotent* if there exists $m\in\mathbb{N}$ such that $\mathcal{C}^m(\mathfrak{g})\equiv\{0\}$.

**Definition 2.4.** A Lie algebra $\mathfrak{g}$ is said to be *abelian* if $[v,w]=0$, for all $v,w\in\mathfrak{g}$.

**Definition 2.5.** The center of a given Lie algebra $\mathfrak{g}$ is the ideal $cen(\mathfrak{g})$ defined as

$$cen(\mathfrak{g})=\{X\in\mathfrak{g}\quad|\quad [X,Y]=0\,,\ \forall Y\in\mathfrak{g}\}.$$

**Definition 2.6.** Fixed and given a Lie algebra, $\mathfrak{g}$, its *maximal abelian dimension*, denoted by $\mathcal{M}(\mathfrak{g})$, is the maximum among the dimensions of its abelian Lie subalgebras.

## 2.2 Lie Algebras $\mathfrak{g}_n$

Let us denote by $\mathfrak{g}_n$ the nilpotent Lie algebra of $n\times n$ strictly upper triangular matrices, with $n\in\mathbb{N}\ \setminus\{1\}$. The expression of the vectors in $\mathfrak{g}_n$ is the following:

$$g_n(x_{r,s})=\begin{pmatrix} 0 & x_{1,2} & \cdots & x_{1,n-1} & x_{1,n}\\ 0 & 0 & \cdots & x_{2,n-1} & x_{2,n}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \cdots & 0 & x_{n-1,n}\\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}\qquad (x_{i,j}\in\mathbb{C}\ ). \tag{1}$$

The dimension of $\mathfrak{g}_n$ is $\frac{n(n-1)}{2}$. The law of $\mathfrak{g}_n$ is expressed as follows:

$$[X_{i,j},X_{j,k}]=X_{i,k},\quad \text{for } 1\leq i<j<k\leq n. \tag{2}$$

with respect to the basis given by the vectors:

$$X_{i,j}=g_n(x_{r,s}),\qquad \text{with } x_{r,s}=\begin{cases} 1, & \text{if } (r,s)=(i,j);\\ 0, & \text{if } (r,s)\neq(i,j)\end{cases} \tag{3}$$

where $i=1,\ldots,n-1$ and $j=i+1,\ldots,n$. Consequently, the center of $\mathfrak{g}_n$ is generated by $\{X_{1,n}\}$, because this vector does not appear as factor in any nonzero brackets.

# 3 Associating Triangular Configurations with Lie Algebras

This section is devoted to show a method which allows us to associate a combinatorial structure with a given Lie algebra. After explaining the method, we apply it to the Lie algebras $\mathfrak{g}_n$. The interested reader can consult [5] for more details about the method explained in this section.

If $\mathfrak{g}$ is a $n$-dimensional Lie algebra with a basis $B = \{e_1, \ldots, e_n\}$, we can write:

$$[e_i, e_j] = \sum_{k=1}^{n} c_{i,j}^k e_k, \tag{4}$$

where $c_{i,j}^k$ are the structure constants of $\mathfrak{g}$. The pair $(\mathfrak{g}, B)$ is associated with a triangular configuration by the following method:

a) Each $e_i \in B$ is represented by one point (vertex) labeled with the index $i$.

b) Given three vertices $i < j < k$, we draw the full triangle with these vertices. The weights $c_{i,j}^k$, $c_{j,k}^i$ and $c_{i,k}^j$ are assigned respectively to the edges connecting the vertices $i$ and $j$, the vertices $j$ and $k$, and the vertices $i$ and $k$. An example is given in Figure 1.

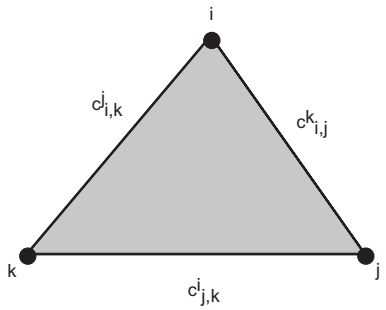



Figure 1: Full triangle.

However, there are some exceptions:

b1) If $c_{i,j}^k = c_{j,k}^i = c_{i,k}^j = 0$, the triangle is not drawn.

b2) If some structure constant is zero, its corresponding edge is drawn by using a discontinuous line (called ghost edge).

b3) If the triangles of vertices $i, j, k$ and $i, j, l$ (with $1 \leq i < j < k, l \leq n$) satisfy that $c_{i,j}^k = c_{i,j}^l$, only one edge is drawn between the vertices $i$ and $j$. This edge is shared by both triangles (see Figure 2).

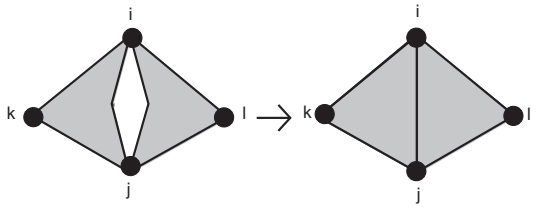



Figure 2: Triangles with a shared edge.

Consequently, given the Lie algebra $\mathfrak{g}$ and the basis $B$, there exists a triangular configuration associated with them. However, the configuration depends on the basis $B$ considered for $\mathfrak{g}$.

If a vertex $i$ is isolated in the triangular configuration of $\mathfrak{g}$, the vector $e_i \in B$ belongs to $cen(\mathfrak{g})$. The converse is not true, because there exist vectors in $cen(\mathfrak{g})$ associated with non-isolated vertices in the triangular configuration.

The following lemma will be used in the next sections, as well as the definitions related to edges and degrees in the configuration.

**Lemma 3.1.** *Fixed and given a triangular configuration with $n$ vertices, all of them of degree $r$, the number of edges in this configuration is equal to $\frac{nr}{2}$.*

**Definition 3.2.** The *degree*, $d_i$ or $deg(i)$, of a vertex $i$ in a triangular configuration is the number of edges incident to $i$. Obviously, all the types of edges (including ghost edges) are considered in this definition.

# 4 Lie Algebras $\mathfrak{g}_n$ and Triangular Configurations

In this section, we study how to obtain the triangular configuration associated with a Lie algebra $\mathfrak{g}_n$ for $n \in \mathbb{N}$.

First, an order is defined in the basis of $\mathfrak{g}_n$ to associate each vector in the basis with a vertex in the triangular configuration. In our case, the order is based on considering the order of the rows and then the order of the columns:

$$\begin{gathered}\{X_{1,2}, X_{1,3}, \ldots, X_{1,n}\} \quad \text{with} \quad \{e_1, e_2, \ldots, e_{n-1}\} \\ \{X_{2,3}, X_{2,4}, \ldots, X_{2,n}\} \quad \text{with} \quad \{e_n, e_{n+1}, \ldots, e_{2n-3}\} \\ \vdots \\ \{X_{n-1,n}\} \quad \text{with} \quad \{e_{\frac{n(n-1)}{2}}\}\end{gathered}$$

Now, we study the relation between the center of $\mathfrak{g}_n$ and its triangular configurations.

As we affirm previously, isolated vertices in a triangular configuration correspond to vectors in the center of the algebra. Next we prove that the converse is not true, using the Lie algebras $\mathfrak{g}_n$ as a counterexample.

**Lemma 4.1.** *The center of $\mathfrak{g}_n$ is associated with a vertex which is only incident to ghost edges.*

*Proof.* The center of $\mathfrak{g}_n$ is generated by $X_{1,n}$, because $X_{1,n}$ does not appear as a factor in any nonzero brackets. The following $n-2$ brackets are the only ones whose result is $X_{1,n}$ in the law of $\mathfrak{g}_n$:

$$[X_{1,j}, X_{j,n}] = X_{1,n} \quad \text{for } 2 \leq j \leq n-1. \tag{5}$$

Consequently, the vector $X_{1,n}$ is associated with the vertex $n-1$, which satisfies the following conditions:

1. It belongs to $(n-2)$ full triangles.
2. It is only incident to ghost edges (because $X_{1,n}$ is not a factor in nonzero brackets).

□

Next, the two following natural numbers are defined:

$$\alpha = \frac{n(n-1)}{2}, \quad \beta = \binom{n}{3};$$

where $n \in \mathbb{N}$ with $n \geq 3$.

We have the following theorems related to the triangular configuration associated with the Lie algebra $\mathfrak{g}_n$.

**Theorem 4.2.** *Let $n \in \mathbb{N}$ be with $n \geq 3$. The Lie algebra $\mathfrak{g}_n$ is associated with a triangular configuration verifying:*

1. *The number of vertices is $\alpha$.*
2. *The degree of each vertex is $2(n-2)$.*
3. *The number of edges is $3\beta$.*

*Proof.* According to the method expounded in Section 3, the triangular configuration associated with $\mathfrak{g}_n$ has $\alpha = dim(\mathfrak{g}_n)$ vertices.

In virtue of previous Lemma, if every vertex has degree $2(n-2)$, the number of edges is:

$$\frac{\alpha\, 2(n-2)}{2} = \alpha\,(n-2) = 3\beta. \tag{6}$$

Consequently, we only have to prove that each vertex in the triangular has degree $2(n-2)$. This is done by considering an iterative procedure.

For $n = 3$, we can check that each vertex has degree $2 = 2(3-2)$. Now, let us suppose that the result is true for $\mathfrak{g}_{n-1}$ ; that is, each vertex in the triangular configuration has degree $2(n-3)$. In virtue of this assumption, we prove the result for $\mathfrak{g}_n$. Remember that a basis of $\mathfrak{g}_n$ is obtained by adding the vectors $\{X_{1,n}, \ldots, X_{n-1,n}\}$ to the basis of $\mathfrak{g}_{n-1}$.

Hence, some new nonzero brackets are added. More concretely, fixed $X_{i,j}$ with $1 \leq i < j \leq n-1$, the bracket $[X_{i,j}, X_{j,n}] = X_{i,n}$ is nonzero for each vector in $\mathfrak{g}_{n-1}$. Therefore, the number of nonzero brackets increases in $\frac{(n-1)(n-2)}{2}$ units.

Translating this to the triangular configurations, a new full triangle appears for each vertex associated with $\mathfrak{g}_{n-1}$. So the degree of each vertex increases in two units. Therefore, by using the iterative assumption, the degree of each vertex is $2(n-3)+2 = 2(n-2)$. □

**Theorem 4.3.** *Let $n \in \mathbb{N}$ be with $n \geq 3$. The Lie algebra $\mathfrak{g}_n$ is associated with a triangular configuration of $\alpha$ vertices, $3\beta$ edges and $\beta$ full triangles with $2\beta$ ghost edges.*

*Proof.* According to the method given in Section 3 to obtain triangular configurations, each nonzero bracket involves a full triangle, formed by 2 ghost edges. □

# 5 Application to Representation Theory

Next, we show an application of triangular configurations for computing minimal faithful matrix representation of nilpotent Lie algebras. Our main goal is the following: Using the properties of the triangular configuration associated with $\mathfrak{g}_n$, we want to find a triangular configuration inside and isomorphic to the given nilpotent Lie algebra.

So, fixed and given a nilpotent Lie algebra $\mathfrak{g}$, there exists $n \in \mathbb{N}$ such that $\mathfrak{g}$ is isomorphic to a subalgebra of $\mathfrak{g}_n$. It would be interesting to know which is the minimal $n \in \mathbb{N}$ such that $\mathfrak{g}$ is contained as a subalgebra in $\mathfrak{g}_n$,but not in $\mathfrak{g}_{n-1}$.

In this sense, Burde [4] studied the minimal dimension for faithful representations of a given Lie algebra $\mathfrak{g}$, considering faithful $\mathfrak{g}$-modules. This invariant was defined as:

$$\mu(\mathfrak{g}) = \min\{\ \dim(M) \mid M \text{ is a faithful } \mathfrak{g}-\text{module}\}. \tag{7}$$

However, we are dealing with a different definition of minimal faithful representation, which is only possible for nilpotent Lie algebras:

$$\widehat{\mu}(\mathfrak{g}) = \min\{\ n \in \mathbb{N} \ \ | \exists \text{ subalgebra of } \mathfrak{g}_n \text{ isomorphic to } \mathfrak{g}\}. \tag{8}$$

In particular, $\mu(\mathfrak{g})$ is less than or equal to $\widehat{\mu}(\mathfrak{g})$.

This section studies and expounds a new method for obtaining minimal faithful representations of a given nilpotent Lie algebra $\mathfrak{g}$ of dimension $r$, by using triangular configurations. A step by step explanation of our method is given next:

1. Draw first the triangular configuration associated with the Lie algebra $\mathfrak{g}$ by using its law.
2. Study several properties of this configuration: Number of vertices, degree of each one, number and types of edges and number of full triangles.
3. Use Theorems 1 and 2 to compute the first natural number $n \in \mathbb{N}$ such that the triangular configuration associated with $\mathfrak{g}_n$ verifies the properties cited in the previous step. This natural number gives the faithful minimal matrix representation of $\mathfrak{g}$ as a subalgebra of $\mathfrak{g}_n$.

By using this method, the triangular configuration associated with $\mathfrak{g}$ is contained in the triangular configuration associated with $\mathfrak{g}_n$ for such a $n$. Consequently, a representative can be computed for the minimal faithful matrix representation of $\mathfrak{g}$.

As an example of how to apply this method, Heisenberg algebras are considered next. These algebras form a special class of nilpotent Lie algebras. For a given $k \in \mathbb{N}$, the *Heisenberg algebra* $\mathfrak{H}_k$ is the $(2k+1)$-dimensional Lie algebra having the following law with respect to a certain basis $\{w_i\}_{i=1}^{2k+1}$:

$$[w_{2i}, w_{2i+1}] = w_1, \quad \forall i = 1, \ldots, k. \tag{9}$$

We prove that the minimal faithful matrix representation of the $(2k+1)$-dimensional Heisenberg algebra is $k+2$.

The minimal dimension was computed in [4] for Heisenberg algebras: $\mu(\mathfrak{H}_k) = k+2$. So we can affirm that $\widehat{\mu}(\mathfrak{H}_k) \geq k+2$. Next we prove that the minimal faithful matrix representation of $\mathfrak{H}_k$ is exactly $k+2$.

The triangular configuration associated with $\mathfrak{H}_k$ is shown in Figure 5. Obviously, this triangular configuration has $2k+1$ vertices and $k$ full triangles. Vertex 1 is the only one of degree $2k$ (the rest of vertices have degree 2) and all the edges incident to it are ghost edges.

Obviously, $\mathfrak{H}_k$ cannot be obtained as a subalgebra of $\mathfrak{g}_k$ because the vertices in the triangular configuration associated with $\mathfrak{g}_k$ have degree $2(k-2)$. In this way, we ask ourselves if the minimal faithful matrix representation of $\mathfrak{H}_k$ is $k+1$. According to Theorem 1, every vertex in the triangular configuration associated with $\mathfrak{g}_{k+1}$ has degree $2(k-1)$. So $\mathfrak{H}_k$ cannot be isomorphic to any subalgebras of $\mathfrak{g}_{k+1}$. Finally, we consider the Lie algebra $\mathfrak{g}_{k+2}$, which satisfies all the necessary conditions.

- The number of vertices is $\frac{(k+1)(k+2)}{2} \geq 2k+1$, because $k \geq 1$.
- The degree of each vertex is $2k$.
- The center of $\mathfrak{g}_{k+2}$ corresponds to Vertex $k+1$, whose incident edges are ghost edges.

Now we compute a representative for the minimal faithful matrix representation for each Heisenberg algebra.

Let us note that the vector $w_1$ corresponds to the center of the Heisenberg algebra $\mathfrak{H}_k$. Consequently, we associate this vector with the generator of $cen(\mathfrak{g}_{k+2})$; i.e. $w_1 = X_{1,k+2}$. As the law of $\mathfrak{g}_{k+2}$ contains the brackets $[X_{1,i}, X_{i,n}] = X_{1,n}$ for $2 \leq i \leq k+1$, we can also associate the vectors $X_{1,i}$ and $X_{i,n}$, for $2 \leq i \leq k+1$, with the remaining vectors in the basis $\{w_i\}_{i=1}^{2k+1}$ of $\mathfrak{H}_k$. In this way, a representative for the minimal faithful matrix representation of $\mathfrak{H}_k$ is given as follows:

$$w_1 = X_{1,k+2},\ w_2 = X_{1,k+1},\ w_3 = X_{k+1,k+2}, \ldots,\ w_j = X_{1,k-j+1},$$

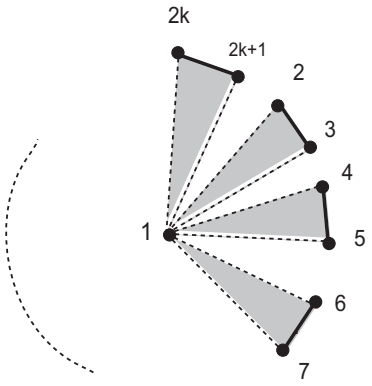



Figure 3: Triangular configuration associated with $\mathfrak{H}_k$.

$$w_{j+1} = X_{k-j+1,k-j+2}, \ldots, w_{2k} = X_{1,2}, w_{2k+1} = X_{2,k+2}$$

So minimal faithful matrix representations of nilpotent Lie algebras can be computed by using triangular configurations and considering properties corresponding to Graph Theory. Let us note that these triangular configurations can be used to study other properties in nilpotent Lie algebras.

# References

[1] J.C. Benjumea, F.J. Echarte, J. Núñez and A.F. Tenorio: A method to obtain the Lie group associated with a nilpotent Lie algebra. Comp. Math. Appl. **51** (2006), 1493–1506.

[2] J.C. Benjumea, J. Núñez and A.F. Tenorio: The Maximal Abelian Dimension of Linear Algebras formed by Strictly Upper Triangular Matrices. Theor. Math. Phys. **152** (2007), 1225–1233.

[3] J.C. Benjumea, J. Núñez and A.F. Tenorio: Minimal linear representations of the low-dimensional nilpotent Lie algebras. Math. Scand. **102** (2008), 17–26.

[4] D. Burde: On a refinement of Ado´s Theorem. Arch. Math. (Basel). **70** (1998), 118-127.

[5] A. Carriazo, L.M. Fernández and J. Núñez: Combinatorial structures associated with Lie algebras of finite dimension. Linear Algebra and Applications. **389** (2004), 43–61.

[6] M. Ceballos, J. Núñez and A.F. Tenorio: Abelian subalgebras in some particular types of Lie algebras. Nonlinear Analysis (2008), doi:10.1016/j.na.2008.11.006.

[7] L.M. Fernández and L. Martín-Martínez: Lie algebras associated with triangular configurations. Linear Algebra and Applications. **407** (2005), 43–63.

[8] J. Heber: Non-compact homogeneous Einstein spaces. Inv. Math. **133** (1998) 279–352.

[9] F. Iachello: Lie algebras and Applications. Lecture Notes in Physics **708**. Springer, Berlin, 2006.

[10] P.J. Olver: Applications of Lie Groups to Differential Equations. Springer, New York, 1986.

[11] H. Stephani, D. Kramer, M. MacCallum, C. Hoenselaers and E. Herlt: Exact solutions of Einstein's field equations. Cambridge University Press, Cambridge, 2003.

[12] V.S. Varadarajan: Lie Groups, Lie Algebras and Their Representations. Springer, New York, 1984.

# Búsqueda de comunidades en grafos grandes mediante configuraciones implícitas de vectores

Victor Muntés-Mulero, Arnau Padrol-Sureda, Guillem Perarnau-Llobet *
Julian Pfeifle **

**Resumen**

Presentamos el algoritmo OCA para buscar comunidades solapadas en grafos grandes, como por ejemplo la Wikipedia con $1{,}6 \times 10^7$ nodos y $1{,}8 \times 10^8$ aristas. OCA se basa en la búsqueda iterativa de subconjuntos localmente óptimos para una función objetivo, representando los subconjuntos como vectores suma de una configuración virtual de vectores. Analizamos el comportamiento de dos funciones objetivo, la Laplaciana asociada a la longitud del vector suma, y la conductividad.

## 1. Introducción

Hasta hace relativamente poco, buscar comunidades en grafos significaba *particionar* en familias disjuntas de nodos un grafo dado $G = (V, E)$, de manera que cada familia consiga una alta puntuación con respecto de alguna medida de calidad, por ejemplo la *conductividad* [3]. Por lo general, los algoritmos para hallar tal partición dividen el grafo recursivamente en dos partes, y por tanto la búsqueda de cada nueva familia está restringida por la existencia de todas las familias halladas hasta ahora. El presente trabajo adopta un enfoque distinto, en tanto que permitimos que las comunidades se puedan solapar entre si. Eso es consistente con la observación que las comunidades pueden presentar de manera natural estructuras tanto solapadas como jerárquicas. Un claro ejemplo son las redes sociales. Hacemos énfasis en que nuestra definición de comunidad es puramente operacional y local: una comunidad es un conjunto (generalmente conexo) de nodos de $G$ que alcanza un óptimo local de cierta función objetivo $\varphi$, en el sentido de que añadir o quitar un nodo de ella empeora el valor de $\varphi$.

La idea subyacente de OCA es representar cada nodo de $G$ mediante un vector en un espacio vectorial, de dimensión posiblemente bastante alta, de manera que los vectores que corresponden a nodos no adyacentes son ortogonales, y todos los ángulos entre vectores "adyacentes" son iguales y agudos. Para nosotros, esta representación será siempre *implícita*, en el sentido de que nunca se llega a construir ni un sólo vector explícitamente. Una búsqueda local, apodada LOCA, encuentra una comunidad a partir de cierto subgrafo inicial, añadiendo o quitando nodos para optimizar la función objetiva. Cada ejecución de LOCA necesita tiempo $O\big(s(\log s + \log n)\Delta\big)$, donde $\Delta$ es el grado máximo y $n$ el número de nodos de $G$, y $s$ el tamaño de la comunidad hallada. El algoritmo global OCA combina los resultados de las búsquedas locales empezadas en diferentes conjuntos iniciales repartidos por todo el grafo $G$. De esta manera, se desvela la estructura global del grafo compuesto por comunidades solapadas y localmente óptimas.

## 2. Configuraciones virtuales de vectores

La inspiración de nuestra propuesta son las *representaciones ortogonales* introducidas por Lovász [5] en 1979. Se trata de una colección de $n$ vectores unitarios $v_1, \ldots, v_n$ de manera que $\langle v_i, v_j \rangle = 0$ para todo $\{i, j\} \notin E$. Nosotros generalizamos este concepto de la siguiente manera:

*DAMA, Universitat Politècnica de Catalunya {vmuntes, apadrol, perarnau}@ac.upc.edu
**DMA II, Universitat Politècnica de Catalunya, julian.pfeifle@upc.edu

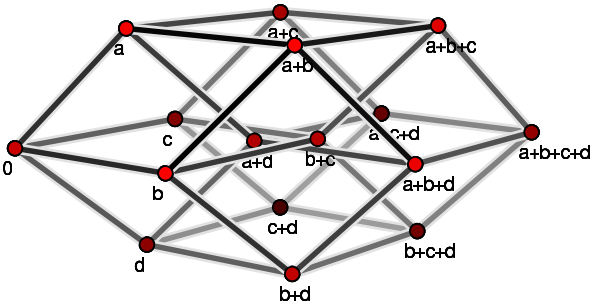
**Definición 2.1.** Un conjunto $\mathcal{V} = \{v_1, \ldots, v_n\}$ de vectores unitarios en un espacio vectorial real es una *representación vectorial virtual* de $G$ si existen $c, d \in (-1, 1)$ de manera que $\langle v_i, v_j \rangle = c$ para todo $\{i, j\} \in E$, y $\langle v_i, v_j \rangle = d$ si $\{i, j\} \notin E$.

El adjetivo *virtual* resalta el hecho que nunca tendremos que construir $\mathcal{V}$ de manera explícita. Ya que los $v_i$ son unitarios, siempre $c, d \in [-1, 1]$; los casos extremos se corresponden con los casos degenerados en los que dos vectores se hacen paralelos (es decir, iguales) o anti-paralelos. El caso más importante para nosotros, que consideramos en exclusiva a partir de ahora, es el de $0 < c < 1$ y $d = 0$.

**Definición 2.2.** El *espacio de búsqueda* $\Gamma = \Gamma(G)$ asociado a $\mathcal{V}$ es el grafo con un nodo para cada subconjunto $S \subseteq V$. Una arista conecta $S$ con $S'$ en $\Gamma$ si $S' = S \cup v$ para algún nodo $v \in V \setminus S$.

Dicho de otro modo, $\Gamma$ es el grafo del cubo de dimensión $|\mathcal{V}|$. El papel de este grafo de $2^n$ nodos, que pensamos también como un *grafo virtual*, es formalizar el proceso de optimizar la función objetivo $\varphi$ sobre todos los subconjuntos de nodos de $G$.






$$H_1(c) = \begin{pmatrix} 1 & c & c & 0 \\ c & 1 & c & c \\ c & c & 1 & c \\ 0 & c & c & 1 \end{pmatrix} \qquad H_2(c) = \begin{pmatrix} 1 & c & 0 & 0 \\ c & 1 & c & 0 \\ 0 & c & 1 & c \\ 0 & 0 & c & 1 \end{pmatrix}$$

El grafo $\Gamma(H_1)$

Figura 1: Dos grafos, las matrices que los representan, y el espacio de búsqueda de uno de ellos.

**Ejemplo 2.3.** La figura 1 muestra la representación vectorial virtual del grafo $H_1$ con $0 < c < 1$ y $d = 0$. Ya que $b$ y $c$ están conectados en $H_1$, pero $a$ y $d$ no, el ángulo $\angle(b, c)$ entre $b$ y $c$ es más pequeño que $\angle(a, d) = \frac{\pi}{2}$, y por tanto $b + c$ es más largo (más alejado de 0) que $a + d$.
Comparamos ahora $H_1$ con $H_2$. Puesto que $d = 0$ implica que dos vectores son ortogonales si sus nodos no están conectados, y $H_1$ está "más conectado" que $H_2$, los vectores que lo representan están más juntos que los de $H_2$. Por tanto, si $H_1$ y $H_2$ son subgrafos de un grafo más grande $G$, la suma de todos los vectores correspondientes a nodos en $H_1$ será más larga que la suma para $H_2$.

Por tanto, (el cuadrado de) la longitud Euclideana parece un buen candidato para la función objetivo para OCA: $\varphi(S) = \left\| \sum_{i \in S} v_i \right\|^2$. El ejemplo siguiente da más credibilidad a esta idea.

**Ejemplo 2.4.** Si $\mathcal{V}$ es una representación vectorial virtual de $G$ y $S = \{1, 2, \ldots, k\}$ indexa un conjunto independiente de tamaño $k$ en $G$, entonces $\varphi(S) = k$. Por otra parte, el vector suma de un $k$-clan $K_k$ tiene longitud cuadrática $ck^2 + (1 - c)k = \Theta(k^2)$. El comportamiento diferente de subgrafos independientes y completos se vuelve más y más pronunciado según crece $c$.

Esperamos pues que comunidades bien conectadas entre sí tengan vectores suma más largos que comunidades más tenues. Sin embargo, y a pesar de todo, esta $\varphi$ decididamente *no* es la elección adecuada, porque el vector suma más largo de todos es *siempre* $\sum_{i \in V} v_i$, que corresponde al grafo entero $G$ (tal y como sugiere la figura 1). Con esta elección de $\varphi$, ¡OCA se comería el grafo $G$ entero!

La lección crucial que podemos aprender del ejemplo 2.4 es el hecho de que valores más grandes de $c$ distinguen las comunidades mejor que valores pequeños. Presentamos a continuación algunos resultados sobre valores extremos de $c$ y cómo aproximarlos.

### 2.1. Cómo calcular el valor más grande posible de $c$

Para ver de qué manera la estructura de $G$ restringe los posibles valores de $c$, codificamos los productos escalares de $\mathcal{V}$ en la matriz $G(c) = I_n + cA$, con $I_n$ la matrix identidad $n \times n$, y $A$ la matriz de adyacencia de $G$ (véase la figura 1).

**Teorema 2.5.** *Para cualquier grafo $G$ existe un único valor máximo para $c$ de manera que $G(c)$ representa los productos escalares entre los miembros de una configuración vectorial virtual de $G$. Este valor más grande es $c = -1/\lambda_n$, con $\lambda_n$ el autovalor más negativo de $A$.*

*Demostración.* Por definición, $G(c)$ representa los productos escalares entre los miembros de una configuración virtual de vectores si y sólo si la matriz se descompone como $G(c) = V^T V$, donde las columnas de la matriz $V$ de tamaño $d \times n$ contienen las coordenadas de los vectores. Esto pasa si y sólo si $G(c)$ es *semidefinida positiva*, y por tanto la *matriz de Gram* de $\mathcal{V}$. Una caracterización de matrices semidefinidas positivas es que todos sus autovalores son no negativos. Puesto que 1 es el único autovalor de $G(0) = I_n$, existe por un lado una representación virtual de cualquier grafo $G$ para $c = 0$, y por otro una representación virtual del grafo vacío. Suponemos pues en adelante que $G$ no es vacío.

En este supuesto, afirmamos que la matriz de adyacencia $A$ de $G$ siempre tiene un autovalor negativo. Para verlo, recuérdese por una parte que los autovalores $\lambda_n \leq \cdots \leq \lambda_1$ de $A$ son todos reales al ser $A$ simétrica, y por otra que la traza de $A$ es la suma de sus autovalores. Ahora, al tratarse de la matriz de adyacencia de un grafo no vacío sin bucles, la traza de $A$ es cero pero no todos los autovalores lo son, y en consecuencia algún autovalor tiene que ser negativo; en particular, $\lambda_n < 0$.

Estamos en condiciones de demostrar la existencia de un valor máximo no nulo para $c$: Los autovalores de $G(c) = I_n + cA$ son $1 + c\lambda_i$, funciones lineales en $c$ que toman valor positivo para $c = 0$. Hemos visto que al menos una de estas rectas tiene pendiente negativa, así que si $c$ aumenta desde 0, llegará el momento en el que algún autovalor de $G(c)$ se anule por primera vez. Este valor es $c = -1/\lambda_n > 0$. □

Para aproximar el autovalor $\lambda_n < 0$, usamos el bien conocido *método de las potencias*, que nos proporciona el autovalor de mayor valor absoluto, con el siguiente resultado.

**Teorema 2.6.** *El autovalor más negativo $\lambda_n$ de cualquier matriz simétrica $A$ se puede aproximar a partir de cualquier cota superior $\lambda_1 \leq \kappa$ para el autovalor mayor.*

*Demostración.* Los autovalores de $B = A - \kappa I_n$ son $\mu_i = \lambda_i - \kappa$, con $\lambda_i$ los autovalores de $A$. Puesto que $\mu_i \leq 0$ para todo $i$, el autovalor de $B$ de máximo valor absoluto es $\mu_n < 0$, lo cual se puede aproximar con el método de la potencia. Para acabar, nótese que $|\lambda_n| = |\mu_n| - \kappa$. □

Una cota superior muy asequible para $\lambda_1$ es $\lambda_1 \leq \Delta$, con $\Delta$ el grado máximo de $G$ [2].

## 3. Funciones objetivo eficientes

Para reparar el fracaso de la función objetivo "longitud de la suma al cuadrado", construimos dos nuevos grafos orientados a partir de $\Gamma$: En $\Gamma^{\uparrow}$, las aristas de $\Gamma$ se orientan hacia el conjunto más grande, mientras que en $\vec{\Gamma}(\varphi)$ una arista se orienta $S \to S'$ si y sólo si $\varphi(S) < \varphi(S')$ (exceptuando casos degenerados). Por tanto, los máximos locales de $\varphi$ corresponden a los sumideros de $\vec{\Gamma}(\varphi)$.

Proponemos dos funciones objetivo. Una es la *Laplaciana dirigida*, una adaptación de la noción de derivada a $\Gamma$. Intenta capturar la intuición que a pesar de que la longitud crece continuamente hasta alcanzar el grafo entero, no lo hace de manera uniforme, y las variaciones en la tasa de crecimiento contienen información sobre las comunidades. Comparamos esta función con la bien conocida *conductividad* [3].

## 3.1. La Laplaciana asociada al cuadrado de la longitud

**Definición 3.1.** El valor en $v$ de la *Laplaciana dirigida* asociada a la función $f$ en el grafo $\Gamma^{\uparrow}$ es

$$\mathcal{L}_{\Gamma^{\uparrow},f}(v) \;=\; f(v) - \frac{1}{\sqrt{\operatorname{indeg}(v)}} \sum_{u:u\to v} \frac{f(u)}{\sqrt{\operatorname{indeg}(u)}}.$$

Por la definición de la orientación en $\Gamma^{\uparrow}$, $\operatorname{indeg}(v) = |S|$ para cualquier nodo $v \in \Gamma^{\uparrow}$, es decir la cardinalidad del subconjunto $S$ que corresponde a $v$ en $\Gamma$. Ponemos $\mathcal{L}_{\Gamma^{\uparrow},f}(v) = 0$ para el nodo $v$ que corresponde al conjunto $S = \emptyset$, para evitar la división por 0.

A partir de ahora, supondremos que $s := |S| > 1$. Entonces la Laplaciana $\mathcal{L}(S) := \mathcal{L}_{\Gamma^{\uparrow},\|\cdot\|^2}(S)$ es

$$\begin{aligned}\mathcal{L}(S) &= \|S\|^2 - \frac{1}{\sqrt{s(s-1)}} \sum_{v\in S} \|S \setminus v\|^2 \\ &= s - \sqrt{s(s-1)} + 2c\,\mathrm{E}_{\mathrm{in}}(S)\left(1 - \frac{s-2}{\sqrt{s(s-1)}}\right),\end{aligned}$$

donde $\mathrm{E}_{\mathrm{in}}(S)$ cuenta las aristas de $G$ con ambos extremos en $S$.

**Teorema 3.2.** *Sea $S$ una comunidad* grande, *$|S| \gg 0$. En este caso, un nodo $u \in V \setminus S$ mejora la Laplaciana $\mathcal{L}$ si y sólo si el número* $\mathrm{nb}_S(u)$ *de vecinos de $u$ en $S$ es mayor que el grado medio* $\operatorname{avdeg}(S)$ *de los nodos en $S$, es decir,* $\mathrm{nb}_S(u) > \operatorname{avdeg}(S)$*. En particular, nodos con un sólo vecino en $S$ solamente son aceptados en $S$ si $S$ es un árbol.*

*Esquema de demostración.* Usando relaciones como $\langle S, u\rangle = c\,\mathrm{nb}_S(u)$, calculamos

$$\begin{aligned}\mathcal{L}(S\cup u) - \mathcal{L}(S) &= 2c\,\mathrm{nb}_S(u)\left(1 - \frac{s-1}{\sqrt{s(s+1)}}\right) \\ &\quad + 2c\,\frac{\mathrm{E}_{\mathrm{in}}(S)}{\sqrt{s}}\left(\frac{s-2}{\sqrt{s-1}} - \frac{s-1}{\sqrt{s+1}}\right) \\ &\quad + 1 - \sqrt{s}\big(\sqrt{s+1} - \sqrt{s-1}\big) \qquad (1)\\ &= c\,\mathrm{nb}_S(u)\left(\frac{3}{s} + O(s^{-2})\right) \\ &\quad + c\,\mathrm{E}_{\mathrm{in}}(S)\left(-\frac{3}{s^2} + O(s^{-3})\right) + O(s^{-2}),\end{aligned}$$

lo cual es no negativo para $s \gg 0$ si y sólo si $\mathrm{nb}_S(u) > \mathrm{E}_{\mathrm{in}}(S)/s$. □

## 3.2. La conductividad

**Definición 3.3.** La *conductividad* de un subconjunto $S \subseteq V$ en $G$ es [3]

$$\phi(S) \;=\; \frac{\mathrm{E}_{\mathrm{out}}(S)}{\mathrm{E}_{\mathrm{out}}(S) + \min\big\{\mathrm{E}_{\mathrm{in}}(S),\, \mathrm{E}_{\mathrm{in}}(V\setminus S)\big\}},$$

donde $\mathrm{E}_{\mathrm{out}}$ cuenta las aristas en $G$ con exactamente un extremo en $S$. Puesto que la conductividad mide la razón entre aristas salientes al total de aristas en el conjunto o su complemento (el más pequeño de los dos), intentaremos minimizarla.

**Teorema 3.4.** *Sea $R = \mathrm{E}_{\mathrm{out}}(S)/\,\mathrm{E}_{\mathrm{in}}(S)$, y supongamos que $\mathrm{E}_{\mathrm{in}}(S) \le \mathrm{E}_{\mathrm{in}}(V\setminus S) - \Delta$, con $\Delta =$* $\operatorname{maxdeg}(G)$*. Entonces un nodo $u \in V \setminus S$ mejora la conductividad $\phi$ si y sólo si* $\mathrm{nb}_S(u) > \frac{\deg(u)}{R+2}$*. En particular, nodos con un sólo vecino en $S$ solamente se aceptarán en $S$ si* $\deg(u) < R + 2$.

*Demostración.* Usando $\mathrm{E}_{\mathrm{out}}(S \cup u) = \mathrm{E}_{\mathrm{out}}(S) + \deg(u) - 2\,\mathrm{nb}_S(u)$, $\mathrm{E}_{\mathrm{in}}(S \cup u) = \mathrm{E}_{\mathrm{in}}(S) + \mathrm{nb}_S(u)$, y $\mathrm{E}_{\mathrm{in}}\big(V \setminus (S \cup u)\big) = \mathrm{E}_{\mathrm{in}}(V \setminus S) - \deg(u) + \mathrm{nb}_S(u)$, hallamos

$$\phi(S \cup u) \;=\; \frac{\mathrm{E}_{\mathrm{out}}(S) - 2\,\mathrm{nb}_S(u) + \deg(u)}{\mathrm{E}_{\mathrm{out}}(S) - \mathrm{nb}_S(u) + \deg(u) + m},$$

donde $m := \min\big\{\,\mathrm{E}_{\mathrm{in}}(S),\, \mathrm{E}_{\mathrm{in}}(V \setminus S) - \deg(u)\big\}$. Ya que $\mathrm{E}_{\mathrm{in}}(S) \le \mathrm{E}_{\mathrm{in}}(V \setminus S) - \deg(u)$, vemos que

$$\phi(S \cup u) - \phi(S) \;=\; \frac{\deg(u) - (R+2)\,\mathrm{nb}_S(u)}{(R+1)\big(T + \mathrm{nb}_{V\setminus S}(u)\big)} \tag{2}$$

con $T = \mathrm{E}_{\mathrm{in}}(S) + \mathrm{E}_{\mathrm{out}}(S)$. Eso termina la demostración. □

# 4. El algoritmo OCA

Presentamos dos variantes del *Overlapping Community Algorithm* OCA: la versión global $\mathsf{OCA}_g$ encuentra todas las comunidades en $G$, mientras que la versión local $\mathsf{OCA}_l$ sirve para explorar las comunidades de un nodo específico. En ambos casos, el motor de búsqueda local, llamado LOCA, es el mismo, y consiste en usar un método de pivot para encontrar un sumidero local en $\Gamma^{\uparrow}$ a partir de cierto conjunto inicial $S_0$. $\mathsf{OCA}_l$ selecciona conjuntos iniciales a partir del nodo a examinar y llama a LOCA. En cambio, $\mathsf{OCA}_g$ selecciona repetidamente conjuntos iniciales distribuidos por el grafo. Escribiremos $\mathrm{Nb}(S) = \big(\bigcup_{u\in S} \mathrm{Nb}_V(u)\big) \setminus S$ para el *conjunto* de vecinos de $S$, donde $\mathrm{Nb}_X(v)$ denota el conjunto de vértices en $X \subseteq V$ adyacentes a $v \in V$. Las estrategias de búsqueda empleadas son las siguientes:

**Regla de pivot para LOCA:** Hemos tenido las mejores experiencias con el método voraz, que en cada paso selecciona el nodo de $S \cup \mathrm{Nb}(S)$ que al suprimir o añadir a $S$ dé el mayor incremento de $\varphi$. Para ello, LOCA mantiene una cola de prioridad $\mathcal{Q}$ con todas las *prioridades* $\Pi_+(u) = \varphi(S \cup u) - \varphi(S)$ para $u \in \mathrm{Nb}(S)$, respectivamente $\Pi_-(u) = \varphi(S \setminus u) - \varphi(S)$ para $u \in S$.

**Selección de vecindario para LOCA:** O bien tomamos todos los nodos a distancia $\le k$ de $S_0$, o bien un subconjunto aleatorio de ellos. Normalmente, usamos $k \le 2$. No hemos implementado criterios sofisticados basados en recorridos aleatorios como en [10].

**Criterio de terminación para OCA:** El número de nodos del grafo ya explorados aumenta mucho al principio, y después el crecimiento se ralentiza. Paramos el algoritmo cuando la tasa de crecimiento del número de nuevos nodos haya decaído debajo de cierto umbral.

**Teorema 4.1.** *Si $G$ tiene $n$ nodos y grado máximo $\Delta$, la búsqueda local LOCA con la Laplaciana $\mathcal{L}$ se puede ejecutar en tiempo $O\big(s(\log s + \log n)\Delta\big)$, donde $s$ es el tamaño de la comunidad encontrada. Para la conductividad $\phi$, este tiempo aumenta a $O\big(s(s + \log n)\Delta\big)$.*

*Demostración.* Solamente consideraremos los pasos necesarios para añadir un nodo $u$ a $S$; quitar nodos es muy similar. Las fórmulas (1) y (2) dicen que para ambas funciones objetivo, LOCA tiene que actualizar los valores de *todos* los $u' \in \Sigma := S \cup \mathrm{Nb}(S)$, porque todas las prioridades dependen de cantidades como $\mathrm{E}_{\mathrm{in}}$ y $s$ que cambian con las propiedades y el tamaño de $S$. Eso lleva a un algoritmo cuadrático en el tamaño del conjunto de la salida. Por (2), eso es lo mejor posible para la conductividad $\phi$. Para la Laplaciana, podemos aumentar la eficiencia de LOCA como sigue. Sea $f_+\big(\mathrm{nb}_S(u), s, \mathrm{E}_{\mathrm{in}}(S)\big)$ el resultado de $\Pi_+(u) = \varphi(S \cup u) - \varphi(S)$ en la fórmula (1). Distinguimos los posibles casos para $u' \in \Sigma \cup \mathrm{Nb}_V(u)$:

1. Si $u' \in \mathrm{Nb}_\Sigma(u)$, reemplazamos $\Pi(u')$ por $f_+\big(\mathrm{nb}_S(u') + 1, s + 1, \mathrm{E}_{\mathrm{in}}(S) + \mathrm{nb}_S(u)\big)$. Eso pasa como mucho $\Delta = \mathrm{maxdeg}(G)$ veces.
2. Si $u' \in \Sigma \setminus \mathrm{Nb}_\Sigma(u)$, reemplazamos $\Pi(u')$ por $f_+\big(\mathrm{nb}_S(u'), s + 1, \mathrm{E}_{\mathrm{in}}(S) + \mathrm{nb}_S(u)\big)$. No sabemos cuántas veces pasa eso, porque no podemos controlar el tamaño de $\Sigma$.
3. Si $u' \in \mathrm{Nb}_V(u) \setminus \mathrm{Nb}_\Sigma(u)$, LOCA ha de calcular $\mathrm{Nb}_V(u')$.

Al desarrollar en potencias de $\frac{1}{s}$, la diferencia $\delta_2$ entre el valor $\Pi_+(u') = f_+\big(\mathrm{nb}_S(u'), s, \mathrm{E}_{\mathrm{in}}(S)\big)$ guardado en $\mathcal{Q}$ y el valor nuevo en el caso 2 resulta ser un orden de magnitud (en potencias de $\frac{1}{s}$) más pequeño que la diferencia análogo $\delta_1$ con el valor nuevo en el caso 1. En particular, $\frac{\delta_2}{\delta_1} < \frac{1}{100}$ para $|S| > 150$. Por tanto, después de un período inicial de duración constante $C$, podemos ignorar el caso 2 por completo. (Omitimos los detalles de este cálculo, y del cálculo análogo para $\varphi(S \setminus u) - \varphi(S)$. Ademas, suponemos que el número total de veces que borramos un nodo es pequeño en comparación con el número de veces que añadimos uno.) Después de este período inicial, el caso 1 efectua a lo sumo $\Delta$ actualizaciones de $\mathcal{Q}$, con un coste de $O(\log j)$ cada uno para una comunidad de tamaño $j$. El caso 3, finalmente, es ejecutado un máximo de $\Delta s$ veces durante el algoritmo. Con las implementaciones que usamos, podemos ejecutar el cálculo de $\mathrm{Nb}_V(u')$ en tiempo $O(\log n)$. El coste total del algoritmo es, por tanto, $C' + s\,O(\log n)\Delta + \sum_{j=C+1}^{s} \Delta O(\log j) = O\big(s(\log n + \log s)\Delta\big)$. □

# 5. Resultados

Hemos ejecutado OCA en tres grafos artificiales y dos grafos reales, descritos a continuación.

| Nombre | tipo | # nodos | # aristas |
|---|---|---|---|
| Erdös-Rényi aleatorio | generado | $10^5$ | 750353 |
| "Word association" | real | 7 211 | 31 798 |
| LFR | generado | $10^4$–$10^6$ | $\sim 10^5$–$10^7$ |
| Árbol de margaritas | generado | $10^5$ | $\sim 4 \cdot 10^5$ |
| Wikipedia | real | 16 986 429 | 176 454 501 |

**Grafos aleatorios Erdös-Rényi.** Se trata de una familia de grafos en la cual con alta probabilidad *no* existen comunidades. Generamos un grafo $G(n, p)$ según el modelo de Erdős-Rényi [1], con $n$ nodos y una arista entre dos de ellos con probabilidad $p$.

**"Word Association Network".** Este conjunto de datos, basado en [7], se ha usado con anterioridad para la detección de comunidades en [8, 9]. Se trata de un grafo dirigido que enlaza palabras relacionadas entre sí, con pesos que reflejan el grado de relación que guardan. Lo hemos usado para evaluar la calidad de $\mathsf{OCA}_l$, porque su propia estructura nos permite evaluar de manera intuitiva la calidad de las comunidades encontradas por OCA. Para nuestros experimentos, lo hemos convertido en un grafo no dirigido, conectando dos vértices si la suma de los pesos de las dos aristas que los unen excede un cierto umbral $\omega^*$. Usando el mismo valor que [8] e ignorando nodos aislados da un grafo con 7 211 nodos y 31 798 aristas. Después de quitar todos los nodos de grado 1, queda un grafo con 5 353 nodos y 29 940 aristas.

**LFR ("Communitized benchmark graphs").** Lacichinetti, Fortunato and Radicchi (en adelante, LFR) proponen en [4] un juego de pruebas para la detección de comunidades. Puesto que se trata de grafos generados por ordenador, podemos determinar de manera cuantitativa tanto el comportamiento de OCA según aumenta el tamaño del grafo, como la calidad de las comunidades halladas. Unos valores típicos que hemos considerado son: $n = 10^5$, grado medio 20, grado máximo 90, tamaño de comunidades en $(25, 150)$, exponente de la ley de potencias para la distribución de grados $\gamma = 2{,}5$, exponente para la distribución de los tamaños de las comunidades $\beta = 1{,}5$, y *mixing parameter* $\mu = 0{,}2$. Este último parametro determina la proporción de aristas que un nodo tiene en común con nodos fuera de su comunidad.

**Árboles de margaritas (comunidades solapadas).** No nos consta publicada ninguna familia de grafos que sirva como estándar para evaluar algoritmos de búsqueda de comunidades solapadas. Por este motivo, aquí proponemos una familia bastante sencilla con este fin. Los nodos vienen numerados por enteros consecutivos. Creamos una arista entre dos nodos $u$ y $v$, con probabilidad $p_1$, si se cumplen las condiciones $f(u)$ y $f(v)$, donde $f(x) = (x = 0 \bmod p) \vee (x = 0 \bmod q)$, para $p, q$ primos entre si. Además, creamos una arista entre cualquier par de nodos $u$ y $v$, con probabilidad $p_2$, si $u = v \bmod p$ y $u, v \neq 0 \bmod q$. El grafo *margarita* resultante tiene $p$ comunidades, una comunidad central que consiste de múltiplos de $p$ y $q$, y $p-1$ comunidades solapadas con aquélla. El parámetro $q$ determina el grado de solapamiento. Nuestro grafo de pruebas será un *árbol de margaritas*, en el que unimos varias margaritas a través de sus hojas.

**Wikipedia.org.** Los artículos en esta conocida enciclopedia tienen enlaces a otras páginas o sus traducciones a otros idiomas. Hemos extraído casi 17 millones de artículos y más de 180 millones de enlaces en 253 idiomas diferentes de los ficheros XML originales.

Hemos llevado a cabo los experimentos en un ordenador con un procesador con 2.33 GHz y 9 GB de RAM, corriendo bajo Linux (versión 2.6.18 del kernel). Los grafos se manejan a través de DEX, un gestor de bases de datos [6]. La implementación actual de DEX tiene dos librerías, una en C++ para manejar mapas de bits y maps, y otra en Java para tareas de alto nivel. La librería C++ contiene un buffer pool para permitir el procesamiento out-of-core de los grafos DEX.

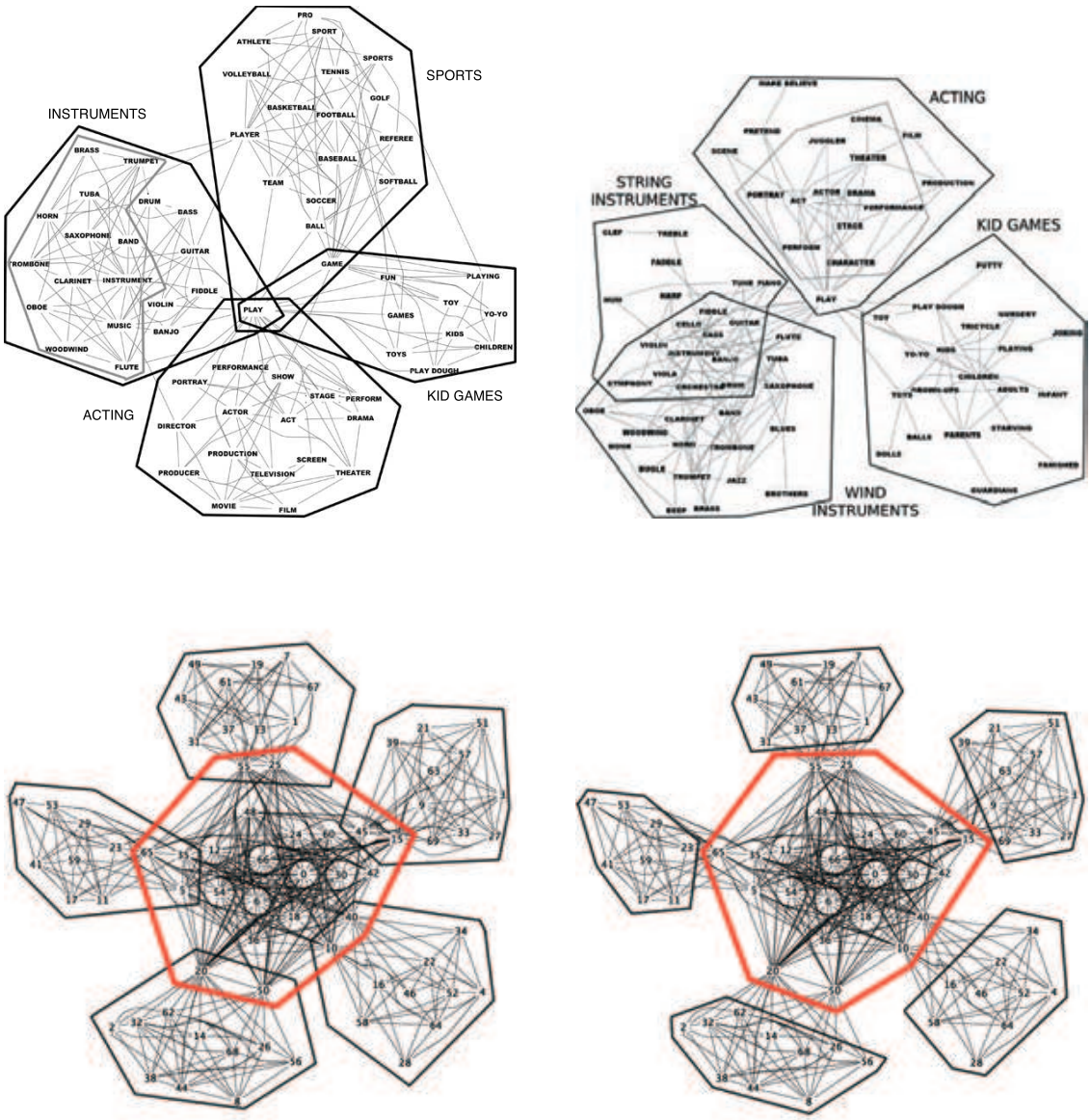



Figura 2: *Arriba:* Comunidades encontradas por LOCA en el grafo de asociaciones de palabras a partir de la palabra PLAY, por la Laplaciana (izquierda) y la conductividad (derecha). *Abajo:* Las comunidades encontradas en un grafo margarita por la Laplaciana (izquierda) y la conductividad (derecha).

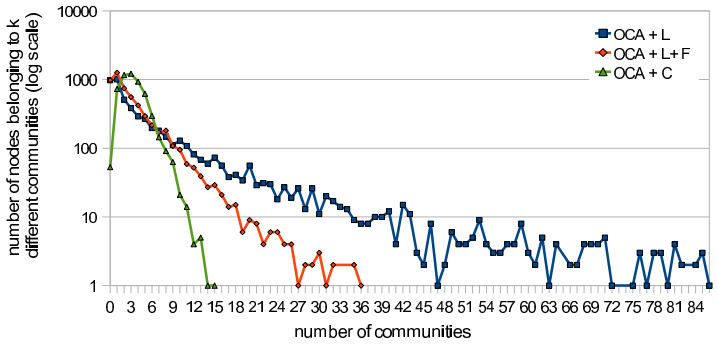



Figura 3: Número de nodos en grafos estándar LFR en función del número de comunidades a las que pertenecen. L y C se refieren a la Laplaciana y la conductividad, mientras que F es un postprocesamiento que consiste en fusionar comunidades muy similares.

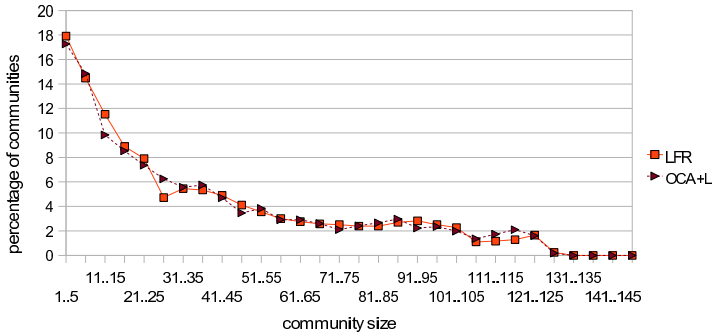



Figura 4: Validación cuántitativa de OCA con la conductividad en los grafos LFR. Mostramos hasta qué punto OCA es capaz de reconstruir la estructura de comunidades en los juegos de prueba.

# Referencias

[1] P. Erdös and A. Rényi, *On random graphs, I*, Publ. Math. (Debrecen), 6 (1959), pp. 290–297.

[2] C. Godsil and G. Royle, *Algebraic graph theory.*, Graduate Texts in Mathematics. 207. New York, NY: Springer. xix, 439 p., 2001.

[3] R. Kannan, S. Vempala, and A. Vetta, *On clusterings: Good, bad and spectral*, Journal of the ACM, 51 (2004), pp. 497–515.

[4] A. Lancichinetti, S. Fortunato, and F. Radicchi, *Benchmark graphs for testing community detection algorithms*, Phys. Rev. E (Statistical, Nonlinear, and Soft Matter Physics), 78 (2008).

[5] L. Lóvasz, *On the Shannon capacity of a graph*, IEEE Trans. Inform. Theory, 25 (1979), pp. 1–7.

[6] N. Martinez-Bazán, J. Nin, S. Gómez-Villamayor, M. A. Sánchez-Martínez, V. Muntés-Mulero, and J. L. Larriba-Pey, *DEX: High-performance exploration on large graphs for information retrieval*, in CIKM '07: Proceedings of the sixteenth ACM conference on information and knowledge management, New York, NY, USA, 2007, ACM, pp. 573–582.

[7] D. L. Nelson, C. L. McEvoy, and T. A. Schreiber, *The University of South Florida word association, rhyme, and word fragment norms*, 1998.

[8] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, *Uncovering the overlapping community structure of complex networks in nature and society*, Nature, 435 (2005), p. 814.

[9] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, *Detect overlapping and hierarchical community structure in networks*, Physica A: Stat. Mech. Appl., 388 (2009), pp. 1706 – 1712.

[10] D. A. Spielman and S.-H. Teng, *A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning.* Preprint http://arxiv.org/abs/0809.3232, 2008.

## Sesión 5, 15:30–16:50

# Algoritmos ACO aplicados a problemas geométricos de optimización*

M. G. Dorzán, E. O. Gagliardi, M. G. Leguizamón, M. T. Taranilla †
G. Hernández ‡

**Resumen**

Muchos problemas de optimización en configuraciones geométricas son NP-duros por lo que interesa obtener soluciones aproximadas. En este trabajo proponemos la utilización de una técnica metaheurística, *Optimización basada en Colonias de Hormigas* (*Ant Colony Optimization - ACO*) para la resolución aproximada de los siguientes problemas para un conjunto de puntos en el plano: triangulación de peso mínimo, pseudotriangulación de peso mínimo y poligonización de perímetro mínimo.

## 1 Introducción

En Geometría Computacional hay numerosos problemas que, o bien son de naturaleza NP-dura, o bien son problemas para los cuales no se conocen soluciones eficientes. De todos modos, resulta de interés encontrar soluciones a tales problemas, aunque las mismas sean aproximadas a las óptimas, por medio de métodos de naturaleza heurística. En particular, es interesante el estudio de problemas de optimización geométrica relacionados con ciertas configuraciones geométricas obtenidas a partir de un conjunto de puntos como son las triangulaciones, las pseudotriangulaciones y las poligonizaciones. En estos problemas se busca optimizar ciertas propiedades que miden la calidad de las configuraciones: peso, perímetro, dilación, factor de carga, etc. Dada la dificultad inherente de dichos problemas, los algoritmos aproximados surgen como candidatos alternativos para su aplicación. Éstos, pueden dar soluciones cercanas a las óptimas y pueden ser específicos para un problema tratado o formar parte de una estrategia general aplicable en la resolución de distintos problemas, como lo son las técnicas metaheurísticas.

Una metaheurística es un proceso de generación iterativo que guía la búsqueda de soluciones combinando inteligentemente diferentes conceptos de diversos campos como: inteligencia artificial [13], evolución biológica [2], inteligencia colectiva [9], sistemas inmunes [3], entre otros. Una metaheurística da un marco algorítmico general que puede ser aplicado en problemas de optimización con pocas modificaciones que lo adapten a un problema específico. Estos métodos son simples de implementar y han demostrado ser exitosos en encontrar de forma eficiente buenas soluciones para problemas de optimización NP-duros [11].

Los algoritmos propuestos en este trabajo derivan de una metaheurística basada en *inteligencia colectiva* (*swarm intelligence*), más específicamente, en el comportamiento de ciertas colonias de hormigas. Este marco presupone un sistema de agentes que obedecen a un conjunto de reglas muy simples, pero que actuando cooperativamente, componen un sistema complejo. En particular, tratamos con la técnica *Ant Colony Optimization* (*ACO*), la cual es un proceso distribuido, en el que un

*Parcialmente subvencionado por Proyecto UPM AL09-PAC-12 y por Proyecto Tecnologías Avanzadas de Bases de Datos N° 22/F614, UNSL

†Departamento de Informática, Facultad de Ciencias Físico Matemáticas y Naturales, Universidad Nacional de San Luis, Argentina {mgdorzan, oli, legui, tarani}@unsl.edu.ar

‡Facultad de Informática, Universidad Politécnica de Madrid, España, gregorio@fi.upm.es

conjunto de agentes (reactivos) actúan en forma independiente y cooperan esporádicamente en forma indirecta, para llevar a cabo un objetivo común.

En este artículo presentamos el diseño de algoritmos *ACO* para la obtención de soluciones aproximadas a los siguientes problemas: Triangulación de Peso Mínimo (*Minimum Weight Triangulation* - $MWT$), Pseudotriangulación de Peso Mínimo (*Minimum Weight Pseudo-triangulation* - $MWPT$) y Poligonización de Perímetro Mínimo (*Minimum Perimeter Polygonization* - $MPP$).

El trabajo se ha organizado de la siguiente forma. En la Sección 2 presentamos los aspectos teóricos y generales de *ACO*. En la Sección 3 describimos los algoritmos *ACO* propuestos para la resolución de los problemas estudiados. En la Sección 4 presentamos algunos aspectos de la experimentación y terminamos en la Sección 5 con las conclusiones y trabajos futuros.

## 2 Optimización basada en Colonias de Hormigas

La Optimización basada en Colonias de Hormigas (Ant Colony Optimization, *ACO* en el resto del trabajo) es una metaheurística inspirada en el comportamiento que siguen las hormigas para encontrar los caminos más cortos entre las fuentes de comida y el hormiguero, surgida a partir del trabajo inicial de Dorigo, Maniezzo y Colorni sobre Sistema de Hormigas (*Ant System*) [4].

Los algoritmos *ACO* son aptos para resolver problemas de optimización combinatoria. Se basan en una colonia de hormigas artificiales, representadas por agentes computacionales simples, que trabajan de manera cooperativa y se comunican mediante rastros de feromona artificiales [5].

Son esencialmente algoritmos constructivos: en cada iteración del algoritmo, cada hormiga construye una solución al problema recorriendo un grafo de construcción. Cada arista (p, q) del grafo representa los posibles pasos que la hormiga puede dar y tiene asociadas dos informaciones que guían el movimiento de la hormiga:

- Información heurística, que mide la preferencia heurística, $\eta_{pq}$, de moverse desde el nodo p hasta el nodo q, o sea, de recorrer la arista. Las hormigas no modifican esta información durante la ejecución del algoritmo.
- Información de los rastros de feromona artificiales, que miden la "deseabilidad aprendida" del movimiento, $\tau_{pq}$, de p a q, imitando a la feromona real depositada por las hormigas reales. Esta información se modifica durante la ejecución del algoritmo dependiendo de las soluciones encontradas por las hormigas.

Una vez que cada hormiga ha generado una solución se evalúa la misma y se puede depositar una cantidad de feromona en función de la calidad de su solución. Esta información representa el sesgo de búsqueda que influye al resto de las hormigas de la colonia en el futuro.

Además, el modo de operación genérico de un algoritmo *ACO* incluye dos procedimientos adicionales, la evaporación de los rastros de feromona y las acciones del demonio. La evaporación de feromona la lleva a cabo el entorno y se usa como un mecanismo que evita el estancamiento en la búsqueda y permite que las hormigas busquen y exploren nuevas regiones del espacio. Las acciones del demonio son acciones opcionales, que no tienen una correspondencia con el comportamiento de las hormigas reales, para implementar tareas desde una perspectiva global. Algunos ejemplos de acciones del demonio son: observar la calidad de todas las soluciones generadas y depositar una nueva cantidad de feromona adicional sólo en las aristas asociadas a algunas soluciones, o aplicar un procedimiento de búsqueda local a las soluciones generadas por las hormigas antes de actualizar los rastros de feromona.

A continuación presentamos los pasos fundamentales de la estrategia de *ACO*.

**Algoritmo 2.1.** ACO-General

```
{
Inicializar                    /* inicialización de parámetros y estructuras */
```

```
Para c=1 hasta NroCiclos
    Para k=1 hasta NroHormigas
        ConstSolucionk              /* hormiga-k construye solución k */
    ActualizarMejorSolución         /* selección de la mejor solución */
    ActualizarRastro                /* actualización de rastro de feromonas */
FinPara
RetornarMejorSolución
}
```

**Inicializar**

Incluye la inicialización de los valores de los parámetros que se consideran en el algoritmo. Entre otros, se deben fijar: el rastro inicial de feromona asociado a cada arista, $\tau_0$, que es un valor positivo pequeño, normalmente el mismo para todas las aristas, el número de hormigas en la colonia, k, y los pesos que definen la proporción en la que afectarán la información heurística y de los rastros de feromonas en la regla de transición probabilística, denominados $\beta$ y $\alpha$ respectivamente.

**Construcción de una solución por la hormiga k**

Se inicia con una solución parcial vacía, que se extiende a cada paso añadiéndole un componente de solución factible elegido entre los vecinos de la solución actual. Esto equivale a encontrar una ruta en el grafo de construcción guiada por el mecanismo que define el conjunto de vecinos factibles con respecto a la solución parcial. La elección de un vecino factible se realiza de manera probabilística en cada paso de la construcción, dependiendo de la variante *ACO* utilizada. En este trabajo, el modelo de probabilidad con retroalimentación utilizado en los algoritmos de construcción es:

$$P_{ij}(k) = \begin{cases} \frac{\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}}{\sum_{h \in F(p_i)} \tau_{ih}^{\alpha}.\eta_{ih}^{\beta}}, & \text{j} \in \text{F(p}_i\text{)}; \\ 0, & \text{en otro caso.} \end{cases} \quad (1)$$

donde:

- F($p_i$) es el conjunto de puntos factibles para el punto $p_i$.
- $\tau_{ij}$ es el valor de feromona asociado a la arista ($p_i$, $p_j$).
- $\eta_{ij} = 1/d_{ij}$ es el valor heurístico asociado a la arista ($p_i$, $p_j$). ($d_{ij}$ es la distancia euclídea entre los puntos $p_i$ y $p_j$).
- $\alpha$ y $\beta$ son parámetros positivos que determinan la importancia relativa de la feromona con respecto a la información heurística.

**Actualización del rastro**

En este paso se aumenta el nivel de feromona de los caminos prometedores y disminuye el de los caminos no tan buenos. Primero, se reducen todos los valores de feromona por medio del proceso de evaporación. Luego, se incrementa el nivel de feromona al conjunto de soluciones buenas. Se utiliza la siguiente fórmula:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \quad (2)$$

donde:

- $\rho \in (0, 1]$ es el factor de persistencia del rastro.
- $\Delta\tau_{ij} = \sum_{k=1}^{NroHormigas} \Delta^k\tau_{ij}$ es la acumulación de rastro, proporcional a la calidad de las soluciones.
- $\Delta^k\tau_{ij} = \begin{cases} 1/L_k, & \text{si la hormiga k utilizó la arista } p_i, p_j; \\ 0, & \text{en otro caso.} \end{cases}$

- $L_k$ representa el valor objetivo de la solución k (aparece en el denominador para problemas de minimización).

La evaporación de feromona evita una convergencia demasiado rápida del algoritmo. Además, esta forma de olvidar permite la exploración de nuevas áreas del espacio de búsqueda.

La actualización del rastro de feromona se puede realizar al menos de dos formas: elitista o no elitista. En el caso elitista, se utiliza la mejor solución encontrada para dar un refuerzo adicional a los niveles de feromona. El no elitista utiliza las soluciones encontradas por todas las hormigas para dar un refuerzo adicional a los niveles de feromona. Nuestra propuesta incluye la aplicación de estas dos formas de actualización del rastro de feromona.

# 3 Algoritmos ACO propuestos

Presentamos la adaptación del algoritmo *ACO*-General a tres problemas de optimización geométrica sobre un conjunto de puntos en el plano, triangulación de peso mínimo, pseudotriangulación de peso mínimo y poligonización de perímetro mínimo.

## 3.1 Triangulación de peso mínimo MWT

Dado un conjunto $S$ de puntos en el plano, una triangulación de $S$ es un conjunto maximal de segmentos cuyos extremos son los puntos de $S$ y tales que dos cualesquiera de esos segmentos no se cortan en puntos interiores. El peso de una triangulación T es la suma de las longitudes euclídeas de todos los segmentos (o aristas) de T. La triangulación que minimiza esta suma se denomina triangulación de peso mínimo de $S$ y se denota por $MWT(S)$. Esta triangulación óptima es útil en problemas de aproximación de datos en dos variables. La complejidad de su cálculo fue uno de los problemas abiertos más interesantes en Geometría Computacional hasta que Mulzer y Rote demostraron en 2006 que la construcción de $MWT(S)$ es un problema NP-duro [12]. En cuanto a resultados aproximados el mejor resultado conocido es un algoritmo con factor constante de aproximación de Krznaric y Levcopoulos [10].

En el algoritmo *ACO*, cada hormiga construye una triangulación, partiendo de un punto inicial cualquiera, agregando aristas a la triangulación mientras no se corten. En caso de no haber aristas factibles desde el punto actual, la elección del próximo punto de referencia se realiza mediante alguno de los siguientes criterios: i) en forma aleatoria; ii) que tenga mayor cantidad de aristas factibles; iii) tenga menor cantidad de aristas factibles.

**Algoritmo 3.1.** ConstSolucionk

```
{
S_k ← ∅                                  /* S_k Solución construida por la hormiga k */
p_i ← SeleccionarPuntoInicial(S)
Mientras CantFactibles(S_k) > 0
    Fp_i ← Factibles(p_i, S_k)
    Si Fp_i = ∅                          /* no existen puntos factibles desde p_i*/
        p_i ← SeleccionarPunto(S, S_k)
        Fp_i ← Factibles(p_i, S_k)
    FinSi
    p_j ← SeleccionarPuntoProbabilisticamente(Fp_i)
    S_k ← S_k∪ {(p_i, p_j)}              /* agrega la arista (p_i, p_j) a la solución */
    ActualizarFactibles(p_i, p_j)
    p_i ← p_j
FinMientras
}
```

donde:
Factibles($p_i$, $S_k$): retorna un conjunto de puntos p $\in S$, tal que la arista ($p_i$, p) no corta a ninguna arista perteneciente a la solución $S_k$.
SeleccionarPuntoInicial($S$): retorna un punto p $\in S$, elegido aleatoriamente.
CantFactibles($S_k$): retorna la cantidad de puntos factibles calculada de acuerdo a las aristas que forman parte de la solución $S_k$ hasta el momento.
SeleccionarPunto($S$, $S_k$): retorna un punto p $\in S$, tal que Factibles(p, $S_k$) debe contener al menos un punto. Esta selección se realiza teniendo en cuenta las aristas que forman parte de la solución $S_k$ y alguno de los siguientes criterios: 1) se elige el punto p en forma aleatoria; 2) se elige el punto p que tenga mayor cantidad de puntos factibles; 3) se elige el punto p que tenga menor cantidad de puntos factibles.
SeleccionarPuntoProbabilisticamente($Fp_i$): retorna un punto $p_j \in Fp_i$ elegido de acuerdo al modelo de probabilidad dado en la ecuación 1.

## 3.2 Pseudo-triangulación de peso mínimo MWPT

Dado un conjunto $S$ de puntos en el plano, una pseudotriangulación es una partición del cierre convexo de $S$ en pseudotriángulos cuyo conjunto de vértices son los puntos de $S$. Un pseudotriángulo es un polígono simple con exactamente tres vértices convexos. El peso de una pseudotriangulación P es la suma de las longitudes euclídeas de todas las aristas de P. La pseudotriangulación que minimiza esta suma se denomina pseudotriangulación de peso mínimo de $S$ y la denotaremos por $MWPT(S)$. El término pseudotriangulación fue introducido por Pocchiola y Vegter en [14] por analogía con los arreglos de pseudorrectas y en [15] hay un excelente resumen de muchos resultados sobre pseudotriangulaciones. En cuanto al peso, Gudmundsson y Levcopoulos presentaron en [8] un algoritmo que construye, en tiempo cúbico, la pseudotriangulación de peso mínimo de un polígono convexo. Y también una solución, en tiempo $O(nlogn)$, que aproxima la solución óptima con un factor $O(logn)$ con respecto al peso del árbol generador de peso mínimo sobre $S$.

En el algoritmo *ACO*, cada hormiga construye una solución, partiendo de la cara formada por el cierre convexo de la nube de puntos, CH($S$). Así se inicia un proceso de partición de regiones o caras hasta que se consigue que todas sean pseudotriángulos vacíos de puntos interiores. En el paso general se considera el conjunto $Caras_k$ donde se mantienen las caras no tratadas. En el análisis de una cara se verifica en primer lugar si es pseudotriángulo vacío de puntos interiores, en cuyo caso pasa a formar parte del conjunto final de pseudotriángulos (PST). En otro caso, se parte en dos nuevas regiones, o bien seleccionando un punto interior y dos del borde, o bien, si no hay puntos interiores, dos del borde, para descomponer la cara inicial en dos nuevas. Se actualiza $Caras_k$ y se repite el proceso hasta que todas las caras de $Caras_k$ se han analizado.

**Algoritmo 3.2.** ConstSolucionk

```
{
PST ← ∅
Mientras CARAS_k ≠ ∅
    Sea F ∈ CARAS_k
    Si F es pseudotriángulo vacío de puntos interiores
        PST ← PST ∪ {F}        /* F es un nuevo pseudotriángulo */
        CARAS_k ← CARAS_k - {F}
    Sino
        ParticionarCara(F)     /* particiona la cara F */
    FinSi
FinMientras
S_k ← Convertir(PST)
}
```

En el procedimiento Inicializar del *ACO*-general, se realiza $CARAS_k \leftarrow \{CH(S)\}$.
Convertir(PST): construye la pseudotriangulación a partir de los pseudotriángulos obtenidos en PST.

```
SubAlgoritmo ParticionarCara(F)
{
/* selecciona los puntos para formar nueva cara */
p_i ← SeleccionarPunto(F)                                  /* interior - borde de F */
p_j ← SeleccionarPuntoProbabilisticamente(F, p_i, 0)       /* del borde de F, respecto de p_i */
Si p_i no pertenece al borde de F
    p_k ← SeleccionarPuntoProbabilisticamente(F, p_i, p_j)
Sino
    p_k ← 0
FinSi
DividirCara(F, p_i, p_j, p_k, F_1, F_2)
CARAS_k ← CARAS_k - F ∪ F_1, F_2
}
```

Donde:
SeleccionarPuntoProbabilisticamente(F, $p_i$, $p_j$): si $p_i$ es cero retorna un punto p ∈ Factibles(F, $p_i$). En otro caso, retorna un punto p ∈ {Factibles(F, $p_i$)-$p_j$}. En ambos casos, p es elegido de acuerdo al modelo de probabilidad dado en la ecuación 1. Factibles(F, $p_i$) es el conjunto de puntos visibles de $p_i$, no unidos a $p_i$ por una arista, en la cara F.
DividirCara(F, $p_i$, $p_j$, $p_k$, $F_1$, $F_2$): construye dos caras $F_1$ y $F_2$, a partir de F y de los puntos $p_i$, $p_j$ y $p_k$.

```
SubAlgoritmo SeleccionarPunto(F)
{
I ← Interiores(F)
Si I ≠ ∅
    p ← SeleccionarPuntoInterior(F)
Sino
    p ← SeleccionarPuntoBorde(F)
FinSi
Retornar p
}
```

Los criterios de selección en SeleccionarPuntoInterior(F) y SeleccionarPuntoBorde(F), pueden ser: i) se elige el punto p en forma aleatoria; ii) se elige el punto p que tenga mayor cantidad de puntos factibles; iii) se elige el punto p que tenga menor cantidad de puntos factibles.

## 3.3 Poligonización de perímetro mínimo MPP

Dado un conjunto finito de puntos $S$ en el plano, se llama poligonización de $S$ a un polígono simple cuyos vértices sean todos los puntos de $S$. Un conjunto $S$ de n puntos en el plano tiene una cantidad exponencial de poligonizaciones por lo que es interesante detectar aquellas que son óptimas respecto de algún criterio, por ejemplo minimizar o maximizar el área o el perímetro del polígono obtenido. Fekete demostró en [6] que la optimización del área es un problema NP-completo. La minimización del perímetro es también un problema NP-completo dada su relación con la versión geométrica del Problema del Viajante, [7].

Presentamos a continuación la estrategia *ACO* para construir una solución aproximada a la poligonización de perímetro mínimo. Cada hormiga construye una solución, utilizando el método Steady Growth propuesto en [1]. El algoritmo de construcción parte de un triángulo con vértices en los puntos de $S$ y sin otros puntos de $S$ en su interior y, en cada paso, construye un polígono con un vértice más del conjunto $S$. Para ello selecciona un punto factible p ∈ $S$, tal que ningún otro punto de $S$ - {p} está contenido en CH($S_k$ ∪ {p}), y tal que existe al menos una arista de $S_k$ (la solución parcial) que sea completamente visible desde p, agregándolo a la solución.

**Algoritmo 3.3.** ConstSolucionk

```
{
S_k ← ConstruirSoluciónInicial(S) /* S_k: Solución construida por la hormiga k */
S ← S - {p / p ∈ S_k}
    AristasUsadas ← ∅        /* conjunto de aristas utilizadas en la construcción de la poligonización */
Mientras S ≠ ∅
    p ← SeleccionarProximoPunto(S)
    AgregarPuntoSolucion(p, S_k)
    S ← S - {p}
FinMientras
}

SubAlgoritmo SeleccionarProximoPunto(S)
{
PuntoNoFactible ← true
Mientras PuntoNoFactible
    p ← SeleccionarPunto(S)
    Si (el interior del CH(S_k ∪ {p}) es vacío de puntos pertenecientes a S) Y (AristasVisibles(p, S_k) ≠ ∅)
        PuntoNoFactible ← false
    FinSi
FinMientras
Retornar p
}
```

donde:
ConstruirSoluciónInicial($S$): retorna el triángulo formado por tres puntos $p_i$, $p_j$, $p_k$ pertenecientes a $S$ elegidos en forma aleatoria, tal que dicho triángulo no contiene puntos interiores.
SeleccionarPunto($S$): retorna un punto $p \in S$ elegido aleatoriamente.
AristasVisibles(p, $S_k$): retorna el conjunto de aristas de $S_k$ completamente visibles desde p.
CH($S_k \cup \{p\}$): calcula el cierre convexo del conjunto de puntos formado por los puntos del polígono $S_k$ y el punto p.
AgregarPuntoSolucion(p, $S_k$): selecciona una arista ($p_j$, $p_k$) $\in$ AristasVisibles(p, $S_k$), para reemplazarla por las aristas ($p_j$, p) y (p, $p_k$) pertenecientes a $S_k$, guardando la arista ($p_j$, $p_k$) en AristasUsadas. La selección se realiza de acuerdo al modelo de probabilidad dado en la ecuación 1.

# 4 Experimentación

La evaluación de heurísticas aplicadas en la resolución de problemas geométricos necesita generar aleatoriamente instancias de prueba, para contar con conjuntos de tamaño y complejidad variados. Se implementó un generador de conjuntos de puntos, utilizando diferentes funciones de generación aleatoria de la librería CGAL. Dichos conjuntos varían en tamaño y distribución en el plano, como por ejemplo puntos generados en un disco, en un cuadrado, en un círculo, entre otros.

Se ha implementado el algoritmo *ACO* para los tres problemas, $MWT$, $MWPT$ y $MPP$ en lenguaje C, encontrándose actualmente en etapa de desarrollo y evaluación de la experimentación. Se están realizando pruebas con conjuntos de 40, 80, 120, 160 y 200 puntos, con 50 hormigas y 1000 iteraciones por ejecución. Los valores utilizados en los parámetros del algoritmo *ACO* son $\rho$: 0.1, 0.25 y 0.5; $\alpha$: 1; y $\beta$: 1 y 5. Se realizan 30 ejecuciones para cada combinación de valores de parámetros, y se obtiene la mediana y el promedio de los resultados obtenidos.

# 5 Conclusiones y trabajos futuros

En este artículo hemos presentado el diseño de algoritmos *ACO* para la obtención de soluciones aproximadas a los siguientes problemas: triangulación de peso mínimo, pseudotriangulación de peso mínimo

y poligonización de perímetro mínimo. La etapa experimental se realizará empleando casos de prueba diversos, que permitan determinar cuáles de los parámetros de los algoritmos *ACO* son los más adecuados para los problemas de optimización presentados.

Pretendemos continuar en la obtención de buenas aproximaciones a las estructuras geométricas presentadas, considerando otras medidas de calidad tales como dilación, número de apuñalamiento, área, etc., utilizando *ACO* y otras técnicas metaheurísticas.

## Referencias

[1] T. Auer y M. Held, *Heuristics for the generation of random polygons.* Proc. 8th Canadian Conference Computational Geometry, pp. 38-44, 1996.

[2] T. Bäck, D. Fogel y Z. Michalewicz, *Handbook of Evolutionary Computation.* IOP Publishing Ltd and Oxford University Press, 1997.

[3] D. Corne, M. Dorigo y F. Glover, *New Ideas in Optimization.* McGraw-Hill, 1997

[4] M. Dorigo, V. Maniezzo y A. Colorni, *The Ant System: An autocatalytic optimizing process.* Technical report 91-016 revised, Dipartimento di Elettronica, Politecnico di Milano, Milan, 1991.

[5] M. Dorigo y T. Stützle, *Ant Colony Optimization.* Massachusetts Institute of Technology, 2004.

[6] S. Fekete, *On simple polygonizations with optimal area.* Discrete and Computational Geometry, 23, pp. 73-110, 2000.

[7] M. Garey y D. Johnson, *Computers and Intractability: a guide of theory of NP-completeness.* Freeman, 1979.

[8] J. Gudmundsson y C. Levcopoulos, *Minimum weight pseudo-triangulations.* Computational Geometry, Theory and applications, 38, pp. 139-153, 2007.

[9] J. Kennedy y R. Eberhart, *Swarm Intelligence.* Morgan Kaufmann Publishers, 2001

[10] D. Krznaric, C. Levcopoulos, *Quasi-greedy triangulations approximating the minimum weight triangulation.* Journal of Algorithms, 27 (2) pp. 303–338, 1998.

[11] Z. Michalewicz y D. Fogel, *How to Solve It: Modern Heuristics.* 2nd Edition, Springer, 2004.

[12] W. Mulzer y G. Rote, *Minimum weight triangulation is NP-hard.* Proceedings of the 22nd Annual ACM Symp. on Computational Geometry. pp. 1-10, 2006.

[13] I. Osman y J. Kelly, *MetaHeuristics: Theory & Application.* Kluwer Academic Publishers, 1996.

[14] M. Pocchiola y G. Vegter, *Pseudo-triangulations: theory and applications.* Proceedings of the 12th Annual ACM Symposium on Computational Geometry, pp. 291-300, 1996.

[15] G. Rote, F. Santos y I. Streinu, *Pseudo-triangulations - a survey.* Surveys on Discrete and Computational Geometry-Twenty Years Later, Contemporary Mathematics, E. Goodman, J. Pach, R. Pollack, eds. American Mathematical Society, 2008.

# Intersección de Segmentos utilizando la tecnología paralela CUDA.

Lidia Ortega Alvarado * M.D. Robles Ortega†

**Resumen**

Numerosas optimizaciones de algoritmos geométricos permiten reducir el orden de complejidad o la carga de datos asociada. A la hora de realizar la implementación se espera que estas mejoras repercutan en el tiempo de ejecución. Sin embargo, no siempre podrá conseguirse en la proporción esperada para todos los tamaños o tipos de datos de entrada. En estos casos la única alternativa suele ser la paralelización del algoritmo.

En este trabajo se presenta la tecnología paralela CUDA para la resolución de un problema clásico en Geometría Computacional, la intersección de segmentos. Las ventajas de CUDA frente al empleo de otras arquitecturas clásicas son numerosas, el bajo coste de los dispositivos para ejecutar estos programas, el relativo poco esfuerzo para adaptar los algoritmos a esta filosofía de trabajo y la alta rentabilidad en tiempo de ejecución que se consigue en numerosos casos.

## 1 Introducción

El objetivo fundamental de la Geometría Computacional es proporcionar algoritmos eficientes para resolver problemas geométricos. A su vez, estos problemas geométricos pueden ser la base para la resolución de otros muchos en diferentes ámbitos de la ciencia. Las nuevas tecnologías en computación no deben pasar desapercibidas para la disciplina encargada de proporcionar los pilares geométricos en otros muchos campos de investigación.

En este trabajo presentamos uno de los problemas clásicos de la Geometría Computacional, el cálculo de intersecciones de segmentos. Una resolución eficiente para este problema, y por ende para la intersección de polígonos, permite resolver otros de mayor índole en numerosos campos. Cualquier SIG (Sistemas de Información Geográficos) necesita solapar en un momento dado la información de las diferentes capas. Encontramos también aplicaciones en Diseño Asistido por Ordenador al igual que en Informática Gráfica, por ejemplo en tratamiento de detección de colisiones.

En este trabajo planteamos la resolución de la intersección de segmentos bajo tres paradigmas. El algoritmo de fuerza bruta en CPU con tiempo de ejecución $O(n^2)$ fue mejorado a $O(n \log n)$ por Bentley-Ottmann [3]. El mejor comportamiento de este método se produce sobre nubes de segmentos dispersas, es decir, con baja probabilidad de intersección entre los segmentos. Sin embargo la mejora del comportamiento del método optimizado baja con respecto al método cuadrático a medida que la proporción de intersecciones aumenta, tal y como hemos comprobado en nuestros experimentos.

Reducir el tiempo de ejecución sería posible únicamente utilizando alguna metodología paralela. El problema del paralelismo ha sido normalmente el sobre-esfuerzo que supone plantear un algoritmo para una determinada arquitectura (software), y el coste en sí de obtener dicha arquitectura de computadoras (hardware). En la actualidad, la tecnología CUDA trabaja sobre una arquitectura barata incorporada en ordenadores personales: las tarjetas gráficas NVidia. Por otro lado el lenguaje de programación para trabajar con CUDA es una extensión de $C++$. Se estima que la curva de aprendizaje para los usuarios de lenguajes de alto nivel es relativamente baja. Además, la arquitectura matricial de las tarjetas gráficas permite adaptar con suma facilidad numerosos algoritmos. A este nuevo paradigma

*Depto. Informática. Universidad de Jaén, lidia@ujaen.es
†Depto. Informática. Universidad de Jaén, mrobles@ujaen.es

de programación se le ha denominado *GPGPU* (General Propose Graphics Programming Units), es decir, el uso de tarjetas gráficas para resolución de problemas de propósito general.

En la Sección 2 de este trabajo introduciremos las características de CUDA (Compute Unified Device Architecture) y su posible adaptabilidad a ciertos algoritmos geométricos. En la Sección 3 describimos los métodos empleados para resolver el cálculo de la intersección de segmentos en CPU y las ventajas computacionales de uno y otro según el tipo de nubes de puntos. En la Sección 4 describimos el método implementado en CUDA y por último en la Sección 4 plantearemos las ventajas y desventajas de la utilización de CUDA, así como posibles mejoras.

## 2 La tecnología paralela CUDA

En los últimos años el hardware gráfico ha experimentado un avance muy significativo. La mayoría de estos dispositivos proporcionaban a sus usuarios la posibilidad de programar la propia tarjeta gráfica o GPU (Graphics Processing Unit). Estos desarrolladores podían controlar de este modo todas las etapas del procesamiento gráfico (pipeline) que culmina en la visualización en pantalla. En los estados iniciales de este proceso se realizan numerosos cálculos a nivel geométrico sobre los objetos $3D$ a visualizar [6, 9].

Esta enorme funcionalidad empezó a ser utilizada por estos desarrolladores para realizar todo tipo de cálculos, no siempre pensando en una inmediata visualización, es decir, utilizando una GPU como si se tratase de una CPU. Pronto los resultados obtenidos se desvincularon totalmente de las aplicaciones gráficas, naciendo la GPGPU. Existían limitaciones puesto que se debía ajustar cualquier tipo de dato al tipo procesable por las etapas del pipeline, sin embargo las ventajas computacionales eran numerosas al contar con un auténtico multiprocesador. Pero la GPGPU tiene aún poco alcance fuera del ámbito de los desarrolladores de hardware gráfico puesto que los lenguajes de programación disponibles seguían estando íntimamente ligados a él: OpenGL o DirectX. Para cualquier programador desconocedor de estos lenguajes y de la arquitectura hardware, no resultaba una alternativa atractiva, y más sabiendo que no todo tipo de problemas se ajustan a una resolución eficiente utilizando GPU.

En el año 2006 NVIDIA presenta la tecnología CUDA en su última generación de tarjetas gráficas, la serie 8. Utiliza una filosofía integradora con un lenguaje de programación genérico como es C++ y la arquitectura paralela de una GPU, desvinculándose además del pipeline gráfico. Trabajar con CUDA resulta asequible, en primer lugar porque es barato, de hecho las tarjetas gráficas NVIDIA están muy extendidas en los ordenadores personales de los aficionados a los vídeo-juegos. En segundo lugar porque su curva de aprendizaje es reducida para aquellos programadores cercanos a lenguajes tipo C o C++.

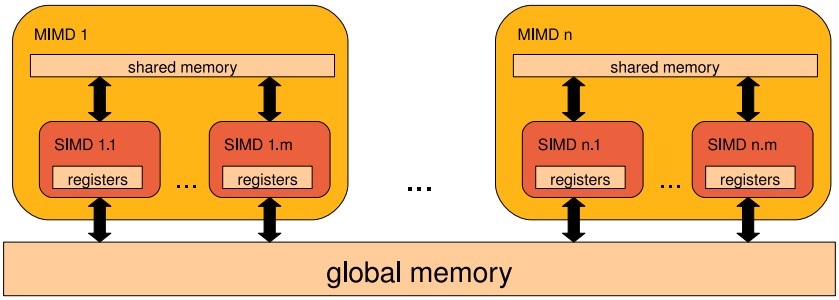



Figura 1: Arquitectura multiprocesador de CUDA.

La arquitectura que presenta CUDA es un conjunto de multiprocesadores MIMD (Múltiple Instructions stream, Multiple Data stream), como vemos en la Figura 1. Cada multiprocesador posee un conjunto de procesadores SIMD (Single Instruction, Multiple Data). Existen numerosos modelos de memoria conviviendo en esta arquitectura: cada procesador SIMD posee una serie de registros a modo de memoria local (sólo accesible para ese procesador), a su vez cada multiprocesador posee una memoria compartida (accesible por todos los procesadores SIMD de un multiprocesador) y finalmente

la memoria global sería aquella accesible por todos los multiprocesadores y, por ende, por todos y cada uno de los procesadores SIMD [7].

**Algoritmo 2.1. void PROCEDIMIENTO_CUDA(CPUdataStructures *data)**

1: Definir y reservar memoria para estructuras de datos en GPU
2: Copiar datos de CPU a GPU
3: Definir las variables *numThreads* and *numBlocks*
4: Ejecutar la función de código en cada hilo
5: Copiar los datos de GPU a CPU
6: Liberar las estructuras de datos en GPU

El modelo de computación vincula el término de procesador SIMD con un *hilo* y cada multiprocesador con un *bloque*. Normalmente para llevar a cabo una implementación en CUDA se parte de una estrategia de solución en CPU. Habrá ocasiones en que la solución CPU sea fácilmente extensible a CUDA, sin embargo en otras se deberá estudiar su viabilidad. Las estructuras de datos en CPU deben adaptarse a otras más simples tipo matriz o vector, para hacerlas compatibles con las de GPU. Posteriormente se sigue el esquema genérico del Algoritmo 2.1, donde se declaran las estructuras de datos en GPU y se habilita memoria para ellas. El siguiente paso consiste en traspasar los datos desde CPU a GPU, para posteriormente repartir estos datos de entrada entre los diferentes hilos (procesadores) y bloques (multiprocesadores). Todos los procesadores SIMD ejecutan el mismo trozo de código pero con diferentes datos. Una vez terminado el proceso, los resultados se devuelven a CPU.

Precisamente, los algoritmos más adecuados para ser ejecutados en cuda son aquellos fácilmente paralelizables, es decir, que ejecutan siempre la misma secuencia de código para todos los datos de entrada. Las ventajas en tiempo con respecto a CPU son mayores cuando:

- *el algoritmo tiene un orden de ejecución cuadrático o superior*: el tiempo necesario para realizar el traspaso de CPU a GPU tiene un gran coste que no suele verse compensado por el bajo coste computacional de un método lineal.
- *es mayor la carga de cálculo computacional en cada hilo*: de nuevo para compensar el tiempo de traspaso de información conviene que cada hilo tenga cierta carga computacional.
- *es menor la dependencia entre los datos para realizar los cálculos*: esta situación se da cuando cada procesador SIMD sólo necesita de datos albergados en memoria local o compartida y no necesita acceder a memoria global, de acceso más lento.
- *es menor el trasiego de información entre CPU y GPU, siendo óptimo cuando este proceso sólo se realiza una vez, al comienzo y al final del proceso*: lo que significa que los datos de entrada al sistema siempre son los mismos y que el proceso no se retroalimenta con la CPU en estados intermedios.
- *no existen secciones críticas, es decir, varios procesos no necesitan escribir en las mismas posiciones de memoria*: mientras la lectura de memoria global y compartida puede ser simultánea, la escritura en la misma posición de memoria plantea un mecanismo de acceso bloqueo y espera que siempre ralentiza el proceso global. Este mecanismo está disponible en CUDA mediante las operaciones atómicas (a partir de CUDA 1.1)

Según lo expuesto anteriormente, aquellos algoritmos geométricos que se ejecutan en tiempo cuadrático mediante un algoritmo de fuerza bruta tendrían más posibilidad de ser eficientemente paralelizables que aquellos que necesitan de estructuras de datos arbóreas. En muchas ocasiones un algoritmo cuadrático optimizado a $O(n \log n)$ emplea este tipo de estructuras, que si bien mejoran el orden del algoritmo, aumentan el valor de la constante asociada, haciéndolos sólo más rápidos con $n$ elevado. En otras ocasiones las características de los datos de entrada pueden determinar realmente las ventajas computacionales de un método cuadrático con respecto a otro $O(n \log n)$. Un ejemplo lo representaría sobre nubes de segmentos muy densas.

En este trabajo hemos realizado un estudio de tres diversos métodos para el cálculo de la intersección de segmentos, en base al algoritmo, la arquitectura utilizada y el tiempo de ejecución del algoritmo. Los resultados nos indican que la tecnología CUDA reduce el tiempo de proceso a las realizadas en CPU utilizando tanto los métodos de fuerza bruta y línea de barrido.

# 3 Intersección de Segmentos en CPU

Según el teorema de Bentley-Ottmann [3], se necesita un tiempo $O((n+k)\log n)$ para encontrar las $k$ intersecciones existentes en un conjunto de $n$ segmentos utilizando una carga de datos de $O(n+k)$. Sobre este planteamiento inicial se han realizado numerosas mejoras en base a la carga de datos [2] y al orden de complejidad del algoritmo [4]. Bajo ciertos supuestos, este tiempo puede ser optimizado, por ejemplo si sólo existen dos tipos de segmentos, rojos y azules, y sólo se contabilizan las intersecciones entre segmentos de diferentes colores [8].

A pesar de que existen algunas mejoras al método de línea de barrido de Bentley-Ottmann, éste sigue siendo el método más empleado por su relativa sencillez. Sin embargo a nivel de tiempo de ejecución, sólo resulta interesante cuando $k$ es bajo, en otro caso podría ser incluso superior al método de fuerza bruta [5], como hemos comprobado en nuestros experimentos.

El algoritmo de fuerza bruta se sintetiza en el Algoritmo 3.1. El tiempo es claramente $O(n^2)$, aunque está algo mejorado para no comprobar la intersección de los segmentos $s_i$ y $s_j$ si previamente se ha comprobado la intersección entre $s_j$ con $s_i$. La función $intersecta(Segmento\, s1, Segmento\, s2, Punto\, pinter)$ determina si el segmento $s1$ intersecta con el segmento $s2$ y en tal caso devuelve el punto de intersección $pinter$. A pesar de su orden de ejecución cuadrático, este algoritmo se caracteriza por su robustez y simplicidad del código y de las estructuras de datos utilizadas. En la práctica, este algoritmo sería el más adecuado para conjuntos pequeños de datos y escasas ejecuciones del algoritmo.

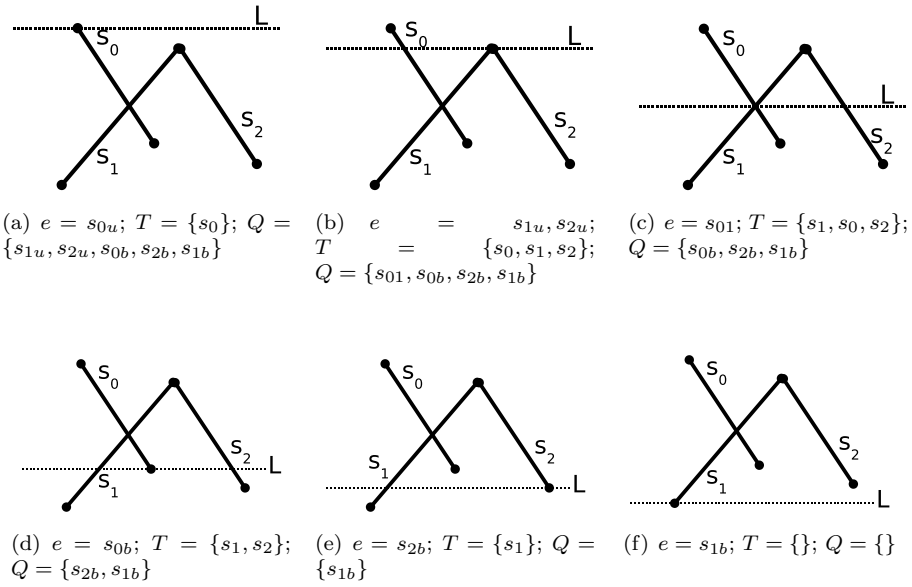
**Algoritmo 3.1.** *void* $PROCEDIMIENTO\_CPU\_Cuadratico(Segmento\, S[n],\ Punto\, Inters[k])$

```
1: INPUT S: array de n segmentos
2: OUTPUT Inters: array con los k puntos de intersección de S
3: LOCAL Punto pinter
4: LOCAL Integer cont = 0
5: for i = 0 TO n do
6:    for j = 0 TO i do
7:       if intersecta (s[i], s[j], pinter)) then
8:          Intersecciones[cont] = pinter
9:          cont = cont + 1
10:      end if
11:   end for
12: end for
```

Sin embargo, cuando $n$ aumenta y el número potencial de intersecciones es bajo, el algoritmo de Bentley-Ottmann permite ejecuciones sustancialmente más rápidas. Este método está descrito en detalle en [5], el Algoritmo 3.2 tan sólo resume los pasos a groso modo. Una línea horizontal $L$ barre el plano de arriba hacia abajo manteniendo ordenados de izquierda a derecha el conjunto de segmentos que atraviesa en una estructura de datos de acceso logarítmico $T$. A medida que dicha línea baja iría encontrando el conjunto de intersecciones a su paso, lo que a nivel de implementación supone una ventaja computacional si sólo se cuestionan posibles intersecciones en parejas de segmentos adyacentes en $T$.

Se necesitan varias estructuras de datos, $T$ de tipo árbol, para mantener ordenados de izquierda a derecha los segmentos atravesador por $L$, y una lista de $Q$ de siguientes eventos. Se considera como evento los extremos de los segmentos y los puntos de corte ya procesados pero aún no superados por $L$. Cada nuevo evento que es alcanzado implica una actualización en $T$, insertar un nuevo segmento, eliminarlo o intercambiar posiciones entre segmentos. Cualquiera de estas acciones supone siempre nuevas posibilidades de intersección en las posiciones de cambio, pero afectando sólo a sus adyacentes. En la Figura 6 se observa el comportamiento de estas estructuras de datos para un ejemplo con 3 segmentos.

Como hemos señalado anteriormente, el algoritmo de línea de barrido mejora al método cuadrático cuando se espera que el número de intersecciones $k$ sea reducido. A medida que éstas aumenta según aparece en la Figura 3, la ventaja computacional disminuye, llegando a ser significativamente más lento. La razón es el esfuerzo adicional para la actualización y consulta de estas estructuras de datos. En esos casos, la única forma de conseguir una mejora computacional consiste en diseñar un algoritmo




(a) $e = s_{0u}$; $T = \{s_0\}$; $Q = \{s_{1u}, s_{2u}, s_{0b}, s_{2b}, s_{1b}\}$

(b) $e = s_{1u}, s_{2u}$; $T = \{s_0, s_1, s_2\}$; $Q = \{s_{01}, s_{0b}, s_{2b}, s_{1b}\}$

(c) $e = s_{01}$; $T = \{s_1, s_0, s_2\}$; $Q = \{s_{0b}, s_{2b}, s_{1b}\}$

(d) $e = s_{0b}$; $T = \{s_1, s_2\}$; $Q = \{s_{2b}, s_{1b}\}$

(e) $e = s_{2b}$; $T = \{s_1\}$; $Q = \{s_{1b}\}$

(f) $e = s_{1b}$; $T = \{\}$; $Q = \{\}$

Figure 2: Ejemplo de ejecución del Algoritmo 3.2

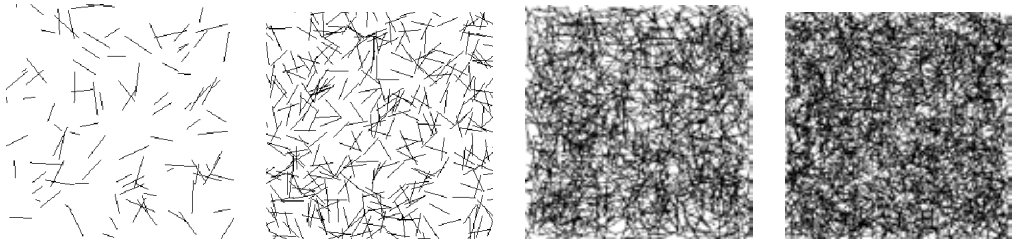


Figure 3: Ejemplos de diferentes concentraciones de nubes de puntos.

paralelo. En la Figura 4 observamos que para concentraciones a partir de 3-15 intersecciones por segmento ya resulta más rápida la intersección de segmentos cuadrática.

**Algoritmo 3.2.** *void PROCEDIMIENTO_CPU_BO(Segmento S[n], Punto Inters[k])*

1: **INPUT** $S$: array de $n$ segmentos
2: **OUTPUT** *Intersecciones*: array con los $k$ puntos de intersección de $S$
3: **LOCAL** $T$: árbol con los segmentos ordenados de izquierda a derecha
4: **LOCAL** $Q \leftarrow S$: lista de eventos ordenados de mayor a menor ordenada
5: **LOCAL** $e \leftarrow$ obtener siguiente evento de Q
6: SI $e$ es el comienzo de un nuevo segmento $s$ (o de varios), entonces $s$ se inserta en $T$ y se cuestiona su posible intersección sus dos vecinos izquierdo y el derecho; en caso positivo la intersección se considera un nuevo evento a intersectar en $Q$.
7: SI $e$ es el final de segmento $s$ (o de varios), en ese caso el hueco que deja en $T$ puede implicar que ahora sus antiguos vecinos intersecten entre sí; en caso positivo la intersección se considera un nuevo evento a intersectar en $Q$.
8: SI $e$ es un punto de intersección entre dos segmentos $s$ y $r$ (también pueden ser varios), en este caso se intercambian las posiciones de $s$ y $r$ en $T$ y se buscan intersecciones de cada uno con el nuevo vecindario.

# 4 Intersección de Segmentos en GPU

La programación paralela de algoritmos geométricos está ampliamente documentada en el ámbito de la Geometría Computacional [1]. Aún así al programador le plantea la necesidad de conocer las diversas arquitecturas paralelas que existen, y cuales de ellas tiene a su alcance. Se estima como óptimo que

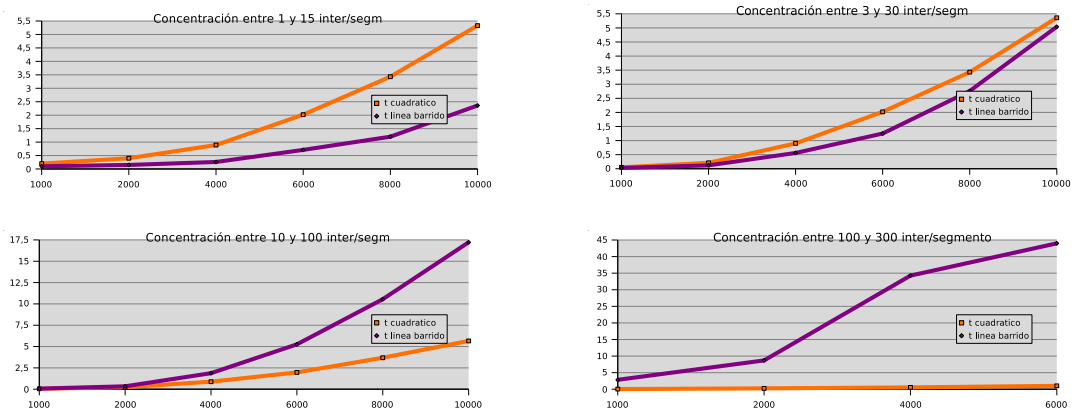



Figure 4: Ejemplos de tiempos de ejecución dependiendo del número de intersecciones por segmento.

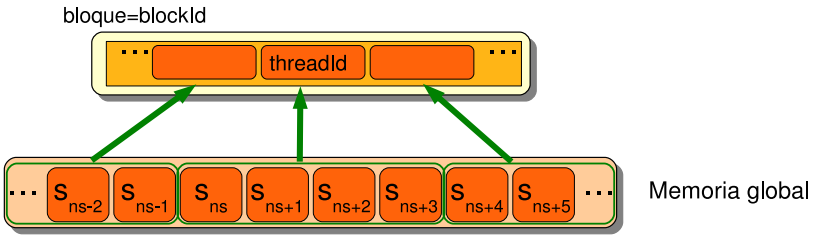



Figura 5: Reparto de segmentos en GPU.

un algoritmo que necesita un tiempo $T(f(n))$ se ejecute en $T(f(n)/M)$ para una arquitectura con $M$ procesadores. Sin embargo en la práctica esto no suele ocurrir porque se necesita tiempo adicional para repartir los datos de entrada entre los diversos procesadores, así como realizar tareas de sincronización entre procesos. Normalmente conseguir un elevado número de procesadores suele implicar acceder a ordenadores de centros de cálculo o investigación, siendo alternativas no siempre accesibles para todas las aplicaciones que requieran, por ejemplo, un masivo cálculo de intersecciones.

Paradógicamente, muchos de los ordenadores personales pensados para vídeo juegos incluyen un verdadero multiprocesador en la tarjeta gráfica, como hemos indicado en la Introducción. Su coste es inferior al del procesador CPU y sin embargo permiten paralelizar problemas obteniendo mejoras cercanas a $200X$. Dependiendo del programador, el esfuerzo para aprender CUDA al nivel necesario puede ser bastante menor que el esfuerzo requerido para implementar y depurar el Algoritmo 3.2. Además, una vez conocida la estrategia de programación, podrá aplicarla más fácilmente a otros problemas futuros.

Se ha escogido el algoritmo 3.1 para adaptarlo a una implementación en CUDA por ser un problema cuadrático optimizado y bien conocido en Geometría Computacional. La simplicidad del código y de las estructuras de datos del Algoritmo 3.1 no necesitan especial adaptación. Un programa en CUDA tiene dos partes, aquella que se ejecuta en CPU que sigue el esquema del Algoritmo 2.1 y otra que se ejecuta individualmente en cada hilo representada en el Algoritmo 4.1 para el caso concreto de la intersección de segmentos.

En un programa CUDA pueden elegirse el número de hilos y el número de bloques, que representan a nivel lógico el número de procesadores y el número de multiprocesadores respectivamente. De este modo, el programador no tiene por qué conocer los detalles concretos de esta arquitectura y un programa puede ejecutarse en cualquier tarjeta soportando la misma versión de CUDA, independientemente del número de procesadores reales que tenga. Para nuestro ejemplo con $n$ segmentos, cada hilo va a cargar localmente y procesar las intersecciones de $SEGMENTOS\_THREAD$ segmentos. Si el número elegido de hilos por bloque es $BLOCKSIZE$, el número total de bloques es $n/(BLOCKSIZE * SEGMENTOS\_THREAD)$. Cada hilo sabe en qué bloque está con el identificador $blockId$ y cual es su índice dentro de ese bloque con $threadId$, lo que le permite conocer cual es la

| segmentos | tcpu (ms) | tgpu (ms) |
|---|---|---|
| 200 | 1,2 | 120 |
| 500 | 7,5 | 114 |
| 1000 | 29 | 115 |
| 2000 | 102 | 121 |
| 4000 | 408 | 145 |
| 8000 | 1638 | 188 |
| 16000 | 6553 | 262 |
| 32000 | 26139 | 439 |
| 64000 | 103896 | 1220 |
| 100000 | 255486 | 2346 |
| 120000 | 577016 | 3166 |
| 150000 | 942578 | 4710 |
| 180000 | 1025072 | 6561 |
| 200000 | 1186250 | 8006 |
| 250000 | 1602760 | 12070 |

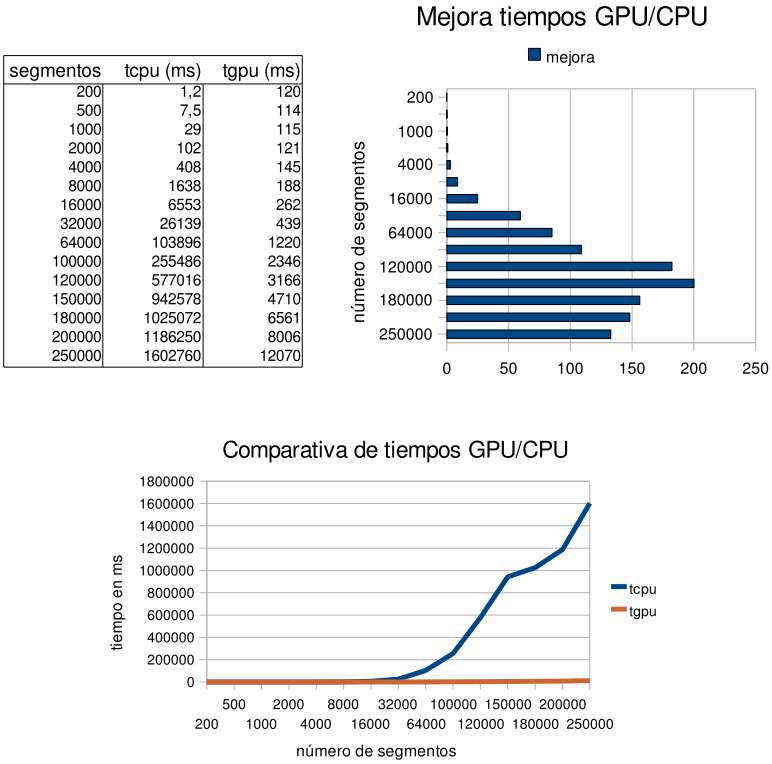



Figure 6: Tiempos de ejecución y mejoras entre CPU y GPU.

posición $ns$ dentro de $S$ donde comienzan los $SEGMENTOS\_THREAD$ asignados a cada hilo (ver Figura 5). El tiempo de ejecución en cada hilo es $O(SEGMENTOS\_THREAD * n)$.

La sentencia 12 del algoritmo merece especial atención puesto que si todos los procesos comparten este vector, añadir los resultados tendrán que escribir simultáneamente en dicho vector incrementando una variable para incrementar paulatinamente la posición donde se va a escribir en el vector. Como todos los procesadores querrán realizar esta acción simultáneamente se necesitan secciones críticas. En CUDA este tipo de operaciones está soportada a partir de procesadores con compatibilidad 1.1 (disponibles desde comienzos de 2008) con las denominadas funciones atómicas.

**Algoritmo 4.1.** *void PROCEDIMIENTO_GPU(Segmento S[n], Punto Inters[k])*

```
1:  INPUT S: array de n segmentos
2:  OUTPUT Inters: array con los k puntos de intersección de S
3:  Integer ns = (blockId * BLOCKSIZE + threadId) * SEGMENTOS_THREAD;
4:  LOCAL Segmento ts[SEGMENTOS_THREAD] ⇐ S[ns, ..., ns + SEGMENTOS_THREAD]
5:  LOCAL Segmento aux
6:  LOCAL Integer cont = 0
7:
8:  for i = 0 TO ns do
9:     aux = S[i]
10:    for j = 0 TO i do
11:       if intersecta (ts[i], s[j], pinter)) then
12:          SEC_CRITICA(Añadir el punto pinter al vector Inters)
13:       end if
14:    end for
15: end for
```

Las pruebas se han llevado a cabo en un ordenador con un procesador de un ordenador Core Quad a $2.33GHz$ con $4GB$ de memoria y una tarjeta gráfica NVIDIA Geforce GTX 280, con 240 procesadores. Los resultados represenados en la Figura 6 nos muestran mejoras de hasta $200X$ de

modo que las intersecciones de 150.000 segmentos se procesan en CPU en más de 15 minutos y en GPU en menos de 5 segundos. En la gráfica de comparativa de tiempos se observa claramente un comportamiento lineal en los tiempos de ejecución en GPU y claramente cuadrático en CPU. Los valores de $BLOCKSIZE = 64$ y $SEGMENTOS\_THREAD = 4$ han sido elegidos experimentalmente.

Incluso para nubes de segmentos dispersas con bajo número de intersecciones, esta implementación mejora en tiempo a la de la línea de barrido para este rango en número de segmentos. Se espera que dicho algoritmo $O(n \log n)$ mejore al de GPU a medida que $n$ se hace infinito, aunque podrían existir problemas de robustez en CPU al utilizar estructuras de datos complejas para tal número de segmentos.

# 5 Conclusiones y Trabajos Futuros

La tecnología CUDA puede convertirse en una herramienta muy interesante para conseguir ejecuciones muy rápidas en numerosos problemas en Geometría Computacional. Existen algunos inconvenientes, como tener tamaños de datos muy elevados o tener que adaptarlos a estructuras de datos tipo vector. El tiempo de ejecución se puede ver ralentizado al necesitar numerosas operaciones atómicas o al realimentar procesos de GPU con CPU sucesivamente. Sin embargo en la gran mayoría de los casos el beneficio en tiempo está garantizado. Las últimas tarjetas que están saliendo al mercado cuentan además con doble precisión, muy importantes para realizar cálculos matemáticos.

# 6 Agradecimientos

Este trabajo ha sido parcialmente subvencionado por el Ministerio de Ciencia y Tecnología de España y por la Unión Europea por medio de los fondos FEDER con el proyecto TIN2004-06326-C03-03, y también por la Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía con los proyectos P07-TIC-02773 y P06-TIC-01403.

# Referencias

[1] S. G. Akl and K. A. Lyons. *Parallel computational geometry.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] I. J. Balaban. An optimal algorithm for finding segments intersections. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 211–219, New York, NY, USA, 1995. ACM.

[3] J. L. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Computers*, 28(9):643–647, 1979.

[4] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39(1):1–54, 1992.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry: algorithms and applications.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[6] R. Fernando. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics.* Pearson Higher Education, 2004.

[7] C. NVIDIA. *NVIDIA CUDA Programming Guide (version 1.0).* NVIDIA Corporation, 2007.

[8] L. Palazzi and J. Snoeyink. Counting and reporting red/blue segment intersections. In *CVGIP: Graph. Models Image Process*, pages 530–540. Springer-Verlag, 1993.

[9] M. Pharr and R. Fernando. *GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation.* Addison-Wesley Professional, March 2005.

# Estrategias de configuración de acciones políticas con valoraciones subjetivas. Búsqueda geométrica.

Rodrigo, Javier * López, Mª Dolores † Lantarón, Sagrario ‡ Caro, Raquel §

**Resumen**

En este trabajo se ponen en práctica diversas técnicas de la Geometría Computacional para analizar un problema de toma de decisiones y configuración de estrategias políticas a lo largo de un mandato, en el que Gobierno y oposición perfilan sus propuestas de actuación sobre dos temas de relevante importancia para los ciudadanos. Se considera una componente variable, tanto en la importancia de los temas a tratar como en la presunción de las estrategias a tomar por los partidos. La finalidad del trabajo se centra en encontrar las estrategias óptimas a seguir por los dos partidos mayoritarios de un país permitiéndoles variar en un cierto grado sus propuestas. Además, el proceso tiene un carácter dinámico ya que se pretende que dichas propuestas vayan modificándose según las previsiones de actuación del partido contrario. Este enfoque junto con la consideración de componentes subjetivas representa la aportación del estudio. El tratamiento del problema se realiza desde un punto de vista geométrico y se desarrolla un algoritmo de búsqueda de estrategias óptimas. Palabras clave: Teoría de Juegos, Competición Política, Algoritmos de Búsqueda, Localización.

## 1 Introducción

Este trabajo aborda la resolución de un tipo de problema de Economía Política ([17], 1999; [20], 2001) a través de herramientas geométricas ([19], 1985). Los puntos del plano, que se denomina plano de políticas, representan las diferentes opciones políticas sobre dos temas de importancia. Se asume que la distancia entre dos puntos da una idea sobre la afinidad de las políticas relativas a esos dos tópicos ([15],1987; [16], 2000). Como la importancia de los temas a tratar no tiene por qué ser igual, ni siquiera estar perfectamente determinada, se propone incorporar un parámetro en ella, la ponderación. Por la subjetividad de la valoración de la importancia de los temas, este parámetro introduce un aspecto variable que ha sido tratado también en distancias como la distancia relativa de Hamming de ponderación convexa ([10], 1996; [12], 1996; [8], 1980). En este trabajo, como enfoque alternativo se considera la distancia euclídea ponderada. La finalidad del estudio se centra en encontrar las estrategias óptimas a seguir por los dos partidos mayoritarios de un país (Gobierno y oposición) permitiéndoles variar en un cierto grado sus propuestas. Además el proceso tiene un carácter dinámico ya que se pretende que dichas propuestas vayan modificándose según las previsiones de actuación del partido contrario. Este enfoque junto con la consideración de componentes subjetivas representa la aportación del trabajo. En cualquier proceso de toma de decisiones, el modelo matemático empleado se verá afectado por los valores numéricos introducidos. Debemos ser conscientes de que la validez de los resultados puede depender de la asignación numérica a parámetros desconocidos, para los que sólo podemos tener en cuenta estimaciones o conjeturas. No obstante, el modelo presentado tiene en cuenta estos factores y arroja resultados fiables que pueden ser aplicados por los partidos políticos. La estructura del trabajo es la siguiente: El modelo y los preliminares del problema se desarrollan en la sección 2. Las estrategias de búsqueda de posiciones óptimas se tratan en la sección 3. La sección 4 presenta el algoritmo de búsqueda de esas estrategias óptimas.

*Departamento de Matemática Aplicada. E.T.S. de Ingeniería. Universidad Pontificia Comillas de Madrid

†Departamento de Matemática Aplicada de la E.T.S.I. Caminos, Canales y Puertos. Universidad Politécnica de Madrid, ma08@caminos.upm.es

‡Departamento de Matemática Aplicada de la E.T.S.I. Caminos, Canales y Puertos. Universidad Politécnica de Madrid

§Departamento de Organización industrial. E.T.S. de Ingeniería. Universidad Pontificia Comillas de Madrid

## 2 El modelo

El problema es una adaptación a este nuevo planteamiento de modelos de la literatura ([1], 2006; [13], 2007). Dicha adaptación se hace de la siguiente forma: Se consideran el plano de políticas definido a partir de dos ítems de actual relevancia, dos partidos políticos que adoptan políticas en esos dos ítems, representadas por los puntos $t_1$ y $t_2$ y la localización de $n$ tipos de votantes representados por los puntos $v_1, \ldots, v_n$ ([20], 2001). Con la consideración de la mediatriz adecuada a la distancia considerada, es posible calcular el número de votantes que elegirían a cada uno de los partidos por proximidad o afinidad a su política. Se acepta que cada partido puede alterar su política en un cierto entorno con el objetivo de obtener el mayor número posible de seguidores. La finalidad es encontrar las situaciones óptimas que le garantizan ese mayor número de seguidores. Suponiendo que ese partido elegiría una de esas posiciones, se desea determinar cuál sería la respuesta del otro partido, es decir, la estrategia óptima a preparar para la posible propuesta de su oponente. Este planteamiento puede ser visto como una versión discreta del Juego de Voronoi. En este juego, dos jugadores se localizan en el plano con la finalidad de ganar el mayor área posible ([11], 2003; [7], 2003; [18], 2004; [6], 2004). En el presente trabajo dos puntos se localizan con el propósito de ganar el mayor número posible de puntos de un conjunto dado, en lugar de mayor área. Modelos similares de localización se han planteado en diferentes campos como la organización industrial, tratamiento de imágenes, movimientos de robots, etc. También se han dado modelos de estrategia óptima para la localización de los partidos en trabajos de la Economía Política. La mayoría de ellos consideran la población como un continuo ([2], 2001). En este artículo, la nueva visión consiste en el trabajo con una población discreta, así como en la aplicación de técnicas de la Geometría Computacional adaptada al problema y en la consideración de entornos y distancias ponderadas y parámetros de incertidumbre. Se considera que a lo largo del periodo electoral existe un debate sobre dos temas detectados de interés en la sociedad. Se asume que cada uno de ellos tiene importancia distinta lo que se representa a través de una ponderación. Sea la política adoptada por Gobierno y oposición en cada uno de los dos temas representada como los puntos $t_1 = \left(t_1^1, t_2^1\right)$ y $t_2 = \left(t_1^2, t_2^2\right)$ y sean $v_i = \left(v_1^i, v_2^i\right)$ con $i = 1, \ldots, n$, las coordenadas que representan las preferencias respecto a estos temas de los $n$ tipos de votantes de una cierta población. Se define en el juego planteado la función de utilidad de una política $t_j$ para cada tipo $v_i$ como: $\gamma\left(t_j, v_i\right) = -d\left(t_j, v_i\right)^2$ donde $d\left(t_j, v_i\right)$ representa la distancia euclídea ponderada entre la postura política $t_j$ y el tipo $v_i$: $d\left(t_j, v_i\right) = \sqrt{\alpha\left(v_1^i - t_1^j\right)^2 + (1-\alpha)\left(v_2^i - t_2^j\right)^2}$. El parámetro $\alpha \in (0,1)$ mide la importancia de cada uno de los temas a tratar. Las funciones de ganancia en el juego presentado vienen dadas por: $\Pi^1\left(t_1, t_2\right) =$ número de puntos $v_i$ tales que $d\left(v_i, t_1\right) \leq d\left(v_i, t_2\right)$, $\Pi^2\left(t_1, t_2\right) =$ número de puntos $v_i$ tales que $d\left(v_i, t_1\right) > d\left(v_i, t_2\right) = n - \Pi^1\left(t_1, t_2\right)$, si $t_1 \neq t_2$. El conjunto de puntos del primer partido será el formado por aquellos tipos que están más cerca de la posición $t_1$ que de la del segundo partido. Para localizar estos tipos se utiliza la mediatriz del segmento que une $t_1$ y $t_2$. Ésta está dada por: $\left\{(x,y) \in \mathbb{R}^2 / \alpha\left(x - t_1^1\right)^2 + (1-\alpha)\left(y - t_2^1\right)^2 = \alpha\left(x - t_1^2\right)^2 + (1-\alpha)\left(y - t_2^2\right)^2\right\}$. Dado que la política es un proceso dinámico, con respuestas de cada partido a las propuestas presentadas por su opuesto, plantearemos el juego de elección de posturas políticas a seguir por cada partido de la siguiente manera: Partiendo de las propuestas planteadas por los partidos mayoritarios de un país sobre los temas a tratar, el partido gobernante busca una estrategia óptima que le acerque al mayor número posible de ciudadanos dentro de un entorno que representa su flexibilidad ideológica. Por su parte, la oposición espera esta reacción del Gobierno y prepara otra estrategia distinta en su entorno de flexibilidad para encontrar la mejor respuesta a cualquiera de las posibles posiciones óptimas tomadas por el Gobierno. Se entra así en una evolución de las posiciones de los partidos dentro de unos entornos de flexibilidad. Estos tipos de juegos secuenciales ya han sido estudiados, con otras técnicas, en el caso continuo ([20], 2001). Los entornos de flexibilidad también se verán afectados por la ponderación otorgada a cada tema. Se definen como:

**Definición 2.1.** Partiendo de la posición inicial de cada partido , $i = 1, 2$, se define su entorno de flexibilidad como: $N_i = \left\{(x,y) \in \mathbb{R}^2 / \alpha\left(x - x_1^i\right)^2 + (1-\alpha)\left(y - x_2^i\right)^2 \leq R_i^2\right\}$, donde $R_i$ con $i = 1, 2$, representa el grado de flexibilidad de cada partido, es decir, $N_i$ es la región interior de la elipse de centro la postura inicial adoptada por el partido y semiejes $\frac{R_i}{\sqrt{\alpha}}$ y $\frac{R_i}{\sqrt{1-\alpha}}$.

Se observa que a menor trascendencia de uno de los temas a considerar, por ejemplo en el primero ($\alpha$ más cercano a cero) la flexibilidad que el partido puede llevar a cabo en él es mayor. Este compor-

tamiento es lógico ya que en temas de gran relevancia, los partidos deben mantenerse más cercanos a su postura ideológica inicial. A lo largo del trabajo se supone que los entornos de cada partido son disjuntos, es decir, $N_1 \cap N_2 = \emptyset$. Esto asegura el que los partidos no ofrezcan las mismas políticas.

## 2.1 Clasificación de los votantes por regiones

El entorno de flexibilidad de cada partido le asegura cierto número de votantes sea cual sea la posición tomada por su oponente dentro de su entorno. De esta forma, los puntos del conjunto de votantes se clasifican en tres regiones:

- Votantes seguros para el primer partido.
- Votantes seguros para el segundo partido.
- Votantes indecisos que pueden ser captados por aquel partido que se situé en la zona adecuada de su entorno. Estos votantes pueden ser decisivos y hacia ellos debe orientarse la campaña y las propuestas políticas.

A continuación se determinan e interpretan las regiones. Figura 1.

- Puntos que el primer partido captura siempre: Los puntos $(v_{i1}, v_{i2})$ que pertenecen al conjunto: $\{(x,y) / \max\{[(x,y),(c_1,c_2), / (c_1,c_2) \in N_1]\} \leq d[(x,y),N_2]\}$ La frontera de este conjunto es: $\sqrt{\alpha (x - x_1^2)^2 + (1-\alpha)(y-x_2^2)^2} - \sqrt{\alpha (x-x_1^1)^2 + (1-\alpha)(y-x_2^1)^2} = R_1 + R_2$ Los votantes seguros para el primer partido resultan aquellos que se sitúan en la región limitada por esta curva en la que se encuentra el partido.
- Puntos que el segundo partido captura siempre: Los puntos $(v_{i1}, v_{i2})$ que pertenecen al conjunto: $\{(x,y) / \max\{[(x,y),(c_1,c_2), / (c_1,c_2) \in N_2]\} \leq d[(x,y),N_1]\}$ La frontera de dicho conjunto es: $\sqrt{\alpha (x - x_1^1)^2 + (1-\alpha)(y-x_2^1)^2} - \sqrt{\alpha (x-x_1^2)^2 + (1-\alpha)(y-x_2^2)^2} = R_1 + R_2$

Los votantes seguros para el segundo partido serán aquellos que se sitúan en la región limitada por esta curva en la que se encuentra el partido. Los votantes indecisos serán aquellos que se sitúan entre las dos curvas determinadas anteriormente.

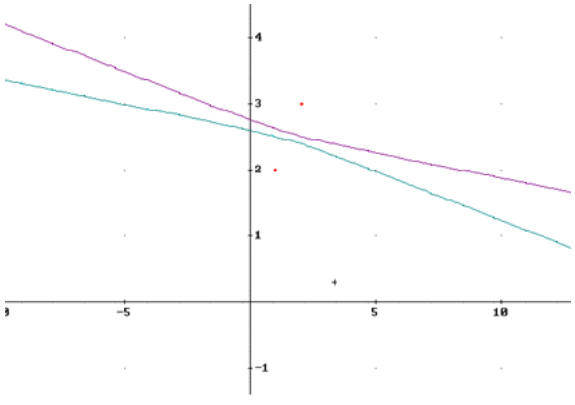



Figura 1: Regiones de captación de votantes.

La consideración de pesos en la distancia como un medidor de la importancia de los temas tratados, resulta relevante. Las curvas que delimitan las regiones que clasifican los votantes van cambiando y por tanto cambian las regiones de captación según los pesos. La figura 2 ilustra estas regiones para el caso particular donde las posiciones iniciales de los partidos coinciden en alguno de los temas tratados. Por ejemplo $\left(x_1^1, x_2^1\right) = (0,0)$, $\left(x_1^2, x_2^2\right) = (2,0)$, $R_1 = R_2 = \frac{1}{2}$ y $\alpha$ varía de 0.2 a 0.9 con paso 0.1.

En estos casos uno de los temas no resulta relevante ya que los dos partidos están inicialmente de acuerdo en él (en el ejemplo se trata del segundo tema). Es este caso, un mayor peso y por tanto, una mayor trascendencia del primer tema, representa que la región de votantes seguros para cada partido aumenta, conteniendo a las regiones que corresponden a pesos inferiores. Con ello el número de votantes indecisos desciende en función de la importancia que tenga el tema considerado.

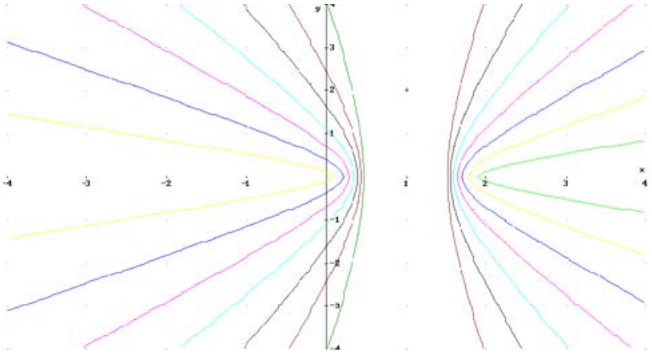


Figura 2: Regiones para la captación de conjuntos de votantes cuando los partidos coinciden en alguna de las políticas a ofrecer y se varía el peso.

# 3 Estrategias para la búsqueda de posiciones óptimas

## 3.1 Búsqueda de posiciones óptimas para el partido gobernante

El resultado presentado por los autores en [1], 2006 donde se considera la distancia euclídea, puede ser generalizado en este caso con la siguiente proposición:

**Proposición 3.1.** *Una postura óptima $t_1$ para el primer partido, para una posición dada $t_2$ del segundo, siempre se encuentra en la frontera de $N_1$ y en el arco de la elipse localizado entre los puntos $p'$, $p''$ intersección con las rectas tangentes a la elipse desde $t_2$ (parte visible de $N_1$ desde $t_2$). Figura* 3.

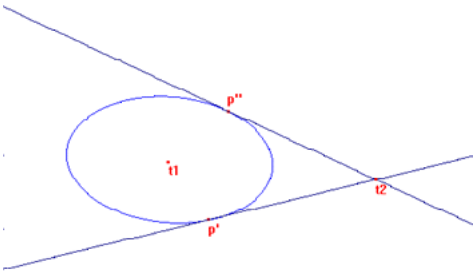



Figura 3: $p', p''$ delimita una zona donde se localiza el arco de máxima captación para $t_1$.

Dentro del arco definido en la proposición 1, llamamos $A$ a la región de máxima captación de votantes por parte del primer partido. El proceso a seguir para la determinación de $A$ se basa en el cálculo de la zona de máxima intersección entre dicho arco y los conjuntos

$$\left\{(x,y) \in \mathbb{R}^2 / \alpha \left(x - v_{i1}\right)^2 + (1-\alpha)\left(y - v_{i2}\right)^2 \leq \alpha \left(t_1^2 - v_{i1}\right)^2 + (1-\alpha)\left(t_2^2 - v_{i2}\right)^2\right\}$$

con $i = 1, \ldots, n$.

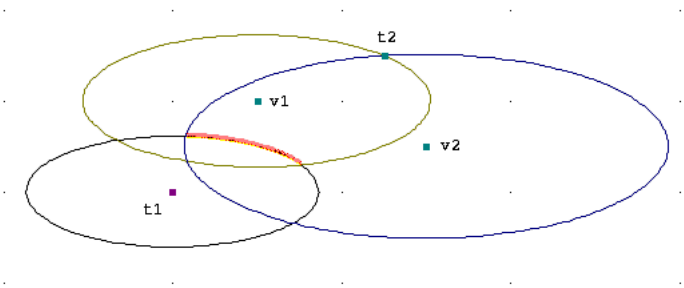



Figura 4: El arco marcado representa la zona de captación para $t_1$ de los puntos $v_1, v_2$ en la frontera de $N_1$.

### 3.2 Estrategias de respuesta para la oposición

Como el partido de la oposición supone que a partir de su propuesta inicial, el Gobierno elegirá una postura que le garantiza un mayor número de seguidores, situada en $A$, la idea ahora es preparar la estrategia de respuesta adecuada. Para ello se supone que seguirá la postura más conservadora, es decir, debe encontrar la posición óptima, la que le garantice un mayor número de seguidores, sea cual sea la posición del primer partido dentro de esa región óptima.

**Proposición 3.2.** *Dentro de $N_2$, llamamos $B$ a la región de máxima captación de votantes por parte del segundo partido suponiendo que el primero se sitúa en $A$. La región $B$ se calcula como la máxima intersección entre $N_2$ y los entornos centrados en los votantes y de radio la mínima distancia de estos a $A$: $\left\{(x,y) \in \mathbb{R}^2 / \alpha\left(x - v_{i1}\right)^2 + (1-\alpha)\left(y - v_{i2}\right)^2 \leq \left(d\left(v_i, A\right)\right)^2\right\}$, con $i = 1, \ldots, n$.*

Calculo de la distancia de $v_i$ a $A$: La distancia de un punto exterior a un entorno se alcanza en la intersección del segmento que une el punto con el centro del entorno y la frontera de dicho entorno. Por tanto, si el segmento que une $v_i$ con el centro de $N_1$ interseca a $A$ (parte de la frontera del entorno), la distancia de $v_i$ a $A$ se alcanza en ese punto intersección. En otro caso se alcanza en uno de los extremos de $A$.

## 4 El algoritmo

Se desarrolla un algoritmo que permite obtener las regiones $A$ y $B$ de máxima captación de votantes para cada partido.

### 4.1 Cálculo de la región $A$

En este apartado, se desarrolla el algoritmo que permite obtener $A$. Se trata de una adaptación del algoritmo presentado en [1], 2006. La complejidad de dicho algoritmo en el peor de los casos es $O\left(n \log n\right)$ ([4], 1997). Procedimiento:

- Paso 1: Encontrar $p'$ y $p''$ (proposición 1). Definir un contador $c'$ con valor inicial $c' = 0$.
- Paso 2: Sea $L$ una lista vacía y $m$ otro contador con valor $m = 0$. Para cada punto $v_i$, se encuentran los puntos de intersección de la frontera de $N_1$ y la frontera del entorno elíptico centrado en $v_i$ que pasa por $t_2$.
    1. Si no existe intersección por estar $N_1$ contenido en el entorno elíptico centrado en $v_i$ que pasa por $t_2$, se incrementa $m$ en una unidad.
    2. Si no existe esa intersección porque los conjuntos son disjuntos, se mantiene el valor $m$.

3. Si existen dos puntos de intersección fuera de la parte visible de $N_1$ desde $t_2$, se incrementa $m$ en una unidad.
4. En cualquier otro caso:
   (a) Si los dos puntos pertenecen a la parte visible de $N_1$ desde $t_2$, entonces se incluyen los dos puntos en $L$.
   (b) Si $x_i'$ pertenece a la parte visible y $x_i^*$ no, se incluye $x_i'$ en $L$.
   (c) Si $x_i^*$ pertenece a la parte visible y $x_i'$ no, se incluye $x_i^*$ en $L$ y se incrementa $c'$ en una unidad.

- Paso 3. Se ordenan los puntos de la lista $L$ según el ángulo respecto $t_1$ (en el sentido horario).
- Paso 4. Sea $c = c' + m$ y $x = p'$. Se avanza en la lista $L$ realizando lo siguiente a cada elemento:
  1. Si se encuentra un elemento $x_i'$ se toma $c = c + 1$, y si $c > m$ entonces se hace $m = c$ y $x = x_i'$
  2. Si se encuentra un $x_i^*$, se hace $c = c - 1$.

Nota: Si $x_i'$ y $x_i^*$ coinciden por ser tangentes las elipses consideradas, se toma $x_i'$ previo a $x_i^*$ en la lista $L$. Cuando el algoritmo completa la ejecución, el contador $m$ indica el máximo número de puntos $v_i$ que el primer partido puede ganar. El extremo inicial del arco de $N_1$ donde este partido debe localizarse, es el punto almacenado en la variable $x$, y el extremo final es el punto que sigue al anterior en la lista $L$.

## 4.2 Cálculo de la región $B$

Se desarrolla un algoritmo que permite obtener la región $B$ y que obtiene el máximo número de votantes del segundo partido para cualquier posición del primero en $A$. La idea del algoritmo es la siguiente: Para todos los puntos $(v_{i1}, v_{i2})$, se considera $D_i$, un entorno elíptico centrado en $(v_{i1}, v_{i2})$ y radio la distancia de $(v_{i1}, v_{i2})$a $A$. Se determina la máxima intersección de $D_i$, $i = 1, \ldots, n$ que interseque con $N_2$. Un procedimiento para obtener dicha intersección se puede realizar a través del siguiente algoritmo desarrollado por los autores: Procedimiento:

- Input: el entorno $N_2$ y las $n$ elipses $D_1, \ldots, D_n$, que suponemos no tangentes a $N_2$, ninguna de ellas contenida en $N_2$ (este caso no se puede dar en nuestro planteamiento), no todas disjuntas de $N_2$, ni todas conteniendo a $N_2$ (en estos dos últimos casos la zona de máxima intersección será $N_2$, y la máxima intersección 0 y $n$ respectivamente)
- Paso 1: Intersecamos las fronteras de $D_1$ y $N_2$, y hallamos los dos puntos de intersección $a_1$, $b_1$ (si las fronteras son disjuntas ir al paso 3).
- Paso 2: Aplicamos el algoritmo de la sección 4.1. para encontrar la zona de máxima intersección de $D_2, \ldots, D_n$ en el arco de $D_1$ que une $a_1$ con $b_1$ dentro de $N_2$, y el número de elipses, $k_1$, que intersecan en ese arco.
- Paso 3: Repetimos los pasos 1 y 2 para $D_2, \ldots, D_n$
- Paso 4: Tomamos, de las zonas halladas en el paso 2, aquéllas en las que se alcanza el $\max_{i=1,\ldots,j} k_i$, donde $j$ es el número de elipses que tienen frontera no disjunta con $N_2$. Para ello se crea una lista $L$ con los puntos $\alpha_i, \beta_i$, extremos de los arcos obtenidos en el paso 2. Los arcos en la lista $L$ no siguen ningún tipo de ordenación.
- Paso 5: Se deben obtener todas las regiones acotadas por los arcos obtenidos en el paso 4 o aquellas zonas acotadas por arcos obtenidos en el paso 4 y la frontera de $N_2$. Existen dos posibilidades:
  1. La zona a encontrar está delimitada únicamente por los arcos obtenidos en el paso 4.
     (a) Se toman los puntos extremos que delimitan el primer arco no utilizado de la lista $L$. Llámense $p1$, $p2$. Se marca el arco como utilizado.

(b) Se hace una búsqueda del punto $p1$ entre los extremos del resto de arcos no utilizados de la lista. Sea el arco de extremos $\alpha_i, \beta_i$ en el que se encuentra la equivalencia, siendo por ejemplo $p1 = \alpha_i$, entonces se asigna nuevo valor a $p1$ ($p1 \leftarrow \beta_i$).

(c) Se repite el proceso de búsqueda las veces necesarias hasta que $p1 = p2$. (Región cerrada)

2. La zona a encontrar está delimitada por los arcos obtenidos en el paso 4 y por el entorno $N_2$.

(a) Se toman los puntos extremos que delimitan el primer arco no utilizado de la lista $L$. Llámense $p1$, $p2$. Se marca el arco como utilizado (idéntico a 5.1.*a*)

(b) Se hace una búsqueda del punto $p1$ entre los extremos del resto de arcos no utilizados de la lista. Si se encuentra en algún extremo de arco, se sigue el proceso detallado en 5.1.*b*., continuando hasta que $p1$ no se encuentre en ningún extremo de arco no utilizado, entonces $p1$ pertenece al entorno $N_2$, llámese $pB1$, asignándose a $p1$ el valor $p2$. ($p1 \leftarrow p2$)

(c) De nuevo se realiza la búsqueda de $p1$ entre los extremos del resto de arcos no utilizados, siguiendo el proceso indicado en 5.2.*b*. En ese momento, el punto $p1$ es punto del entorno $B$, llámese $pB2$. El arco $pB1$, $pB2$, del entorno $N_2$ pertenece a la zona de máxima intersección. (Región cerrada)

- Output: Las regiones halladas en el paso 5 son las zonas de máxima intersección en $N_2$ del conjunto de elipses, siendo la máxima intersección: $\max_{i=1,\ldots,j} k_i + 1$.

**Observación 4.1.** La complejidad del algoritmo viene dominada por la aplicación $n$ veces del algoritmo para el cálculo de la región $A$, presentado en la sección 4.1. Como éste tiene complejidad $n \log n$, la complejidad del algoritmo desarrollado es $n^2 \log n$. Puede observarse que el procedimiento presentado tiene la misma complejidad que el desarrollado por [3], 1979 para el cálculo de máxima intersección de un conjunto de círculos, constituyendo así una alternativa para el mismo en el caso de elipses. En cualquier caso existe un algoritmo que calcula la máxima intersección por medio de la construcción de un arreglo de círculos, ligeramente más rápido ([9], 1992; [21], 1995, [5], 2005). No obstante, el aquí desarrollado resulta más fácil de implementar y por ello más práctico.

# 5 Conclusiones

En este trabajo se aborda el diseño de estrategias políticas óptimas ante cualquier cambio importante de la situación en la que resulta necesario flexibilizar las exigencias. El estudio se realiza a través de un modelo geométrico discreto basado en técnicas de la Geometría Computacional. Es sabido que conceptos de la Geometría Computacional pueden ser aplicados a algunos campos de la Economía, permitiendo resolver ciertos problemas en el área de la Economía Política. En este estudio, las técnicas empleadas presentan una nueva visión donde las aportaciones más relevantes se centran, por un lado en la consideración subjetiva y variable de la importancia de los elementos a tratar en cada momento de la actualidad política de cada país, por otro en la consideración de entornos que representan la flexibilidad política que los partidos pueden permitirse en cada uno de los temas a tratar. Dicha flexibilidad también se ve afectada por un factor de importancia para dichos temas que puede considerarse indeterminado o impreciso. Estas técnicas desarrolladas permiten diseñar estrategias de respuesta jugando con las opciones que el adversario político es probable que vaya a realizar. Además, se da un algoritmo que obtiene las soluciones óptimas para la captación máxima de votantes en ese proceso de variación de propuestas, por medio del cálculo de la zona de máxima intersección en un conjunto de elipses. Aunque existen algoritmos para hallar la región de máxima intersección en un arreglo de curvas de complejidad ligeramente inferior al presentado, éste presenta la ventaja de resultar más sencillo de implementar.

# Referencias

[1] M. Abellanas, I. Lillo, M. López y J. Rodrigo. *Electoral strategies in a dynamical democratic system: geometric models*. European Journal of Operational Research 175, 870–878, 2006.

[2] H. Ahn, S. Cheng, O. Cheong, M. Golin, y R. Oostrum. *Competitive Facility Location along a Highway.* 7th Annual International Computing and Combinatory Conference 2108 of LNCS, 2001.

[3] J.L. Bentley y T.A. Ottmann. *Algorithms for reporting and counting geometric intersections.* IEEE Trans Comput. C-28, 643-647, 1979.

[4] M. de Berg, M. van Kreveld, M. Overmars y O. Schwarzkopf. *Computational Geometry*, Algorithms and Applications. Springer. New York, 1997.

[5] S. Cabello, J.M. Díaz-Báñez, S. Langerman, C. Seara y I. Ventura. *Reverse facility location problems.* In Proceedings of the XI EGC´05, 263-270, 2005.

[6] O. Cheong, S. Har-Peled, N. Linial y J. Matousek. *The One-Round Voronoi Game.* Discrete and Computational Geometry 31 (1), 125-138, 2004.

[7] F. Dehne, R. Klein y R. Seidel. *Maximizing a Voronoi Region: The Convex Case.* Lectures Notes in Computer Science. Proc ISAAC 2518; 624-634, 2002.

[8] D. Dubois y H. Prade. *Fuzzy sets and Systems.* Theory and Applications. Academic Press, New York, 1980.

[9] H. Edelsbrunner, L. Guibas, J. Pach, R. Pollack, R. Seidel y M. Sharir. *Arrangements of curves in the plane-topology, combinatorics and algorithms.* Theoretical computer science, 92, 319.336, 1992.

[10] T.A. Estlin y R.J. Mooney. *Multistrategy learning of search control for partial-order planning.* In Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI Press/MIT Press volume I, 843-848, 1996.

[11] S. Fekete y H. Meijer. *The One-Round Voronoi Game Replayed.* Workshop on Algorithms and Data Structures, Springer Lecture Notes in Computer Science 2748, 150-161, 2003.

[12] F. Herrera, V. Herrera y J.L. Verdegay. *A model of consensus in group decision making under linguistic assessments.* Fuzzy Sets and Systems 78, 73-87, 1996.

[13] I. Lillo, M. López y J. Rodrigo. *A Geometric study of the Nash equilibrium in a weighted case.* Applied Mathematical Sciences 55 (1), 2715-2725, 2007.

[14] H. Llavador. *Voting with preference over margins of victory.* Mathematical Social Science. Doi: 10.1016/ J.mathsocsci. 2008-05.009, 2008.

[15] A. Okabe y A. Suzuki. *Stability of Spatial Competition of Many Firms in a Bounded Two-Dimensional Space.* Environment and Planning A 19, 1067-1082, 1997.

[16] A. Okabe, B. Boots, K. Sugihara, y S. Chiu. *Spatial Tessellations Concepts and Applications of Voronoi diagrams.* John Wiley and Sons. Chichester, 2000.

[17] T. Persson y G. Tabellini. *Political Economics and Public Finance.* NBER Working Papers Series. Handbook of Public Economics, vol III, 1999.

[18] F. Plastria y E. Carrizosa. *Optimal location and design of a competitive facility.* Mathematical Programming 100, 247-265, 2004.

[19] F. Preparata y M. Shamos. *Computational Geometry. An introduction.* Springer-Verlag. New York, 1985.

[20] J. Roemer. *Political Competition.* Harvard University Press, 2001.

[21] M. Sharif y P.K. Agargual. *Davenport-Schinzel sequences and their geometric applications.* Cambridge University Press, 1995.

# Búsqueda geométrica del equilibrio en un juego con restricciones de entorno

Abellanas Oar, Manuel * López González, María Dolores †
Rodrigo Hitos, Javier ‡

**Resumen**

Se propone el estudio del equilibrio de Nash en un juego bidimensional en el que dos jugadores, representados por dos puntos, buscan captar el mayor número posible de puntos pertenecientes a un conjunto finito dado. Los dos jugadores tienen restringidas sus posiciones mediante dos entornos circulares disjuntos. Se aplica este juego a la discretización de un modelo de competición política. Palabras clave: Equilibrio de Nash; Arreglo de círculos; Máxima intersección.

**Abstract**

In this paper a study of the Nash equilibrium in a geometric game is performed. The two players are represented by two points in the plane and their pay-off is the number of points they get from a finite subset of the plane. The two players have a restriction in their positions in the plane by means of two disjoint circular neighbourhoods. This game is applied to create models of political competition. Key words: Nash Equilibrium; Arrangement of circles; Maximum intersection.

## 1 Introducción

El equilibrio de Nash ha sido estudiado como un modelo general de competición. Son posiciones para los jugadores que garantizan que el competidor no puede incrementar su ganancia moviéndose. Fue establecido por primera vez por John Forbes Nash en su conferencia: Non-cooperative games ([10], 1951), como una forma de obtener una estrategia óptima para juegos con dos ó más jugadores. [13] (1997), [8] (1973), [9] (1976), entre otros, han demostrado que generalmente no existen posiciones de equilibrio de Nash cuando la competición tiene lugar en un espacio de más de una dimensión. En este trabajo se desarrollan técnicas geométricas para la búsqueda de posiciones de equilibrio en un juego bidimensional discreto (en el sentido de que el número de posibles ganancias es finito). La aportación del trabajo consiste en la consideración de restricciones en los movimientos de los dos jugadores, lo que supone una forma de evitar el problema de la no existencia en general de equilibrio comentada anteriormente. Las restricciones en el conjunto de estrategias a tomar se reflejan en la consideración de entornos para los dos jugadores que compiten por ganar el mayor número de puntos dentro de un conjunto finito. De esta forma, los jugadores que se enfrentan, $p$ y $q$, tienen restringidos sus movimientos a dos entornos circulares disjuntos, $B = C(x_0, r)$ y $B^{'} = C(x_1, r)$. Este planteamiento asegura el que ambos jugadores no puedan situarse en la misma posición, evitando así posiciones de equilibrio no deseadas de la forma $(t, t)$. Una simplificación de este modelo ya ha sido tratada en anteriores trabajos, en los que sólo se permitía adopción de estrategias a uno de los jugadores ([1], 2006), para el que se buscaba la estrategia óptima. La estructura del trabajo es la siguiente: En la sección 2 se introduce el problema y se desarrolla el modelo matemático, poniendo énfasis en su análisis geométrico. En la sección 3 se presentan condiciones necesarias ó suficientes para que existan

*Departamento de Matemática Aplicada de la Facultad de Informática, Universidad Politécnica de Madrid, mabellanas@fi.upm.es

†Departamento de Matemática e Informática Aplicadas a la Ingeniería Civil de la E.T.S.I. Caminos,Universidad Politécnica de Madrid, ma08@caminos.upm.es

‡Departamento de Matemática Aplicada. E.T.S.I., Universidad Pontificia Comillas, jrodrigo@upcomillas.es

posiciones de equilibrio y se ilustra con ejemplos algunos casos particulares. En la sección 4 se crea un algoritmo para encontrar posiciones de equilibrio en el juego planteado. En la sección 5 se da una interpretación del juego dentro de la economía política.

## 2 El modelo

El juego que se plantea puede ser interpretado como una versión discreta del juego de Voronoi. El juego está definido en un espacio bidimensional: Los dos jugadores eligen sus posiciones en el plano para atraer el mayor número posible de puntos de entre $n$ puntos fijos en el plano. Cada jugador captura los puntos que están más cerca de su posición que de la del otro jugador. Así, la mediatriz de las localizaciones de los dos jugadores divide el plano en dos regiones: cada jugador ganará los puntos que están en su semiplano, y el ganador será el jugador cuya región contenga más puntos ([15], 1994; [17], 1997; [2], 2000; [12], 2000). Se formaliza el juego de la siguiente manera: Se denota a los dos jugadores por $p$ y $q$, y a sus localizaciones en el plano se las llama generalmente $t_1$, $t_2$. Los puntos del plano que se disputan $p$ y $q$ son denotados por $p_1, \ldots, p_n$. Las funciones de ganancias del juego vienen dadas por: $\Pi^1(t_1, t_2) =$ número de puntos $p_i$ tales que $d(p_i, t_1) \leq d(p_i, t_2)$ $\Pi^2(t_1, t_2) =$ número de puntos $p_i$ tales que $d(p_i, t_1) > d(p_i, t_2) = n - \Pi^1(t_1, t_2)$ donde $d(t, p_i)$ es la distancia euclídea entre los puntos $t$ y $p_i$. Permitiremos a los dos jugadores $p$ y $q$ variar su posición dentro de un entorno, es decir; los jugadores que se enfrentan tienen restringidos sus movimientos a dos entornos circulares disjuntos, $B = C(x_0, r)$, $B^{'} = C\left(x_1, r^{`}\right)$, donde $C(x_0, r)$ es el círculo cerrado de centro $x_0$ y radio $r$. Ese punto $x_0$ se puede interpretar como la posición inicial de $p$. Una forma de asegurar esta restricción es asignar ganancias nulas a cada uno de los jugadores si están fuera de sus entornos, es decir, definir $\Pi^1(t_1, t_2) = 0$ si $t_1 \notin B$, $\Pi^2(t_1, t_2) = 0$ si $t_2 \notin B^{`}$, con las ganancias definidas anteriormente si $t_1 \in B$, $t_2 \in B^{`}$. Con este modelo, los autores en [1], 2006, determinan las posiciones óptimas (zonas de máxima captación) para cada jugador dentro de su entorno (ver figura 1).

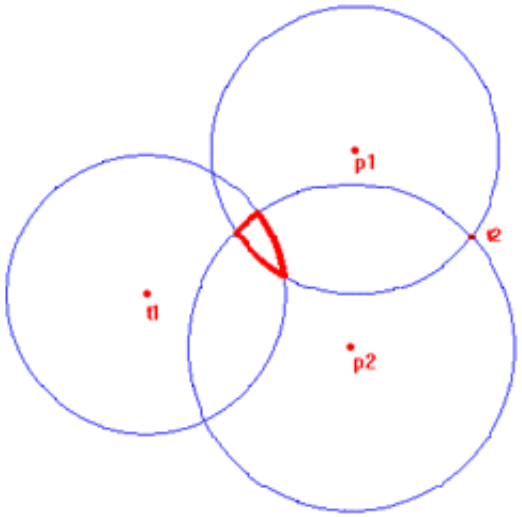



Figura 1: La región limitada por los arcos marcados representa la zona de máxima captación para el jugador p en su entorno dada una posición t2 de q considerando los puntos $p_1$, $p_2$

El concepto de equilibrio que se considera es el de Nash. Su definición adaptada al juego planteado es:

**Definición 2.1.** $(t_1^*, t_2^*)$ es una posición de equilibrio de Nash si

$\Pi^1(t_1, t_2^*) \leq \Pi^1(t_1^*, t_2^*), \Pi^2(t_1, t_2^*) \leq \Pi^2(t_1^*, t_2^*), \forall t_1 \in B, t_2 \in B^{`}$ En la siguiente sección se establecen condiciones necesarias ó suficientes para la existencia de equilibrio en el juego.

# 3 Equilibrio con restricciones

## 3.1 Condiciones de existencia

Empezamos viendo una proposición que es prácticamente la traducción de la definición de equilibrio de Nash a las ganancias del juego planteado.

**Proposición 3.1.** *Las posiciones de equilibrio en el juego planteado serán las posiciones* $(t_1, t_2)$ *de manera que* $t_1$ *está situado en la zona en* $B$ *de máxima intersección de círculos de centro la posición de algún* $p_i$ *y radio la distancia de éste a* $t_2$*, y* $t_2$ *está situado en la zona en* $B^{'}$ *de máxima intersección de círculos de centro la posición de algún* $p_i$ *y radio la distancia de éste a* $t_1$ *([1], 2006).*

*Demostración.* Al estar $q$ localizado en $t_2$, que está en la región de máxima captación de puntos cuando $p$ está en $t_1$, si $q$ se mueve a $t_2^0$ se cumplirá que $\Pi^2\left(t_1, t_2^0\right) \leq \Pi^2\left(t_1, t_2\right)$, y lo mismo pasará con $p$ □

Damos ahora una condición suficiente para que las posiciones $(t_1, t_2)$ definidas en la proposición anterior existan:

**Proposición 3.2.** *Sean* $p_1, \ldots, p_n$ *las posiciones en el plano de los* $n$ *puntos. Llamamos* $p_{i_1}^1, \ldots, p_{i_k}^1$ *a los puntos del conjunto anterior tales que* $d\left(p_{i_j}^1, B\right) \leq d\left(p_{i_j}^1, B^{'}\right)$*,* $p_{i_{k+1}}^1, \ldots, p_{i_n}^1$ *a los que cumplen que* $d\left(p_{i_j}^2, B\right) > d\left(p_{i_j}^2, B^{'}\right)$*, y llamamos* $B_j^1 = C\left(p_{i_j}^1, d\left(p_{i_j}^1, B^{'}\right)\right) \cap B$*,* $B_j^2 = C\left(p_{i_j}^2, d\left(p_{i_j}^2, B\right)\right) \cap B^{'}$*. Entonces, si* $\cap_{j=1}^k B_j^1 \neq \emptyset$ *y* $\cap_{j=k+1}^n B_j^2 \neq \emptyset$*, cualquier posición* $(t_1, t_2)$ *con* $t_1 \in \cap_{j=1}^k B_j^1$*,* $t_2 \in \cap_{j=k+1}^n B_j^2$ *es de equilibrio.*

*Demostración.* En esas posiciones, $p$ gana los puntos $p_{i_j}^1$, $j = 1, \ldots, k$, se ponga donde se ponga $q$ dentro de su entorno, y $q$ gana los puntos $p_{i_j}^2$, $j = k+1, \ldots, n$, se ponga donde se ponga $p$. Entonces si se mueve $p$ no puede ganar más de los $k$ puntos que tiene estando $q$ posicionado en $t_2$, y tenemos la misma situación para $q$, luego la posición $(t_1, t_2)$ es de equilibrio. □

**Observaciones 3.3.** 1.- El resultado establecido en la proposición anterior garantiza, en los casos en que se da la condición suficiente, la existencia de infinitas posiciones de equilibrio. Dicho resultado contrasta con la unicidad de posiciones de equilibrio, cuando existen, en el planteamiento usual de los juegos sin restricciones. 2.- Para verificar la condición establecida en la proposición 3.2, se hace uso de la construcción geométrica conocida como arreglo de círculos ([4], 1997). Este arreglo puede realizarse mediante un algoritmo incremental con un tiempo esperado de ejecución de $O\left(n^2\right)$. ([6], 1992; [16], 1995; [5], 2005). Cuando los círculos que componen los arreglos tienen intersección no vacía, se cumplirá la condición. Cuando ésta no se cumple, asociando una etiqueta a cada celda de los arreglos con el número de círculos que la contienen, obtenemos información de la zona de máxima captación de puntos para cada jugador esté donde esté el otro. Esto puede realizarse mediante un algoritmo estándar de barrido ([3], 1979), con un tiempo esperado de ejecución de $O\left(n^2 \log n\right)$. Se puede reducir ligeramente este tiempo de ejecución utilizando el algoritmo incremental antes citado. 3.- Como en caso de equidistancia entre las posiciones de los dos jugadores se asigna el punto al primero (ver [1]), para asegurar que la posición sea de equilibrio, $t_2$ debe situarse en el interior de $\cap_{j=k+1}^n B_j^2$. Se comenta a continuación la no unicidad en las posiciones de equilibrio en el caso de existencia del mismo.

**Proposición 3.4.** *Si existe alguna posición de equilibrio* $(t_1, t_2)$*, entonces el equilibrio no es único*

*Demostración.* Supongamos que $p_{j_1}, \ldots, p_{j_k}$ son los puntos que gana $t_1$ a $t_2$, y $p_{j_{k+1}}, \ldots, p_{j_n}$ los que gana $t_2$ a $t_1$, entonces, al estar $t_2$ en el interior de $C\left(p_{j_i}, d\left(p_{j_i}, t_1\right)\right)$ para todo $i = k+1, \ldots, n$, podemos buscar una posición $t^{'}$ para el segundo jugador en $B^{'}$ lo suficientemente cerca de $t_2$ para que $t^{'}$ esté en el interior de $C\left(p_{j_i}, d\left(p_{j_i}, t_1\right)\right)$ para todo $i = k+1, \ldots, n$, y por tanto en la zona de máxima ganancia de puntos si el primer jugador está en $t_1$, y que $t_1$ esté en el interior de $C\left(p_{j_i}, d\left(p_{j_i}, t^{'}\right)\right)$ para todo $i = 1, \ldots, k$ (lo que puede ser aunque $t_1$ esté en la frontera de $C\left(p_{j_i}, d\left(p_{j_i}, t_2\right)\right)$ para algún $i = 1, \ldots, k$. Basta con tomar $t^{'}$ en la recta que une $p_{j_i}$ con $t_2$, dentro de $B^{'}$), siendo ésta la zona en $B$ de máxima ganancia estando el segundo jugador en $t^{'}$. Se cumple entonces que $\left(t_1, t^{'}\right)$ es también posición de equilibrio. □

### 3.2 Ejemplos

El modelo presentado aporta la ventaja de que, como vimos en la observación 1 de la proposición 2, puede haber situaciones en las que hay infinitas situaciones de equilibrio para los jugadores. Esto enriquece las posibilidades de variación de posición de los competidores. No obstante, en esta sección se verá que hay también ejemplos de no existencia de posiciones de equilibrio o de existencia de equilibrio único para uno de los jugadores, característica común con los juegos bidimensionales sin restricciones.

**Ejemplo 3.5.** Tomamos la situación con las siguientes posiciones iniciales de los jugadores $x_0, x_1$ (centros de los entornos) y los dos puntos $p_1, p_2$: Sean $x_0 = (0,0)$, $x_1 = (4,0)$, $p_1 = (0,3)$, $p_2 = (0,-3)$, $r = 1$, $r' = \frac{29}{10}$, (ver figura 2). Se puede ver que, siempre que $C(p_1, R)$ interseca con $C(p_2, S)$ para ciertos $R \geq d\left(p_1, B'\right)$, $S \geq d\left(p_2, B'\right)$ (para que puedan intersecar con $B'$), hay algún punto de la intersección en $B$. Entonces, para cualquier posición $t_2$ del segundo jugador en $B'$, se cumple que $C(p_1, d(p_1, t_2)) \cap C(p_2, d(p_2, t_2))$ contiene algún punto de $B$, ya que $R = d(p_1, t_2) \geq d\left(p_1, B'\right)$, $S = d(p_2, t_2) \geq d\left(p_2, B'\right)$, y $C(p_1, d(p_1, t_2)) \cap C(p_2, d(p_2, t_2)) \neq \emptyset$ ($t_2$ está en la intersección). Entonces, el segundo jugador conseguirá los dos puntos de la nube colocándose en algún punto de $B$ en esa intersección, luego cualquier posición $(t_1, t_2)$ con $\Pi^1(t_1, t_2) < 2$ no es de equilibrio. Pero si $\Pi^1(t_1, t_2) = 2$ la posición $(t_1, t_2)$ tampoco es de equilibrio. Por tanto no hay equilibrio en este ejemplo.

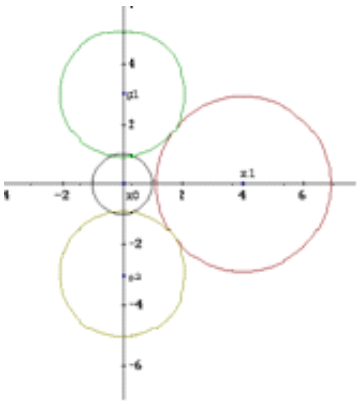



Figura 2: Ejemplo de situación de no existencia de equilibrio

**Ejemplo 3.6.** Se presenta un ejemplo en el que no se da la condición suficiente establecida en la proposición 3.2, y sin embargo hay infinitas posiciones de equilibrio, demostrando que la condición no es necesaria: Sean $x_0 = (0,0)$, $x_1 = (3,0)$, $p_1 = (3,9)$, $p_2 = (3,-9)$, $r = 1$, $r' = \frac{1}{2}$, se cumple que: $d(p_1, B) = d(p_2, B) = 3\sqrt{10} - 1 < d\left(p_1, B'\right) = d\left(p_2, B'\right) = \frac{17}{2}$, y que $B_1^1 \cap B_2^1 = \emptyset$, (ver figura 3) luego no se cumple la condición de la proposición 3.2. Si el segundo jugador se sitúa en $x_1$, y el primero en, por ejemplo, $B_1^1$, entonces la posición es de equilibrio por la proposición 3.1.

**Ejemplo 3.7.** En este ejemplo se trata un caso en el que hay unicidad en las posiciones de equilibrio para el primer jugador, y en el que no hay ninguna posición de equilibrio en la que los dos jugadores estén posicionados en las fronteras de sus entornos. Sean $x_0 = (0,0)$, $x_1 = \left(\sqrt{7}, 0\right)$, $p_1 = (0,3)$, $p_2 = (0,-3)$, $r = r' = 1$, se cumple que $d(p_1, B) = d(p_2, B) = 2 < d\left(p_1, B'\right) = d\left(p_2, B'\right) = 3$, y que $B_1^1 \cap B_2^1 = \{x_0\} \neq \emptyset$, (ver figura 4). En este ejemplo las únicas posiciones de equilibrio son $\left(x_0, t'\right)$ con $t' \in B'$, al cumplir la condición suficiente de la proposición 3.2, y cumplirse que en cualquier posición $(t_1, t_2)$ con $t_1 \neq x_0$ y con $\Pi^1(t_1, t_2) = 2$, se cumplirá que, por ejemplo, $d(p_1, t_1) > 3$, luego situándose el segundo jugador en $B(p_1, d(p_1, t_1)) \cap B'$ ganará a $p_1$ y mejorará su ganancia, por lo que $(t_1, t_2)$ no es de equilibrio.

## 4 Algoritmo para encontrar posiciones de equilibrio

En esta sección se desarrolla un algoritmo para encontrar posiciones de equilibrio de Nash en el juego planteado. Para ello se ve primero una condición necesaria y suficiente de existencia de equilibrio, que

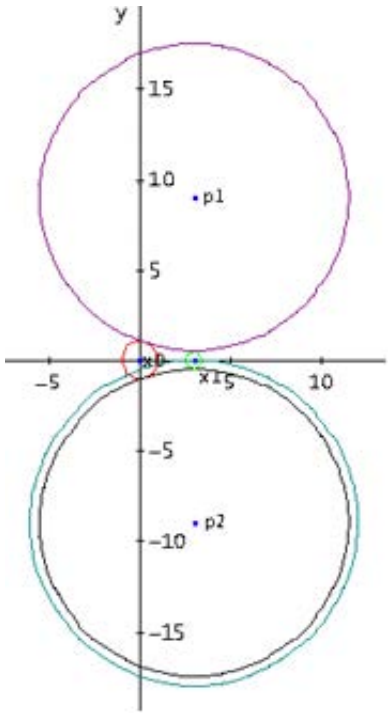



Figura 3: Ejemplo de infinitas posiciones de equilibrio no cumpliéndose la condición suficiente

es una traducción al juego planteado de las condiciones de equilibrio de von Neumann para juegos de suma constante ([11], 1944):

**Proposición 4.1.** *Se consideran los números:* $m_1 = \min_{t \in B}$ *(máxima intersección en* $B^{'}$ *de* $C\left(p_i, d\left(p_i, t\right)\right), i = 1, \ldots, n$ *(máxima ganancia mínima para el segundo jugador),* $m_2 = \min_{t^{'} \in B^{'}}$ *(máxima intersección en* $B$ *de* $C\left(p_i, d\left(p_i, t^{'}\right)\right), i = 1, \ldots, n$*), entonces existe equilibrio en el juego planteado si y sólo si* $m_1 + m_2 = n$*. Las posiciones de equilibrio serán los* $(t_1, t_2)$ *tales que* $t_1$ *es un punto de* $B$ *donde se alcanza* $m_1$ *y* $t_2$ *es un punto de* $B^{'}$ *donde se alcanza* $m_2$ *.*

Entonces, para localizar posiciones de equilibrio, si hay, necesitamos un algoritmo que halle las regiones de $B$ y $B^{'}$ donde se alcanzan esas máximas intersecciones mínimas, ó al menos zonas dentro de esas regiones. Para ello necesitamos unos resultados previos:

**Proposición 4.2.** *Sea una región* $R \subset B$*, entonces para todo* $x \in R$ *se cumple que* $M_R \geq l_x \geq m_R$*, donde* $M_R$ *es la máxima intersección en* $B^{'}$ *de* $C\left(p_i, \max\left\{d\left(p_i, y\right) / y \in R\right\}\right)$*,* $l_x$ *es la máxima intersección en* $B^{'}$ *de* $C\left(p_i, d\left(p_i, x\right)\right)$*,* $m_R$ *es la máxima intersección en* $B^{'}$ *de* $C\left(p_i, d\left(p_i, R\right)\right)$*,* $i = 1, \ldots, n$

*Demostración.* Sea $x \in R$, supongamos que $C\left(p_{i_1}, d\left(p_{i_1}, R\right)\right)$, ..., $C\left(p_{i_m}, d\left(p_{i_m}, R\right)\right)$ tienen intersección no vacía en $B^{'}$. Entonces, como $d\left(p_{i_j}, R\right) \leq d\left(p_{i_j}, x\right)$ para todo $j = 1, \ldots, m$, se cumple que $C\left(p_{i_j}, d\left(p_{i_j}, R\right)\right) \subset C\left(p_{i_j}, d\left(p_{i_j}, R\right)\right)$, por lo que: $\cap_{j=1}^{m} C\left(p_{i_j}, d\left(p_{i_j}, x\right)\right) \cap B^{'} \supset \cap_{j=1}^{m} C\left(p_{i_j}, d\left(p_{i_j}, R\right)\right) \cap B^{'} \neq \emptyset$ y entonces la máxima intersección en $B^{'}$ de $C\left(p_i, d\left(p_i, x\right)\right)$ será mayor ó igual que $m_R$, por lo que $l_x \geq m_R$. De igual modo, se puede ver que $M_R \geq l_x$. □

**Proposición 4.3.** *Si existe una descomposición de* $B$ *como unión de conjuntos* $R_i$*,* $i = 1, \ldots, k$ *de manera que, para algún* $i_0$*,* $M_{R_{i_0}} = m_{R_{i_0}}$ *y se cumple que* $m_{R_i} \geq m_{R_{i_0}}$ *para todo* $i = 1, \ldots, k$*, entonces* $m_1$ *se alcanza para cualquier* $x \in R_{i_0}$ *(*$M_{R_i}$*,* $m_{R_i}$ *siguen la definición de* $M_R$*,* $m_R$ *en la proposición* 4.2*, considerando la región* $R_i$*)*

*Demostración.* Tenemos que, para todo $x \in R_{i_0}$, $M_{R_{i_0}} \geq l_x \geq m_{R_{i_0}}$ (proposición 4.2), por lo que $l_x = m_{R_{i_0}}$ por la primera suposición. Además, para todo $x \in B - R_{i_0}$, se cumple que $x \in R_i$ para

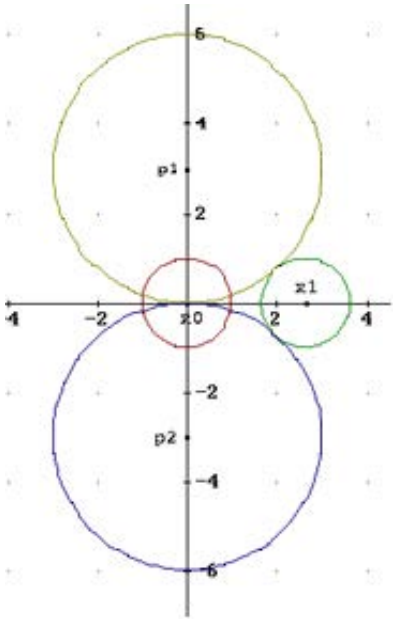



Figura 4: Ejemplo de posición única de equilibrio (no en la frontera del entorno) para uno de los dos jugadores

algún $i \neq i_0$, por lo que, por la proposición 4.2 y la segunda suposición, tenemos que $l_x \geq m_{R_i} \geq m_{R_{i_0}}$ y entonces $m_1 = m_{R_{i_0}}$, alcanzándose ese valor seguro en $R_{i_0}$ □

Vemos ya el algoritmo (su representación gráfica está en la figura 5):

## 4.1 Algoritmo para encontrar zonas donde se alcanzan $m_1$, $m_2$ y posiciones de equilibrio si existen

- Paso 1. Tomar un cuadrado que circunscriba a $B$: $Q = \lfloor x_0^1 - r, x_0^1 + r \rfloor \times \lfloor x_0^2 - r, x_0^2 + r \rfloor$, donde $x_0^1$, $x_0^2$ son las coordenadas de $x_0$ (centro de $B$). Hacer una partición de $Q$ en $k^2$ cuadrados de la forma $Q_{ij} = \left[x_0^1 - r + \frac{2r}{k} i, x_0^1 - r + \frac{2r}{k}(i+1)\right] \times \left[x_0^2 - r + \frac{2r}{k} j, x_0^2 - r + \frac{2r}{k}(j+1)\right]$ con $i = 0, \ldots, k-1$, $j = 0, \ldots, k-1$, para cierto natural $k$. Considerar de estos cuadrados sólo los que intersecan con $B$ (esto se puede hacer restringiendo los valores de $i$ en función de $j$, para conseguir que al menos uno de los vértices de cada cuadrado esté en $B$)

- Paso 2. En los cuadrados considerados en el paso anterior, hallar para todo $i, j$ la máxima intersección en $B^{'}$ de $C\left(p_s, d\left(p_s, Q_{i,j}\right)\right)$, $s = 1, \ldots, n$ $(m_{Q_{ij}})$ y la máxima intersección en $B^{'}$ de: $C\left(p_s, \max\left\{d\left(p_s, Q_{i,j}\right) / x \in Q_{ij}\right\}\right)$, $s = 1, \ldots, n$ $(M_{Q_{ij}})$.

- Paso 3. Hallar el $\min_{i,j} m_{Q_{ij}}$ y comprobar si para algún $i_0$, $j_0$ para el que se alcanza ese mínimo, $m_{Q_{i_0 j_0}} = M_{Q_{i_0 j_0}}$. En ese caso ir al paso 5 y almacenar los $Q_{i_0 j_0}, m_{Q_{i_0 j_0}}$. En caso contrario, ir al paso 4.

- Paso 4. Repetir los pasos 1, 2 y 3 cambiando $k$ por $2k$

- Paso 5. Repetir los pasos del 1 al 4 para un cuadrado $Q^{'}$ que circunscriba a $B^{'}$ y así encontrar los $Q^{'}_{i_0 j_0}$ donde se alcanza el $\min_{i,j} m^{'}_{Q^{'}_{ij}}$ y tales que $m^{'}_{Q^{'}_{i_0 j_0}} = M^{'}_{Q^{'}_{i_0 j_0}}$. Almacenar estos $Q^{'}_{i_0 j_0}$ y $m^{'}_{Q^{'}_{i_0 j_0}}$ ($m^{'}_{Q^{'}_{i_0 j_0}}$ y $M^{'}_{Q^{'}_{i_0 j_0}}$ siguen la definición de la proposición 4.2, pero para regiones $Q^{'}_{i_0 j_0}$ de $B^{'}$ y haciendo las máximas intersecciones en $B$)

- Paso 6. Hallar $m_{Q_{i_0 j_0}} + m^{'}_{Q^{'}_{i_0 j_0}}$. Si este número es igual a $n$, existe equilibrio en el juego planteado y cualquier $\left(t, t^{'}\right)$ con $t \in Q_{i_0 j_0} \cap B$, $t^{'} \in Q^{'}_{i_0 j_0} \cap B^{'}$ es posición de equilibrio (en los puntos de

$Q_{i_0j_0} \cap B$ se alcanza $m_1$, y en $Q^{'}_{i_0j_0} \cap B^{'}$ se alcanza $m_2$. Se cumple además que $m_1 = m_{Q_{i_0j_0}}$, $m_2 = m^{'}_{Q^{'}_{i_0j_0}}$ por la proposición 4.3). Si el número anterior no es $n$, no existe equilibrio en el juego planteado.

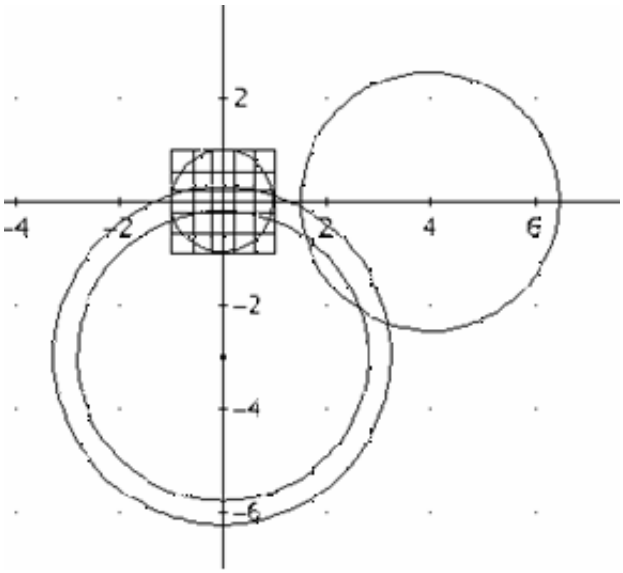



Figura 5: Idea gráfica del algoritmo para un votante en la posición $(0, -3)$

Se demuestra ahora que si los interiores de las zonas en que se alcanzan $m_1$ y $m_2$ son no vacíos, para $k$ suficientemente grande se cumple la condición de parada del paso 3, por lo que el algoritmo es correcto:

**Proposición 4.4.** *Si los interiores de las zonas $D$, $D^{'}$ en que se alcanzan $m_1$ y $m_2$ son no vacíos, el algoritmo expuesto da posiciones de equilibrio en el juego, si existen.*

*Demostración.* En este caso, es posible tomar sin pérdida de generalidad un punto $u$ en el interior de $D$ de manera que $C(p_i, d(p_i, u))$ no sean tangentes a $B^{'}$ para ningún $i = 1, \ldots, n$, y que la intersección de $Fr(C(p_i, d(p_i, u)))$, $Fr(C(p_j, d(p_j, u)))$ no contenga a ningún punto de $Fr\left(B^{'}\right)$ para ningún $i \neq j$, donde $Fr$ es la frontera. Esta elección de $u$ hace que, para un cuadrado $Q_{i_0j_0}$ suficientemente pequeño que contenga a $u$, no haya variación de $m_{Q_{i_0j_0}}$, $M_{Q_{i_0j_0}}$ respecto a $l_u$, por lo que se da la condición de parada del paso 3. □

**Observaciones 4.5.** 1.- La condición de que el interior de $D$ sea no vacío es también necesaria para que se dé la condición del paso 3, ya que si se da esta condición, $Q_{i_0j_0}$ está dentro de $D$. 2.- El algoritmo no da las regiones $D$, $D^{'}$, sino zonas dentro de estos conjuntos. Si se toma $k$ grande, los cuadrados $Q_{i_0j_0}$, $Q^{'}_{i_0j_0}$ pueden perfilar el contorno de estos conjuntos. 3.- La complejidad del algoritmo, en el mejor de los casos (si para el primer valor de $k$ para el que se hace la partición ya se cumple la condición de parada y no hay que refinar) es $O\left(k^2 n^2 \log n\right)$, ya que hay que hacer la máxima intersección de dos arreglos de $n$ círculos, lo que tiene una complejidad del orden de $n^2 \log n$, en los $k^2$ cuadrados de $B$ y en los de $B^{'}$ (aunque en general se toman menos de $k^2$ cuadrados, el número de cuadrados que se toma en función de $k$ es del orden de $k^2$. En cualquier caso, se puede rebajar ligeramente la complejidad como vimos en la observación 2 de la proposición 3.2). Como los $Q_{i_0j_0}$ que da el algoritmo están en $D$, para acertar con el primer valor de $k$ que se pruebe ha de ser necesariamente $diam(Q_{i_0j_0}) = \sqrt{2}\frac{2r}{k} \leq diam(D)$, por lo que $k \geq \frac{2\sqrt{2}r}{diam(D)}$

Queda entonces por analizar el caso en que el interior de $D$ es vacío. Este caso puede darse, como prueba el ejemplo 3.7. Para afrontar estas situaciones hay que tener en cuenta los arreglos de círculos degenerados. Un amplio estudio de dichos arreglos se realiza en [7], 2003. Como se vio en la demostración de la proposición 4.4, los círculos problemáticos en el caso en que el interior de $D$ es vacío

son los centrados en algún $p_i$ y tangentes a $B^{'}$ y los pares de círculos centrados en puntos $p_i$, $p_j$ y cuya intersección sólo contiene un punto de $B^{'}$. Si alguna familia de estos círculos interseca en un solo punto de $B$, este punto será también candidato a zona donde se alcanza $m_1$. Hay que modificar entonces el algoritmo para que contemple también el caso en que el interior de $D$ es vacío de la siguiente manera:

### 4.2 Modificación del algoritmo

Añadir al paso 1: Hallar para los $i$, $j$ tales que $1 \leq i$ ,$j \leq n$, la curva de todos los puntos $t \in B$ tales que $C\left(p_i, d\left(p_i, t\right)\right)$ , $C\left(p_j, d\left(p_j, t\right)\right)$ intersecan en un solo punto de $B^{'}$ (estas curvas son arcos de circunferencia). Hallar los puntos $x$ intersección de estas curvas con la frontera de $B$ ó de estas curvas entre sí. Almacenar estos puntos $x$. Hallar, para todos los puntos $x$ almacenados, $t_x$ (el conjunto de puntos almacenados es finito) Añadir al paso 3: Comprobar si existen unos $i_0$, $j_0$ tales que $x \in Q_{i_0 j_0}$, $t_x = m_{Q_{i_0 j_0}}$ y $m_{Q_{i_0 j_0}} = \min_{i,j} m_{Q_{ij}}$. En ese caso ir al paso 5 y almacenar los $x$, $t_x$ que cumplen la propiedad anterior. Añadir al principio del paso 6: Hallar $t_x + m^{'}_{Q^{'}_{i_0 j_0}}$. Si este número es igual a $n$, existe equilibrio en el juego planteado y cualquier $\left(x, t^{'}\right)$ con $t^{'} \in Q^{'}_{i_0 j_0} \cap B^{'}$ es posición de equilibrio (en $x$ se alcanza $m_1$). Esta modificación aumenta la complejidad del algoritmo, ya que hay que hallar todas las posibles intersecciones en un conjunto de $n^2 + 1$ curvas, pero tiene la ventaja de que cubre también los casos degenerados.

## 5 Aplicaciones a la economía política

Se puede dar una interpretación en términos políticos al juego planteado de la siguiente manera: Los dos jugadores son dos partidos políticos que compiten en unas elecciones. Las posiciones que eligen en el plano son las políticas que adoptan en dos determinados ítems importantes en el proceso electoral. Los centros de los entornos en los que eligen posición se pueden interpretar como las políticas centrales de los partidos en los dos temas y los radios como los grados de flexibilidad que los partidos pueden permitirse en sus políticas para captar más puntos. Estos puntos que se disputan los partidos son entonces los votantes en las elecciones. Se puede ver una versión sin restricciones de este juego, suponiendo que el conjunto de votantes es un continuo, en [14], 2001, con un análisis sobre la no existencia general de posiciones de equilibrio cuando el espacio de políticas es bidimensional. La consideración de entornos es entonces una aportación para evitar esta falta de equilibrio.

## 6 Conclusiones

En el análisis del equilibrio de la mayoría de los juegos de competición multidimensionales, se encuentra que no existen posiciones de equilibrio salvo en casos singulares, por lo que en general no existen posiciones para los jugadores que garanticen que el competidor no pueda incrementar su ganancia moviéndose. En este trabajo se ha propuesto un modelo de competición bidimensional y se han desarrollado estrategias geométricas que encuentran las posiciones de equilibrio, si existen. Como se ha establecido en la sección 3, existen casos en los que la existencia de posiciones de equilibrio, no sólo no son únicas sino que son regiones en el plano, es decir, existen infinitas posiciones, lo que aporta al juego de mayores posibilidades y contrasta con los resultados de posiciones únicas de equilibrio, si existe, donde los dos partidos deberían adoptar una misma postura, típicos de los juegos de competición multidimensionales sin restricciones de entorno.

## Referencias

[1] M. Abellanas, I. Lillo, M. López y J. Rodrigo. *Electoral strategies in a dynamical democratic system: geometric models.* European Journal of Operational Research, 175, pp. 870–878, 2006.

[2] R. Aurenhammer y R. Klein. *Voronoi diagrams. In: Sack, J.-R. and Urrutia*, J. (eds.), Handbook of Computational Geometry. Elsevier, Amsterdam, 2000.

[3] J.L. Bentley y T.A. Ottmann *Algorithms for reporting and counting geometric intersections.* IEEE Trans Comput., C-28, pp. 643–647, 1979.

[4] M. de Berg, M. van Kreveld, M. Overmars y O. Schwarzkopf. *Computational Geometry, Algorithms and Applications.* Springer. New York, 1997.

[5] S. Cabello, J.M. Díaz Báñez, S. Langerman, C. Seara y I. Ventura. *Reverse facility location problems.* Actas de los XI Encuentros de Geometría Computacional, pp. 263-270,2005.

[6] H. Edelsbrunner, L. Guibas, J. Pach, R. Pollack, R. Seidel y M. Sharir. *Arrangements of curves in the plane-topology, combinatorics and algorithms.* Theoretical computer science, 92, pp. 319-336, 1992.

[7] D. Halperin y E. Leiserowitz, *Controlled perturbation for arrangement of circles.* International Journal of Computational Geometry and Applications, 14, pp. 277-310, 2004.

[8] G.H. Kramer. *On a class of equilibrium conditions for majority rule.* Econometrica, 42, pp. 285–297, 1973.

[9] R.D. Mckelvey. *Intransitivities in multidimensional voting models and implications for agenda control.* Journal of Economic Theory, 12, pp. 472–482, 1976.

[10] J. Nash. *Non-cooperative games.* Annals of Mathematics, 54, pp. 286–295, 1951.

[11] J. von Neumann y O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press. Princeton, 1944.

[12] A. Okabe, B. Boots, K. Sugihara y S. Chiu. *Spatial Tessellations Concepts and Applications of Voronoi diagrams.* John Wiley and Sons. Chichester, 2000.

[13] C.R. Plott. *A notion of equilibrium and its possibility under majority rule.* American Economic Review, 57, pp. 787–806, 1967.

[14] J. Roemer. *Political Competition.* Harvard University Press. Harvard, 2001.

[15] D. Serra y C. Revelle. *Market capture by two competitors: the preemptive location problem.* Journal of Regional Science, 34, pp. 549–561, 1994.

[16] M. Sharir y P. Agarwal. *Davenport-Schinzel sequences and their geometric applications.* Cambridge University Press. Cambridge, 1995.

[17] M. Smid. *Closest-point problems in computational geometry.* In Sack, J.-R., and Urrutia, J. (eds.) Handbook of Computational Geometry. Elsevier. Amsterdam, 1997.

# Miércoles, 1 de julio

## Sesión 6, 11:30–13:30

# On some partitioning problems for two-colored point sets

Clara I. Grima* Carmen Hernando†¶ Clemens Huemer‡¶
Ferran Hurtado§¶

**Abstract**

Let $S$ be a two-colored set of $n$ points in general position in the plane. We show that $S$ admits at least $2\left\lfloor \frac{n}{17} \right\rfloor$ pairwise disjoint monochromatic triangles with vertices in $S$ and empty of points of $S$. We further show that $S$ can be partitioned into $3\left\lceil \frac{n}{11} \right\rceil$ subsets with pairwise disjoint convex hull such that within each subset all but at most one point have the same color. A lower bound on the number of subsets needed in any such partition is also given.

## 1 Introduction

An important class of problems in Computational Geometry deals with two-colored point sets [13] motivated, surely, by representing two different properties, facilities,... On the other hand, partitions of point sets have been the focus of an extensive research, see [8] for example. Regarding the union of both concepts, there can also be found results on partitioning a two-colored point set into monochromatic convex sets, that is, all points of a convex set belong to the same color class [10]. We study the following variants of this problem. In Section 2 we consider convex sets with a fixed number of vertices on its convex hull. In particular, we investigate how many monochromatic triangles can be found in a partition of $S$. Here, the monochromatic triangles do not contain points of $S$ in the interior.

In Section 3 we consider partitions into "almost" monochromatic parts. For this case, the number of vertices on the convex hull of a part can be arbitrary, but in each convex set all but at most $k$ of the points - the stains- have the same color, where $k$ is a fixed value. This kind of study is potentially useful in noise reduction in digital signal processing [15]. Salt and pepper noise is a form of noise typically seen on images, where the image contains dark and white dots. Generally this type of noise will only affect a small number of image pixels. We derive bounds on the number of convex pieces that are needed for a partition of $S$ into convex sets with at most one stain. A related problem on two-colored point sets (without considering convexity) is considered in [3], where the authors look for geometric monochromatic 2-factors, possibly adding some extra Steiner points.

## 2 Disjoint monochromatic triangles

Given a two-colored point set $S$ of $n$ points in general position in the plane, let $\kappa(S, m)$ be the maximum number of monochromatic convex $m$-gons that can be constructed with vertices in $S$, such that their convex hulls are pairwise disjoint and have interiors empty of points from $S$. Let

$$\kappa(n, m) = \min\{\kappa(S, m) \mid S \subset \mathbb{R}^2 \text{ is in general position}, |S| = n\}.$$

*Departamento de Matemática Aplicada I, Universidad de Sevilla grima@us.es, Partially supported by projects MTM2008-05866-C03-01 and P06-FQM-01649
†Departament de Matemàtica Aplicada I, Universitat Politècnica de Catalunya, carmen.hernando@upc.edu
‡Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, clemens.huemer@upc.edu
§Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, ferran.hurtado@upc.edu
¶Partially supported by projects MEC MTM2006-01267 and DURSI 2005SGR00692

For $m = 2$, $\kappa(S, 2)$ is the number of segments in the largest noncrossing matching in $S$ such that every edge connects points of the same color. Dumitrescu and Steiger proved that there are sets of $n$ points in general position such that any monochromatic matching consists of fewer than $(1 - \epsilon)n/2$ edges, for some constant $\epsilon$ and gave a lower bound for the constant [11], later improved by Dumitrescu and Kaye [9] who showed $3n/7 - O(1) \leq \kappa(n, 2) \leq 47n/95 + O(1)$.

For $m = 4$ it has been conjectured that $\kappa(S, 4) \geq 1$ provided that $|S|$ is large enough [7]; an example of 32 points with no monochromatic empty convex quadrilateral was given in [16]. Only recently it has been shown that each set of at least 5044 two-colored points contains an (not necessarily convex) empty monochromatic quadrilateral [2]. Arbitrarily large two-colored sets without monochromatic empty convex pentagons were described in [7], therefore $\kappa(n, m) = 0$ for all $m \geq 5$.

Here we focus on the number $\kappa(n, 3)$. As every set of 10 points contains an empty convex pentagon, and at least three of its vertices will have the same color, we immediately get

$$\left\lfloor \frac{n}{10} \right\rfloor \leq \kappa(n, 3).$$

A related result was proved in [7]: every two-colored set of $n$ points in general position admits $\lceil n/4 \rceil - 2$ empty monochromatic *compatible* triangles, which is tight, where compatible means that any two of them have disjoint interiors and, for example all of them might share a vertex. For $\kappa(n, 3)$ we require a much stronger disjointness condition.

Let us recall a result that will be used repeatedly later:

**Lemma 2.1.** [12] *Let $S$ be a set of $n$ points in general position in the plane, and let $G_x$ be the graph with vertex set $S$, two of them being adjacent when the line they define has exactly $\lceil \frac{n-2}{2} \rceil$ points on one side and $\lfloor \frac{n-2}{2} \rfloor$ on the other side. Then, for any $n$ odd, the graph $G_x$ is connected.*

We are now ready for proving the main result of this section:

**Theorem 2.2.** *Let $\kappa(n, 3)$ be the largest number such that any two-colored set of $n$ points in general position in the plane admits $\kappa(n, 3)$ triangles with vertices in $S$, pairwise disjoint and empty of points of $S$. Then we have*

$$2 \left\lfloor \frac{n}{17} \right\rfloor \leq \kappa(n, 3) \leq \begin{cases} \left\lfloor \frac{n-2}{6} \right\rfloor & \text{if } n \text{ is even} \\ \left\lfloor \frac{n-1}{6} \right\rfloor & \text{if } n \text{ is odd} \end{cases}$$

*Proof.* We first show that any set $S_9$ of 9 black and white points contains an empty monochromatic triangle. Let $W$ denote the set of white points, $B$ the set of black points, and assume $|W| > |B|$. $W$ admits a triangulation into $2|W| - |conv(W)| - 2$ white triangles with pairwise disjoint interiors; $conv(W)$ denotes the convex hull of $W$. If $|W| \geq 6$ or $|conv(W)| = 3$ then the triangulation of $W$ contains at least one white triangle also empty of black points. Assume that there is no triangulation of $W$ that contains a triangle empty of points of $B$. This implies that $|W| = 5$ and $|conv(W)| \geq 4$. If $|conv(W)| = 5$ then at least 3 black points lie in the interior of $conv$(W) and form at least one empty triangle. Finally, if $|conv(W)| = 4$ then the 4 black points lie in the interior of $conv$(W). Triangulating $B$ we obtain at least two black triangles, at most one of them contains a white point in its interior.

We now show that any set $S_{17}$ of 17 black and white points contains two pairwise disjoint empty monochromatic triangles. This immediately implies the lower bound. Thereto, we use that $S_{17}$ admits a segment joining one point of each color and leaving 8 points on one side and 7 on the other. Indeed, for each of the 17 points there exists another point of the set such that the line they define leaves 8 points on one side and 7 on the other. By Lemma 2.1, the graph defined by all these line segments is connected. Consequently not all of these line segments connect two points of the same color. Let $x$ and $y$ denote the two points of $S_{17}$ of such a separating line segment, where $x$ and $y$ have different color. There is an empty monochromatic triangle in the set formed by the set of 7 points together with $\{x, y\}$. Since $x$ and $y$ have different color, this triangle uses at most one of $x$ and $y$. Assume $y$ is not used. We thus can find another empty monochromatic triangle in the set formed by the 8 points together with $y$. These two triangles are pairwise disjoint.

To prove the upper bound, consider the set in Figure 1. The set $W$ of white points is in convex position and no two of them lie on a horizontal line. Near every white point, but the topmost and lowermost, a black point is placed such that it lies on the horizontal line through the white point and in the interior of $conv(\mathrm{W})$. Then this $n$-point set has no empty white triangle, because each white triangle can be stabbed by a horizontal line through its second white point, ordered by y-coordinates, thus containing a black point. Then, there are $\lfloor \frac{n-2}{6} \rfloor$ pairwise disjoint black triangles. For the case $n$ is odd, remove the topmost white point in Figure 1. We remark that this example also appears in problems related to guarding triangles [6]. □

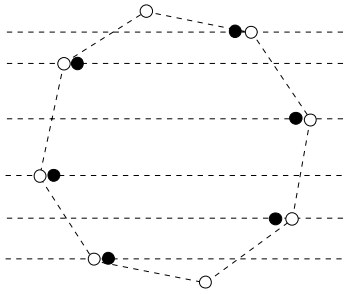


Figure 1: A point set $S$ with $\kappa(S,3) = \lfloor \frac{|S|-2}{6} \rfloor$.

## 3 At most one stain

Given a two-colored point set $S$ of $n$ points in general position in the plane, let $\sigma(S,k)$ be the minimum number of subsets in a partition of $S$, such that their convex hulls are pairwise disjoint and all the points in each subset have the same color with the possible exception of at most $k$ of them (the "stains"). Let

$$\sigma(n,k) = \max\{\sigma(S,k) \mid S \subset \mathbb{R}^2 \text{ is in general position}, |S| = n\}.$$

For $k = 0$, $\sigma(S,0)$ is the cardinality of the smallest partition of $S$ into monochromatic subsets whose convex hulls are pairwise disjoint. Dumitrescu and Pach [10] proved that

$$\sigma(n,0) = \left\lceil \frac{n+1}{2} \right\rceil.$$

For generic $k$ there is a trivial bound

$$\sigma(n,k) \leq \left\lceil \frac{n}{2k+1} \right\rceil,$$

because every subset of $2k+1$ points has at most $k$ stains.

Here we focus on the number $\sigma(n,1)$ and prove the following result:

**Theorem 3.1.** *Let $\sigma(n,1)$ be the smallest number such that any two-colored set of $n$ points in general position in the plane can be partitioned into $\sigma(n,1)$ subsets with pairwise disjoint convex hulls, each one having at most one stain. Then we have*

$$\left\lceil \frac{n+1}{4} \right\rceil \leq \sigma(n,1) \leq \begin{cases} 3\lfloor \frac{n}{11} \rfloor & \text{if } n' = 0 \\ 3\lfloor \frac{n}{11} \rfloor + \left\lceil \frac{n'+1}{4} \right\rceil & \text{if } n' \neq 0 \end{cases}$$

*where $n'$ is the residue of dividing $n$ by 11.*

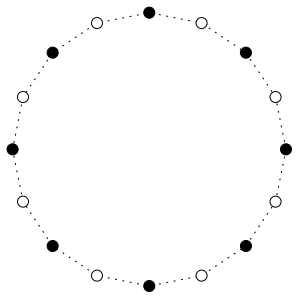


Figure 2: A point set $S$ with $\sigma(S,1) = (|S|+2)/4$.

*Proof.* For the lower bound, we first consider the case $n$ is even. Let $S$ be a two-colored set of $n$ points in convex position such that any two neighbored points on the convex hull have different colors, see Figure 2. We show that $\sigma(S,1) \geq \left\lceil \frac{n+1}{4} \right\rceil$. Let $\Pi$ be a partition of $S$ into the minimum number of subsets, such that their convex hulls are pairwise disjoint and such that each subset contains at most one stain. Define a directed tree $T$, whose nodes are the subsets of $\Pi$, as follows: Choose some subset $B_s$ of $\Pi$ as the source vertex of $T$. Draw directed arcs from $B_s$ to all subsets of $\Pi$ that can be "seen" from $B_s$, and take each of these subsets as the root of a subtree, defined iteratively. More formally, a subset $B_i$ is a descendant of $B_s$ if $conv(B_s \cup B_i)$ does not intersect any other subsets. Similarly, a subset $B_j$ is a descendant of a subset $B_k \neq B_s$ if $B_k$ is a descendant of some subset $B_h$, $B_j$ is not a descendant of $B_h$ and $conv(B_k \cup B_j)$ does not intersect any other subsets of $\Pi$. Figure 3 shows an example.

Assume that $\Pi$ contains at least one subset $B_s$ that contains at least four points, as the statement is obviously true otherwise. Choose $B_s$ as the root of $T$.

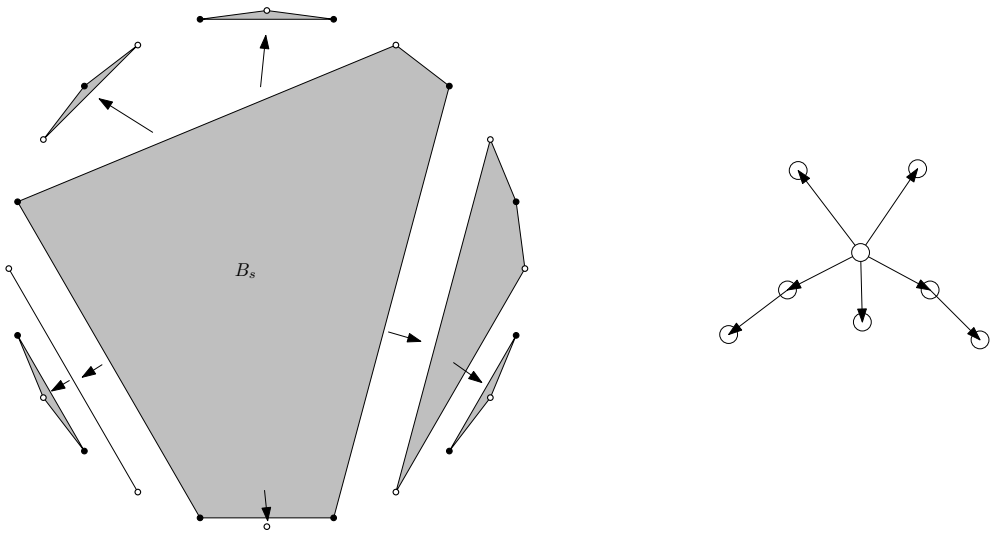



Figure 3: A partition of $S$ and the associated directed tree.

Let $n_k$ denote the number of interior nodes of $T$ that correspond to subsets of $\Pi$ containing exactly $k$ points, and let $h$ denote the number of leaves of $T$. Note that an interior node of $T$ corresponds to a subset of $\Pi$ of at least two points, and a leaf of $T$ corresponds to a subset of $\Pi$ of at most three points. In the following we give a lower bound on $h$ in terms of the number of interior nodes of $T$. Thereto, we assign each leaf of $T$ to a unique interior node of $T$, where a leaf $\ell$ can be assigned to an interior node $v$ if there is a directed path from $v$ to $\ell$ in $T$. We define this assignment recursively by first assigning leaves to the source vertex $B_s$ and then considering the assignment for the children of $B_s$. A subset of $\Pi$ contains at most two edges of the convex hull. We thus can assign at least $|B_s| - 2$ leaves to $B_s$. For $k \geq 5$, to each subset $B_p \neq B_s$ of $\Pi$ containing $k$ points assign at least $k-4$ leaves: $B_p$ has at least

$k-3$ outgoing edges in $T$, where one of them is counted for a leaf assigned to an antecessor of $B_p$. We thus obtain

$$h \geq n_5 + 2n_6 + 3n_7 + \cdots + (t-4)n_t + 2, \tag{1}$$

where $t$ is the number of points of the largest subset, and the last summand 2 comes from the two additional leaves assigned to $B_s$. On the other hand, the number of leaves of $T$ is at least

$$h \geq \frac{n - 2n_2 - 3n_3 - 4n_4 - \cdots - tn_t}{3}. \tag{2}$$

Taking the sum of (1) and three times (2) we obtain

$$4h \geq n - 2n_2 - 3n_3 - 4n_4 - 4n_5 - 4n_6 - \cdots - 4n_t + 2.$$

Thus, we also have

$$4h \geq n - 4n_2 - 4n_3 - 4n_4 - 4n_5 - 4n_6 - \cdots - 4n_t + 2,$$

which implies that

$$\sigma(S,1) \geq h + n_2 + n_3 + n_4 + \cdots + n_t \geq \left\lceil \frac{n+2}{4} \right\rceil = \left\lceil \frac{n+1}{4} \right\rceil,$$

because $n$ is even.

The case when $n$ is odd can be treated the same way, just remove one point from $S$, apply the bound for the obtained even point set and add the removed point to a piece.

For the upper bound, we first show that $\sigma(S_7,1) \leq 2$. Any set $S_7$ of 7 black and white points admits a segment joining one point of each color leaving 3 points on one side and 2 on the other. Indeed, for each of the 7 points there exists another point of the set such that the line they define leaves 3 points on one side and 2 on the other. By Lemma 2.1, the graph defined by all these line segments is connected. Consequently not all of these line segments connect two points of the same color. Then, the 3 points on one side of the segment together with one of the endpoints of the segment form a piece with at most one stain. The remaining three points form the other piece of the partition.

We are going to prove now that for any set $S_{11}$ of 11 black and white points we have $\sigma(S_{11},1) \leq 3$; from this and $\sigma(S_7,1) \leq 2$ the upper bound is immediately derived. Let $W$ and $B$ be the set of white and black points in $S_{11}$, respectively, and let us assume, without loss of generality, that $|W| > |B|$.

If $|B| \leq 3$, the claim is obvious; if $|B| = 4$ we can always find two black points $b_1$ and $b_2$ such that the other two black points lie in opposite halfplanes with respect to the line $b_1b_2$; then the two sets of white and black points lying in the same halfplane and $\{b_1, b_2\}$ give the claimed partition. Therefore, we are only left with the case $|B| = 5$.

For the case $|B| = 5$ observe that whenever we can separate with a line a set of four points with at most one stain, then the seven remaining points would require at most two pieces and we would be done.

Let us consider first the case in which at least one of the white points, $w$, is an extreme point, and let us denote by $x_1, \ldots, x_{10}$ the other points in counterclockwise radial order around $w$, in such a way that $x_1$ and $x_{10}$ are neighbors of $w$ on the convex hull $conv(S_{11})$. Assume that the set $\{x_1, x_2, x_3\}$ contains exactly two black points and one white point, as otherwise we would be done, because we would be able to isolate the subset $\{w, x_1, x_2, x_3\}$ with at most one stain. For the same reason we assume that $x_4$ is a white point, as otherwise we could separate the subset $\{x_1, x_2, x_3, x_4\}$. By symmetry reasons we also assume that $\{x_8, x_9, x_{10}\}$ contains exactly two black points and one white point and that $x_7$ is a white point. But now $\{x_5, x_6\}$ must consist of one white point and one black point, and we can take the partition $\{x_1, x_2, x_3\}, \{w, x_4, x_5, x_6, x_7\}, \{x_8, x_9, x_{10}\}$ (Figure 4).

Let us switch to the case in which none of the white points is an extreme point. If there are four or five black points that are extreme, we are done, because we can define two black monochromatic

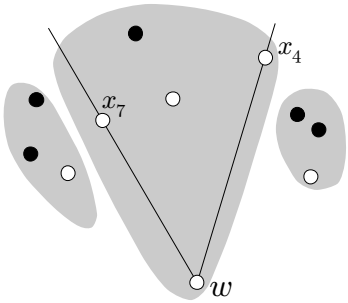
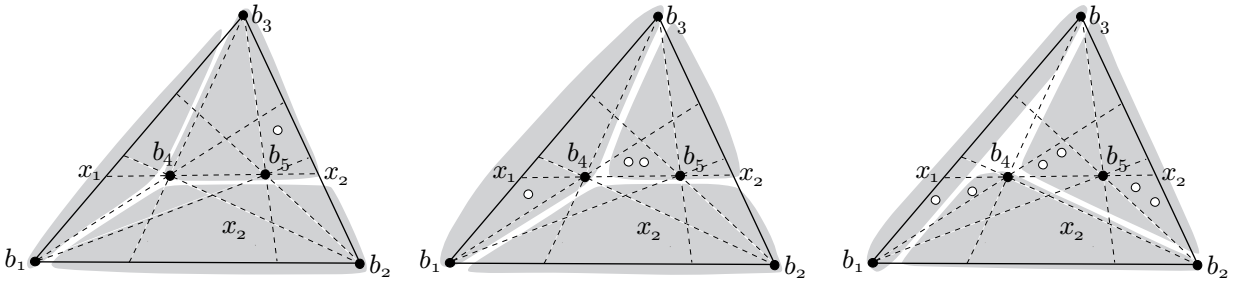



Figure 4: The case $|B| = 5$ with at least one white extreme point.

subsets by taking the endpoints of two disjoint convex hull edges, and the remaining black point would join as a stain the whole subset of white points.

Hence we are left with the situation in which the convex hull is a triangle with three black corners, which we denote by $\{b_1, b_2, b_3\}$. The line through the other two black points, $b_4$ and $b_5$, will cut two convex hull edges; for ease of description we assume that the line $b_1b_2$ is the horizontal base of the convex hull and that the line $b_4b_5$ is parallel to $b_1b_2$, which is no restriction of generality from the combinatorial viewpoint (refer to Figure 5).




(a) (b) (c)

Figure 5: The cases $|B| = 5$ with no white extreme point.

If the halfplane above the line $b_4b_5$ contains less than two white points, we can separate them together with $b_4$, $b_5$ and $b_3$, and get $\sigma(S_{11}, 1) = 3$; the same happens if the halfplane above the line $b_4b_5$ contains more than two white points, which we would separate together with $b_3$. Let us assume therefore that there are exactly two white points above $b_4b_5$. If the triangle $b_3b_4x_2$ contains at most one of them we can take it as part together with $b_4$, $b_5$ and $b_3$, a second part would consist of $b_1$ and any white point interior to $b_1b_4b_3$ and a third part would consist of $b_2$ and any white point interior to the quadrilateral $b_1b_2x_2b_4$ (Figure 5(a)), this gives $\sigma(S_{11}, 1) = 3$. The same argument applies to the triangle $b_5b_3x_1$, hence we assume hereafter that the two white points above $b_4b_5$ lie inside the triangle $b_3b_4b_5$.

Now, if the triangle $b_1b_4b_3$ contains at most one white point, we can take it together with $b_1$, $b_4$ and $b_3$ as a part, a second part would consist of $b_5$ and the two white points above $b_4b_5$, and a third part would consist of $b_2$ and all the white points inside the quadrilateral $b_1b_2x_2b_4$; again, we get $\sigma(S_{11}, 1) = 3$ (Figure 5(b)).

The same reasoning as in the preceding paragraph applies to the triangle $b_5b_2b_3$, hence we are finally left with the situation in which two white points are interior to $b_1b_4x_1$, two are interior to $b_5b_2x_2$ and two are interior to $b_3b_4b_5$ (Figure 5(c)). Now we can take $b_1$ and $b_3$ together with the white point closest to the line $b_1b_3$, the other white point inside $b_1b_4b_3$ joins $b_2$ and $b_4$ in a second subset, and the white points interior to $b_3b_4b_2$ together with $b_5$ is the third part. Again, we get $\sigma(S_{11}, 1) = 3$, which concludes the proof. □

# 4 Future work

We have presented some combinatorial studies on $\kappa(n, 3)$ (the number of monochromatic empty disjoint triangles in any two–colored $n$–set) and $\sigma(n, 1)$ (the smallest number of pieces with at most one stain in any two–colored $n$-set), so an algorithmic approach to both problems would be interesting. On the other hand, since in the first case we have that that any two–colored $n$–set admits a triangulation containing at least $2\left\lfloor \frac{n}{17} \right\rfloor$ monochromatic triangles, one can be interested, thinking in interpolation problems, in to decide, given a two–colored $n$–set of points, whether there exists a triangulation of it without monochromatic triangles. Our first steps in this direction lead us to suspect that this will be an $NP$–complete problem. A related question to determine $\kappa(n, 3)$ is to count the number of *all* (not necessarily disjoint) monochromatic empty triangles in a two-colored point set. This problem has been addressed in [1, 14]. While for uncolored point sets there always exists a quadratic number of empty triangles [4], determining the right asymptotic value for the two-colored case represents an intriguing open problem.

# Acknowledgments

We thank Mercè Claverol, Alberto Márquez, Carlos Seara and David Wood for useful discussions on the topic considered in this paper.

# References

[1] O. Aichholzer, R. Fabila-Monroy, D. Flores-Peñaloza, T. Hackl, C. Huemer, J. Urrutia, *Empty monochromatic triangles*, 20th Canadian Conference on Computational Geometry, 75–78, 2008.

[2] O. Aichholzer, T. Hackl, C. Huemer, F. Hurtado, B. Vogtenhuber, *Large bi-colored point sets admit empty monochromatic 4-gons*, 25th European Workshop on Computational Geometry, 133–136, 2008.

[3] M.N. Atienza, C. Cortés, D. Garijo, M.A. Garrido, C.I. Grima, A. Márquez, M.A. Moreno, J.R. Portillo, P. Reyes, R. Robles, M.E. Suárez, J. Valenzuela, M.T. Villar, *k-Factores en nubes bicromáticas*, XII Encuentros de Geometría Computacional, Valladolid, 53–57, 2007.

[4] I. Bárány, Z. Füredi, *Empty simplices in Euclidean space*, Canadian Mathematical Bulletin 30, 436–445, 1987.

[5] P. Brass, W. Moser, J. Pach, *Research problems in discrete geometry*, Springer-Verlag, New York, 2005.

[6] J. Czyzowicz, E. Kranakis, J. Urrutia, *Guarding the Convex Subsets of a Point Set*, 12th Canadian Conference on Computational Geometry, 47–50, 2000.

[7] O. Devillers, F. Hurtado, G. Károlyi, C. Seara, *Chromatic variants of the Erdős-Szekeres Theorem*, Computational Geometry: Theory and Applications 26, 193–208, 2003.

[8] R. Ding, K. Hosono, M. Urabe and C. Xu *Partitioning a Planar Point Set into Empty Convex Polygons*, Lecture Notes in Computer Science 2866, 129–134, 2003.

[9] A. Dumitrescu, R. Kaye, *Matching colored points in the plane: some new results*, Computational Geometry: Theory and Applications 19, 69–85, 2001.

[10] A. Dumitrescu, J. Pach, *Partitioning colored point sets into monochromatic parts*, International Journal of Computational Geometry and Applications 12, 401–412, 2002.

[11] A. Dumitrescu, W. Steiger, *On a matching problem in the plane*, Discrete Mathematics 211, 183–195, 2000.

[12] P. Erdős, L. Lovász, A. Simmons, E. G. Straus, *Dissection graphs of planar point sets*, in A Survey of Combinatorial Theory, (J.N. Srivastava et al., eds.), North-Holland, 139–149, 1973.

[13] A. Kaneko, M. Kano, *Discrete geometry on red and blue points in the plane – A survey*, in Discrete and Computational Geometry, The Goodman–Pollack Festschrift, Algorithms and Combinatorics 25, (B. Aronov et al., eds.), Springer-Verlag, 551–570, 2003.

[14] J. Pach, G. Tóth, *Monochromatic empty triangles in two-colored point sets*, in Geometry, Games, Graphs and Education, The Joe Malkevitch Festschrift, (S. Garfunkel, R. Nath, eds.), COMAP, 195–198, 2008.

[15] A. Rosenfeld, *Picture Processing by Computer*, New York: Academic Press, 1969.

[16] R. Van Gulik, *32 Two-Colored Points with No Empty Monochromatic Convex Fourgons*, Geombinatorics XV, 32–33, 2005.

# The number of generalized balanced lines*

David Orden† Pedro Ramos‡ Gelasio Salazar §

**Abstract**

Let $S$ be a set of $r$ red points and $b = r + 2\delta$ blue points in general position in the plane. A line $\ell$ determined by them is said to be balanced if in each open half-plane bounded by $\ell$ the difference between the number of red points and blue points is $\delta$. We show that every set $S$ as above has at least $r$ balanced lines. The main techniques in the proof are rotations and a generalization, sliding rotations, introduced here.

## 1 Introduction

Let $B$ and $R$ be, respectively, sets of blue and red points in the plane, and let $S = B \cup R$ be in general position. Let $b = |B|$ and $r = |R| = b + 2\delta$. Furthermore, we are given weights $\omega(p) = +1$ for $p \in B$ and $\omega(q) = -1$ for $q \in R$. Given a halfplane $H$, its weight is then defined as $\omega(H) = \sum_{s \in S \cap H} \omega(s)$. Here and throughout this paper, halfplanes are open unless noticed.

**Definition 1.1.** A line $\ell$ determined by two points of $S$ is said to be *balanced* if the two halfplanes it defines have weight $\delta$. Observe that this implies the two points having different colors.

For $\delta = 0$, we obtain the original result, as conjectured by George Baloglou, and proved by Pach and Pinchasi via circular sequences:

**Theorem 1.2** ([2])**.** *Let $|R| = |B| = n$. Every set $S$ as above determines at least $n$ balanced lines. This bound is tight.*

Tightness is shown, e.g., by placing $S$ on a convex $2n$-gon in such a way that $R$ is separated from $B$ by a straight line.

The general result was proved by Sharir and Welzl in an indirect manner, via an equivalence with a very special case of the Generalized Lower Bound Theorem. This motivates them to ask for a more direct and simpler proof.

**Theorem 1.3** ([3])**.** *Let $B$ and $R$ be, respectively, sets of blue and red points in the plane, and let $S = B \cup R$ be in general position. Let $b = |B|$ and $r = |R| = b + 2\delta$. The number of lines that pass through a point in $B$ and a point in $R$, and such that the two induced halfplanes have weight $\delta$ is at least $r$. This number is attained if $R$ and $B$ can be separated by a line.*

In this paper we give a simple proof of Theorem 1.3 using elementary geometric techniques. Therefore, via the results in [3], we provide a geometric proof of the following very special case of the Generalized Lower Bound Theorem:

Let $\mathcal{P}$ be a convex polytope which is the intersection of $d + 4$ halfspaces in general position in $\mathbb{R}^d$. Let its edges be oriented according to a generic linear function (edges are directed from smaller to larger value; "generic" means that the function evaluates to distinct values at the vertices of $\mathcal{P}$).

*This work started at the 6th Iberian Workshop on Computational Geometry, in Aveiro, and was concluded while Gelasio Salazar was visiting Departament of Mathematics of Alcalá University under the program Giner de los Ríos.

†Departamento de Matemáticas, Universidad de Alcalá, david.orden@uah.es, partially supported by grant MTM2008-04699-C03-02.

‡Departamento de Matemáticas, Universidad de Alcalá, pedro.ramos@uah.es, partially supported by grant MTM2008-04699-C03-02.

§Instituto de Física, Universidad Autónoma de San Luis Potosí, Mexico, gsalazar@dec1.ifisica.uaslp.mx

**Theorem 1.4** ([3]). *The number of vertices with $\lceil \frac{d}{2} \rceil - 1$ outgoing edges is at most the number of vertices with $\lceil \frac{d}{2} \rceil$ outgoing edges.*

All proofs in this paper can be easily translated to the more general setting of circular sequences.

## 2 Geometric tools

We assume that coordinate axis are chosen in such a way that all points have different abscissa. The tools we use are inspired in the rotational movement introduced by Erdős et al. [1].

**Definition 2.1.** Let $P \subseteq S$. A *$P^k$-rotation* is a family of directed lines $P_t^k$, where $t \in [0, 2\pi]$ is the angle measured from the vertical axis, defined as follows: $P_0^k$ contains a single point of $P$, and as $t$ increases, it rotates counterclockwise in such a way that

(i) $|P \cap P_t^k| = 1$ except for a finite number of events, when $|P \cap P_t^k| = 2$, and

(ii) whenever $|P \cap P_t^k| = 1$, there are exactly $k$ points of $P$ to the right of $P_t^k$.

The common point $P \cap P_t^k = \{p\}$ is called the *pivot*, and it changes precisely when $|P \cap P_t^k| = 2$. Observe that $P_0^k = P_{2\pi}^k$.

**Definition 2.2.** Let $\ell^+$ and $\ell^-$ denote, respectively, the halfplanes to the right and to the left of $\ell$. Let $\omega(\ell)$ be the weight of $\ell^+$. Given a $P^k$-rotation, we say that $P^k \geq \delta$ if $\omega(P_t^k) \geq \delta$ for every $t \in [0, 2\pi]$, and similarly for the rest of inequalities. A rotation $B^k$ is *$\delta$-preserving* if either $B^k \geq \delta$ or $B^k < \delta$. Symmetrically, $R^k$ is *$\delta$-preserving* if either $R^k \leq \delta$ or $R^k > \delta$.

**Lemma 2.3.** *In a $R^k$-rotation, transitions $\delta \rightsquigarrow \delta + 1$ and $\delta + 1 \rightsquigarrow \delta$ in $\omega(R_t^k)$ are always through a balanced line. In a $B^k$-rotation, transitions $\delta \rightsquigarrow \delta - 1$ and $\delta - 1 \rightsquigarrow \delta$ in $\omega(B_t^k)$ are always through a balanced line.*

*Proof.* When a red point is found during a $R^k$-rotation, the weight of the halfplane is preserved because the pivot point changes. Therefore, the change $\delta \rightsquigarrow \delta + 1$ happens when a blue point is found in the head of $R_t^k$ (Figure 1, left), while $\delta + 1 \rightsquigarrow \delta$ happens when a blue point is found in the tail of $R_t^k$ (Figure 1, right). In both cases, the points define a balanced line. For a $B^k$-rotation, the proof is identical. □


Figure 1: Transitions in a $R^k$-rotation are always through a balanced line.

Claim 8.1 in [2] has now a more direct proof:

**Lemma 2.4.** *If $r$ is odd, there exists a balanced line which is a halving line of $S$.*

*Proof.* Let $k = \lfloor \frac{r}{2} \rfloor$ and consider a $R^k$-rotation. If $R_0^k \leq \delta$, then $R_\pi^k > \delta$, and conversely. Therefore, there exist transitions $\delta \rightsquigarrow \delta + 1$ and $\delta + 1 \rightsquigarrow \delta$ in $\omega(R_t^k)$ which, from Lemma 2.3 are always through a balanced line. Observe that both transitions are through the same balanced line, with angles $t_0$ and $t_0 + \pi$. □

Let us observe that Theorem 1.4 in [2], which states that Theorem 1.2 is true when $R$ and $B$ are separated by a line $\ell$, has now a trivial proof: if we start $R^k$-rotations with a line parallel to $\ell$, it is obvious that no $R^k$-rotation is $\delta$-preserving. Therefore, there exist transitions $\delta \rightsquigarrow \delta+1$ and $\delta+1 \rightsquigarrow \delta$ in $\omega(R_t^k)$ which, from Lemma 2.3, correspond always to a balanced line. If $r$ is even, there are 2 balanced lines for $k = 0, \ldots, \frac{r}{2}-1$, for a total of $r$ balanced lines, while if $r$ is odd there are 2 balanced lines for $k = 0, \ldots, \lfloor \frac{r}{2} \rfloor - 1$ and 1 balanced line for $k = \lfloor \frac{r}{2} \rfloor$.

**Remark 2.5.** Lemmas 2.3 and 2.4 conclude the proof of Theorem 1.2 if no $R^k$-rotation is $\delta$-preserving or if no $B^k$-rotation (with $k \geq \delta$) is $\delta$-preserving. Hence, in the following we assume that there exists either at least one $R^k$-rotation or one $B^k$-rotation (with $k \geq \delta$) which is $\delta$-preserving.

**Lemma 2.6.** *Let $0 \leq j \leq \lfloor \frac{r}{2} \rfloor$. If $R^j > \delta$ then $B^{j+\delta} \geq \delta$ while if $B^{j+\delta} < \delta$ then $R^j \leq \delta$.*

*Proof.* Consider the line $R_{t_0}^j$. The halfplane $(R_{t_0}^j)^+$ contains $j$ red points and $b > j+\delta$ blue points. Therefore, the line $B_{t_0}^{j+\delta}$ is to the right of $(R_{t_0}^j)^+$ and contains at most $j$ red points. Then, $\omega(B_{t_0}^{j+\delta}) \geq \delta$. The proof of the second statement is analogous. □

The next definition generalizes the concept of $P^k$-rotation in two different ways: parallel movements are permitted and the number of points to the right of the line can change.

**Definition 2.7.** A *$P$-sliding rotation* consists in moving a directed line $\ell$ continuously, starting with an $\ell_0$ which contains a single point $p_0 \in P$, and composing rotation around a point of $P$ (the pivot) and parallel displacement (in either direction) until the next point of $P$ is found. Furthermore, after a $2\pi$ rotation is completed, the line $\ell_0$ must be reached again.

This movement is clearly a continuous curve in the space of lines in the plane. For instance, if a line is parameterized as a point in $S^1 \times \mathbb{R}$, a $P$-sliding rotation describes a (non-strictly) angular-wise monotone curve, with vertical segments corresponding to parallel displacements.

Let $\Sigma$ be a $P$-sliding rotation. Let us denote by $\Sigma_t$ the line with angle $t$ with respect to the vertical axis defined as follows: if there is no parallel displacement at angle $t$, then $\Sigma_t$ denotes the corresponding line. Otherwise, it denotes the leftmost line corresponding to angle $t$.

**Definition 2.8.** A $P$-sliding rotation $\Sigma$ is *positively oriented* if $\Sigma_{t+\pi}$ is to the left of $\Sigma_t$ for all $t \in [0, \pi)$.

That $\Sigma \geq \delta$, as well as the rest of inequalities, is defined exactly as in Definition 2.2. Similarly, a $B$-sliding rotation $\Sigma$ is $\delta$-preserving if $\Sigma \geq \delta$, while a $R$-sliding rotation is $\delta$-preserving if $\Sigma \leq \delta$. The following definition is the crux of the rest of the paper.

**Definition 2.9.** Let $\mathcal{S}$ be the set of all positively oriented, $\delta$-preserving $B$-sliding rotations and $R$-sliding rotations. The *waist* of a $P$-sliding rotation $\Sigma \in \mathcal{S}$ is

$$\min_{t \in [0,\pi]} |P \cap \Sigma_t^- \cap \Sigma_{t+\pi}^-|.$$

We denote by $\Gamma$ the sliding rotation of $\mathcal{S}$ with the smallest waist.

Note that the set $\mathcal{S}$ is non-empty because we have assumed that there exist $\delta$-preserving $B^k$- or $R^k$-rotations, which are a particular type of sliding rotations. Furthermore, the waist takes only a finite number of values, so it has a minimum. If the minimum is not unique, we can pick any of the sliding rotations achieving it.

# 3 Main result

Assume that $\Gamma$ is a $\delta$-preserving $R$-sliding rotation (i.e. $\Gamma \leq \delta$). In this case, we will manage to prove that there exist at least $r$ balanced lines. For the case of $\Gamma$ being a $\delta$-preserving $B$-sliding rotation, the same arguments would show that there exist at least $b$ balanced lines.

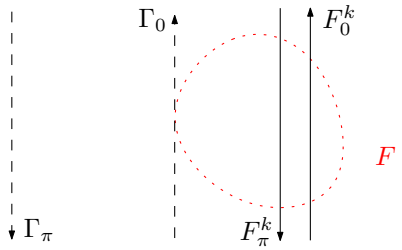



Figure 2: Illustration of the proof of Lemma 3.1.

**Lemma 3.1.** *Let $\Gamma_0$ and $\Gamma_\pi$ be the lines achieving the waist of $\Gamma$, let $\overline{\Gamma}_0^+$ be the closed halfplane to the right of $\Gamma_0$ and let $F = R \cap \overline{\Gamma}_0^+$. For every $k \in \{0, \ldots, |F|-1\}$, during a $F^k$-rotation a balanced line is found. Similarly, let $H = R \cap \overline{\Gamma}_\pi^+$. For every $k \in \{0, \ldots, |H|-1\}$, during an $H^k$-rotation a balanced line is found.*

*Proof.* Figure 2 illustrates the situation. On the one hand, $F_0^k$ is to the right of $\Gamma_0$ and, since $\Gamma$ is positively oriented, $F_\pi^k$ is to the left of $\Gamma_\pi$. This implies that there is a $t_1 \in [0, \pi]$ such that $F_{t_1}^k = \Gamma_{t_1}$ and therefore $\omega(F_{t_1}^k) \leq \delta$. On the other hand, $F_0^k$ is to the left of $\Gamma_\pi$ and $F_\pi^k$ is to the right of $\Gamma_0$, therefore, there exists a $t_2 \in [0, \pi]$ such that $F_{t_2}^k$ and $\Gamma_{t_2+\pi}$ are the same line with opposite directions. Since $\omega(\Gamma_{t_2+\pi}) \leq \delta$, then $\omega(F_{t_2}^k) \geq \delta$. If $\omega(\Gamma_{t_2+\pi}) = \delta$ and the line contains a blue point, then it is a balanced line found in a transition $\delta \rightsquigarrow \delta + 1$. In other case, $\omega(F_{t_2}^k) > \delta$ and hence a transition $\delta \rightsquigarrow \delta + 1$ has occurred for a $t \in (t_1, t_2)$.

Now, observe that $R \setminus F \subset {\Gamma_0}^-$. Hence, in the $F^k$-rotation for $t \in [0, \pi]$, all the points in $R \setminus F$ are found by the head of the line. This implies that a change $\delta \rightsquigarrow \delta + 1$ in the weight of the right halfplane can only occur when a blue point is found in the head of the ray (as in Figure 1, left), hence defining a balanced line. The proof for $H$ is identical. □

Before going on, let us point up that the $|F| + |H|$ balanced lines given by Lemma 3.1 are different, because they have exactly $k$ points of $F$, respectively $H$, to the right. Let now $C_t^\Gamma$ be the *central region* defined by the sliding rotation $\Gamma$ at instant $t$, defined as $C_t^\Gamma = \Gamma_t^- \cap \Gamma_{t+\pi}^-$. Observe that, for the corresponding $t$, the transitions $\delta \rightsquigarrow \delta + 1$ in the proof of Lemma 3.1 correspond to balanced lines inside or in the boundary of the central region.

**Lemma 3.2.** *Let $G = R \setminus (F \cup H)$. For $k \in \{0, \ldots, \lceil |G|/2 \rceil - 1\}$, every $G^k$-rotation has at least two transitions between $\delta$ and $\delta + 1$ which correspond to lines inside or in the boundary of the central region., i.e., for the corresponding $t$, $G_t^k \in C_t^\Gamma$.*

*Proof.* Let us consider first the case when $r$ is odd and $k = \lfloor |G|/2 \rfloor$. $G_0^k$ and $G_\pi^k$ are the same line with opposite directions. Therefore, if $\omega(G_0^k) \leq \delta$ then $\omega(G_\pi^k) > \delta$ and there must be at least two transitions as stated. These transitions correspond to lines in the central region because $\Gamma$ is positively oriented.

For the rest of cases, observe that, by construction, $G_0^k \in C_0^\Gamma$. According to the value of $\omega(G_0^k)$, we distinguish two cases:

- $\omega(G_0^k) \leq \delta$. If there exist some values for which $G_t^k = \Gamma_t$, let $t_1$ and $t_2$ be, respectively, the minimum and maximum of them. If there is no such value, take $t_1 = t_2 = 2\pi$. If $G^k$ takes the value $\delta + 1$ in the interval $(0, t_1)$ it must have transitions $\delta \rightsquigarrow \delta + 1$ and $\delta + 1 \rightsquigarrow \delta$, and the same is true for $(t_2, 2\pi)$. Finally, observe that $G^k$ must take the value $\delta + 1$ at least once, because in other case the sliding rotation obtained by concatenating $G^k$ in $(0, t_1)$, $\Gamma$ in $(t_1, t_2)$ and $G^k$ in $(t_2, 2\pi)$ would be a $\delta$-preserving sliding rotation of waist smaller than the waist of $\Gamma$.
- $\omega(G_0^k) > \delta$. If there exist some values for which $G_t^k = \Gamma_t$, let $t_1$ and $t_2$ be, respectively, the minimum and maximum of them. $G_t^k$ takes the value $\delta$ in the intervals $(0, t_1)$ and $(t_2, 2\pi)$ and therefore the lemma follows. In other case, if $G_t^k$ takes the value $\delta$ in the central region, it must

have also transition $\delta \rightsquigarrow \delta + 1$. Finally, if $\omega(G_t^k) > \delta$ for all $t \in [0, 2\pi]$ we could construct a sliding rotation $\Sigma$ contradicting the choice of $\Gamma$: for each $t$, consider as $\Sigma_t$ the parallel to $G_t^k$ which passes through the first blue point to the right of $G_t^k$. It is easy to see that $\Sigma_t \geq \delta$, because between $\Gamma_t$ and $G_t^k$ there are always at least two blue points. □

The following lemma, which already appeared as Claim 6.4 in [2], will be enough to conclude the proof of Theorem 1.2.

**Lemma 3.3.** *Transitions $\delta \rightsquigarrow \delta+1$ and $\delta+1 \rightsquigarrow \delta$ in a $G^k$-rotation are always either a balanced line or a $\delta+1 \rightsquigarrow \delta$ transition in an $F^j$-rotation, $j \in \{0, \ldots, |F|-1\}$ or an $H^j$-rotation, $j \in \{0, \ldots, |H|-1\}$.*

*Proof.* On the one hand, a balanced line is achieved if there is such a transition because a blue point is found. See Figure 1. On the other hand, if the point inducing the transition is $r \in R$, then necessarily $r \in R \setminus G$ (since the $G^k$-rotation changes pivot whenever a point of $G$ is found). Figure 3 shows that a $\delta + 1 \rightsquigarrow \delta$ transition appears for an $F^j$-rotation with pivot $r$, both if $r \in F$ is found in the tail (left picture) or if $r \in F$ is found in the head (right picture). Note that in the right picture


$\delta \rightsquigarrow \delta + 1$ in $G^k$ $\quad$ $\delta + 1 \rightsquigarrow \delta$ in $G^k$

$\delta + 1 \rightsquigarrow \delta$ in $F^j$ $\quad$ $\delta + 1 \rightsquigarrow \delta$ in $F^j$

Figure 3: Transitions when a point $f \in F \subset R$ found in a $G^k$-rotation induces a $\delta + 1 \rightsquigarrow \delta$ transition in an $F^j$-rotation.

the weight to the right of the horizontal line is $\delta + 1$ both above and below. The case in which the point found is $r \in H$ works similarly. □

The following simple observations show that the number of balanced lines is at least $r$ which, together with the tightness of the bound, finishes the proof of Theorem 1.3:

i) Lemma 3.1 gives $|F| + |H|$ different balanced lines.

ii) Lemmas 3.2 and 3.3 give $|G|$ lines which are, either a balanced line, or a $\delta + 1 \rightsquigarrow \delta$ transition at the central region for an $F^j$- or $H^j$-rotation.

iii) Each transition in ii) forces a new $\delta \rightsquigarrow \delta + 1$ transition at the central region for an $F^j$- or $H^j$-rotation which correspond, as in the proof of Lemma 3.1, to a new balanced line.

## 4 Acknowledgements

The authors thank Jesús García for helpful discussions.

## References

[1] P. Erdős, L. Lovász, A. Simmons, E.G. Strauss. Dissection graphs on planar point sets. In *A Survey of Combinatorial Theory*, North Holland, Amsterdam, (1973), 139–149.

[2] J. Pach and R. Pinchasi. On the number of balanced lines, *Discrete and Computational Geometry*, 25 (2001), 611–628.

[3] M. Sharir and E. Welzl. Balanced Lines, Halving Triangles, and the Generalized Lower Bound Theorem, In *Discrete and Computational Geometry — The Goodman-Pollack Festschrift*, B. Aronov, S. Basu, J. Pach and M. Sharir (Eds.), Springer-Verlag, Heidelberg, 2003, pp. 789–798.

# The Maximum Box Problem for Moving Points on the Plane

S. Bereg * J.M. Díaz-Báñez † P. Pérez-Lantero ‡ I. Ventura §

**Abstract**

Given a set $R$ of $r$ red points and a set $B$ of $b$ blue points on the plane, the static version of the Maximum Box Problem is to find an isothetic box $H$ such that $H \cap R = \emptyset$ and the cardinality of $H \cap B$ is maximized. In this paper, we consider a kinetic version of the problem where the points in $R \cup B$ move according to algebraic functions. We design a compact and local quadratic-space kinetic data structure (KDS) for maintaining the optimal solution in $O(r \log r + r \log b)$ time per each event. We also study the general static problem where the maximum box can be arbitrarily oriented. This is an open problem in [1]. We show that our approach can be used to solve this problem in $O((r+b)^2(r \log r + r \log b))$ time. Finally we propose an efficient data structure to maintain an approximated solution of the kinetic Maximum Box Problem.

## 1 Introduction

In Pattern Recognition and Classification problems, a natural method for selecting prototypes that represent a class is to perform cluster analysis on the training data [7]. Typically, two sets of points $X^+$ and $X^-$ are given, and one would like to find patterns which intersect exactly one of these sets. The clustering can be obtained by using simple geometric shapes such as circles or boxes. Recent papers deal with the *Maximum Box Problem*, where the clustering is due by considering maximum boxes, i.e., boxes containing the maximum number of points in the given data set. See [9, 10]. The basic problem is the following: given a finite set of blue points $B = X^+$ and a finite set of red points $R = X^-$ on the plane, compute a box (axis-aligned rectangle) containing the maximum number of points of $B$ but avoiding points of $R$. In many applications, the data are given in a dynamic scenario. For instance, in fixed wireless telephony access and driving assistance, detection and recognition of patterns for moving objects are key functions [2]. In fact, with the continued proliferation of wireless communication and advances in positioning technologies, algorithms to efficiently solve optimization problems about large populations of moving data are gaining interest. New devices offer to the companies an opportunity of providing a diverse range of e-services, many of which will exploit knowledge of the user's changing location. This results in new challenges to database and classification technology [4].

In this paper, we introduce and investigate the dynamic version of the Maximum Box Problem where the dataset is modeled by points moving according to specified algebraic functions. We present a *Kinetic Data Structure* (KDS) to efficiently maintain the maximum box for a bi-colored set of moving points. A KDS is used for keeping track the attributes of interest in a system of moving objects [8]. The plan of the paper is as follows: In section 2 we give an algorithm for the static case that will be useful for the dynamic version. A new data structure for maintaining the maximum box in a kinetic setting is proposed in section 3. In section 4 we give an algorithm for the Maximum Box Problem in a static setting for which the box can be arbitrarily oriented. Finally, in section 5 we present an efficient data structure to maintain an approximated solution of the Maximum Box Problem when the points move.

*University of Texas at Dallas, besp@utdallas.edu

†Departamento Matemática Aplicada II, Universidad de Sevilla, dbanez@us.es, Partially supported by Grant MEC MTM2006-03909

‡Departamento de Computación, Universidad de La Habana, pablo@matcom.uh.cu, Partially supported by Grant MAEC-AECI and MEC MTM2006-03909

§Departamento Matemática Aplicada II, Universidad de Sevilla, iventura@us.es, Partially supported by Grant MEC MTM2006-03909

## 2 The Static Version

The static version of the maximum box problem in the plane has been already studied. In [9], an $O(b^2 \log b + br + r \log r)$-time algorithm has been presented, where $r = |R|$ and $b = |B|$. We propose here a similar algorithm which is used in the dynamic case for the designment of our data structure. First, we observe that a maximum box for $B$ and $R$ can be enlarged until each of its sides either touches a red point or reaches the infinity. Thus, the maximum box can be transformed to one of the following isothetic objects with red points on its boundary and no red points inside: a rectangle, a half-strip, a strip, a quadrant or a half-plane (see Fig. 1).

We show how to compute the maximum box of type 1), 2) or 3). The cases of a quadrant 4) and a half-plane 5) are even easier and can be addressed by applying the techniques based on the *Dynamic Bentley's Maximum Subsequence Sum Problem* that were presented in [6]. Both cases can be solved in time $O(n \log n)$.

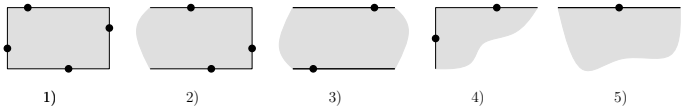



Figure 1: Configurations of the maximum box.

We proceed as follows. For every red point $p_r$, we compute a rectangle $H(p_r)$ such that: $(i)$ the top side contains $p_r$, $(ii)$ the interior contains the maximum number of blue points, and $(iii)$ the boundary has only red points. Then, we take the best of all $H(p_r)$. To do this for each red point $p_r$, we draw a horizontal line $l$ passing through $p_r$ and sweep the points below $l$ by moving $l$ downwards.

Let $h_r$ be the horizontal line that passes through $p_r$. During the sweeping we maintain two balanced binary search trees $T_R$ and $T_B$ such that $T_R$ (resp. $T_B$) contains all the red (resp. blue) points that lie between $h_r$ and $l$ sorted by $x$-coordinate. In the tree $T_B$, for each node $v$ we also maintain a counter of the number of blue points on the subtree rooted at $v$. When $l$ passes through a point $p$ we do the following. Let $left(p)$ (resp. $right(p)$) be the rightmost (resp. leftmost) red point in $T_R$ located to the left (resp. right) of the vertical line that passes through $p$, and let $x_{left}$ (resp. $x_{right}$) be its $x$ coordinate. If $left(p)$ (resp. $right(p)$) does not exist then $x_{left}$ (resp. $x_{right}$) is $-\infty$ (resp. $+\infty$). We only process $p$ if the $x$ coordinate of $p_r$ lies in the interval $(x_{left}, x_{right})$. In that case, $p$ is inserted in $T_R$ or $T_B$ according to its color and if $p$ is a red point, we consider a candidate $H(p_r)$ as the isothetic rectangle (half-strip or strip) whose sides pass through $left(p)$, $right(p)$, $p_r$ and $p$. The count of blue points inside $H(p_r)$ is equal to the count of blue points in $T_B$ whose $x$ coordinate lies in $(x_{left}, x_{right})$.

In the above procedure $left(p)$, $right(p)$ and the count of blue points inside $H(p_r)$ are obtained in logarithmic time each, thus the top-down sweeping from $p_r$ is done in $O(r \log r + r \log b + b \log b)$ time, giving then an overall $O(r^2 \log r + r^2 \log b + rb \log b)$-time and $O(r + b)$-space process. In many applications such as medical diagnosis the negative data, (the red points) has few elements with respect the positive data (the blue points). It means that $r \ll b$ and thus our complexity is essentially $O(b \log b)$. Within the same time we can find the smallest axis-parallel rectangle (box) that maximizes the number of covered points from $B$ and does not contain any point from $R$, improving the $O(n^3 \log^4 n)$-time algorithm given by [10].

## 3 The Maximum Box Problem for moving points

In this section we introduce a new kinetic data structure (KDS), for maintaining the maximum box. A KDS is a structure that maintains an attribute of a set of continuously moving objects [8]. It consists of two parts: a combinatorial description of the attribute and a set of certificates. The certificates are elementary tests on the input objects with the property that as long as their outcomes do not change, the attribute does not.

**Lemma 3.1.** *Let $X$ (resp. $Y$) be the elements of $R \cup B$ sorted by abscissa (resp. ordinate). The maximum box for $R$ and $B$ is univocally determined by $X$ and $Y$, and it does not change combinatorially over time as long as $X$ and $Y$ do not.*

Our KDS is named as *Maximum Box Kinetic Data Structure* (MBKDS) and we will proceed now with its ingredients. In our problem, the set of moving objects is $R \cup B$ and the attribute is its maximum box. According to the previous lemma, we consider as a certificate the condition that two consecutive points in $X$ (resp. $Y$) satisfy the $x$-order (resp. $y$-order). An event is the failure of a certificate at some instant of time $t$ and it implies an update of the MBKDS. We denote each event as a pair $\langle c, t\rangle$ where $c$ is the certificate and $t$ is the instant of time where $c$ is violated. We name an event as *flip*.

A KDS is *compact* if it has *few* certificates and *local* if no object participates in *too many* certificates. With this it can be updated easily when the flight plan of an object changes (see [8]). It is easy to prove that the MBKDS is compact and local.

The MBKDS is composed by the event queue $\mathcal{E}$ and the structure $\mathcal{D} = \{\mathcal{D}(p_r) : p_r \in R\}$ where $\mathcal{D}(p_r)$ is a data structure for each red point $p_r$. When an event corresponding to the certificate of two consecutive elements of $X$, say $[X_i, X_{i+1}]$, occurs it is removed from $\mathcal{E}$. Then, we remove from $\mathcal{E}$ the events corresponding to $[X_{i-1}, X_i]$ and $[X_{i+1}, X_{i+2}]$. After that, we swap $X_i$ and $X_{i+1}$ and insert in $\mathcal{E}$ new events for $[X_{i-1}, X_i]$, $[X_i, X_{i+1}]$ and $[X_{i+1}, X_{i+2}]$. The event time is computed based on our knowledge of the motions of $X_{i-1}$, $X_i$, $X_{i+1}$ and $X_{i+2}$. $\mathcal{E}$ is designed as a priority queue with its basic operations in logarithmic time. The certificates $[Y_i, Y_{i+1}]$ are treated similarly.

The definition and details of $\mathcal{D}(p_r)$ are as follows. Consider first the horizontal line $h_r$ passing through the red point $p_r$ and let $\mathrm{Red}(h_r)$ be the set of red points lying below $h_r$ (see Fig. 2). Denote as $x(p)$ (resp. $y(p)$) the abscissa (resp. ordinate) of $p$. For each $p \in \mathrm{Red}(h_r)$ we define:

- $\mathcal{V}(p)$ as the vertical half-line for which $p$ is its top-most point.
- $\mathcal{I}(p)$ as the maximum-length horizontal segment that contains $p$ and does not intersect any other $\mathcal{V}(q)$ for $q$ in $\mathrm{Red}(h_r) \setminus \{p\}$.
- $\mathrm{Blue}(s)$ as the set of blue points that lie on the rectangle whose bottom side is the horizontal segment $s$ and its top side lies on $h_r$.
- $left(p)$ (resp. $right(p)$) as the rightmost (resp. leftmost) point in $\mathrm{Red}(h_r)$ located to the left (resp. right) of the vertical line that passes through $p$ and whose $y$ coordinate is greater than $y(p)$ (see Fig. 2). Notice that the end points of $\mathcal{I}(p)$ lie on $\mathcal{V}(left(p))$ and $\mathcal{V}(right(p))$.
- the set $candidates(p_r)$ of the points $p \in \mathrm{Red}(h_r)$ such that $\mathcal{I}(p)$ intersects the vertical line that passes through $p_r$. Observe that, according to the previous section, $H(p_r)$ is the box whose top side lies on $h_r$ and its bottom side is the interval $\mathcal{I}(p)$ such that $p \in candidates(p_r)$ and $|\,\mathrm{Blue}(\mathcal{I}(p))|$ is maximized.

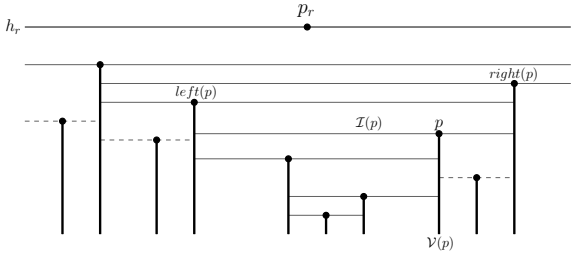



Figure 2: $\mathcal{D}(p_r)$. Candidate points are those whose $\mathcal{I}(p)$ is drawn with a continuous line.

The key idea for $D(p_r)$ is to dynamically compute $H(p_r)$ when two points that lie below $h_r$ make a flip. $\mathcal{D}(p_r)$ is designed as follows:

- The elements of $candidates(p_r)$ are stored in the leaves of a dynamic balanced binary search tree $\mathcal{Q}$ in decreasing order of ordinate. Every node $v$ is labeled with $a$ such that $|\operatorname{Blue}(\mathcal{I}(p))|$ is equal to the sum of the $a$ labels of the nodes in the unique simple path from the leaf that represents $p$ to the root. In addition, every internal node $v$ is labeled with $b$ whose value is the candidate $p$ in the subtree rooted at $v$ that maximizes $|\operatorname{Blue}(\mathcal{I}(p))|$. With this, the label $b$ of the root is the candidate red point that determines $H(p_r)$.

- A balanced binary search tree $\mathcal{T}_R$ whose elements are the elements of $\operatorname{Red}(h_r)$ in increasing order of abscissa. Every node $v$ is labeled with $y_{max}$ that is the element that has the maximum ordinate on the subtree rooted at $v$. It permits us to obtain in logarithmic time $left(p)$ and $right(p)$ for any $p \in \operatorname{Red}(h_r)$.

- For each $p \in \operatorname{Red}(h_r)$ we divide $\mathcal{I}(p)$ at $p$ obtaining the horizontal segments $\mathcal{I}_{left}(p)$ and $\mathcal{I}_{right}(p)$ (see Fig. 3). Let $\mathcal{T}_{left}(p) = \{\mathcal{I}_{right}(q) : q \in \operatorname{Red}(h_r) \setminus \{p\} \wedge right(q) = p\}$ and $\mathcal{T}_{right}(p) = \{\mathcal{I}_{left}(q) : q \in \operatorname{Red}(h_r) \setminus \{p\} \wedge left(q) = p\}$. The set $\mathcal{T}_{left}(p)$ is represented in a dynamic balanced binary search tree where its elements $\mathcal{I}_{right}(q)$ are stored at the leaves in decreasing order of $y(q)$. Every node $v$ is labeled with $a$ in a similar way as we did for $\mathcal{Q}$. In this case $|\operatorname{Blue}(\mathcal{I}_{right}(q))|$ is the sum of the $a$ labels of the nodes in the path from the leaf that represents $\mathcal{I}_{right}(q)$ to the root. Also, every node $v$ is labeled with the boolean $c$ indicating if all the elements $\mathcal{I}_{right}(q)$ in the subtree rooted at $v$ satisfy that $q$ is a candidate red point. $\mathcal{T}_{left}(p)$ is designed to support the operators *join* and *split* both in logarithmic time [5]. The set $\mathcal{T}_{right}(p)$ is represented in the same manner.

- The set $\{\mathcal{I}(p) : p \in \operatorname{Red}(h_r)\} \cup \{\mathcal{V}(p) : p \in \operatorname{Red}(h_r)\}$ determines a partition $\mathcal{P}$ of the lower half-plane of $h_r$. For each cell $\mathcal{C}$ of $\mathcal{P}$ we store in a balanced binary search tree $\mathcal{T}_B(\mathcal{C})$ (supporting operators *join* and *split*) the elements of $B \cap \mathcal{C}$ in decreasing order of ordinate. Each node $v$ is labeled with the count of blue points on the subtree rooted at $v$ in such a way the label of the root is $|B \cap \mathcal{C}|$. Each blue point $p$ below $h_r$ has a reference $\mathcal{T}_{\mathcal{C}}(p)$ to the tree $\mathcal{T}_B(\mathcal{C})$ where $\mathcal{C}$ is the cell that contains $p$. Also, we associate to each $p \in \operatorname{Red}(h_r)$ the tree $\mathcal{T}_B(p)$, which is the tree of blue points that corresponds to the cell of $\mathcal{P}$ that is bounded below by $\mathcal{I}(p)$, and the tree $\mathcal{T}_{B_{right}}(p)$ that corresponds to the cell of $\mathcal{P}$ bounded by $\mathcal{V}(p)$ and $\mathcal{V}(q)$ and has no bottom boundary, where $q$ is the element that follows $p$ on $\mathcal{T}_R$. (see Fig. 3).

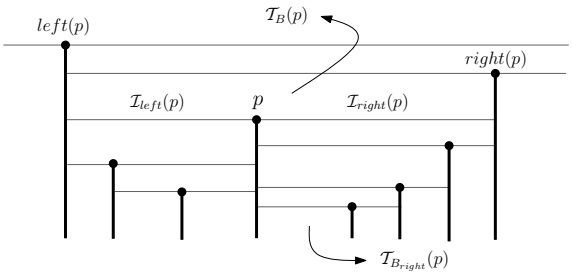



Figure 3: Details of $\mathcal{D}(p_r)$.

Because of the similarity between the structure of $\mathcal{D}(p_r)$ and the operations of the algorithm of section 2, a similar algorithm to build $\mathcal{D}(p_r)$ can be designed. We establish the following results,

**Theorem 3.2.** *Given $R$, $B$ and $p_r$ such that $|R| = r$, $|B| = b$ and $p_r \in R$, $\mathcal{D}(p_r)$ requires $O(r + b)$ space and it can be built in $O(r \log r + r \log b + b \log b)$ time.*

**Corollary 3.3.** *Given $R$ and $B$ such that $|R| = r$ and $|B| = b$, the data structure $\mathcal{D}$ has space complexity $O(r^2 + rb)$ and it can be built in $O(r^2 \log r + r^2 \log b + rb \log b)$ time.*

In the following we show all the possible types of flips that can occur when the elements of $R \cup B$ move. The details on how to process any of them depend on its type and are given in the full version. We say that a flip is *horizontal* (resp. *vertical*) when the two involved points change their $y$-order (resp. $x$-order).

**Flip type (A)**: Two blue points make a vertical flip. Nothing to do.

**Flip type (B)**: Two blue points $b_1$ and $b_2$ make a horizontal flip. Exchange $b_1$ and $b_2$ in every tree of blue points $\mathcal{T}_B(.)$ or $\mathcal{T}_{B_{right}}(.)$ that contains both.

**Flip type (C)**: A red point $r_1$ and a blue point $b_2$ make a horizontal flip. Consider five cases as depicted in Figure 4. Note that case 5) occurs in only one structure $\mathcal{D}(\cdot)$ and it can be processed by rebuilding $\mathcal{D}(\cdot)$.

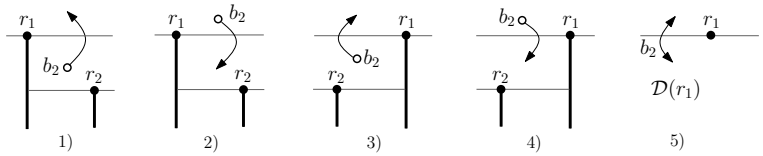



Figure 4: Horizontal flip cases between a red point $r_1$ and a blue point $b_2$.

**Flip type (D)**: A red point $r_1$ and a blue point $b_2$ make a vertical flip. Consider two cases as depicted in Figure 5, and their symmetric cases.

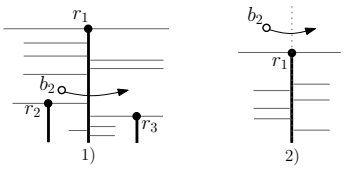



Figure 5: Vertical flip cases between a red point $r_1$ and a blue point $b_2$.

**Flip type (E)**: Two red points $r_1$ and $r_2$ make any flip: we have to consider two cases as depicted in Figure 6, and their symmetric cases. When the flip is vertical the operators *join* and *split* [5] are useful.

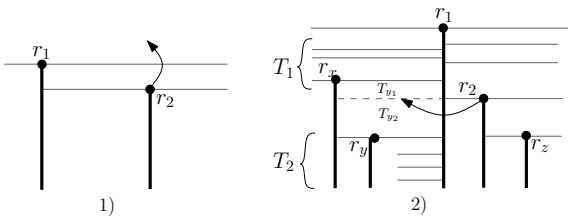



Figure 6: Flip cases between two red points $r_1$ and $r_2$. 1) Horizontal 2) Vertical.

Each $\mathcal{D}(p_r)$ is updated in $O(\log r + \log b)$ time, then it implies an $O(r \log r + r \log b)$-time update for $\mathcal{D}$. The basic operations of the procedures for solving the flip cases are done in logarithmic time each. Each of the trees that are used in $\mathcal{D}(p_r)$ can be implemented by using *Red-Black Trees* where all the elements are stored in the leaves [5]. We arrive to the following result.

**Theorem 3.4.** *Given a set of moving red points $R$ and a set of moving blue points $B$ such that $|R| = r$ and $|B| = b$, whenever two points in $R \cup B$ make a flip, the data structure MBKDS can be updated in $O(r \log r + r \log b)$ time.*

# 4 The Arbitrarily Oriented Maximum Box Problem

In this section we consider, as an application of our method for the dynamic case, the problem of finding the Maximum Box for a static set $R \cup B$ on the plane, when the box can be oriented according with any angle (direction) in $[0, \pi]$.

It is easy to see that there are $O((r+b)^2)$ critical directions and the method of section 2 can be applied for each of them. With this we obtain an $O((r+b)^2(r^2 \log r + r^2 \log b + rb \log b))$-time algorithm. However, we can do it better by iterating all the critical directions and computing dynamically the

Maximum Box per each. Start with the direction given by the angle $\theta = 0$. Compute the isothetic Maximum Box and the data structure $\mathcal{D}$. Then make a rotational sweeping of the coordinate axis maintaining the $x$-order and the $y$-order of the elements of $R \cup B$. The $x$-order (resp. $y$-order) changes whenever two points are at the same distance to the $y$-axis (resp. $x$-axis), implying the swap (*flip*) of two consecutive elements and the appearance of a new critical orientation $\theta$. Thus, the Maximum Box that is oriented according to $\theta$ is computed by making the $O(r \log r + r \log b)$-time update in $\mathcal{D}$. The overall rotation angle is at most $\pi$ radians and we establish the following result.

**Theorem 4.1.** *Given a set of red points $R$ and a set of blue points $B$ on the plane such that $|R| = r$ and $|B| = b$, the Arbitrarily Oriented Maximum Box Problem can be solved in $O((r+b)^2(r \log r + r \log b))$ time and $O(r^2 + rb)$ space.*

# 5 The approximated version

In this section we study how to maintain an approximated maximum box by using sub-quadratic space. An $O(n \log^2 n)$-time algorithm to compute the 1/2-approximated maximum box on a static setting is presented in [9]. The key idea is as follows: take the median point $p$ of the $x$-order and compute the exact maximum box $H_p$ that has $p$ on either its left side or its right side. Then, return the best box between $H_p$ and the ones that are returned by the recursive call to the sets $\{q \in R \cup B | x(q) < x(p)\}$ and $\{q \in R \cup B | x(q) > x(p)\}$.

We use a similar procedure to extent MBKDS to a new KDS named as Apx-MBKDS. First of all, we study the dynamic operations on $\mathcal{D}(p_r)$, that is, how to perform efficiently insertions an deletions of red an blue points on it. After that, we consider a new data structure $\mathcal{D}^*(p_r)$ as an extension of $\mathcal{D}(p_r)$ and it is used with a binary tree that has the points of $R$ sorted by $y(\cdot)$.

## 5.1 Dynamic operations in $\mathcal{D}(p_r)$

In this subsection we prove that inserting or deleting either a red point or a blue point in $\mathcal{D}(p_r)$ can be done in $O(r + b)$ time. Both actions are based on the following primitives.

- *GetPriorities*: Given any $\mathcal{D}(p_r)$ calculates $|\operatorname{Blue}(\mathcal{I}(p))|$ for all the red points $p \in \mathcal{D}(p_r)$. It takes all such $p$ and applies a top-bottom traversal in $\mathcal{T}_{left}(p)$ and in $\mathcal{T}_{right}(p)$ to obtain, by using the $a$-marks, $|\operatorname{Blue}(\mathcal{I}_{right}(q))|$ if $right(q) = p$ or $|\operatorname{Blue}(\mathcal{I}_{left}(q))|$ if $left(q) = p$.
- *IntersectLists*$(v)$: Given any $\mathcal{D}(p_r)$ and a point $v$ returns two lists $L_{left}$ and $L_{rigth}$ such that: *i)* $L_{left}$ (*resp.* $L_{rigth}$) contains the red points $q \in \mathcal{D}(p_r)$ such that $\mathcal{I}(q)$ intersects the vertical half-line emanated from $v$ downwards and $x(q) < x(v)$ (*resp.* $x(v) < x(q)$) *ii)* the elements in $L_{left}$ and $L_{rigth}$ are sorted in decreasing order of $y(\cdot)$. It can be done in linear time as follows: Denote by $l_v$ the vertical half-line emanated from $v$ downwards and set $L_{left}$ and $L_{rigth}$ to two empty lists. Iterate the red points $q \in \mathcal{D}(p_r)$ by traversing the leaves of $\mathcal{T}_R$ from left to right. If $\mathcal{I}(q)$ intersects $l_v$ then insert $q$ at the end of $L_{left}$ if $x(q) < x(v)$, otherwise insert it at the beginning of $L_{rigth}$.
- *RebuildCandidates*: This is the operation that given $\mathcal{D}(p_r)$ rebuilds in linear time the set $\mathcal{Q}$ associated to $\mathcal{D}(p_r)$. First, apply *GetPriorities* and obtain the candidates red points by merging on a list $L$ the two lists returned by invoking *IntersectLists*$(p_r)$. After that, apply a bottom-up building of $\mathcal{Q}$ from $L$.

By using above primitives insertions and deletions can be done as follows. Inserting a blue point $v$ in $\mathcal{D}(p_r)$ can be done in linear time: First, find the cell $\mathcal{C}$ of the partition $\mathcal{P}$ determined by $\mathcal{D}(p_r)$ and insert $v$ in $\mathcal{T}_B(\mathcal{C})$. After, in a naive way: Apply *GetPriorities* and, for all red point $q \in \mathcal{D}(p_r)$ increment $|\operatorname{Blue}(\mathcal{I}_{left}(q))|$ (*resp.* $|\operatorname{Blue}(\mathcal{I}_{right}(q))|$) in one if $v$ is above $\mathcal{I}_{left}(q)$ (*resp.* $\mathcal{I}_{right}(q)$). At the end invoke *RebuildCandidates*.

Inserting a red point $v$ requires a different approach.

1. Apply *GetPriorities* to obtain for all $p \in R \cup B$, $|\,\mathrm{Blue}(\mathcal{I}_{left}(p))|$ and $|\,\mathrm{Blue}(\mathcal{I}_{right}(p))|$ and $IntersectLists(v)$ to get the two lists of red points $L_{left} = \{l_1, l_2, \ldots, l_{k_1}\}$ and $L_{right} = \{r_1, r_2, \ldots, r_{k_2}\}$ where $l_i$ $(1 \leq i \leq k_1)$ and $r_j$ $(1 \leq j \leq k_2)$ are the left and right red points as shown in Figure 7.

2. Include $v$ in $\mathcal{D}(p_r)$ by inserting $v$ in $\mathcal{T}_R$. This implies the calculation of $|\,\mathrm{Blue}(\mathcal{I}_{left}(v))|$ and $|\,\mathrm{Blue}(\mathcal{I}_{right}(v))|$ to insert $\mathcal{I}_{left}(v)$ and $\mathcal{I}_{right}(v)$ in the trees $\mathcal{T}_{right}(left(v))$ and $\mathcal{T}_{left}(right(v))$ respectively.

3. Suppose now w.l.o.g. that $Y(l_1) > Y(r_1)$. Determine for all red point $p$ in $L_{left}$ or $L_{rigth}$ $(p \neq l_1)$ the lists of blue points $L_1(p)$ and $L_2(p)$ whose members are sorted by $Y$ in a decreasing way and $L_1(p)$ (resp. $L_2(p)$) contains the blue points of $\mathcal{T}_B(p)$ such that $x(p) < x(v)$ (resp. $x(v) < x(p)$).

4. For the new cells $\mathcal{C}_v$, $\mathcal{C}_{l_1}$ and $\mathcal{C}_{r_1}$, whose bottom sides are $\mathcal{I}(v)$, $\mathcal{I}(l_1)$ and $\mathcal{I}(r_1)$ respectively, compute the corresponding trees $\mathcal{T}_B(v) = \mathcal{T}_B(\mathcal{C}_v)$, $\mathcal{T}_B(l_1) = \mathcal{T}_B(\mathcal{C}_{l_1})$ and $\mathcal{T}_B(r_1) = \mathcal{T}_B(\mathcal{C}_{r_1})$.

5. The elements of $L_{left}$ are the left neighbors of $\mathcal{V}(v)$ and to build $\mathcal{T}_{left}(v)$ we have to compute $|\,\mathrm{Blue}(\mathcal{I}_{right}(l_i))|$ for all $l_i$ in $L_{left}$. It can be done as follows: calculate first $|\,\mathrm{Blue}(\mathcal{I}_{right}(l_1))|$, and after for each $2 \leq i \leq k_1$ build $\mathcal{T}_B(l_i)$ from the concatenation of the lists $L_1(r_j)$ where $Y(l_{i-1}) < Y(r_j) < Y(l_i)$, then $|\,\mathrm{Blue}(\mathcal{I}_{right}(l_i))|$ is calculated by using the following equation:

$$|\,\mathrm{Blue}(\mathcal{I}_{right}(l_i))| = |\,\mathrm{Blue}(\mathcal{I}_{right}(l_{i-1}))| - |\,\mathrm{Blue}(\mathcal{I}_{left}(l_i))| + |\mathcal{T}_B(l_i)|$$

   where $|\mathcal{T}_B(l_i)|$ is the number of elements in $\mathcal{T}_B(l_i)$. Proceed similarly to build $\mathcal{T}_{right}(v)$ from $L_{right}$.

6. Compute $T_{B_{right}}(l_{k_1})$ and $T_{B_{right}}(v)$ and apply *RebuildCandidates*.

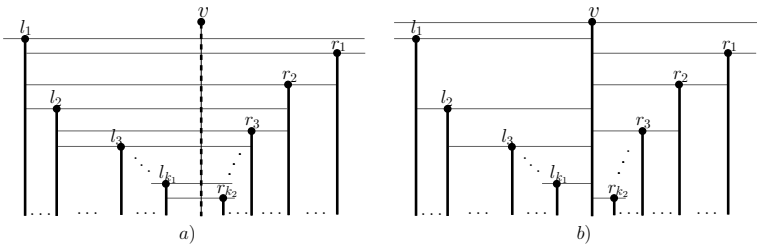



Figure 7: Inserting a red point $v$ in $\mathcal{D}(p_r)$. *a)* Before insertion, *b)* After insertion.

The deletion can be done similar to the insertion. It is easy to show that both operations spend $O(r + b)$ time thus we obtain the following result:

**Theorem 5.1.** *Inserting or deleting either a red point or a blue point in the data structure $\mathcal{D}(p_r)$ can be done in $O(r + b)$ time.*

## 5.2 Apx-MBKDS

In this section we extend or MBKDS to support the maintenance of the 1/2-approximated maximum box. For this consider the extended version $\mathcal{D}^*(p_r)$ of $\mathcal{D}(p_r)$ that takes care the points that are not only below the line $h_r$ that passes through $p_r$ but also above it. Those points above $h_r$ are structured symmetrically to the points that lie below (See Figure 8 (a)). Now suppose using in the 1/2-approximated maximum box algorithm presented in [9] the median point of the $y$-order instead of the $x$-order one's. Then our Apx-MBKDS is a Two-Level data structure that is composed in the first level by a balanced binary search tree $T_{1/2}$ having the elements of $R$ sorted by $Y$ and in the second level by instances of $\mathcal{D}^*(.)$. Refer to Figure 8 (b). The recursive definition is as follows:

Given $R$ and $B$, let $p_r$ be the median point of $R$ in the $y$-order. The root node $v$ stores $p_r$ and $\mathcal{D}^*(v) = \mathcal{D}^*(p_r)$ and it is built from $R$ and $B$. The left child of $v$ is recursively constructed from the sets $R_l = \{q \in R | y(q) < y(p_r)\}$ and $B_l = \{q \in B | y(q) < y(p_r)\}$. Similarly, the right child from $R_r = \{q \in R | y(p_r) < y(q)\}$ and $B_r = \{q \in B | y(p_r) < y(q)\}$.

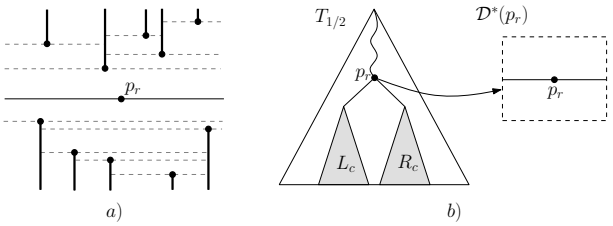



Figure 8: *a)* $\mathcal{D}^*(p_r)$, *b)* The Two-Level Data Structure for the approximated version.

Additionally we associate to every node $v$ of $T_{1/2}$ the best maximum box $H(v)$ of the elements of the subtree rooted at $v$. It is the best box between the solution provided by $\mathcal{D}^*(v)$ and $H(lc(v))$ and $H(rc(v))$ where $lc(v)$ and $rc(v)$ are the left and right child of $v$ respectively. The box $H(root)$ of the root node *root* is the 1/2-approximated maximum box. It is easy to show that Apx-MBKDS uses $O((r+b)\log r)$ space and can be built in $O((r+b)\log^2(r+b))$ time. Now we describe the processing of flips.

Processing the horizontal flip between two blue points $p_1$ and $p_2$ or the vertical flip between a red point $p_1$ and a blue point $p_2$ can be done as follows: Let $v$ be the node in $T_{1/2}$ where the search paths of $p_1$ and $p_2$ splits. Apply the $O(\log r + \log b)$-time update in $\mathcal{D}^*(w)$ for every node $w$ in the path from the parent of $v$ to the root. Thus this operation takes $O(\log^2 r + \log r \log b)$ time. The horizontal flip between a red point $p_1$ and a blue point $p_2$ can be processed as above but in addition: Let $v$ be the node that stores $p_1$, update $\mathcal{D}^*(v)$ by applying the insertion/deletion method of a blue point described in the previous subsection and update $\mathcal{D}^*(w)$ in every node $w$ in the path from $v$ to the predecessor (resp. successor) of $v$ in $T_{1/2}$. This process spends $O(r+b)$ time. The horizontal flip between two red points $p_1$ and $p_2$ can be processed similarly but adding the following: Let $v_1$ and $v_2$ be the nodes that store $p_1$ and $p_2$ respectively and suppose w.l.o.g. that $v_2$ belongs to the subtree rooted at $v_1$. Update $\mathcal{D}^*(v_1)$ by applying the insertion/deletion method of a red point in $\mathcal{D}(.)$. We obtain the main result of this section:

**Theorem 5.2.** *Apx-MBKDS is a compact and local kinetic data structure for maintaining the* 1/2*-approximated maximum box over a set of moving points. It uses* $O((r+b)\log r)$ *memory and can be built in* $O((r+b)\log^2(r+b))$ *time. The events can be processed in* $O(r+b)$ *time in the worst case.*

# References

[1] B. Aronov and S. Har-Peled. On approximating the depth and related problems. SIAM J. Comput. Vol. 38, 899–921, 2008.

[2] P. Bonnet, J. Gehrke, P. Seshadri. Towards Sensor Database Systems. In Proc. of the Second International Conf. on Mobile Data Management, Hong Kong. Lecture Notes Comp. Sci., vol. 1987, 3 – 14, 2001.

[3] B. Chazelle and L. J. Guibas. Fractional cascading: A data structuring technique, Algorithmica, 1 , 133–162, 1986.

[4] A. Civilis, C. S. Jensen, S. Pakalnis. Techniques for Efficient Road-Network-Based Tracking of Moving Objects. IEEE Transactions On Knowledge And Data Engineering, Vol. 17, No. 5, 698–712, 2005.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest. *Introduction to Algorithms*, Second Edition. MIT Press, McGraw-Hill, 2001.

[6] C. Cortés, J. M. Díaz-Báñez, P. Pérez-Lantero, C. Seara, J. Urrutia, I. Ventura. Bichromatic separability with two boxes: a general approach. Journal of Algorithms. InPress. 2009.

[7] R. Duda, P. Hart, D. Stork. *Pattern Classification. John Wiley and Sons, Inc.*, New York, 2001.

[8] L.J. Guibas. Kinetic Data Structures: A State of the Art Report. In *Robotics: The Algorithmic Perspective.* A. K. Peters, Ltd 191–209, 1998.

[9] Y. Liu, M. Nediak. Planar Case of the Maximum Box and Related Problems. In Proc. Canad. Conf. Comp. Geom., Halifax, Nova Scotia, August 11–13, 2003.

[10] M. Segal. Planar Maximum Box Problem. Journal of Mathematical Modeling and Algortihms. 3, 31–38, 2004.

# $L$-corredor $k$-cromático

C. Bautista-Santiago* J. Cano-Vila† J. M. Díaz-Báñez‡
H. González-Aguilar§ D. Lara¶ J. Urrutia‖

**Resumen**

Sea $S$ un conjunto $k$-coloreado de $n$ puntos en posición general en el plano, $k \leq n$. Decimos que una región en el plano es $k$-coloreada, con respecto a $S$, si ésta contiene al menos un punto de cada color. Sea $p = (a, b)$ un punto en el plano, definimos $C_p^+ = \{(x, y) : a \leq x, b \leq y\}$ y $C_p^- = \{(x, y) : a < x, b < y\}$. Un $L$-corredor es un conjunto del plano de la forma $C_p^+ \setminus C_q^-$ tal que $p = (a, b)$ y $q = (c, d)$ con $a < c$ y $b < d$.

En este trabajo proponemos un algoritmo con complejidad $O(n^2 \log k)$ que resuelve el problema de calcular el $L$-corredor $k$-cromático que minimice $(c-a)$, bajo la condición que $(c-a) = (d-b)$. Además, damos un algoritmo de complejidad $O(n^2 k)$ que encuentra el $L$-corredor $k$-cromático, tal que éste minimiza la suma de $(c - a) + (d - b)$. Así también, consideramos el problema donde el $L$-corredor $k$-cromático no tiene una orientación fija, y proporcionamos un algoritmo con complejidad $O(n^3 \log n)$ para poder encontrarlo.

## 1 Introducción

Dado un punto $p = (a, b)$ del plano, sean $C_p^+ = \{(x, y) : a \leq x, b \leq y\}$ y $C_p^- = \{(x, y) : a < x, b < y\}$ los cuadrantes cerrado y abierto, respectivamente, definidos por $p$. Un $L$-corredor del plano es un conjunto de puntos $\mathcal{C}_{p,q}$ de la forma $C_p^+ \setminus C_q^-$ tales que si $p = (a, b)$ y $q = (c, d)$, entonces $a < c$ y $b < d$.

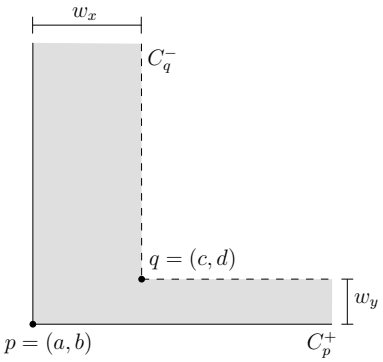



Figura 1: Definición de un $L$-corredor.

Dado un $L$-corredor $\mathcal{C}_{p,q}$, podemos asociarle dos valores: $w_x = c - a$ y $w_y = d - b$. A $w_x$ y $w_y$ les llamaremos los *anchos* de los *pasillos* de $\mathcal{C}_{p,q}$. Cuando $w_x = w_y$ diremos que $w_x$ es el *ancho* de $\mathcal{C}_{p,q}$.

*Posgrado en Ciencia e Ingeniería de la Computación, UNAM, México, crevel@uxmcc2.iimas.unam.mx.

†Posgrado en Ciencia e Ingeniería de la Computación, UNAM, México, j_cano@uxmcc2.iimas.unam.mx.

‡Departamento de Matemática Aplicada II, Universidad de Sevilla, España, dbanez@us.es. Financiado parcialmente por el proyecto MTM2006-03909 del MEC.

§Instituto de Matemáticas, UNAM, México, hernan@matem.unam.mx. Financiado parcialmente por el proyecto CONACyT CB-2007/80268.

¶Posgrado en Ciencia e Ingeniería de la Computación, UNAM, México, dlara@uxmcc2.iimas.unam.mx.

‖Instituto de Matemáticas, UNAM, México, urrutia@matem.unam.mx. Financiado parcialmente por los proyectos MTM2006-03909 del MEC y CONACyT CB-2007/80268.

Problemas sobre $L$-corredores han sido estudiados en varios artículos, ver [2, 3, 4, 5]. En algunos de esos trabajos, los $L$-corredores tienen cuatro orientaciones distintas rotando los $L$-corredores definidos aquí 90°, 180° y 270° (corredores isotéticos), o por cualquier ángulo entre 0 y $2\pi$.

En [2], se estudia el problema de encontrar un $L$-corredor de ancho máximo, no necesariamente isotético, cuyo interior no contenga ningún punto de una familia dada de puntos. En [5] se busca encontrar un $L$-corredor con las mismas condiciones, pero permitiendo que el ángulo entre sus pasillos sea distinto a $\pi/2$.

Sea $S \subset \mathbb{R}^2$ un conjunto de $n$ puntos en posición general, de manera que cada punto $p \in S$ tiene asociado un color $c_i$ de entre $k$ colores, es decir, $S$ es un conjunto *$k$-coloreado*. Decimos que un $L$-corredor $\mathcal{C}_{p,q}$ está $k$-coloreado, con respecto al conjunto de puntos $S$, si $\mathcal{C}_{p,q} \cap S$ contiene al menos un punto de cada color. En tal caso diremos simplemente que $\mathcal{C}_{p,q}$ está $k$-coloreado, o que es $k$-cromático.

Existen varios algoritmos que encuentran objetos geométricos $k$-coloreados. En [1], por ejemplo, se dan algoritmos para encontrar la *banda* $k$-coloreada más angosta en $O(n^2 \log n + n^2\alpha(k) \log k)$ tiempo, así como el *rectángulo isotético* $k$-coloreado de menor área (o perímetro) en $O(n(n-k)\log^2 k)$ tiempo. La complejidad de dichos algoritmos fue mejorada en [3], quedando en $O(n^2 \log n)$ y $O(n(n-k)\log k)$ respectivamente; además se plantea y resuelve el problema de encontrar el rectángulo $k$-coloreado de área mínima con orientación arbitraria, proponiendo un algoritmo de $O(n^3 \log k)$ tiempo. Por otra parte, en [6] se establece un algoritmo que en $O(kn \log n)$ tiempo da una solución a los problemas de reportar el *círculo* $k$-coloreado de radio mínimo, así como el *cuadrado* isotético $k$-coloreado de menor área (o perímetro).

En este trabajo, proponemos algoritmos para el problema de encontrar un $L$-corredor $k$-coloreado, que minimice $w_x + w_y$, así como un $L$-corredor de ancho mínimo, esto es, que minimice $w_x$ dado que $w_x = w_y$. También damos un algoritmo para el problema de hallar un $L$-corredor $k$-coloreado de ancho mínimo, sin que éste tenga una orientación dada.

## 2 Encontrando el $L$-corredor $k$-cromático de ancho mínimo

Dado $S$, en esta sección presentaremos un algoritmo para encontrar un $L$-corredor $\mathcal{C}_{p,q}$ $k$-cromático de ancho mínimo, esto es, un $L$-corredor, en el cual $w_x = c - a = d - b = w_y$.

Para cada punto $p = (a, b)$ del plano, sea $\vec{h_p}$ el conjunto de puntos $(x, b)$ tales que $x \geq a$, y $\vec{v_p}$ el conjunto de puntos $(a, y)$ con $y \geq b$, esto es, $\vec{h_p}$ y $\vec{v_p}$ son los rayos horizontal y vertical que parten de $p$ hacia la derecha y hacia arriba, respectivamente.

Claramente la frontera de un $L$-corredor $\mathcal{C}_{p,q}$ está formada por los cuatro rayos $\vec{h_p}$, $\vec{v_p}$, $\vec{h_q}$ y $\vec{v_q}$, a los que llamaremos los rayos de $\mathcal{C}_{p,q}$. Dado un $L$-corredor $\mathcal{C}_{p,q}$, decimos que un punto $r$ en su frontera es *esencial*, si no existe otro punto del mismo color que $r$ en el interior de $\mathcal{C}_{p,q}$.

Las siguientes observaciones serán de utilidad:

Supongamos que $\mathcal{C}_{p,q}$ es un $L$-corredor $k$-cromático de ancho mínimo. Es fácil ver que al menos tres de los cuatro rayos que forman la frontera de $\mathcal{C}_{p,q}$ deben contener al menos un elemento de $S$. Además podemos suponer que cada uno de los rayos $\vec{h_p}$, $\vec{v_p}$ contiene al menos un punto de $S$, el cual puede ser el mismo para ambos rayos, esto es, que coincidan con $p$. Por ejemplo, si $\vec{h_p}$ no contiene ningún elemento de $S$, podemos desplazar a $p$ y a $q$ simultáneamente hacia arriba hasta que $\vec{h_p}$ encuentre un elemento de $S$, obteniendo así otra solución a nuestro problema en la que tanto $\vec{h_p}$ como $\vec{v_p}$ contengan al menos un elemento de $S$. Esto puede resumirse de la siguiente manera:

**Observación 2.1.** Siempre existe un $L$-corredor de ancho mínimo $\mathcal{C}_{p,q}$ tal que cada uno de los rayos $\vec{h_p}$ y $\vec{v_p}$ contienen al menos un punto de $S$. Además al menos uno de los rayos $\vec{h_q}$ y $\vec{v_q}$ contiene un elemento de $S$.

**Observación 2.2.** Si $\mathcal{C}_{p,q}$ es un $L$-corredor $k$-cromático de ancho mínimo, entonces $p$ y $q$ están sobre una recta con pendiente 1.

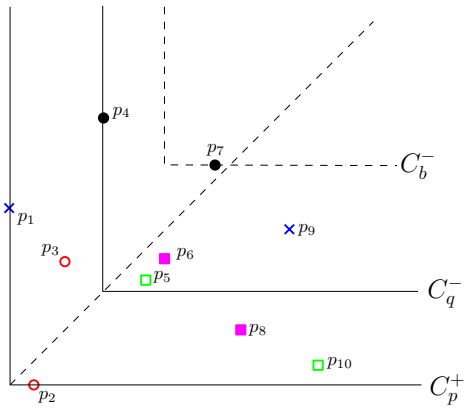



Figura 2: Ilustración de la observación 2.3.

Dado un cuadrante $C_p^+$, sea $H = \{h_1, \ldots, h_k\}$, donde $h_i$ es el punto de color $c_i$ en el interior de $C_p^+$ más cercano a $\vec{h_p}$. De manera análoga definimos al conjunto $V = \{v_1, \ldots, v_k\}$, pero tomando como referencia al rayo vertical $\vec{v_p}$. Asociamos a $C_p^+$ el cuadrante $C_b^+$ donde $b$ es el punto de intersección de la horizontal por el elemento $h_i$ de $H$ más alejado de $\vec{h_p}$, y la vertical por el elemento $v_j$ de $V$ más alejado de $\vec{v_p}$. La siguiente observación es obvia:

**Observación 2.3.** Si $\mathcal{C}_{p,q}$ es un $L$-corredor de ancho mínimo, entonces el punto $q$ se encuentra en $C_p^+ \setminus C_b^-$. Además al menos uno de $\vec{v_b}$ o $\vec{h_b}$ contiene un punto de $S$. Figura 2.

Dado un cuadrante $C_p^+$ y $r$ un punto en $C_p^+$, definimos a $d_p(r)$ como el mínimo entre las distancias de $r$ a $\vec{h_p}$ y a $\vec{v_p}$. Para cada color $c_i$, sea $m_i$ el punto de $S$ de color $c_i$ en $C_p^+$ que minimiza $d_p(m_i)$, y sea $\mathcal{M} = \{m_1, m_2, \ldots, m_k\}$. La siguiente observación será útil: sea $\mathcal{C}_{p,q}$ el $L$-corredor $k$-cromático de ancho mínimo, y sea $m_i$ el elemento de $\mathcal{M}$ con mayor $d_p(m_i)$, entonces $m_i$ pertenece a $\vec{v_q}$, o a $\vec{h_q}$, en otras palabras, $m_i$ determina $C_q^-$. Vemos que para cualquier $m_i$, tenemos que $m_i$ está en $H$ o en $V$, por lo que $\mathcal{M}$ puede ser construido a partir de $H$ y $V$ en $O(k)$. Observemos que dado un cuadrante $C_p^+$, podemos encontrar los conjuntos $H$ y $V$ asociados a dicho cuadrante en tiempo lineal, lo cual, junto con la observación 2.3, da lugar inmediatamente a un algoritmo de tiempo $O(n^3)$.

Realicemos un barrido de linea, con una horizontal, parándonos en cada punto $r = (a, b)$ de $S$ en orden decreciente con respecto al valor de $b$, a cada parada de este barrido le llamaremos *parada horizontal*. Por cada parada horizontal, encontraremos el $\mathcal{C}_{p,q}$ $k$-cromático de ancho mínimo tal que $r \in \vec{h_p}$. Para esto visitaremos los puntos $p_i = (x_i, y_i)$ en $S$, tales que $b \leq y_i$ y $a \geq x_i$ en orden decreciente con respecto a $x$, nos referiremos a este recorrido como el *barrido vertical*. Supongamos sin pérdida de generalidad que estos puntos están etiquetados como $p_1, \ldots, p_t$, tales que si $i < j$ entonces $x_j < x_i$. Entonces cuando visitemos a $p_i$, podremos encontrar el $\mathcal{C}_{p,q}$ solución tal que $p = (x_i, b)$, claramente $r$ estará contenido en $\vec{h_p}$ y $p_i$ en $\vec{v_p}$, a este tipo de parada le llamaremos *parada vertical*.

Observemos que al pasar de $p_i$ a $p_{i+1}$, cada uno de los conjuntos $H$ y $V$ cambiará a lo más en un elemento con respecto a los que se tenían cuando visitamos a $p_i$, ya que sólo podrían diferir en $p_{i+1}$ y un elemento $e$ de alguno de los $H$ y $V$ de la parada anterior, tal que $e$ tenga el mismo color que $p_{i+1}$. Claramente $V$ siempre cambiará, y $H$ sólo en algunas ocasiones. De esta forma, $H$ y $V$, pueden ser actualizados en tiempo constante al pasar de $p_i$ a $p_{i+1}$. Nuestro problema ahora es que en cada parada vertical, $\mathcal{M}$ puede cambiar drásticamente con respecto a la parada anterior, figura 3, y recalcular $\mathcal{M}$ puede tomar $O(k)$. Esto induce un algoritmo de tiempo $O(n^2k)$ que da solución a este problema. Sin embargo, siendo más cuidadosos podemos mejorar esta complejidad.

Como ya se observó anteriormente, los elementos de $\mathcal{M}$ están en $V$ o en $H$, definimos los conjuntos $V' = V \cap \mathcal{M}$ y $H' = H \cap \mathcal{M}$. Hagamos la siguiente observación con respecto a los elementos de $V'$. Consideremos la parada vertical en la que visitamos al punto $p_i$, claramente $p_i$ entrará a $V'$. Digamos que $p_i$ tiene color $c$, entonces si en la parada anterior teníamos otro elemento de color $c$ en $V'$, éste tendrá que salir y no volverá a formar parte de $V'$. Ahora sea $p_j$ el elemento de color $c$ en $H$ y

supongamos que en las siguientes paradas no habrá otro punto de color $c$ a visitar. Claramente, dado que $p_i \in V'$, tenemos que $d_p(p_i) \leq d_p(p_j)$. Sin embargo, conforme continúe nuestro barrido vertical, $d_p(p_i)$ se irá incrementando hasta que eventualmente $d_p(p_i) > d_p(p_j)$, por lo que $p_i$ no podrá estar en $V'$ y entonces $p_j$ formará parte de $H'$. Claramente $p_i$ no podrá volver a ser parte de $V'$. Esto lo podemos resumir en la siguiente observación:

**Observación 2.4.** Sea $p_i$ cualquier elemento de $S$. Entonces $p_i$ entrará y saldrá de $V'$ a lo más una vez.

Sea $m \in \mathcal{M}$ tal que $d_p(m)$ es máxima, esto es, el elemento de $\mathcal{M}$ que define a $C_q^-$. Es fácil ver que $m$ será el elemento de $V'$ más alejado de $\vec{v_p}$ o el elemento de $H'$ más alejado de $\vec{h_p}$. Claramente $m$ es el único elemento de $\mathcal{M}$ que nos interesa para nuestra solución, el cual lo podemos decidir a partir de $H'$ y $V'$. Estas observaciones son la clave para nuestro algoritmo.

Para facilitar nuestra exposición, supongamos que $k$ es potencia de 2. Representemos a $H'$ como una cola de prioridad en forma de un árbol binario balanceado con $k$ hojas, donde las hojas serán los elementos de $H$, tal que la $i$-ésima hoja será el elemento de color $c_i$ en $H$. La llave de cada hoja será su distancia a $\vec{h_p}$. Además las hojas podrán ser marcadas como activas o inactivas, donde el estar activa significa que dicho elemento está en $\mathcal{M}$ y por tanto en $H'$. Entonces en cada nodo $v$ del árbol tendremos la llave de su descendiente activo con llave de mayor tamaño. De esta forma la raíz contendrá al elemento de $H'$ más alejado de $\vec{h_p}$. Se sabe que una estructura de este tipo, se puede actualizar haciendo recorrido desde la hoja modificada a la raíz, lo cual toma $O(\log k)$.

Análogamente podemos representar a $V'$ como una cola de prioridad, sólo que ahora las hojas serán los elementos de $V$. Nótese que la distancia de un elemento de $V$ a $\vec{v_p}$ aumentará conforme avanza el barrido vertical, sin embargo, lo que nos interesa es tener en la raíz el elemento de $V'$ más alejado de $\vec{v_p}$, el cual será el de coordenada $x$ mayor, por lo que ahora las llaves de los elementos de $V'$ serán sus coordenadas $x$, ver figura 4. De esta forma, cuando visitemos a $p_i$ en el barrido vertical, $p_i$ entrará en $V'$, por lo que habrá que actualizar a $V'$ y posiblemente también a $H'$, ya que $p_i$ podría ser un elemento de $H$, que al entrar en $V'$, haga salir un elemento de $H'$.

Consideremos el momento del barrido vertical en el que nos movemos de $p_{i-1}$ a $p_i = (x_i, y_i)$. Sea $c_j$ el color de $p_i$. Supongamos que el $\mathcal{M}$ de la parada actual sólo difiere al de la parada anterior en $p_i$ y otro punto de color $c_j$. Sea $h_j$ el elemento en $H$ de color $c_j$ y sea $d$ la distancia de $h_j$ a $\vec{h_p}$. Como ya se observó anteriormente, si continuamos el barrido vertical y no ha entrado un nuevo elemento de color $c_j$, entonces llegará un momento a partir del cual $p_i$ desaparecerá de $V'$ y $h_j$ formará parte de $H'$. Ese instante lo podemos reconocer en el momento que entra $p_i$ a $V'$, ya que será cuando la coordenada $x$ del punto de parada en el barrido vertical sea menor que $x_i - d$.

Veremos que podemos actualizar $H'$ y $V'$ en el momento adecuado. Construiremos otra cola de prioridad $E$, de forma similar a las usadas para $H'$ y $V'$, donde almacenaremos los eventos en los que sale un elemento de $V'$ y entra uno a $H'$. Por cada color $c_i$, tendremos un elemento $v_i = (x_i, y_i)$ de $V$ y un elemento $h_i$ en $H$, tales que $v_i$ y $h_i$ tienen el mismo color. Sea $d_i$ la distancia de $h_i$ a $\vec{h_p}$. Entonces

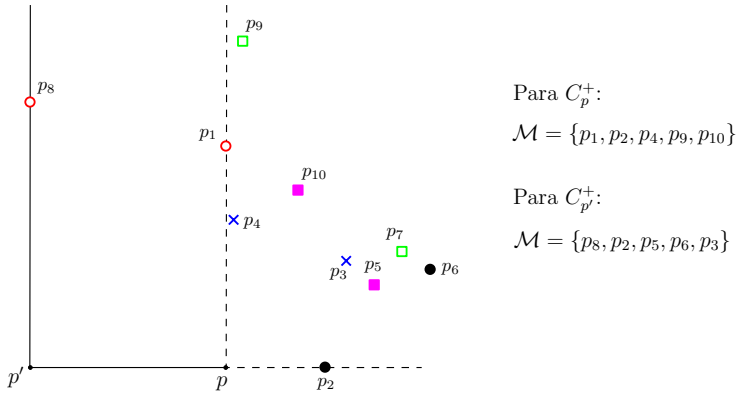



Figura 3: Un caso en el que $\mathcal{M}$ puede cambiar drásticamente respecto al paso anterior.

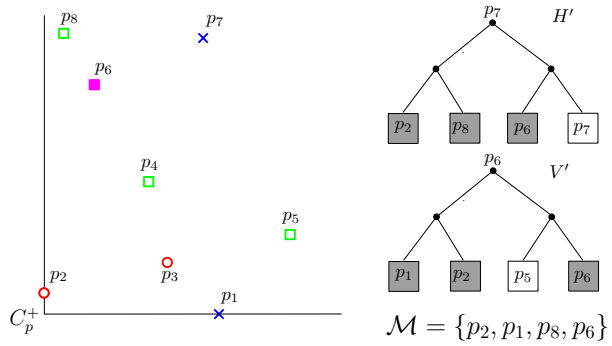



Figura 4: Las colas de prioridad correspondientes a $H'$ y $V'$.

cada hoja representará a uno de los $k$ colores, y su llave será el valor de $x_i - d_i$ correspondiente a dicho color y estará activa si $v_i$ está en $V'$, de modo que la raíz contendrá la coordenada $x$ más próxima en el barrido, a partir de la cual algún $v_i$ dejará de estar en $V'$.

De este modo, antes de avanzar a la siguiente parada del barrido verificaremos si esta parada ocurre antes o después que el elemento máximo de $E$, esto es, si el $p_i$ de la siguiente parada tiene coordenada $x$ menor o mayor que el valor contenido en la raíz de $E$. Si está antes, entonces podemos continuar con la siguiente parada de forma normal, de lo contrario, habrá que sacar algún elemento de $V'$ y agregar otro a $H'$, y posteriormente actualizar $E$. Repetimos esto hasta que la coordenada $x$ de $p_i$ sea mayor al valor contenido en la raíz de $E$. Por las observaciones anteriores, ésto puede suceder a lo más un número lineal de veces al barrer una vertical de $p_1$ a $p_t$.

Entonces reconocemos tres tipos de eventos que nos pueden ocurrir por cada punto en $S$ durante una parada horizontal. Primero, que dicho punto entre en $H$, lo cual implica actualizar $H'$; que dicho punto entre en $V$, entonces habría que actualizar $V'$, $E$ y posiblemente $H'$; y finalmente que dicho punto salga de $V'$, por lo que habría que actualizar las tres colas de prioridad. Vemos que estos eventos tienen un costo de $O(\log k)$ operaciones, y es fácil ver que cada uno de estos eventos ocurre a lo más una vez por punto en cada parada horizontal. De esto se sigue que el costo total de cada parada horizontal es de $O(n \log k)$, además de lo que nos toma construir las colas de prioridad al inicio de la parada horizontal. Esto fácilmente puede ser hecho en tiempo $O(n)$, por lo que el costo total de cada parada horizontal es de $O(n \log k)$. Dado que tenemos un número lineal de paradas horizontales podemos enunciar el siguiente resultado:

**Teorema 2.5.** *Dado un conjunto $S$ de $n$ puntos en el plano, $k$-coloreado y en posición general, podemos encontrar un $L$-corredor $\mathcal{C}_{p,q}$ $k$-cromático de ancho mínimo en tiempo $O(n^2 \log k)$.*

# 3 Encontrando el $L$-corredor $k$-cromático que minimiza $w_x + w_y$

En esta sección proponemos un algoritmo para dar solución al siguiente problema: dado $S$ un conjunto de $n$ puntos en el plano, en posición general y $k$-coloreado, obtener un $L$-corredor $\mathcal{C}_{p,q}$ $k$-cromático, con respecto a $S$, de tal forma que $w_x + w_y$ sea mínima.

Supongamos que $C_{p,q}$ es un $L$-corredor $k$-coloreado, tal que la suma $w_x + w_y$ es mínima. A diferencia del problema anterior, en este caso podemos observar lo siguiente:

**Observación 3.1.** Cada rayo $\vec{h_q}$ y $\vec{v_q}$ de $C_{p,q}$, contiene un punto de $S$, además, dichos puntos son distintos.

No obstante, es fácil notar que las observaciones acerca de $C_p^+$ siguen siendo válidas para este problema, por lo que podemos encontrar una solución para este problema usando la misma idea del barrido planteada en la sección anterior.

Para este problema definiremos a $H$ y $V$ de forma similar que en el problema anterior, sólo que ahora sus elementos estarán ordenados con respecto a su distancia a $\vec{h_p}$ y $\vec{v_p}$, respectivamente. De

nuevo $H$ y $V$ cambiarán, con respecto a la parada anterior en a lo más en un elemento y al manejarlos como listas ordenadas podrán ser actualizadas en tiempo $O(\log k)$. Ahora describiremos brevemente como podemos encontrar en tiempo $O(k)$, el $L$-corredor $\mathcal{C}_{p,q}$ $k$-cromático que minimice $w_x + w_y$, a partir de $H$ y $V$.

Supongamos entonces que $\mathcal{C}_{p,q}$ es el $L$-corredor que buscamos. Sean $e$ y $e'$ puntos por los que pasan $\vec{h_q}$ y $\vec{v_q}$ respectivamente, debido a la observación 3.1, $e \neq e'$. Claramente $e \in V$ y $e' \in H$, y ningún elemento que esté antes que $e$ en el orden de $V$ tiene el mismo color que $e'$. De igual forma ningún elemento que esté antes que $e'$ en el orden de $H$, tiene el mismo color que $e$. Es fácil ver que si no existen dos puntos con estas propiedades, entonces el $L$-corredor buscado no existe, o no minimiza $w_x + w_y$. Claramente para cada elemento $e$ de $V$ podemos encontrar su correspondiente $e'$ en tiempo $O(k)$. No es difícil ver que si procesamos los elementos de $V$ en orden, podemos calcular para todos los elementos $e \in V$ su correspondiente $e' \in H$ en tiempo amortizado $O(k)$.

Por tanto tenemos:

**Teorema 3.2.** *Dado un conjunto $S$ de $n$ puntos en el plano, $k$-coloreado y en posición general, podemos encontrar un $L$-corredor $\mathcal{C}_{p,q}$ $k$-cromático que minimice $w_x + w_y$ en tiempo $O(n^2 k)$.*

# 4 $L$-Corredor $k$-cromático no orientado

En esta sección, damos un procedimiento para resolver el problema de minimizar la anchura de un $L$-corredor $k$-coloreado cuando la orientación es una variable libre.

Al igual que en el caso orientado, es fácil ver que siempre existe un corredor óptimo que contiene un punto en cada semirecta de la frontera exterior ($\vec{h_p}$ y $\vec{v_p}$ en las secciones anteriores), o bien un punto $p$ en la intersección de las dos semirectas, al que llamamos ápice. Aquí explicamos, sin entrar en detalle, un algoritmo eficiente para encontrar el corredor no orientado óptimo que contiene un punto en cada semirecta exterior. El segundo caso es, de hecho, un caso degenerado y más simple que éste.

Sean $\vec{x}$ y $\vec{y}$ dos rectas ortogonales y orientadas. Definimos el cuadrante como la intersección del semiplano izquierdo de $\vec{x}$ (digamos $H_x$) y de $\vec{y}$ ($H_y$). Observemos que si la frontera exterior del cuadrante pasa por dos puntos $p$ y $q$, su ápice puede estar en cualquier punto del semicírculo con extremos en $p$ y $q$, ver Figura 5.

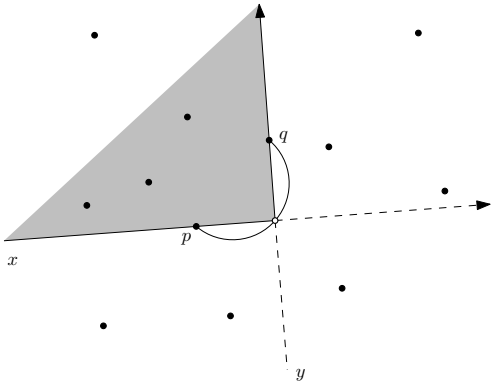



Figure 5: Cuadrantes con dos puntos fijos en la frontera.

Básicamente, lo que nos interesa hacer es un barrido con un cuadrante anclado en $p$ y $q$, lo cual es equivalente a mover el ápice sobre el semicírculo desde $p$ hasta $q$. Lo que necesitamos saber, en cada instante y para cada uno de los colores, es el punto en el interior del cuadrante más cercano a la frontera exterior de éste. Conocido ésto, la solución estará dada por el color que minimice la distancia al color más lejano en todo el recorrido.

A continuación indicamos, brevemente, cómo podemos resolver este problema usando la dualidad geométrica punto-recta.

Dado un punto $p$ y una recta $\ell$, denotamos como $p*$ y $\ell*$ sus duales correspondientes. Dado el arreglo de rectas del espacio dual, mover el ápice del cuadrante sobre el semicírculo es igual a barrer el espacio dual con dos rayos verticales, los cuales parten de $\vec{x}*$ y $\vec{y}*$ y cuya distancia horizontal es exactamente $\pi/2$.

La idea clave para dar un algoritmo eficiente es que, en lugar de hacer un barrido con dos lineas verticales en el dual, usaremos una subestructura, que no es más que una envolvente inferior de un arreglo de semirectas que contiene la información que nos interesa, es decir, cuál es el punto más cercano de cada color en todo el intervalo de barrido.

Claramente un punto $s \in S$ entra y sale del cuadrante una única vez. Entonces, podemos construir un arreglo de semirectas $A_i^x$, para cada color $i$, de la siguiente forma. Para cada $s$, tomaremos el rayo de $s*$ compuesto por los puntos de la forma $(a, b)$, tales que cuando $\vec{x}$ tiene pendiente $a$, $s$ está dentro del cuadrante. Claramente nuestro arreglo contendrá $O(n)$ rayos y el punto en el interior del cuadrante, cuando $\vec{x}$ tiene pendiente $a$, tal que su distancia a $\vec{x}$ es mínima, está dado por la envolvente inferior de tal arreglo. Análogamente podemos construir el arreglo $A_i^y$. Nuestro objetivo es superponer los dos subarreglos $A_i^x$ y $A_i^y$, de forma que la envolvente inferior del arreglo resultante nos proporcione el punto más cercano a la frontera exterior del cuadrante en cada instante del barrido. Para ello, hay que hacer con cuidado una transformación, obteniendo los espacios duales con $p$ y $q$ en el origen de coordenadas de forma independiente y sobreponer los arreglos correspondientes, dando lugar a un nuevo arreglo asociado a la frontera del cuadrante, que denotamos como $A_i$. De esta forma, para una orientación $a$ de $\vec{x}$, si $s$ es el punto de color $i$ más cercano a la frontera del cuadrante, entonces, el punto de $s*$ con abscisa $a$, será parte de la envolvente inferior de $A_i$.

Es fácil ver que la envolvente inferior de cada arreglo $A_i$, $L(A_i)$, $i = 1, \ldots, k$, tiene complejidad lineal y ésta puede calcularse en tiempo $O(n \log n)$ [7]. De esta forma, considerando el arreglo $\mathcal{A}$, compuesto por las envolventes inferiores $L(A_i)$ para todos los colores $i = 1, \ldots, k$, podemos obtener la solución a nuestro problema al determinar su envolvente superior.

El arreglo $\mathcal{A}$ puede verse como un arreglo con un número lineal de segmentos y puede calcularse en tiempo $O(n \log n)$. Por tanto, ejecutando el algoritmo anterior para cada pareja de puntos en $S$, podemos establecer el siguiente resultado:

**Teorema 4.1.** *Dado un conjunto $S$ de $n$ puntos en el plano, $k$-coloreado y en posición general, podemos calcular el $L$-corredor $k$-cromático no orientado con menor ancho en tiempo $O(n^3 \log n)$.*

# Referencias


[1] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop y V. Sacristán. *Smallest Color-Spanning Object*. Lecture Notes in Computer Science, 2161:278-289, 2001.

[2] S. W. Cheng. *Widest empty L-shaped corridor*. Information Processing Letters, 58:277-283, 1996.

[3] S. Das, P. P. Goswami y S. C. Nandy. *Recognition of Minimum Width Color-Spanning Corridor and Minimum Area Color-Spanning Rectangle*. Lecture Notes in Computer Science, 3480:827-837, 2005.

[4] G. K. Das, D. Mukhopadhyay y S. C. Nandy. *Improved Algorithm for a Widest 1-Corner Corridor*. Lecture Notes in Computer Science, WALCOM '09: Proceedings of the 3rd International Workshop on Algorithms and Computation, 5431:83-92, 2009.

[5] J.M. Díaz-Bañez, M. A. López y J. A. Sellarès . *On finding a widest 1-corner corridor*. Information Processing Letters, 98:199-205, 2006.

[6] D. P. Huttenlochar, K. Kadem y M. Sharir. *The upper envelope of Voronoi surfaces and its application*. Discrete and Computational Geometry, 8:267-291, 1993.

[7] M. Sharir and P.K. Agarwal *Davenport-Shinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

# Measuring the error of linear separators on linearly inseparable data

Boris Aronov * Delia Garijo † Yurai Núñez-Rodríguez‡ David Rappaport §
Carlos Seara ¶ Jorge Urrutia‖

**Abstract**

Given a set of red points $R$ and blue points $B$ we seek to determine a linear separator of the sets, although the sets may be linearly inseparable. Thus we obtain a partition into two parts, where each part may contain some misclassified points. We determine the error of the partition based on the amount of work needed to move the misclassified points. We consider several different measures of work and provide algorithms to find linear separators that minimize the error under these different measures.

## 1 Introduction

Let $R = \{r_1, \ldots, r_p\}$ be a set of $p$ red points and $B = \{b_1, \ldots, b_q\}$ a set of $q$ blue points in $\mathbb{R}^d$. Let $n = p+q$ and assume that the points are *disjoint* and in general position, that is, no $d+1$ of the points lie in the same hyperplane in $\mathbb{R}^d$. We say that $R$ and $B$ are *linearly separable* if there is a hyperplane, a *linear separator*, that partitions $\mathbb{R}^d$ such that each part contains only red or only blue points. If there is no linear separator for $R$ and $B$, then we say that the sets are *linearly inseparable*. Given sets $R$ and $B$ we attempt to find a linear separator, and if no such separator exists we would like to produce an *approximate separator* with the "best" approximation. The notion of "best" is left intentionally informal as the precise properties that should be optimized are application dependent. In this paper we will examine several different criteria for choosing an approximate separator.

Let $P = \{p_1, \ldots, p_n\} = R \cup B$. Let $H$ denote a hyperplane which misclassifies $P$ and partitions $P$ into two non-empty subsets. It will be useful to call one of the subsets the *left* subset and say that red points in it are *well classified* and blue points in it are *misclassified*. This language is hardly a loss of generality as we can always just label the majority color in some side red, and name that side the *left side*. The other subset is refereed to as *right* and plays a complementary role. Given $P$ and $H$ we may be left with a set, $\Xi$, the subset of $P$ that is misclassified by $H$, that is, $H$ is an approximate separator. We use $s(H)$ to represent the quality of $H$ as an approximate separator, a cost of $H$. Our goal is to find a hyperplane that minimizes the cost under one of the following assumptions.

1. $s(H)$ is the maximum Euclidean distance from $H$ to a point in $\Xi$: MinMax criterion.
2. $s(H)$ is the sum of the Euclidean distances from $H$ to every point in $\Xi$: MinSum criterion.
3. $s(H)$ is the sum of squares of the Euclidean distances from $H$ to every point in $\Xi$: MinSum$^2$ criterion.
4. $s(H)$ is simply the cardinality of $\Xi$: MinMis criterion.

*Dept. of Computer and Information Science, Polytechnic University, USA, aronov@cis.poly.edu.
†Depto. de Matemática Aplicada I, Universidad de Sevilla, dgarijo@us.es.
‡School of Computing, Queen's University, Kingston, Canada, yurai@cs.queensu.ca.
§School of Computing, Queen's University, Kingston, Canada, daver@cs.queensu.ca.
¶Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, carlos.seara@upc.edu.
‖Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx.

# 2 MinMax

In this section we concentrate in what may be considered as the simplest version of the problem as our solutions are straightforward and succinct. We will also be setting a pattern that we follow in all subsequent versions. We first consider the problem in $\mathbb{R}$. In each case the one-dimensional algorithm for finding an optimal approximate separator offers intuitive insight to solutions in higher dimensions. This phenomenon is easily explained by the following observation.

**Observation 2.1.** Suppose we have obtained an optimal approximate separator $H$ for $P$ in $\mathbb{R}^d$. Let $P^-$ ($H^-$) denote a projection orthogonal to $H$ of $P$ ($H$) to $\mathbb{R}^{d-1}$. Maintaining the same notation, repeatedly project into a lower dimension until we get to $\mathbb{R}$. Then $H^-$ is an optimal approximate separator of $P^-$ and furthermore $s(H) = s(H^-)$.

Thus every optimal solution in higher dimensions has an equivalent one-dimensional solution, but the number of possible candidate solutions that we can evaluate to determine an optimal one usually increases with the dimension, i.e., bigger dimension implies more possible candidates.

## 2.1 $d = 1$

**Observation 2.2.** An optimal MinMax approximate separator is the mean of the leftmost blue point and the rightmost red point and is found in $\Theta(n)$ time.

The observation is justified by the fact that the cost $s(H)$ is realized by the misclassified point that is furthest away from $H$. Placing $H$ at the mean of the extremes is obviously the cost minimizing approximate separator.

## 2.2 $d = 2$

As in the one-dimensional version of this problem the optimal approximate separator is the mean of extreme points. In $\mathbb{R}^2$ we must consider combinations of an extreme point of $R$ paired with an extreme and antipodal extreme point of $B$. We then pick the pair that are closest together. When closest antipodal extreme points come from the same set they determine the width of the set. In our case the pairs come from two different sets, but it is a routine matter to show that in $\mathbb{R}^2$ there are only $O(n)$ antipodal pairs that need to be examined. These points can be found using the well known algorithmic technique that is described as the method of *rotating calipers*. See for example Tousaint [12].

Thus the crux of our method examines antipodal pairs from $R$ and $B$. We begin by computing the convex hulls $\mathrm{CH}(B)$ and $\mathrm{CH}(R)$ represented as the lists of the extreme points of $B$ and $R$ respectively stored in order around the boundary of the corresponding convex hulls. There are many well known convex hull algorithms for a set of $n$ points and the most efficient use $O(n \log h)$ time where $h$ represents the number of extreme points of the set. See for example Chan's algorithm [2]. Once we have the hulls we pick fixed antipodal directions for $\mathrm{CH}(B)$ and $\mathrm{CH}(R)$, and then systematically march around each of the boundaries. As we never backtrack along either of the hulls, we make the full revolution in $O(n)$ steps. At each step we maintain the distance between the antipodal points, and when we complete a full revolution we have the overall minimum antipodal distance. Thus we have the following observation.

**Observation 2.3.** We can obtain a MinMax optimal approximate separator in $O(n \log n)$ time, using $O(n)$ space. If we use $h$ to denote the sum of the number of the blue and red extreme points then we can say that the complexity is in $O(n \log h)$ time.

When the number of extreme points of $R$ and $B$ are in $\Omega(n)$ then we can demonstrate that $\Omega(n \log n)$ is a lower bound for find a best MinMax optimal approximate separator. We use a reduction from the Max–Gap problem for points on the first quadrant of the unit circle [11] which is known to have an $\Omega(n \log n)$ lower bound in the algebraic computation tree model.

### 2.3 $d \geq 3$

The minimum width problem for a set of $n$ points $P$ in $\mathbb{R}^3$ can be solved in $O(m+n\log n)$ time and $O(n)$ space, where $m$ is the number of antipodal edge-edge pairs of $CH(P)$ (see Houle and Toussaint [9]). Houle and Toussaint perform an exhaustive analysis and show that there are polyhedra that have $\Theta(n^2)$ antipodal edge-edge pairs. Thus it follows that with separate polyhedra with a sum total of $n$ edges, we may have $\Theta(n^2)$ edge pairs as well. Thus the search space for determining an antipodal pair with minimal distance is in $\Theta(n^2)$, and by using the techniques developed by Houle and Toussaint we can obtain an optimal approximate MinMax separator in $O(m+n\log n)$ time. With respect to the problem of computing the width of a set of points, a series of papers derived improved sub-quadratic algorithms, the best of which requires $O(n^{3/2+\delta})$ expected time (see [3]). For $d \geq 4$, the $O(n^{\lceil d/2 \rceil})$ time algorithm can be achieved by realizing the solution space as a convex polytope in $d+1$ variables/dimensions, and applying an optimal half-space intersection algorithm.

## 3 MinSum

In this section we study the problem of minimizing the sum of distances of those points that have to be moved for $R$ and $B$ to become linearly separable, that is, using the MinSum criterium. Most of the proofs in dimension $d = 1$ can be extended for $d \geq 2$ as a consequence of Observation 2.1.

### 3.1 $d = 1$

Let $R$ and $B$ be inseparable sets of red and blue points on a line. We want to find the optimal separator $a$ according to the MinSum criterium. Assume that the red (blue) region is on the left (right) of $a$, and that $a$ is strictly between two consecutive points. The following lemma characterizes the solution of the MinSum problem.

**Lemma 3.1.** *An optimal separator point $a$ lies between two consecutive points of $R \cup B$ such that the number of misclassified blue points is equal to the number of misclassified red points.*

**Observation 3.2.** If there exists an optimal separator $a$ between consecutive points, say $s$ and $t$, then any point $p \in (s,t)$ is an optimal separator. In fact, we can take the closed interval $[s,t]$ treating $s$ as belonging to the left region and $t$ belonging to the right region.

**Lemma 3.3.** *Let $|R| = p$, $|B| = q$. The* MinSum *problem in $\mathbb{R}$ has infinitely many optimal solutions. An optimal separator is any point $a$ of the closed interval defined by the two consecutive points of $R \cup B$ in the positions $p$ and $p+1$ counting from left to right.*

**Theorem 3.4.** *The 1-dimensional* MinSum *problem can be solved in $\Theta(n)$ time.*

### 3.2 $d = 2$

Let $\ell$ be an the optimal separator line for $R$ and $B$ according to the MinSum criterion. Let $\ell^+$ ($\ell^-$) be the open half-plane above (below) $\ell$. The following two lemmas are straightforward consequence from Observation 2.1 and Lemmas 3.1 and 3.3.

**Lemma 3.5.** *The optimal direction to move the misclassified points is the orthogonal direction to $\ell$.*

**Lemma 3.6.** *The number of misclassified blue points on $\ell^+$ is equal to the number of misclassified red points on $\ell^-$. The 2-dimensional* MinSum *problem has an infinite number of optimal separators.*

Let $\ell := ax + y + b = 0$ and $p_i = (x_i, y_i) \in \ell^+$ ($p_j = (x_j, y_j) \in \ell^-$) a point of $P = R \cup B$ which satisfies $ax_i + y_i + b > 0$ ($ax_j + y_j + b < 0$). Assign a value $\delta_i$ to each point $p_i$, red or blue, such that $\delta_i = 0$ for each red point in $\ell^-$ (well classified) and $\delta_i = 1$ for each blue point in $\ell^-$ (misclassified). Analogously, assign $\delta_j = 0$ to each blue point in $\ell^+$ (well classified) and $\delta_j = 1$ to each red point in $\ell^+$

(misclassified). A red or blue point $p_i = (x_i, y_i)$ on $\ell$ satisfies $ax_i + y_i + b = 0$ and it is always consider well classified, i.e., $\delta_i = 0$. If the number of misclassified points on $\ell^-$ and $\ell^+$ are equal, then the sum of distances of the points of $P = R \cup B$ to $\ell$ is the function:

$$s(\ell) = \frac{1}{\sqrt{a^2+1}} \left( a(\sum(\delta_i x_i - \delta_j x_j)) + \sum(\delta_i y_i - \delta_j y_j) \right).$$

Let us write $A = \sum(\delta_i x_i - \delta_j x_j)$ and $B = \sum(\delta_i y_i - \delta_j y_j)$, which are constants for a fixed bipartition. Then, the function $s(\ell)$ only depends on the parameter $a$, the slope of $\ell$, so we write $s(\ell) = s(a)$.

$$s(\ell) = s(a) = \frac{Aa + B}{\sqrt{a^2+1}}, \qquad s'(a) = \frac{-Ba + A}{(a^2+1)^{3/2}}.$$

The value $a = A/B$ is a minimum of the function $s(a)$, so line $\ell_0 := A/Bx + y + b = 0$ is the optimal separator for the bipartition defined by $\ell$. This fixes the slope of $\ell_0$ but we still need to determine a value (or range of values) of $b$. The family of parallel lines $A/Bx + y + b = 0$, $b \in [b_1, b_2]$, is defined by the supporting lines with slope $A/B$ of the left and right part of the bipartition of $P$ produced by $\ell$.

Algorithm 1: 2d-MinSum-optimal-separator

1. Compute the $p$ and $(p+1)$-levels in $\mathcal{A}(P)$ and the sequence $(a_1, \ldots, a_{n_p-1})$. Let $e_i$ and $e'_i$ be the lines which intersection point is $a_i$.
2. Let $\ell_1 := \mathcal{D}(a_1)$. In $O(n)$ time do the following: (1) compute in the primal the bipartition $\{P_{1,1}, P_{1,2}\}$ of $P$ produced by $\ell_1$: $P_{1,1}$ is the subset of points of $P$ above $\ell_1$ including the point $\mathcal{D}(e_1)$, and $P_{1,2}$ is the subset of points of $P$ below $\ell_1$ including the point $\mathcal{D}(e'_1)$; (2) compute the function $s(a)$, the value $A/B$ which minimizes $s(a)$, check that $A/B \leq a_1$; and (3) do $m := A/B$, $s(m) := s(A/B)$, and $ind := 1$ the index of the cell $C_1$.
3. **For** $i := 2$ to $n_p$ (sweeping the $p$-level from left to right and stopping at $a_i$) **do**:
   (a) Update the changes of the function $s(a)$: only the two points $\mathcal{D}(e_{i-1})$ and $\mathcal{D}(e'_{i-1})$ interchange their half-planes in the bipartition $\{P_{i,1}, P_{i,2}\}$ given by $\ell_i := \mathcal{D}(a_i)$ with respect to the bipartition $\{P_{i-1,1}, P_{i-1,2}\}$ working with the neighbor cells $C_{i-1}$ and $C_i$. Thus, only two values of $\delta_j$ in the function $s(a)$ has to be updated.
   (b) Compute the new value $A/B$ which minimizes the function $s(a)$, and check that $a_{i-1} \leq A/B \leq a_i$ (if $i = n_p$ check that $a_{n_p} \leq A/B$). Compute the value $s(A/B)$. If $s(A/B) < s(m)$, then do $m := A/B$, $s(m) := s(A/B)$, and $ind := i$.
4. Let $m$, $s(m)$, and $ind$ be the current values. Compute the intersection points of the vertical line $x = m$ with the boundary of the cell $C_{ind}$. Let $(m, b_1)$ and $(m, b_2)$ be these intersection points. Any line of the family of parallel lines $\ell = mx + y + b = 0$, $b \in [b_1, b_2]$ is an optimal separator for $P$ and MinSum $= s(m)$.

**Theorem 3.7.** *The* 2*-dimensional* MinSum *problem can be solved deterministically in time* $O(n \log n + n^{4/3} \log^{1+\epsilon} n)$ *for an arbitrarily small constant* $\epsilon > 0$ *or in* $O(n \log n + n^{4/3})$ *expected time.*

## 3.3 $d = 3$

Let $\pi$ a plane which is an optimal separator for $P = R \cup B$ according to the MinSum criterion. By the same reasoning, an optimal separator plane $\pi$ with normal vector $\vec{v}$ can be translated along the direction given by $\vec{v}$ until it bumps into a point in $P$, and all these planes are optimal.

Let $\pi := ax + by + z + c = 0$ a plane with normal vector $(a, b, 1)$ and $p_i = (x_i, y_i, z_i) \in \pi^+$ ($p_j = (x_j, y_j, z_j) \in \pi^-$) a point of $P = R \cup B$ which satisfies $ax_i + by_i + z_i + c > 0$ ($ax_i + by_i + z_i + c < 0$). Let $\delta_i$ and $\delta_j$ be defined as in the subsection above depending on whether the point is well or misclassified. We recall that a point $p_i = (x_i, y_i, z_i)$ on $\pi$ satisfies $ax_i + by_i + z_i + c = 0$ and it is always consider well

classified, so $\delta_i = 0$. If the number of misclassified points in both half-spaces of $\pi$ are equal, the sum of distances between misclassified points of $P$ and the plane $\pi$ is the function:

$$s(\pi) = \frac{1}{\sqrt{a^2+b^2+1}} \left( a(\sum(\delta_i x_i - \delta_j x_j)) + b(\sum(\delta_i y_i - \delta_j y_j)) + \sum(\delta_i z_i - \delta_j z_j) \right).$$

Let us write $A = \sum(\delta_i x_i - \delta_j x_j)$, $B = \sum(\delta_i y_i - \delta_j y_j)$, and $C = \sum(\delta_i z_i - \delta_j z_j)$. These values are constants for a fixed bipartition. Then, the function $s(\pi)$ only depends on the parameters $a$ and $b$, e.i., it only depends on the normal vector $(a, b, 1)$ of $\pi$, so we write $s(\pi) = s(a, b)$.

$$s(\pi) = s(a, b) = \frac{Aa + Bb + C}{\sqrt{a^2+b^2+1}}.$$

Algorithm 2: 3D-MinSum-optimal-separator

1. Compute the $p$ and $(p+1)$-levels in $\mathcal{A}(P)$ and the sequence $(a_1, \ldots, a_{n_p})$. Let $e_i$, $e_i'$, and $e_i''$ be the three planes which intersection point is $a_i$.
2. Let $\pi_1 := \mathcal{D}(a_1)$. In $O(n)$ time compute in the primal the bipartition $\{P_{1,1}, P_{1,2}\}$ of $P$ produced by $\pi_1$: $P_{1,1}$ is the subset of points of $P$ in $\pi_1^+$ including the point $\mathcal{D}(e_1)$, and $P_{1,2}$ is the subset of points of $P$ in $\pi_1^-$ including the points $\mathcal{D}(e_2')$ and $\mathcal{D}(e_3'')$ (analogously for $\mathcal{D}(e_2)$ in $\pi_1^+$ and $\mathcal{D}(e_1')$ and $\mathcal{D}(e_3'')$ in $\pi_1^-$, and for $\mathcal{D}(e_3)$ in $\pi_1^+$, and $\mathcal{D}(e_1')$ and $\mathcal{D}(e_2'')$ in $\pi_1^-$, corresponding to the two neighbor cells sharing $a_1$). For each case compute the function $s(a, b)$ and the pair which minimizes $s(a, b)$. Let $(a_1, b_1)$ be this pair. Do $(m_1, m_2) := (a_1, b_1)$, $s(m_1, m_2) := s(a_1, b_1)$, and *ind* the index of the current cell.
3. **For** $i := 2$ to $n_p$ **do**:
   (a) Update the changes of the function $s(a, b)$: only the two points interchange their half-spaces in the bipartition $\{P_{i,1}, P_{i,2}\}$ given by $\pi_i := \mathcal{D}(a_i)$ with respect to the bipartition $\{P_{i-1,1}, P_{i-1,2}\}$ working with the neighbor cells. Thus, only two values of $\delta_j$ in the function $s(a, b)$ has to be updated.
   (b) Compute the pair $(a_i, b_i)$ which minimizes the function $s(a, b)$, compute the value $s(a_i, b_i)$. If $s(a_i, b_i) < s(m_1, m_2)$, then do $(m_1, m_2) := (a_i, b_i)$, $s(m_1, m_2) := s(a_i, b_i)$, and *ind* the index of the current cell.
4. Let $(m_1, m_2)$, $s(m_1, m_2)$, and *ind* be the current values. To check that $(m_1, m_2, 1)$ is a normal vector between the ones defined by the cell $C_{ind}$ cost the complexity of the cell. To determine the plane $\pi$ and the infinite solutions we compute the extreme points, $(m_1, m_2, c_1)$ $(m_1, m_2, c_2)$, in the the boundary of the cell $C_{ind}$ with the vertical line passing through the point $(m_1, m_2, 0)$. Any plane of the family of parallel planes $\pi = m_1 x + m_2 y + z + c = 0$, $c \in [c_1, c_2]$ is an optimal separator for $P$ and MinSum $= s(m_1, m_2)$.

**Theorem 3.8.** *The* 3*-dimensional* MinSum *problem can be solved in* $O(n^{5/2} \log^6 n)$ *expected time.*

## 3.4 $d \geq 4$

For dimension $d \geq 4$, the upper bound for the size of the $p$-level is only slightly better than the $O\left(n^{\lfloor d/2 \rfloor} p^{\lceil d/2 \rceil}\right)$ [5]. More concretely, an upper bound is $O(n^{d-\alpha_d})$ for a very small $\alpha_d = 1/(4d-3)^d$. As Agarval et al. [1] observed, the bound can be made sensitive to $p$, namely $O(n^{\lfloor d/2 \rfloor} p^{\lceil d/2 \rceil - \alpha_d})$. Matoušek et al. [10] give an $O(n^{4-2/45})$ upper bound for $d = 4$.

**Theorem 3.9.** *The* $d$*-dimensional* MinSum *problem can be solved deterministically in time*

$$O\left( n^{\lfloor d/2 \rfloor} p^{\lceil d/2 \rceil} \left( \frac{\log n}{\log p} \right)^{O(1)} \right) = O(n^d).$$

# 4 MinSum$^2$

We now consider the criterion of minimizing the sum of the squared distances of those points that have to be moved until $R$ and $B$ become linear separable, i.e., using the MinSum$^2$ criterion.

## 4.1 $d = 1$

Let $R$ and $B$ be two inseparable sets of red and blue points on the $x$-axis. Assume that an optimal separator point $a$ determines the red region on its left and the blue region on its right. It means that the misclassified red and blue points have to be moved to the left and to the right of $a$, respectively. Given an optimal separator point $a$, let $b_1, \ldots, b_k$ be the $x$-coordinates of the misclassified blue points and $r_1, \ldots, r_m$ the $x$-coordinates of the misclassified red points. The sum of the squared distances of the misclassified points to $a$ is given by

$$s(a) = (a - b_1)^2 + \cdots + (a - b_k)^2 + (r_1 - a)^2 + \cdots + (r_m - a)^2$$

$$s(a) = (k + m)a^2 - 2a\left(\sum_{i=1}^{k} b_i + \sum_{i=1}^{m} r_i\right) + \left(\sum_{i=1}^{k} b_i^2 + \sum_{i=1}^{m} r_i^2\right).$$

If we put

$$A = \sum_{i=1}^{k} b_i + \sum_{i=1}^{m} r_i, \qquad B = \sum_{i=1}^{k} b_i^2 + \sum_{i=1}^{m} r_i^2,$$

$s(a) = (k + m)a^2 - 2aA + B$, and $s'(a) = (k + m)a - 2A = 0$ which implies that $a = \frac{A}{k+m}$. This value is a minimum since there is at least one misclassified point and so $s''(a) = k + m > 0$. Observe that

$$a = \frac{A}{k + m} = \frac{\sum_{i=1}^{k} b_i + \sum_{i=1}^{m} r_i}{k + m}$$

is the arithmetic median of the misclassified points. Thus, the value $a$ gives us a measure of how much the points are misclassified with respect to their arithmetic median. A two-dimension interpretation is the regression line of the misclassified points.

Let $S = (s_1, \ldots, s_n)$ denote $P = R \cup B$ with $n = p + q$, where the points are ordered from left to right. We use *interval* $i$ to denote the interval between $s_i$ and $s_{i+1}$. As a shorthand it will be convenient to use $\Xi_i$ to denote the set of misclassified points and $N_i$ the number of misclassified points when the separator is in interval $i$. The average value of the misclassified points at interval $i$ is

$$a_i = \frac{1}{N_i} \sum_{s \in \Xi_i} s.$$

We say that an average $a_i$, is *correct* if $s_i < a_i \leq s_{i+1}$. We prove the following theorem.

**Theorem 4.1.** *There is a unique correct average that defines an optimal solution for the one dimensional* MinSum$^2$ *problem, and it can be found in linear time.*

## 4.2 $d = 2$

Let $\ell := ax + y + b = 0$ be the optimal separator line according to the MinSum$^2$ criterion. Let $p_i = (x_i, y_i)$ ($p_j = (x_j, y_j)$) be a point of $R \cup B$ in $\ell^+$ ($\ell^-$) which satisfies $ax_i + y_i + b > 0$ ($ax_j + y_j + b < 0$). Assign a value $\delta_i$ to each point $p_i$, red or blue, such that $\delta_i = 0$ for each red point in $\ell^+$ (well classified) and $\delta_i = 1$ for each blue point in $\ell^+$ (misclassified). Analogously, assign $\delta_j = 0$ to each blue point in $\ell^-$ (well classified) and $\delta_j = 1$ to each red point in $\ell^-$ (misclassified). A point $p_i = (x_i, y_i)$ on $\ell$ satisfies $ax_i + y_i + b = 0$ and it is consider well classified, i.e., $\delta_i = 0$. The sum of squared distances between the points of $R \cup B$ and the line $\ell$ is the function:

$$s(\ell) = \sum_{p_i \in \ell^+} \delta_i \frac{(ax_i + y_i + b)^2}{a^2 + 1} + \sum_{p_j \in \ell^-} \delta_j \frac{(ax_j + y_j + b)^2}{a^2 + 1} =$$

$$= \frac{1}{a^2+1}\left(\sum_{p_i \in R \cup B} \delta_i(a^2x_i^2 + y_i^2 + b^2 + 2ax_iy_i + 2abx_i + 2by_i)\right).$$

Let us write $A = \sum \delta_i x_i^2$, $B = \sum \delta_i y_i^2$, $C = \sum \delta_i$, $D = \sum \delta_i x_i y_i$, $E = \sum \delta_i x_i$, and $F = \sum \delta_i y_i$ which are constants for a given partition. Thus, $s(\ell)$ only depends on $a$ and $b$, so we write $s(\ell) = s(a, b)$,

$$s(a,b) = \frac{Aa^2 + B + Cb^2 + 2Da + 2Eab + 2Fb}{a^2+1}.$$

PROCEDURE: MinSum$^2$-OPTIMAL-SEPARATOR

1. Dualize the point set $P = R \cup B$ as red and blue lines obtaining the arrangement $\mathcal{A}(P)$.
2. Visit all the cells in $\mathcal{A}(P)$ and for on each cell do the following calculations:
   (a) Update the changes of the function $s(a, b)$: only a point change in the bipartition $\{P_1, P_2\}$, thus update two values of $\delta_i$.
   (b) Compute the the pair $(a_1, b_1)$ which minimizes $s(a, b)$. Check that the point obtained by dualizing the line $\ell_1 := a_1x + y + b_1 = 0$ is inside the current cell.
   (c) Compute and update the minimum value of $s(a, b)$ and its corresponding optimal separator $\ell := ax + y + b = 0$.

**Theorem 4.2.** *The two-dimensional* MinSum$^2$ *problem can be solved in* $O(n^2)$ *time.*

For $d \geq 3$, similar computations can be extended obtaining an $O(n^d)$ time algorithm.

# 5 MinMis

Another way of thinking into reach linear separability is to delete the misclassified points, i.e., compute the minimum number of points that have to be deleted in order to get the linear separability of the remaining red and blue points. This problem is equivalent to the problem of computing an optimal separator for $B$ and $R$ which minimize the sum of misclassified points [7].

## 5.1 $d = 1$

Let $R$ and $B$ be set of red and blue points on a line, respectively. In order to compute the separator $a$ given the minimum number of misclassified points, i.e., the separator $a$ that minimize the sum $s$ of misclassified points in both sides, we sort the points in $O(n \log n)$ time, obtained a linear number of intervals defined by two consecutive points. Any point in an interval gives the same value of $s$. Thus do a sweep from left to right updating the value $s$ in constant time each time we a point (red or blue) changes from right to left of the current $a$. Notice that for a given separator $a$ we have $r$ red points and $b$ blue points on its left and $n - r$ red points and $n - b$ blue points on its right. It is clear that we get an overall $O(n \log n)$ time algorithm for computing the optimal separator (either point or interval). We prove that this algorithm is optimal using a reduction to the $\epsilon$-distance problem for points on a line which has an $\Omega(n \log n)$ time lower bound in the algebraic decision tree model [11].

**Theorem 5.1.** *The one-dimensional* MinMis *problem can be solved in optimal* $\Theta(n \log n)$ *time.*

## 5.2 $d = 2$

For the two-dimensional problem we can proceed as for the 2-dimensions MinSum$^2$ problem counting the number of misclassified points in both sides of the line, updating the minimum of the sum of them. Houle [8] gave an $O(n^2)$ time algorithm for this problem. For a given $k$, Cole, Sharir and

Yap [6] presented an $O(nk \log n)$ time algorithm to compute the separators misclassifying at most $k$ points. Everett et al. [7] improve upon these results if the number of misclassified points is not too large, getting a $O(nm \log m + n \log n)$ time algorithm, where $m$ is the number of misclassified points. Chan [4] present an algorithm that find a line $\ell$ that minimizes $k$, the total number of red points above $\ell$ and the blue points below $\ell$, in $O((n + k^2) \log n)$ expected time and $O(n + k^2)$ space.

**Theorem 5.2.** *The two-dimensional* MinMis *problem can be solved in* $O(n^2)$ *time.*

For $d \geq 3$ we can proceed as for the 2-dimensions MinSum$^2$ problem counting the number of misclassified points in both sides of the hyperplane. Thus, the $d$-dimensional MinMis problem can be solved in $O(n^d)$ time.

Table 1: Summary of results.

| Dimension | MinMax | MinSum | MinSum$^2$ | MinMis |
|---|---|---|---|---|
| $d = 1$ | $\Theta(n)$ | $\Theta(n)$ | $\Theta(n)$ | $\Theta(n \log n)$ |
| $d = 2$ | $\Theta(n \log n)$ | $O(n \log n + n^{4/3} \log^{1+\epsilon} n)$<br>$O(n \log n + n^{4/3})$ $(*)$ | $O(n^2)$ | $O(n^2)$ |
| $d = 3$ | $O(n^2)$ | $O(n^{5/2} \log^6 n)$ $(*)$ | $O(n^3)$ | $O(n^3)$ |
| $d \geq 4$ | $O(n^{\lfloor d/2 \rfloor})$ | $O(n^d)$ | $O(n^d)$ | $O(n^d)$ |

$(*)$ expected time

# References


[1] P. K. Agarwal, B. Aronov, T. M. Chan, and M. Sharir. On levels in arrangements of lines, segments, planes, and triangles. 2001

[2] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete and Computational Geometry*, Vol. 16,1996, pp.361–368.

[3] T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. *Sixteenth Annual Symposium on Computational Geometry*, 2000, pp. 300–309.

[4] T. M. Chan. Low-Dimensional Linear Programming with Violations. 2004. A preliminary version of this paper appeared in *Proc. 43rd IEEE Sympos. Found. Comput. Sci.*, 2002.

[5] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4, 1989, pp. 387–421.

[6] R. Cole, M. Sharir, and C. K. Yap. On $k$-hulls and related problems. *SIAM J. Comput.*, 16, 1987, pp. 61–77.

[7] H. Everett, J.-M. Robert, and M. van Kreveld. An optimal algorithm for the $(\leq k)$-levels, with applications to separation and transversal problems. *International Journal of Computational Geometry and Applications*, 6, 1996, pp. 247–261.

[8] M. E. Houle. Algorithms for weak and wide separation of sets. *Discrete Applied Mathematics*, Vol. 45, N. 2, 1993, pp. 139-159.

[9] M. E. Houle and G. T. Toussaint. Computing the width of a set. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 761–765, 1988.

[10] J. Matoušek, M. Sharir, S. Smorodinsky, and U. Wagner. On $k$-sets in four dimension. Manuscript, 2004.

[11] V. Sacristán. Lower bounds for some geometric problems. *Technical Report MA-IR-98-0034, Universitat Politècnica de Catalunya*, Departament de Matemàtica Aplicada II, 1998.

[12] G. T. Toussaint. Solving geometric problems with the rotating calipers. *Proceedings of IEEE MELECON'83*, Athens, Greece, May 1983, pp. A10.02/1–4.

# Bichromatic Discrepancy via Convex Partitions

S. Bereg * J.M. Díaz-Báñez † D. Lara ‡ P. Pérez-Lantero § C. Seara ¶
J. Urrutia ‖

**Abstract**

Let $R$ be a set of red points and $B$ a set of blue points on the plane. In this paper we introduce a new concept for measuring how mixed the elements of $S = R \cup B$ are. The discrepancy of a set $X \subset S$ is $||X \cap R| - |X \cap B||$. We say that a partition $\Pi = \{S_1, S_2, \ldots, S_k\}$ of $S$ is convex if the convex hulls of its members are pairwise disjoint. The discrepancy of a convex partition of $S$ is the minimum discrepancy of the sets $S_i$. The discrepancy of $S$ is the discrepancy of the convex partition of $S$ with maximum discrepancy. We study the problem of computing the discrepancy of a bichromatic point set. We divide the study in general convex partitions for both general set of points and points in convex position, and also when the partition is given by a line. In this case we prove that this problem is 3SUM-hard.

## 1 Introduction

In this paper we study a parameter that measures how *mixed* two point sets are. In the rest of this paper $S = R \cup B$ will always be a set of points on the plane in general position whose elements are colored either red (the elements of $R$), or blue (the elements of $B$). We will also assume that $R$ and $B$ are non-empty and have $r$ and $b$ elements respectively. Intuitively speaking $R$ and $B$ are *well mixed* if for any convex region $C$ of the plane the proportion of red elements of $S$ is aproximately $\frac{r}{b+r}$, e.g. if $r = 2b$, we would expect $C$ to contain twice as many red points as blue. It is clear that we must be careful on how we define well mixed sets of points, as since $S$ is in general position, we can always find many convex sets containing only two points of $S$ with the same color. In this paper we introduce a parameter that seems like a good candidate to measure how well mixed a bicolored point set is, we call this parameter the *discrepancy* of $S$.

For any set of points $P$ on the plane, let $CH(P)$ denote the convex hull of $P$. Let $S = R \cup B$ be a bicolored point set, and $X \subset S$. The discrepancy of $X$ is defined as $\nabla(X) = ||X \cap R| - |X \cap B||$. We say that a partition $\Pi = \{S_1, S_2, \ldots, S_k\}$ of $S$ is a *convex partition* if $CH(S_i) \cap CH(S_j) = \emptyset$ for all $1 \leq i < j \leq k$. The *discrepancy of $S$ with respect to* $\Pi$ is defined as $d(S, \Pi) = \min_{i=1..k} \nabla(S_i)$. The *discrepancy of* $S$, $d(S)$, is defined as the largest $d(S, \Pi)$ over all the convex partitions $\Pi$ of $S$.

Notice that if the discrepancy of a set is large, then there is a convex partition $\Pi$ of $S$ in which each element of $\Pi$ has large discrepancy. If $d(S) = 1$ *any convex partitioning* of $S$ has at least one element with discrepancy one. For example if $S = R \cup B$ is separable, that is there is a line $\ell$ that leaves all the elements of $R$ on one of the half-planes it determines, and all the elements of $B$ on the other, then the discrepancy of $S$ is at least the minimum of $r$ and $b$.

*University of Texas at Dallas, besp@utdallas.edu
†Departamento Matemática Aplicada II, Universidad de Sevilla, dbanez@us.es, Partially supported by Grant MEC MTM2006-03909
‡Instituto de Matemáticas, Universidad Nacional Autónoma de México, dlara@uxmcc2.iimas.unam.mx
§Departamento de Computación, Universidad de La Habana, pablo@matcom.uh.cu, Partially supported by Grant MAEC-AECI and MEC MTM2006-03909
¶Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, carlos.seara@upc.edu, Partially supported by Grants MEC MTM2006-01267 and DURSI 2005SGR00692
‖Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx, Partially supported by Grant MEC MTM2006-03909

If we restrict ourselves to convex partitions of $S$ with exactly $k$ elements, we obtain the $k$-discrepancy of $S$, which will be denoted as $d_k(S)$. When $k = 1$ then the partition $\Pi$ has only one element, and thus $d_1(S) = \nabla(S) = |r - b|$. If $k = 2$ then we have what we call *linear discrepancy*, that is the discrepancy obtained by partitions of $S$ induced by lines that split $S$ into two subsets.

Our concept of discrepancy has applications in Data Analysis and Clustering in sets of data of two classes, say red and blue. We can state that a red-blue dataset is not good for clustering when its discrepancy is low. Hence our concept can be used as a priori tester to a dataset for clustering. The extreme case is when $d(S) = 1$, in this case we say that $S$ is *locally balanced*. Many of the results of this paper focus on the hardness of deciding if a bicolored point set is locally balanced or not.

The discrepancy between two objects is a measure of how different the objects are. In [1] they study a concept of discrepancy for hypergraps, they study the problem of assigning weight $+1$ or $-1$ to the vertices of a given hypergraph in such a way the maximum weight of a edge (i.e. the absolute value of the sum of the weights of its vertices) is minimized. Geometric Discrepancy Theory [4] studies how uniform nonrandom structures can be. For example, how to color $n$ points in the plane to minimize the difference between the number of red points and the number of blue ones within any disk. In [2, 6, 7] the concept of bichromatic discrepancy is considered by computing the object (e.g. box, triangle, strip, convex polygon, etc.) of maximum absolute difference between red and blue points inside it. In this paper we introduce a new parameter to measure the discrepancy of bicolored point sets.

The outline of this work is as follows. In section 2 we provide combinatoric results for measuring discrepancy for general convex partitions. We consider two cases, when the points are in convex position and when they are not. In section 3 we give combinatoric and hardness proofs on measuring discrepancy by using partitions by a line.

# 2 General Convex partitions

In this section we explore sets of red and blue points with discrepancy equal to one. A point set $P$ is in *convex position* if the elements of $P$ are the vertices of a convex polygon. We consider two cases, points in convex position and points in general position. We start with the following lemma which is straightforward to prove.

**Lemma 2.1.** *Let $S = R \cup B$ then, $d(S) \geq 1$. Moreover if $d(S) = 1$ then $|r - b| \leq 1$.*

## 2.1 Point Sets in Convex Position

**Definition 2.2.** Suppose that the elements of $S$ are in *convex position* and that $|r - b| \leq 1$. We say that they form an *alternating convex chain* if we can label them $p_1, p_2, \ldots, p_{r+b}$ in the counterclockwise order around $CH(S)$ such that, for every $1 \leq i \leq r + b - 1$, $p_i$ and $p_{i+1}$ have different color.

**Lemma 2.3.** *If $S$ is in convex position then $d(S) = 1$ if and only if $S$ is an alternating convex chain.*

*Proof.* Suppose that $d(S) = 1$ and that $S$ is not an alternating convex chain. By Lemma 2.1 $|r-b| \leq 1$. If $r = b$ and for some $i$ we have that $p_i$ and $p_{i+1}$ have the same color, the partition $\Pi = \{S_1, S - S_1\}$ where $S_1 = \{p_i, p_{i+1}\}$ has discrepancy 2. If $r = b + 1$ and there is an $i$ such that $p_i$ and $p_{i+1}$ are red points, then if $S_1 = \{p_i, p_{i+1}\}$, $\nabla(S_1) = 2$, $\nabla(S - S_1) = 3$, and the partition $\Pi = \{S_1, S - S_1\}$ is such that $d(S) \geq d(S, \Pi) = 2$. It follows by contradiction that $S$ is an alternating convex chain.

Suppose now that $S$ is an alternating convex chain. It is easy to see that in any convex partition $\Pi = \{S_1, S_2, \ldots, S_k\}$ of $S$, there is at least one $S_i$ $(1 \leq i \leq k)$ such that $S_i$ is an alternating convex chain, and thus has discrepancy less than or equal to one. Thus $\nabla(S_i) \leq 1$, and that $d(S, \Pi) \leq 1$ for all $\Pi$. Therefore $d(S) = 1$. □

To make some of our proofs easier, for any bicolored point set $X$ with $r'$ red points, and $b'$ blue points, let $\nabla'(X) = r' - b'$. Observe that $\nabla(X) = |\nabla'(X)|$. Next we prove:

**Theorem 2.4.** *If $S$ in convex position then $d(S) = \max_{k=1,2,3} d_k(S)$.*

*Proof.* Let $d = d(S)$, observe that in particular $0 \leq \nabla(S) \leq d$. Assume w.l.o.g. that $0 \leq \nabla'(S) \leq d$. Let $\Pi = \{S_1, S_2, \ldots, S_k\}$ be a convex partition of $S$ with minimum cardinality such that $d(S, \Pi) = d$. By definition, $\nabla(S_i) \geq d$ $(1 \leq i \leq k)$. Suppose that $k > 3$. Then $S$ has at least two elements, say $S_1$ and $S_2$, such that both of them contain only consecutive elements of $S$ along the boundary of $CH(S)$.

If any of $S_1$ or $S_2$, say $S_1$, is such that $\nabla'(S_1) \leq -d$ then $\nabla'(S - S_1) = \nabla'(S) - \nabla'(S_1) \geq 0 + d = d$, and thus $\nabla(S - S_1) = |\nabla'(S - S_1)| \geq d$. This is a contradiction because the convex partition $\Pi' = \{S_1, S - S_1\}$ has cardinality 2 and $d(S, \Pi') \geq d$. Suppose then that $\nabla'(S_1) \geq d$ and $\nabla'(S_2) \geq d$. Observe that $\nabla'(S - S_1 - S_2) = \nabla'(S) - \nabla'(S_1) - \nabla'(S_2) \leq d - d - d = -d$, and thus $\nabla(S - S_1 - S_2) = |\nabla'(S - S_1 - S_2)| \geq d$. This is a contradiction because $\Pi'' = \{S_1, S_2, S - (S_1 \cup S_2)\}$ has cardinality 3 and $d(S, \Pi'') \geq d$. □

This implies:

**Theorem 2.5.** *The discrepancy of a bicolored point set in convex position can be computed in polynomial time.*

## 2.2 Point Sets in General Position

In this section we deal with bicolored point sets in general position. The following lemmas are straightforward.

**Proposition 2.6.** *If $S$ has at most four elements and $d(S) = 1$ then $S$ is an alternating convex chain.*

**Proposition 2.7.** *There are two combinatorially different point configurations of a bichromatic point set $S$ with five points such that $d(S) = 1$ (see Fig. 1).*

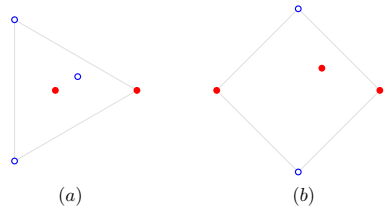



Figure 1: Two different point configurations with five points with $d(S) = 1$.

We now show configurations of points in general position that are locally balanced.

**Proposition 2.8.** *For all $n \geq 4$ there are bichromatic point sets, not in convex position, with $d(S) = 1$.*

*Proof.* The proof is based on the following constructions. For $n = 2m + 1$ let $Q$ be a regular convex polygon with $2m$ vertices such that their colors alternate blue and red along the boundary of $Q$. Let $S$ be the set of vertices of $Q$ plus a red point $p$ close to the center of $Q$ (see Fig. 2 $(a)$). Let $\Pi = \{S_1, S_2, \ldots, S_k\}$ be any convex partition of $S$. If $k = 1$ then $d(S, \Pi) = 1$. Suppose that $k > 1$, then there is some $S_i \in \Pi$ $(1 \leq i \leq k)$ such that $r \notin S_i$ and $S_i$ contains a set of consecutive vertices of $Q$. Then $\nabla(S_i) \leq 1$ and therefore $d(S, \Pi) \leq 1$.

If $n = 2m + 2$, place two points $p$ and $q$ in the interior of $Q$ such that $p$ and $q$ are close enough to the middle of an edge $e$ of $Q$, and the line joining then is almost parallel to $e$. It is easy to see now that $d_2(S) = 1$ and that $d(S, \Pi) \leq 1$ for all the convex partitions $\Pi$ of $S$ (see Fig. 2(b)). □

Constructing point sets with large discrepancy is straightforward. As we already mentioned in the introduction of this paper, if $R$ and $B$ are linearly separable then $d(S) \geq min\{r, b\}$.

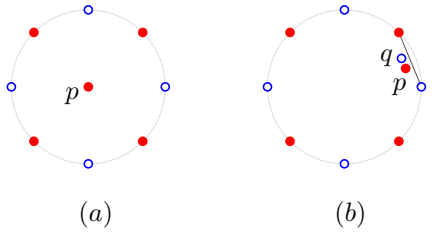



Figure 2: Point sets with $n$ points and $d(S) = 1$. $(a)$ $n$ is odd. $(b)$ $n$ is even.

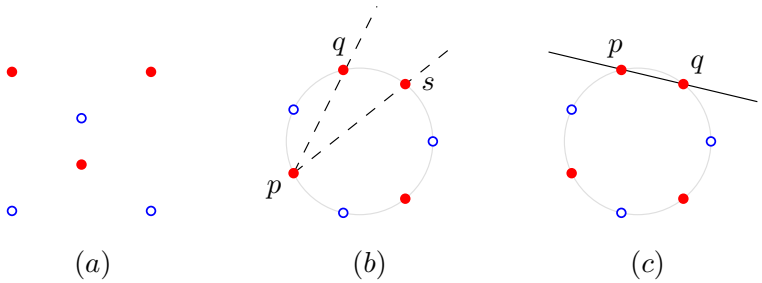



Figure 3: $(a)$ There are no three consecutive points of the same color in the projection on any line and $d_2(S) = 2$. $(b)$ The red points $q$ and $s$ are consecutive in the angular sorting of $S - \{p\}$ with respect to $p$ and $d_2(S) = 1$. $(c)$ The number of red and blue points in the half-plane above the line through $p$ and $q$ is zero and $d_2(S) = 1$ because $S$ is an alternating convex chain.

## 3 Partitions with a line

The problem of deciding the discrepancy of point sets in general position seems to be non-trivial. At this point, we are unable even to characterize point sets with discrepancy one. In this section we characterize sets with linear discrepancy one and show how to decide if the linear discrepancy of a bicolored point set $S$ is equal to $d$. We introduce the following notation.

Let $\Pi_{\ell^+}$ and $\Pi_{\ell^-}$ be the open half-planes bounded below and above respectively by a non vertical line $\ell$. Let $S_{\ell^+} = S \cap \Pi_{\ell^+}$, $S_{\ell^-} = S \cap \Pi_{\ell^-}$ and $\Pi_\ell = \{S_{\ell^+}, S_{\ell^-}\}$. The *linear discrepancy of* $S$ is $d_2(S) = \max_\ell d(S, \Pi_\ell)$ where the lines $\ell$ contain no point in $S$.

**Proposition 3.1.** *Let $S = R \cup B$ such that $r = b$ and $d_2(S) = 1$. Then the following properties hold:*

1. *The convex hull of $S$ is an alternating chain.*
2. *When projected on any line, the points of $S$ form a sequence such that no three consecutive points have the same color.*
3. *For every $p \in S$ on the convex hull of $S$, the angular ordering of the elements of $S - \{p\}$ with respect to $p$ is a sequence with alternating colors.*
4. *For every line $\ell$ passing through two points of the same color, say red, the number of red points in each of $S_{\ell^+}$ and $S_{\ell^-}$ is exactly one less than the number of blue points in $S_{\ell^+}$ and $S_{\ell^-}$ respectively.*

Property 2 in Proposition 3.1 is not sufficient to guarantee that $d(S) = 1$, e.g. see Fig. 3 $(a)$. If $r \neq b$ properties 3 and 4 are not necessarily true, see Fig. 3 $(b)$ and $(c)$. We now show that if $r = b$, Property 4 is sufficient.

The next result, proven in [5] will be useful:

**Theorem 3.2.** *Let $P$ and $Q$ be two disjoint convex polygons on the plane. Then there is at least one edge $e$ of $P$ or $Q$ such that the line $\ell_e$ containing $e$ separates the interior of $P$ from the interior of $Q$.*

**Lemma 3.3.** *If $r = b$ then the following two conditions are equivalent: (a) $d_2(S) = 1$, (b) for every line $\ell$ passing through two points of $S$ with the same color $\nabla(S_{\ell^+}) = \nabla(S_{\ell^-}) = 1$.*

*Proof.* It is easy to prove that $(a)$ implies $(b)$. We show here that $(b)$ implies $(a)$. Suppose that $d_2(S) = d \geq 2$. We now show that there exists a line $\ell$ containing two points of the same color of $S$ such that $\{\nabla(S_\ell^-), \nabla(S_\ell^+)\} = \{d, d-2\}$.

Let $\ell_0$ be a line containing no elements of $S$ such that $d_2(S) = d(S, \Pi_{\ell_0}) = d$. Assume w.l.o.g. that $\ell_0$ is horizontal. Since $r = b$ we have that $d_2(S) = d(S, \Pi_{\ell_0}) = \nabla(S_{\ell_0}^+) = \nabla(S_{\ell_0}^-) = d$ and $\nabla'(S_{\ell_0}^+) = -\nabla'(S_{\ell_0}^-)$. Assume w.l.o.g. that $\nabla'(S_{\ell_0}^+) > 0$ (i.e. $S_{\ell_0}^+$ has more red points than blue, and $S_{\ell_0}^-$ has more blue points than red).

Let $P$ and $Q$ be the polygons induced by the convex hulls of $S_{\ell_0}^+$ and $S_{\ell_0}^-$ respectively. Let $p$ be a vertex of $P$ such that there is a line $\ell'$ passing trough $p$ that separates $P$ from $Q$. Then $p$ must be a red point, for otherwise by translating $\ell'$ up by a small distance, we obtain a partitioning $\Pi'$ of $S$ with discrepancy $d+1$. Similarly any point $q$ in $Q$ such that there is a line trough $q$ that separates $P$ from $Q$ must be blue.

By Theorem 3.2 there is an edge $e$ of $P$ or $Q$, with vertices $p$ and $q$, such that the line $\ell_e$ containing $e$ separates $P$ from $Q$. If $e$ is an edge of $P$ then $p$ and $q$ are red by using the above observation. Thus we have that $\{\nabla(S_{\ell_e}^+), \nabla(S_{\ell_e}^-)\} = \{d, d-2\}$. A symmetric argument works when $e$ belongs to $Q$. □

**Proposition 3.4.** *Let $S = R \cup B$, then we have,*

$$\max\left\{1, \lfloor\frac{|r-b|}{2}\rfloor\right\} \leq d_2(R \cup B) \leq \max\left\{\lfloor\frac{|r-b|}{2}\rfloor, \min\{r, b\}\right\}.$$

*Furthermore, both bounds are tight.*

*Proof.* Suppose w.l.o.g. that $r \geq b$. By the *Ham Sandwich Cut Theorem* [10] there exists a line $\ell$ passing through at most one red point and at most one blue point such that $|S_{\ell^+} \cap R| = |S_{\ell^-} \cap R| = \lfloor\frac{r}{2}\rfloor$ and $|S_{\ell^+} \cap B| = |S_{\ell^-} \cap B| = \lfloor\frac{b}{2}\rfloor$. Four cases arise depending on the parities of $r$ and $b$. We show only the case when $r$ and $b$ are even. The other cases can be solved in a similar way.

If $r = 2a$ and $b = 2c$ then $\ell$ contains no point of $S$, and $\nabla(S_{\ell^+}) = \nabla(S_{\ell^-}) = a - c = \lfloor\frac{r-b}{2}\rfloor$. Thus $\lfloor\frac{r-b}{2}\rfloor = d(S, \Pi_\ell) \leq d_2(S)$.

If $|r-b| \geq 2$ then $d_2(S) \geq \lfloor\frac{|r-b|}{2}\rfloor \geq 1$ thus it is missing to prove that $d_2(S) \geq 1$ when $|r-b| \leq 1$. Suppose w.l.o.g. that $b \leq r \leq b+1$. If there is a blue point $p$ in the convex hull of $S$ take a line $\ell$ separating $p$ from $S - \{p\}$ and suppose that $p \in S_{\ell^+}$, then $\nabla(S_{\ell^+}) = 1$, $\nabla(S_{\ell^-}) = r - b + 1 \geq 1$ and $d_2(S) \geq d(S, \Pi_l) = 1$. If no such $p$ exists then there are two consecutive red points $p$ and $q$ in the convex hull of $S$, then take a line $\ell$ separating $p$ and $q$ from $S - \{p, q\}$ and suppose that $p, q \in S_{\ell^+}$, then $\nabla(S_{\ell^+}) = 2$, $\nabla(S_{\ell^-}) = b - r + 2 \geq 1$ and $d_2(S) \geq d(S, \Pi_l) = 1$. This proves the lower bound.

We show now that this lower bound is tight. Suppose w.l.o.g. that $r > b$ and let $X$ be a set composed by $r$ red points and $r$ blue points, and let $Y$ be a set of $r-b$ red points. Put the elements of $X$ on an alternating convex chain and the elements of $Y$ in the interior of the convex hull of $X$ in such a way there is a line $\ell_e$ such that $d(X, \Pi_{\ell_e}) = 0$ and $\ell_e$ splits $Y$ into two subsets of cardinality $\lfloor\frac{|Y|}{2}\rfloor$ and $\lceil\frac{|Y|}{2}\rceil$ respectively. Let $S = X \cup Y$ and observe that $d(S, \Pi_{\ell_e}) = \lfloor\frac{|Y|}{2}\rfloor = \lfloor\frac{r-b}{2}\rfloor$. For any line $\ell$ we have that $d(X, \Pi_\ell) \in \{0, 1\}$ (by Lemma 2.3). If $d(X, \Pi_\ell) = 0$ then $d(S, \Pi_\ell) = d(X \cup Y, \Pi_\ell) = d(Y, \Pi_\ell) \leq \lfloor\frac{|Y|}{2}\rfloor = \lfloor\frac{r-b}{2}\rfloor$. If $d(X, \Pi_\ell) = 1$ then $d(X \cup Y, \Pi_\ell) = \min\{x-1, (r-b)-x+1\}$ where $x$ is such that $l$ splits $Y$ in $x$ and $(r-b)-x$ points respectively. It is easy to prove that $\min\{x-1, (r-b)-x+1\} \leq \lfloor\frac{r-b}{2}\rfloor$. Then $d_2(S) = d(S, \Pi_{\ell_e}) = \lfloor\frac{r-b}{2}\rfloor$.

To prove the upper bound suppose w.l.o.g. that $r \geq b$ (i.e. $b = \min\{r, b\}$). We have to show that $d(S, \Pi_\ell) > b \Rightarrow d(S, \Pi_\ell) \leq \lfloor\frac{r-b}{2}\rfloor$ for every line $\ell$. Let $\ell$ be a line such that $d(S, \Pi_\ell) > b$. Then we have that $\nabla'(S_{\ell^+}) > 0$ and $\nabla'(S_{\ell^-}) > 0$. In fact, suppose that $\nabla'(S_{\ell^+}) < 0$, then $\nabla(S_{\ell^+}) = |S_{\ell^+} \cap B| - |S_{\ell^+} \cap R| \leq |S_{\ell^+} \cap B| \leq b$ thus $d(S, \Pi_\ell) \leq b$, a contradiction. Now, $\nabla'(S_{\ell^+}) > 0$ and $\nabla'(S_{\ell^-}) > 0$ imply that $d(S, \Pi_\ell) \leq \lfloor\frac{r-b}{2}\rfloor$. In fact, suppose the contrary, $\nabla(S_{\ell^+}) \geq \lfloor\frac{r-b}{2}\rfloor + 1$ and

$\nabla(S_{\ell^-}) = (r-b) - \nabla(S_{\ell^+}) \geq \lfloor \frac{r-b}{2} \rfloor + 1$, thus $r - b \geq 2\lfloor \frac{r-b}{2} \rfloor + 2$, a contradiction. If $\lfloor \frac{r-b}{2} \rfloor \leq b$ the upper bound is tight if we take separable sets $R$ and $B$. If $\lfloor \frac{r-b}{2} \rfloor > b$ we have shown above how to build a set of points $S$ with $d_2(S) = \lfloor \frac{r-b}{2} \rfloor$. □

**Corollary 3.5.** *Let $S = R \cup B$ such that $|r - b| \geq 2$. If $r \geq 3b$ or $b \geq 3r$ then $d_2(R \cup B) = \lfloor \frac{|r-b|}{2} \rfloor$.*

*Proof.* Suppose that $r \geq 3b$, then $r - b \geq 2b \Rightarrow \frac{r-b}{2} \geq b \Rightarrow \lfloor \frac{r-b}{2} \rfloor \geq b$. Thus the upper and lower bounds of $d_2(R \cup B)$ in Proposition 3.4 are equal. □

## 3.1 Hardness

**Theorem 3.6.** *Given an integer $d \geq 1$ it is 3SUM-hard to decide if $d_2(S) = d$.*

*Proof.* We will use a reduction from the 3SUM-problem similar to the 3SUM-hardness proof of the 3-POINTS-ON-LINE-problem [9]. Consider the set $X = \{x_1, \ldots, x_n\}$ of $n$ integer numbers (positive's and negative's) an instance of 3SUM-problem and assume w.l.o.g. that $x_1 < \ldots < x_j < 0 < x_{j+1} < \ldots < x_n$ $(1 \leq j < n)$. Let $M = \max\{|x_1|, |x_n|\}$. If $d = 1$ put a blue point in $(-2M, 0)$ and a red point in $(2M, 0)$. If $d > 2$ then for each $1 \leq i \leq d-2$ put a red point in $(-2M - i + 1, 0)$ and a blue point in $(2M + i - 1, 0)$. Let $\varepsilon$ be a small real positive number such that $\varepsilon < \frac{1}{6M}$. For each $1 \leq i \leq n$ put a red point $r_i$ in $(x_i - \varepsilon, x_i^3)$ and a blue point $b_i$ in $(x_i + \varepsilon, x_i^3)$ (see Fig. 4). We can prove that there exists a line separating three distinct pairs $(r_i, b_i)$, $(r_j, b_j)$ and $(r_k, b_k)$ if and only if $(x_i, x_i^3)$, $(x_j, x_j^3)$ and $(x_k, x_k^3)$ are collinear (i.e. $x_i + x_j + x_k = 0$). Let $S$ be the set of red an blue points as above. We have that $d_2(S) \geq d$ because $d_2(S, \Pi_\ell) = d$ for every line $\ell$ separating exactly two distinct pairs $(r_i, b_i)$ and $(r_j, b_j)$. If $d_2(S) > d$ then there is a line separating more than two pairs implying that three elements in $X$ sum to zero. Therefore, three elements in $X$ sum to zero if and only if $d_2(S) \neq d$. □

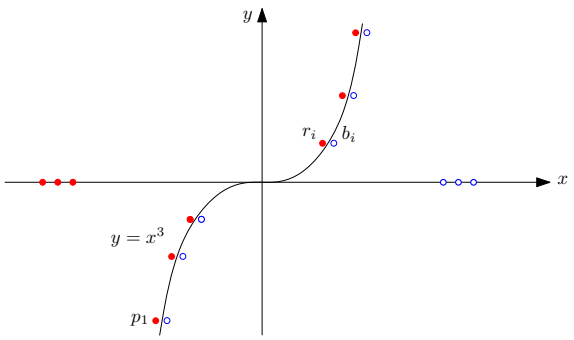



Figure 4: Reduction from 3SUM-problem when $d = 5$.

**Theorem 3.7.** *Computing the linear discrepancy of a bichromatic point set is 3SUM-hard and it can be done in $O(n^2)$ time.*

*Proof.* The hardness is an implication of Theorem 3.6 and we can use duality for computing $d_2(S)$. □

## 3.2 The Weak Separator problem

Given a bichromatic set of points in the plane the *Weak Separator problem* (WS-problem) looks for a line that maximizes the number of blue points on one of its sides plus the number of the red ones on the other. The WS-problem can be solved in $O(n^2)$ [11] or in $O(nk \log k + n \log n)$ [8] where $k$ is the number of misclassified points. Recently an $O((n + k^2) \log n)$ expected time algorithm has been presented in [3]. We now prove that the WS-problem is 3SUM-hard.

**Lemma 3.8.** *Let $S = R \cup B$ such that $r = b$. Solving the WS-problem for $S$ is equivalent to finding a line $\ell$ such that $d(S, \Pi_\ell) = d_2(S)$.*

*Proof.* Let $\ell$ be any line such that $d(S, \Pi_\ell) = d_2(S)$. Since $r = b$ then $\nabla'(S_{\ell^+}) = -\nabla'(S_{\ell^-})$. Suppose w.l.o.g. that $d_2(S, \Pi_\ell) = \nabla'(S_{\ell^+}) = |S_{\ell^+} \cap R| - |S_{\ell^+} \cap B| > 0$. We have that $|S_{\ell^+} \cap R| + |S_{\ell^-} \cap B| = |S_{\ell^+} \cap R| + |B| - |S_{\ell^+} \cap B| = b + |S_{\ell^+} \cap R| - |S_{\ell^+} \cap B|$. Hence $|S_{\ell^+} \cap R| + |S_{\ell^-} \cap B|$ is maximum if and only if $|S_{\ell^+} \cap R| - |S_{\ell^+} \cap B| = d_2(S)$ is maximum. □

The next result follows:

**Theorem 3.9.** *The WS-problem is 3SUM-hard.*

# 4 Acknowledgements

The problems studied here, were introduced and partially solved during the *Fourth Spanish Workshop on Geometric Optimization*, June 2008, El Rocío, Huelva, Spain. The authors would like to thank other workshop participants for helpful comments.

# References

[1] P.K. Agarwal and J. Pach. Combinatorial Geometry. Wiley-Interscience Series in Discrete Mathematics and Optimization, 16:267–290, 1995.

[2] C. Bautista-Santiago, J.M. Díaz-Báñez, D. Lara, P. Pérez-Lantero, J. Urrutia and I. Ventura, Computing Maximal Islands. Proc. 26th European Workshop on Computational Geometry - EWCG'09.

[3] T. M. Chan. Low-Dimensional Linear Programming with Violations. SIAM Journal on Computing, 34(4), 879–893, 2005.

[4] B. Chazelle. The Discrepancy Method in Computational Geometry. Handbook of Discrete and Computational Geometry, CRC Press 44, 983–996, 2004.

[5] J. Czyzowicz, E. Rivera Campo, J. Urrutia and J. Zaks. Separating Convex Sets. Discrete and Computational Geometry 7:189-195, 1992.

[6] D. P. Dobkin and D. Gunopulos. Concept learning with geometric hypotheses. In Proc. 8rd Annu. Conference on Comput. Learning Theory. ACM Press, 1995.

[7] D. P. Dobkin, D. Gunopulos, and W. Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. J. Computer and Systems Sciences, 52(3), 453–470, 1996.

[8] H. Everett, J.M. Robert, M. Kreveld. An Optimal Algorithm for Computing ($\leq k$)-Levels, with Applications to Separation and Transversal Problems. Int. J. Comput. Geom. Appl., 6:247–261, 1996.

[9] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. Computational Geometry: Theory & Applications, 5:165–185, 1995.

[10] J. Goodman and J. O'Rourke. Handbook of Discrete and Computational Geometry. Edited by CRC Press, 211, 1997.

[11] M. E. Houle. Algorithms for weak and wide separation of sets. Proc. Int. Workshop on Discr. Alg. and Comp, 61–68, 1989.

## Sesión 7, 17:00–18:20

# Stabbers of line segments in the plane *

M. Claverol† D. Garijo‡ C. I. Grima§ A. Márquez¶ C. Seara‖

**Abstract**

The problem of computing a representation of the stabbing lines of a set $S$ of $n$ line segments in the plane was solved by Edelsbrunner et al. with an $\Theta(n \log n)$ time and $O(n)$ space algorithm. We present a study of different types of stabbers such as wedges, double-wedges, 2-level trees, and zigzags; providing efficient algorithms whose time and space complexities depend on the number of combinatorially different extreme lines $h_S$ or critical lines $c_S$, and the number $k_S$ of different slopes that appear in $S$.

## 1 Introduction

Let $S = \{s_1, \ldots, s_n\}$ be a set of line segments (or segments) in the plane. For convenience, we require that if $p$ and $q$ are endpoints of a segment, then $p \neq q$, and consequently, lines, rays, and points are not considered to be segments. In order to avoid tedious case analysis, we assume that the endpoints of the segments are in general position. Nevertheless, the results presented in the paper can be extended to arbitrarily segment sets.

A line is a *transversal* of (or *stabs*) $S$ if it intersects each segment of $S$. Edelsbrunner et al. [7] presented an $\Theta(n \log n)$ time and $O(n)$ space algorithm for solving the problem of constructing a representation of all traversal lines or stabbing lines of $S$. See Edelsbrunner [6] for an analysis of this problem from both a combinatorial and computational point of view. The lower bound from Edelsbrunner et al. [7] does not apply to the decision problem: *determining if there exists a line stabber for* $S$. Avis et al. [2] presented an $\Omega(n \log n)$ time lower bound in the fixed order algebraic decision tree model to determine the existence of a line stabber for $S$. For a set of $n$ vertical segments, a stabbing line can be computed in $O(n)$ time.

A stabbing line $\ell$ for $S$ classify the endpoints of the segments in two classes: endpoints above $\ell$, say red points; and endpoints below $\ell$, say blue points. The endpoint on $\ell$ is classified according to the other endpoint. Thus, we can see the problem of stabbing $S$ as a problem of classifying the endpoints of the segments into disjoint monochromatic red and blue regions defined by the stabber, i.e., as a separability problem.

Since we want that the stabbers for $S$ classify the endpoints of the segments in that way, we can consider the condition that there is no segment stabbed by more than one element of the stabber. We call this condition the *separability condition*. So we look for stabbers for $S$ such that we can assign red and blue colors to the endpoints of the segments and split the plane into disjoint monochromatic regions, i.e., obtaining a red/blue classification of the endpoints of the segments. Hurtado et al. [11] classified red and blue points in the plane with separators which are similar to our stabbers.

Following this line of research, we deal with the problem of finding different kinds of "simple" stabbers when there exists no stabbing line for $S$. Concretely, we shall consider the structures shown in Figure 1. The goal is to design efficient algorithms for computing these stabbers for $S$ with or without the separability condition.

*Supported by projects MEC MTM2006-01267 and DURSI 2005SGR00692.

†Dept. de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, Spain, merce@ma4.upc.edu

‡Depto. de Matemática Aplicada I, Universidad de Sevilla, Sevilla, Spain, dgarijo@us.es

§Depto. de Matemática Aplicada I, Universidad de Sevilla, Sevilla, Spain, grima@us.es

¶Depto. de Matemática Aplicada I, Universidad de Sevilla, Sevilla, Spain, almar@us.es

‖Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, carlos.seara@upc.edu

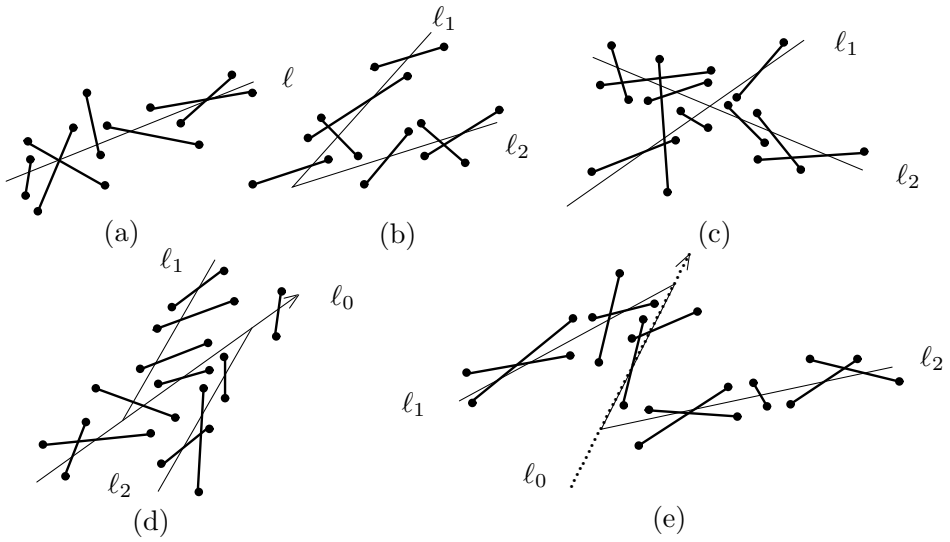



Figure 1: Stabbers from (a) to (e): line, wedge, double-wedge, 2-level tree, zigzag.

A standard geometric tool which will be used throughout this work is *duality* [7]: the geometric transform denoted by $\mathcal{D}$ which maps a point into a non-vertical line and vice versa. Given a point $p := (a, b)$ and a line $\ell := y = cx + d$, we have $\mathcal{D}(p) := y = ax + b$ and $\mathcal{D}(\ell) := (-c, d)$. A segment $s_i \in S$ is determined by its endpoints. The endpoints are transformed by $D$ into two lines. If $s_i$ is not vertical, $\mathcal{D}(s_i)$ is a double-wedge which does not contain a vertical line in its interior. Thus, the double-wedge is formed by two upper rays and two lower rays. If $s_i$ is a vertical segment, $\mathcal{D}(s_i)$ is a *strip*. The set of endpoints of the segments in $S$ is transformed by $\mathcal{D}$ into an arrangement of $2n$ lines denoted by $\mathcal{A}(S)$.

The transform $\mathcal{D}$ satisfies the following properties: (i) the transform $\mathcal{D}$ maintains the relative position (above/below) of points and lines; (ii) a line $\ell$ intersects a segment $s_i$ if and only if the point $\mathcal{D}(\ell)$ lies in the double-wedge $\mathcal{D}(s_i)$; (iii) the stabbing lines of $S$ stand in one-to-one correspondence with the intersection points of their double-wedges, i.e., $\bigcap_{s_i \in S} \mathcal{D}(s_i)$.

**Related works**. Claverol [5] as a part of her PhD thesis initiated the study here developed. In this paper we improve the complexities she obtained and some other stabbing problems are also considered. Atallah and Bajaj [1] presented an $O(n\alpha(n) \log n)$ algorithm for line stabbing $n$ simple objects in the plane, where $\alpha(n)$ is the inverse of the Ackerman's function. A simple object is an object which has an $O(1)$ store description and for which common tangents and intersections can be computed in $O(1)$ time. Edelsbrunner, Guibas and Sharir [8] showed how to construct a representation of the line stabbers of convex polygons with a total of $n$ vertices in $O(n\alpha(n) \log n)$ time. Later improved to $O(n \log n)$ using an $O(n \log n)$ time algorithm from Hershberger [10] for finding the lower envelope of a segment set in the plane. O'Rourke [14] presented an algorithm for finding (if it exists) a stabbing line of vertical line segments. Goodrich and Snoeyink [9] presented a natural variant considering another type of stabbers different from the lines by solving the problem of computing a transversal convex polygon for a set of parallel segments in $O(n \log n)$ time. Bhattacharya et al. [3] worked on the problem of computing the shortest transversal segment for a set of lines in the plane and also for a set of convex polygons. Lyons et al. [12] studied the problem of computing the minimum perimeter convex polygon which stabs a set of isothetic line segments. Rappaport [15] considered the problem of computing a simple polygon with minimum perimeter which stabs or contains a set of line segments. Bhattacharya et al. [4] and Mukhopadhyay et al. [13] considered the problem of computing the minimum area convex polygon which stabs a set of parallel line segments.

## 2 Ideas and tools

Some ideas and tools are shared by most of our results. In this section we present them in a unified context. The lines containing the segments of $S$ can have different slopes. Denote by $m_i$ the slope of (the line containing) the segment $s_i \in S$. The complexity of many of the algorithms that we present here depends on the number of different slopes of the lines containing the segments of $S$, written as $k_S$. It is due to the following fact: the endpoints of the segments fall into two classes determined by a stabbing line $\ell$, endpoints above $\ell$ and endpoints below $\ell$, say red and blue respectively. The endpoint of a segment on $\ell$ is classified according to its other endpoint. Thus, for a given slope of $\ell$ our problem of stabbing the set $S$ can be viewed as a red-blue separability problem of classifying the endpoints of the segments into disjoint monochromatic regions of the plane determined by the stabber. In fact, it is not difficult to show that there exist as many different classifications of the endpoints of $S$ as $k_S$.

A relevant property about our stabbers is that not all the lines can be candidate to define one of these structures. For instance, consider a ray $\ell$ which is part of a wedge that stabs $S$, and its extension denoted by $\ell'$. We have that all the segments that are not intersected by $\ell$ must lie on the same half-plane defined by $\ell'$. Thus, we say that a line $\ell'$ is an *extreme line* for $S$ if $\ell'$ stabs a subset of segments $S_1 \subseteq S$, $S_1 \neq \emptyset$, and the remaining segments $S_2 = S \setminus S_1$ are in only one of the open half-planes defined by $\ell'$. Otherwise, the line $\ell'$ is said to be a non-extreme line for $S$. Thus, as we have mentioned before, our interest in extreme lines comes from the following fact: *the line containing any ray of a stabbing wedge for $S$ is extreme line for $S$.*

In this paper, extreme lines are studied from two different points of view: computational and combinatorial view. The reason is that our algorithms depend on the computation of the set of extreme lines for $S$. Thus, we say that two non-vertical lines, $\ell_1$ and $\ell_2$, are *combinatorially different* with respect to $S$ if either: (1) the subset of segments $S_1 \subseteq S$ stabbed by $\ell_1$ is different from the subset of segments $S_2 \subseteq S$ stabbed by $\ell_2$; or (2) if $S_1 = S_2$ then either: (i) the subset of endpoints of segments of $S$ above $\ell_1$ is different from the subset of endpoints of segments of $S$ above $\ell_2$, or (ii) the subset of endpoints of segments of $S$ below $\ell_1$ is different from the subset of endpoints of segments of $S$ below $\ell_2$. If $h_S$ is the number of combinatorially different extreme lines of $S$, we compute a representation of the combinatorially different extreme lines for $S$ in $O(h_S + n \log n)$ time and $O(h_S + n)$ space. Observe that the number of combinatorially different extreme lines for $S$ is at most $O(n^2)$, but depending on the properties of the stabbing problem, we can consider only a subset of them named the *critical extreme lines* which size $c_S$ is at most linear.

## 3 Stabbing wedges

Our first aim is to study the problem of deciding whether the set $S$ can be stabbed by a wedge, and computing this structure in case of existence. Obviously, it is assumed that the set $S$ is not stabbed by a line. We distinguish two cases: stabbing wedges $W$ satisfying the separability condition (described in Section 1) or those that do not satisfy such condition. In the first case, we provide an $O(h_S k_S \log n + n \log n)$ time and $O(h_S + n)$ space algorithm. The range for $h_S$ is from $O(1)$ to $O(n^2)$, and the range for $k_S$ is from $O(1)$ to $O(n)$. In the second case, we design an $O(c_S k_S \log n + n \log n)$ time and $O(n)$ space algorithm, with range for $c_S$ in between $O(1)$ and $O(n)$.

We now introduce some useful notation for our purpose. Given a line $\ell$ and a segment $s$, we can classify the endpoints of $s$ with respect to $\ell$ whenever $\ell$ and the line containing $s$ are not parallel. It suffices to do a parallel sweep with $\ell$ until it crosses $s$, leaving one endpoint in $\ell^+$, and the other one in $\ell^-$. These endpoints are denoted by $e^+$ and $e^-$, respectively.

Let $W$ be a stabbing wedge for $S$. The two rays that form $W$ are denoted by $\ell_1$ and $\ell_2$, and $W$ is written as $W = \{\ell_1, \ell_2\}$. The line containing $\ell_i$ for $i = 1, 2$ is denoted by $\ell_i'$. The half-planes defined by $\ell_i$ are written as $\ell_i'^+$ and $\ell_i'^-$. Suppose that $\ell_1$ stabs a subset of segments $S_1 \subsetneq S$, $S_1 \neq \emptyset$, and the set $S_2 = S \setminus S_1$ is stabbed by $\ell_2$.

### 3.1 Stabbing wedges satisfying the separability condition

Since by definition we consider $W$ as a separator structure, a segment can not be stabbed by both rays. Let $S_1^+$ ($S_1^-$) be the set of endpoints of the segments of $S_1$ classified as $e^+$ ($e^-$) with respect to $\ell_1'$. Analogously, $S_2^+$ ($S_2^-$) is the set of endpoints of the segments of $S_2$ classified as $e^+$ ($e^-$) with respect to $\ell_2'$. Thus, $S_1^-$ and $S_2^+$ are contained inside the wedge $W$, and $S_1^+$ and $S_2^-$ are located outside it. Notice that these assignments $\{+,-\}$ depend on the relative position of the lines $\ell_1'$ and $\ell_2'$, i.e., on the slope and the aperture angle of the stabbing wedge. We concentrate in a particular case but the remaining constant number of cases can be handle in a similar way.

**Lemma 3.1.** *If $W = \{\ell_1, \ell_2\}$ is a stabbing wedge for $S$, $\ell_1'$ and $\ell_2'$ are extreme lines for $S$ and at least half of the segments of $S$ are stabbed by either $\ell_1$ or $\ell_2$.*

The next lemma assumes the following conditions: (i) let $\ell_1'$ be an extreme line for $S$ where $S_1 \subsetneq S$, $S_1 \neq \emptyset$, is the subset of segments stabbed by $\ell_1'$. Let $S_1^+$ and $S_1^-$ be the classification of the endpoints of the segments of $S_1$ given by $\ell_1'$, and let $S_2 = S \setminus S_1$. (ii) Let $m$ be a fixed slope and let $S_2^+$ and $S_2^-$ be the classification of the endpoints of the segments of $S_2$ by sweeping a line with slope $m$.

**Lemma 3.2.** *There exists a stabbing wedge $W = \{\ell_1, \ell_2\}$ for $S$ with $\ell_1$ contained in $\ell_1'$ if and only if $S_2^-$ is line separable from $S_1^- \cup S_2^+$. The locus of apices of the stabbing wedges for $S$ respecting this classification of endpoints is a (possible unbounded and degenerate) convex quadrilateral $Q$ defined by the following four lines: the interior supported lines between $CH(S_1^+)$ and $CH(S_1^- \cup S_2^+)$ and the interior supported lines between $CH(S_1^- \cup S_2^+)$ and $CH(S_2^-)$.*

Lemmas 3.1 and 3.2 are the key tools to design an algorithm that proves the following result.

**Theorem 3.3.** *The set of combinatorially different stabbing wedges for $S$ with the separability condition, together with a representation of them formed by the locus of apices of the stabbing wedges can be computed in $O(h_S k_S \log n + n \log n)$ time and $O(h_S + n)$ space.*

### 3.2 Stabbing wedges not satisfying the separability condition

Let $W = \{\ell_1, \ell_2\}$ be a stabbing wedge for $S$ not verifying the separability condition. A property that only holds for this type of wedges is the following: *there always exists a stabbing wedge $W$ for $S$ (not satisfying the separability condition) formed by two rays both anchored on fixed points of $S$, i.e., both rays are contained in critical extreme lines.* This property let us prove the following result.

**Theorem 3.4.** *The set of combinatorially different stabbing wedges for $S$ not satisfying the separability condition can be computed in $O(c_S k_S n + n \log n)$ time and $O(n)$ space.*

Next we show an $\Omega(n \log n)$ lower bound for the problem of deciding if there exists a stabbing wedge for a set of arbitrarily segments. We reduce the decision of the stabbing wedge problem to the problem of deciding whether there exists a stabbing line for a segment set, which has an $\Omega(n \log n)$ time lower bound in the fixed order algebraic decision tree model [2]).

**Theorem 3.5.** *Deciding whether there exists a stabbing wedge for an arbitrary segment set requires $\Omega(n \log n)$ time in the fixed order algebraic decision tree model.*

## 4 Stabbing wedges for parallel segments with equal length

Let $S = \{s_1, \ldots, s_n\}$ be a set of $n$ parallel segments in the plane *with equal length* which are not stabbed by a line. We now consider the problem of computing a stabbing wedge $W$ for $S$ satisfying the separability condition.

Up to symmetry with respect to either the $x$-axis or the $y$-axis, we distinguish three types of wedges according to the relative position of the rays $\ell_1$ and $\ell_2$ of the possible stabbing wedge $W = \{\ell_1, \ell_2\}$ for $S$. Let $\alpha_W$ be the *aperture angle* or *interval direction* defined by the rays $\ell_1$ and $\ell_2$ of $W$. The three types are the following: (a) $\alpha_W$ contains the vertical direction; (b) $\alpha_W$ contains the horizontal direction; and (c) both rays $\ell_1$ and $\ell_2$ of $W$ have positive slope.

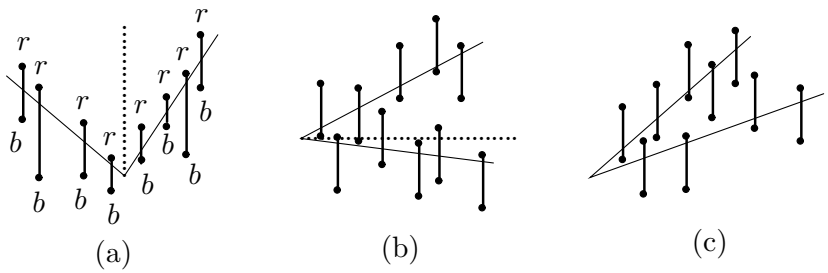



Figure 2: The three types of stabbing wedges for a parallel segment set.

## 4.1 Type (a)

When $\alpha_W$ contains the vertical direction, it is possible to design the following $O(n \log n)$ time algorithm for computing a stabbing wedge for $S$. It is based on an $O(n \log n)$ time and $O(n)$ space algorithm for deciding the wedge separability of a red-blue point set in the plane [11].

1. In $O(n)$ time, classify the endpoints of the segments of $S$ as follows. For each segment $s$, color red the endpoint of $s$ with bigger $y$-coordinate and color blue the other endpoint. Let $R$ and $B$ be the sets of red and blue endpoints.
2. In $O(n \log n)$ time, decide whether there exists a separating wedge for $R$ and $B$, and compute it in case of existence. In the same time, we can compute the locus of apices of all the separating wedges formed by convex quadrilaterals.

**Theorem 4.1.** *Given the set $S$, a stabbing wedge of type (a) for $S$ can be computed in $O(n \log n)$ time and $O(n)$ space.*

The assumption that the segments have equal length is not used in the algorithm. If the slopes of the rays of the wedge are known, there exists an $O(n)$ time algorithm to compute a stabbing wedge for $S$ using the median of the $x$-coordinates of the endpoints of the segments.

## 4.2 Type (b)

We have also obtained an $O(n \log n)$ time algorithm for computing a stabbing wedge for $S$ when $\alpha_W$ contains the horizontal direction. For each segment $s_i \in S$, consider its midpoint $\rho_i$. In $O(n \log n)$ time, sort these midpoints by decreasing $y$-coordinate, and let $\preceq_y$ denote this order. Denote by $S^* = \{s_1^*, \ldots, s_n^*\}$ the set of segments of $S$ sorted by the $\preceq_y$ order of their midpoints. The key tools to design our algorithm are given by the following lemma.

**Lemma 4.2.** *If there exists a stabbing wedge for $S$, $W = \{\ell_1, \ell_2\}$, such that $\alpha_W$ contains the horizontal direction, then the midpoints of the segments stabbed by $\ell_1$ appear before in the $\preceq_y$ order than the midpoints of the segments stabbed by $\ell_2$.*

*Let $\ell_1'$ be a line with positive slope that stabs $S_1 \subsetneq S$, $S_1 \neq \emptyset$, and let $\ell_2'$ be a line with negative slope that stabs $S_2 = S \setminus S_1$. Consider the classification of the endpoints of $S$ provided by $\ell_1'$ and $\ell_2'$. There exists a stabbing wedge $W = \{\ell_1, \ell_2\}$ for $S$ if and only if both $S_1^+$ and $S_2^-$ are line separable from $S_1^- \cup S_2^+$.*

**Theorem 4.3.** *Given the set $S$, a stabbing wedge of type (b) for $S$ can be computed in $O(n \log n)$ time and $O(n)$ space.*

## 4.3 Type (c)

When both rays $\ell_1$ and $\ell_2$ of $W$ have positive slope, the main problem is to obtain a consistent classification of the midpoints of the segments according to the possible stabbing wedge of type (c). Denote by $d$ the length of the segments of $S$, and recall that $\rho_i$ is the midpoint of the segment $s_i \in S$.

Assume that there exists a stabbing wedge of type (c) for $S$, denoted by $W = \{\ell_1, \ell_2\}$. Suppose also that $\alpha_W$ is known. Let $\ell'_i$ for $i = 1, 2$, be the line containing the ray $\ell_i$. Denote by $\ell''_1$ the line below and parallel to $\ell'_1$, such that the *vertical distance* between the two lines $\ell''_1$ and $\ell'_1$ is exactly $d/2$. Similarly, $\ell''_2$ is the line above and parallel to $\ell'_2$ such that the vertical distance between $\ell''_2$ and $\ell'_2$ is also $d/2$. Let $\ell$ be the bisector of the angle defined by $\ell'_1$ and $\ell'_2$ or any line with slope between the slopes of $\ell'_1$ and $\ell'_2$ (Figure 3).

Obviously the angle defined by $\ell''_1$ and $\ell''_2$ is $\alpha_W$. Let $\ell$ be the bisector line of $\alpha_W$. In fact, as $\ell$ we can take any line with slope within the slope interval defined by $\alpha_W$. Consider the double-wedge $DW$ formed by the lines $\ell''_1$ and $\ell''_2$, and the corresponding upper rays and lower rays of $DW$. By definition of $\ell''_1$ and $\ell''_2$, all the midpoints of the segments of $S$ stabbed by $\ell_1$ ($\ell_2$) are above (below) or over the upper rays (lower rays) of $DW$. Let $\preceq_\ell$ be the order of the midpoints of the segments of $S$ according to a sweeping by the bisector line $\ell$ of $\alpha_W$. Thus, if we know that $\alpha_W \geq \alpha$, for some given $\alpha$, we can compute a constant number $\lceil \frac{\pi}{2\alpha} \rceil = t$ of slope candidates for line $\ell$ and check each one in $O(n \log n)$ time and $O(n)$ space. For stabbing wedges with very small aperture angle $\alpha_W$, the value $t$ can dominate $n$.

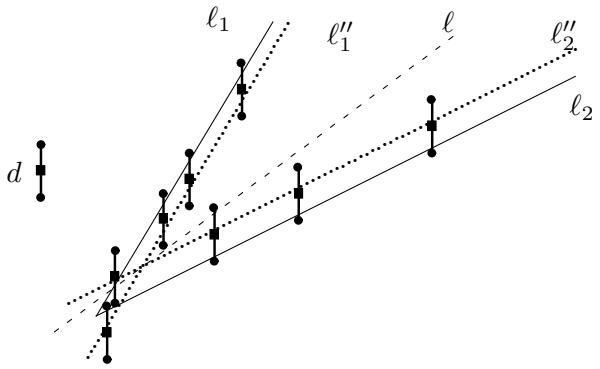



Figure 3: Both rays $\ell_1$ and $\ell_2$ of a stabbing wedge have positive slope.

**Theorem 4.4.** *Given the set $S$ and a positive value $\alpha$, a stabbing wedge of type (c) for $S$ with aperture angle $\alpha_W \geq \alpha$ can be computed in $O(nt \log n)$ time and $O(n)$ space.*

# 5 Other stabbers

Table 1 summarizes the obtained results for the decisional problems of the stabbers we have considered. By (sc) and (nsc) we denote *satisfying separability condition* and *not satisfying the separability condition*, respectively.

| **Stabber** | **Time** | **Space** |
|---|---|---|
| Wedge (sc) | $O(h_S k_S \log n + n \log n)$ | $O(h_S + n)$ |
| Wedge (nsc) | $O(c_S k_S n + n \log n)$ | $O(n)$ |
| Double-wedge (sc) | $\min\{O(n^4), O(n^3 k_S \log n)\}$ | $O(n^2)$ |
| Double-wedge (nsc) | $O(n^2 k_S \log n)$ | $O(n^2)$ |
| 2-level tree (nsc) | $O(n^2 k_S \log n)$ | $O(n^2)$ |
| Zigzag (sc) | $O(n^2 k_S \log n)$ | $O(n^2)$ |
| Zigzag (nsc) | $O(n^3 k_S)$ | $O(n^2)$ |

Table 1: Summary of results of decision problem.

**Acknowledgements**. We want to give special thanks to M. Abellanas and F. Hurtado for their help in the first study of the stabbing problems which was part of the Claverol's PhD thesis. E. Arkin and J. S. B. Mitchell also deserve our thanks for their comments.

# References

[1] M. Atallah and C. Bajaj. Efficient algorithms for common transversal. *Inform. Process. Lett.*, 25, (1987), 87–91.

[2] D. Avis, J.-M. Robert, and R. Wenger. Lower bounds for line stabbing. *Inform. Process. Lett.*, 33, (1989), 59–62.

[3] B. K. Bhattacharya, J. Czyzowicz, P. Egyed, G. Toussaint, I. Stojmenovic, and J. Urrutia. Computing shortest transversals of sets. *Proc. 7th. Ann. ACM Sympos. Comput. Geom.*, (1991), 71–80.

[4] B. Bhattacharya, C. Kumar, and A. Mukhopadhyay. Computing an area-optimal convex polygonal stabber of a set of parallel line segments. *Proc. 5th. Canad. Conf. Comput. Geometry*, (1993), 169–174.

[5] M. Claverol. Problemas geométricos en morfología computacional. *PhD Thesis. Universitat Politècnica de Catalunya*, 2004.

[6] H. Edelsbrunner. *Algorithms in combinatorial geometry*. EATCS Monographs on Theoretical Computer Science, Vol. 10, Springer-Verlag, (1987).

[7] H. Edelsbrunner, H. A. Maurer, F. P. Preparata, A. L. Rosenberg, E. Welzl, and D. Wood. Stabbing line segments. *BIT*, Vol. 22, (1982) pp. 274–281.

[8] H. Edelsbrunner, L. J. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: algorithms and applications. *Report No. UIUCDCS-R-87-1390*, University of Ilinois, (1987).

[9] M. T. Goodrich and J. S. Snoeyink. Stabbing parallel segments whit a convex polygon. *Proc. 1st Workshop Algorithms Data Struct. Lecture notes Comput. Sci.*, 382 (1989), 231–242, Springer-Verlag.

[10] J. Hershberger. Finding the upper envelop of $n$ line segments in $O(n \log n)$ time. *Information Processing Letters*, 33, 1989, pp. 169–174.

[11] F. Hurtado, M. Noy, P. A. Ramos, and C. Seara. Separating objects in the plane by wedges and strips. *Discrete Applied Mathematics*, Vol. 109, (2000), pp. 109–138.

[12] K. A. Lyons, Henk Meijer, and David Rappaport. Minimum polygon stabbers of isothetic line segments. *Department of Computing and Information Science*, Queen's University, Ontario, Canada, (1990).

[13] A. Mukhopadhyay, C. Kumar, E. Green, and B. Bhattacharya. On intersecting a set of parallel line segments with a convex polygon of minimum area. *Inform. Process. Lett.*, 105, (2008), 58–64.

[14] J. O'Rourke. An on-line algorithm for fitting straigt lines between data ranges. *CACM*, 24, (1981), 574–578.

[15] D. Rappaport. Minimum polygon transversals of line segments. *International Journal of Computational Geometry & Applications*, Vol. 5, No. 3, (1995), 243–256.

# Maximum Covered Path within a Region*

Manuel Abellanas† Antonio Bajuelos‡ Inês Matos‡

**Abstract**

Let $S$ be a set of sensors with a given transmission range $r \in \mathbb{R}^+$. The objective of this paper is to calculate the maximum coverage of a path between two points $p$ and $q$ on a region $R$. This problem is solved for four cases: $R$ is the line segment $\overline{pq}$, $R$ is a planar graph and $p$ and $q$ are two of its nodes, $R$ is a polygonal region containing $p$ and $q$ and, to conclude, $R$ is the whole plane. A maximum covered path between $p$ and $q$ is also found on the second case.

## 1 Introduction and Related Work

Problems related to wireless ad-hoc networks (or just sensor networks) have emerged in the last few years given the fast development of technology. Sensor networks are used to solve a great diversity of problems that range from battlefield monitoring to weather detection, museums' security and even wildlife protection [5, 12]. The problems introduced in the following are related to *coverage*. Since each sensor can be located anywhere within a specific region, coverage measures the quality of this placement. Let $R$ be a region that is surveilled by a given wireless sensor network. Following the approach to coverage studied by Huang and Tseng [8], the objective of the next set of problems is to decide whether a path within $R$ is covered. If so, find the path's maximum coverage provided by such network. Coverage can be looked at from two opposite perspectives. In the worst-case coverage there is an attempt to locate the regions of $R$ that are hidden from the sensors, that is, not surveilled. The best-case coverage is characterised as an attempt to locate the areas that are highly surveilled, thus identifying the "best" surveilled regions of $R$. The objective above clearly is included in this second case.

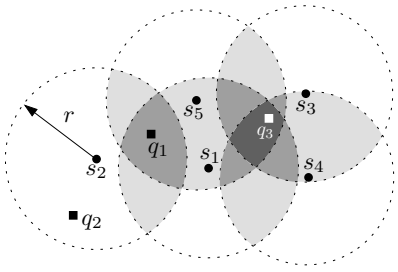



Figure 1: Set $S$ with range $r$ is represented by dots. Point $q_1$ has $\mathrm{MC}_S(q_1) = 3$ since it is a point of $D(s_1, r) \cap D(s_2, r) \cap D(s_5, r)$; $q_2$ is 1-covered by $S$ since it is only interior to $D(s_2, r)$. Point $q_3$ has $\mathrm{MC}_S(q_3) = 4$ since it is interior to four disks.

Let $S$ be a set of $n$ points on the plane that represent the location of $n$ devices which are able to send or receive some sort of wireless signal, like sensors. Also suppose that the devices of $S$ are homogeneous, that is, they all have the same power transmission range $r \in \mathbb{R}^+$ which is fixed. Assuming

*When this paper was finished, the third author was supported by a FCT fellowship, grant SFRH/BD/28652/2006

†Facultad de Informática, Universidad Politécnica de Madrid, co-supported by Project Consolider Ingenio 2010 i-MATH C3-0159 and MICINN Project MTM2008-05043 mabellanas@fi.upm.es

‡Departamento de Matemática & CEOC, Universidade de Aveiro, supported by CEOC through *Programa* POCTI, FCT, co-financed by EC fund FEDER {leslie,ipmatos}@ua.pt

all distances are euclidean, the distance between a point $q$ on the plane and a set $S$ of points is defined as the minimum distance between $q$ and every point of $S$. Point $q$ is said to be covered by a set $S$ with transmission range $r$ if the distance from $q$ to $S$ is less or equal to $r$. A point covered by $k$ or more sensors of $S$ is said to be *$k$-covered* by $S$. Let $\mathrm{MC}_S(q)$ denote the maximum coverage of point $q$ provided by $S$. For example, in Figure 1 point $q_1$ is 3-covered by $S$ since it is within reach of three sensors, $\mathrm{MC}_S(q_1) = 3$. Let $D = \{D(s, r) : \forall s \in S\}$ be the set of all disks of radius $r$ centred at sensors of $S$. In the same figure, point $q_3$ has $\mathrm{MC}_S(q_3) = 4$ since it is interior to four disks of $D$. A path between two points is called a $k$-covered path or a $k$-path if every point of such path is $k$-covered by $S$. As the sensors' transmission range is a given parameter, the algorithms proposed in this paper aim to find the maximum $k \in \mathbb{N}$ such that a $k$-path within a region $R$ exists.

There are several works related to this subject. For Kaučič and Žalikm [9], $k$-guarding a surface patch is having it guarded by at least $k$ guards. According to that definition, they solve two optimisation problems to $k$-guard a polyhedral terrain using heuristics. However, for Belleville et al. [4], $k$-guarding a polygon $P$ means that it is possible to find a collection of guards on the edges of $P$ (at most one guard per edge) such that every point in $P$ is visible to, at least, $k$ guards. Concerning the geometric optimisation of the sensors' range, the breakthrough paper is acknowledged to Meguerdichian et al. [12]. Others followed either by studying different versions of the coverage problem or by improving and proving previous results [5, 10, 13]. The authors in [3, 13] study routes which are always close (or always far) to a given set of sensors on the plane, whereas the authors in [1, 2] study this optimisation problem while working with 2-covering.

As previously stated, the work by Huang and Tseng [8] is relatively close to the subject focused on in this paper. They show how to determine whether every point in a polygonal region $R$ or in the service area of the sensor network is $k$-covered. The solution proposed in [8] is based on the boundary of the area covered by each sensor, that is, on the boundary of $D(s, r)$ for $s \in S$. They define $s$ as *$k$-perimeter-covered* if every point on the boundary of $D(s, r)$ is $k$-covered by $S \backslash \{s\}$ (see Figure 2). Region $R$ is $k$-covered if every sensor whose range reaches $R$ (or is within $R$) is $k$-perimeter-covered. In the example in Figure 2, $R$ is 1-covered by $S$. So, supposing $d$ is the number of sensors that are within reach of sensor $s$, the perimeter-coverage of $s$ can be calculated in $\mathcal{O}(d \log d)$ time. Overall, the perimeter-coverage of every sensor is found in $\mathcal{O}(nd \log d)$ time, which may add up to $\mathcal{O}(n^2 \log n)$ time.

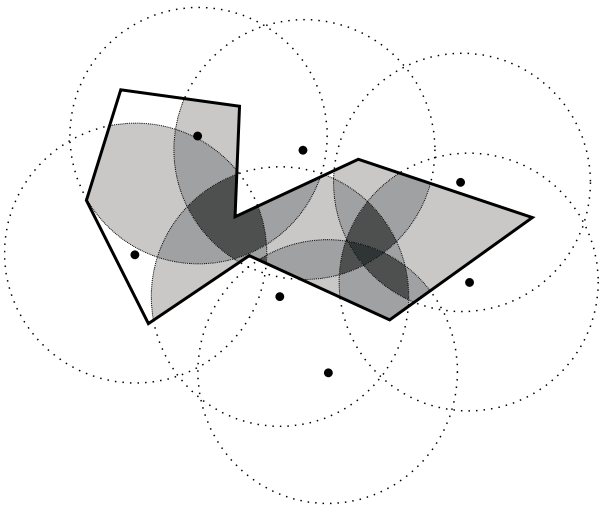


Figure 2: The coverage of the region is given by the $k$-perimeter-coverage of every sensor. The faces of the arrangement $D \cap R$ are coloured according to their coverage, each darker level of gray means one more sensor is covering that face.

This paper is organised as follows: in Section 2 it is shown how to calculate the maximum $k \in \mathbb{N}$ such that the line segment connecting points $p$ and $q$ on the plane is $k$-covered. Such algorithm will then be used in Section 3 to compute the maximum coverage of path between two nodes of a graph. Moreover, it is also shown how to compute such a path. In its turn, the idea introduced in Section 3 helps to solve the problems presented in Sections 4 and 5. The maximum coverage of a path between two points within a polygonal region is solved in Section 4 and the case where the path exists on the

plane is solved in Section 5. The paper concludes in Section 6 with the results.

## 2 Maximum Coverage of a Line Segment

As before, let $S$ a set of $n$ sensors with transmission range $r \in \mathbb{R}^+$. Given two points $p$ and $q$ on the plane, the next algorithm calculates the maximum $k \in \mathbb{N}$ such that the line segment connecting both points, $\overline{pq}$, is $k$-covered. First, $\overline{pq}$ is divided in several pieces and then each piece is analysed separately. Let $I$ be the sorted set of the intersection points between $\overline{pq}$ and $D$, the set of disks of radius $r$ centred at sensors of $S$ (see Figure 3). Each pair of consecutive points of $I$ defines a piece of $\overline{pq}$. It is easy to see that each of these pieces is interior to the same disks. In other words, the number of disks covering each piece of $\overline{pq}$ is constant. Therefore, it suffices to count the number of disks at each intersection point. For each $i \in I$, $\mathrm{MC}_S(i)$ can be calculated from the previous one since the number of disks covering two adjacent pieces of $\overline{pq}$ only differs by 1. Consequently, it is only needed to calculate the maximum number of disks covering the first point of $I$, the rest follows by adding or subtracting 1. The next algorithm calculates the maximum coverage of $\overline{pq}$, $\mathrm{MC}_S(\overline{pq})$.

ALGORITHM **Maximum Coverage of a Line Segment**

INPUT: Set $S$ of sensors with range $r$, points $p$ and $q$
OUTPUT: $\mathrm{MC}_S(\overline{pq})$

1. Compute set $I$ of the intersection points between $\overline{pq}$ and $D(s,r), \forall s \in S$.
2. $I \leftarrow I \cup \{p,q\}$. Sort $I$ along $\overline{pq}$: $I = \{i_0, \ldots, i_f\}$.
3. $\mathrm{MC}_S(i_0) \leftarrow |\{D(s,r), \forall s \in S : i_0 \in D(s,r)\}|$.
4. From $j = 1$ to $f$ do
   If a new disk covers $i_j \in I$
   Then $\mathrm{MC}_S(i_j) \leftarrow \mathrm{MC}_S(i_{j-1}) + 1$.
   Otherwise $\mathrm{MC}_S(i_j) \leftarrow \mathrm{MC}_S(i_{j-1}) - 1$.
5. $\mathrm{MC}_S(\overline{pq}) \leftarrow \min\{\mathrm{MC}_S(i_0), \ldots, \mathrm{MC}_S(i_f)\}$.

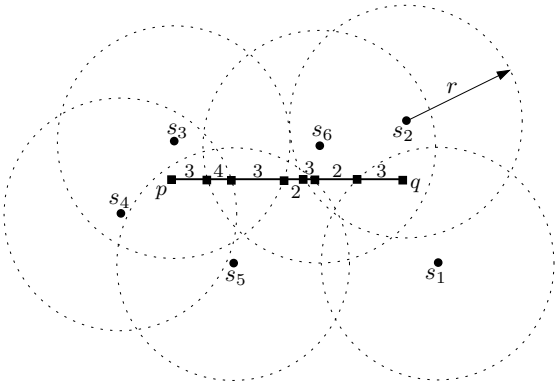



Figure 3: Set $S = \{s_1, \ldots, s_6\}$ with range $r$ is represented by dots. Line segment $\overline{pq}$ is divided in seven pieces and each piece has its coverage number above it, $\mathrm{MC}_S(\overline{pq}) = \min\{3,4,3,2,3,2,3\} = 2$.

**Theorem 2.1.** *Given a set $S$ of $n$ sensors with transmission range $r$ and two points $p$ and $q$, $\mathrm{MC}_S(\overline{pq})$ can be calculated in $\mathcal{O}(n \log n)$ time.*

*Proof.* Since the set $S$ is formed by $n$ sensors on the plane, set $D$ also has $n$ disks. There is a linear number of intersection points between the arrangement of disks and $\overline{pq}$ since each disk can only intersect

$\overline{pq}$ twice. Let $I$ be that set of intersection points plus $p$ and $q$. Consequently, sorting $I$ takes $\mathcal{O}(n \log n)$ time. While being computed, each intersection point $i_j \in I$ is flagged to indicate if a new disk is covering the next piece of $\overline{pq}$ or not. Calculating $\mathrm{MC}_S(i_0)$ can be done in linear time. The number of disks covering the remaining points of $I$ can be easily calculated in constant time per intersection point. Therefore, $\mathrm{MC}_S(\overline{pq}) = \min\{\mathrm{MC}_S(i_0), \ldots, \mathrm{MC}_S(i_f)\}$ can be calculated in linear time. Overall, the time complexity for this procedure is $\mathcal{O}(n \log n)$. □

If there are many queries regarding the same set of sensors, there is another solution which is more efficient. In this case, it is convenient to preprocess the arrangement of disks $D$ which can be done in $\mathcal{O}(n^2)$ time [6]. That way, each line segment connecting two points can only cross a linear number of disks (at most, it crosses the same disk twice). The number of disks that cover each intersection point is given by the faces of the arrangement so the number of disks covering each piece of the line segment is controllable in $\mathcal{O}(n)$ time.

# 3 Maximum Covered Path on a Planar Graph

Let $G = (N, E)$ be a connected geometric planar graph. Calculating the maximum coverage of the whole graph $G$ can be easily done using the next proposition that is a generalisation of Theorem 2.1.

**Proposition 3.1.** *Let $S$ be a set of $n$ sensors with transmission range $r$, $G = (N, E)$ a connected geometric planar graph and $|E| = m$. The maximum coverage of $G$ can be calculated in $\mathcal{O}(mn \log n)$ time.*

Let $n_p$ and $n_q$ be two arbitrary nodes of $G$, a path connecting them using only the edges of $G$ that are $k$-covered by $S$ is a $k$-path (see Figure 4(a)). The main problem in this section is to calculate the maximum $k \in \mathbb{N}$ such that a $k$-path between nodes $n_p$ and $n_q$, $P(n_p, n_q)$, exists on $G$. Such value is denoted by $\mathrm{MC}_{S,G}(P(n_p, n_q))$ or $\mathrm{MC}_S(P(n_p, n_q))$ if graph $G$ is clear from the context.

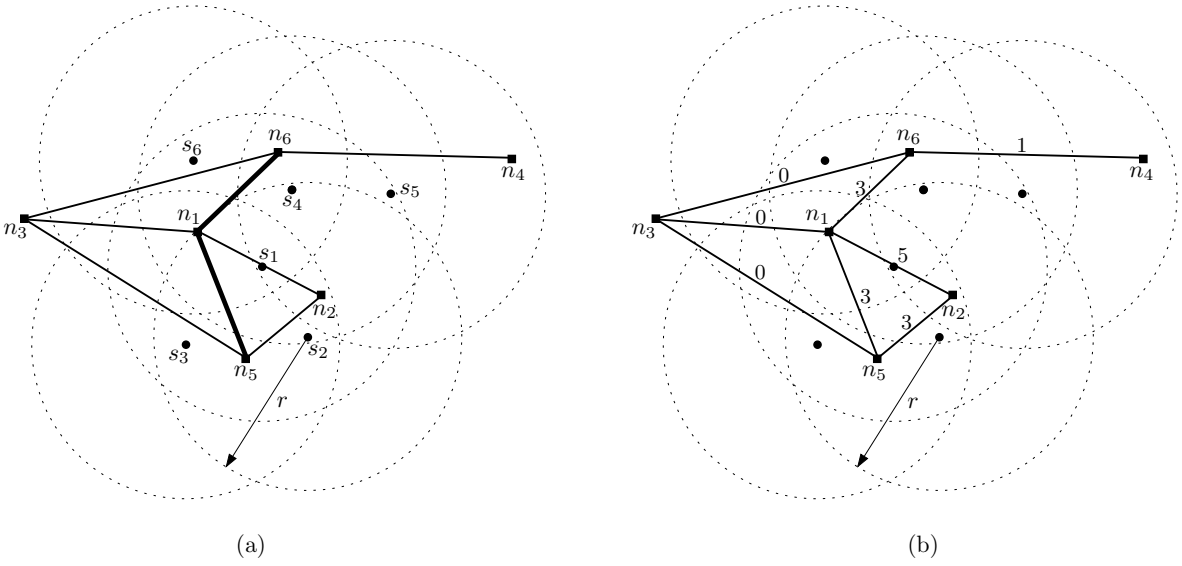



Figure 4: Nodes of graph $G$ are represented by squares. (a) The darker path connecting $n_5$ and $n_6$ is a 3-covered path on $G$. (b) Weighted graph $G_w$, where each edge $e$ has weight $\mathrm{MC}_S(e)$.

Follows an algorithm to calculate $\mathrm{MC}_S(P(n_p, n_q))$. First graph $G$ is transformed into edge-weighted graph $G_w = (N, E_w)$ by assigning the weight $w(e) = \mathrm{MC}_S(e)$ to each edge $e \in E_w$ (see Figure 4(b)). Computing the weights of the edges of $E_w$ has the same time complexity as computing the maximum covering of $G$. Therefore, according to Proposition 3.1 such procedure takes $\mathcal{O}(|E| \times n \log n)$ time. A $k$-path on $G_w$ between two nodes of $N$ can only traverse edges whose weight is greater or equal to $k$. Consequently, a $k$-path between $n_p$ and $n_q$ that maximises the value of $k$ is a path whose lightest

edge has the maximum weight possible. If the problem was the opposite, find a $k$-path connecting two nodes of $G_w$ that minimises the weight of its heaviest edge, then it could be solved by computing a minimum-weight spanning tree (MST). Therefore, the original problem can be seen as the reverse of an MST which naturally is a maximum-weight spanning tree (MaxST). An algorithm to find such tree is well known and its strategy passes through the construction of a new graph by inverting the weights of the original graph. Therefore, an MST of this new graph is then a MaxST of the original graph. Let $T_w$ be a MaxST of $G_w$. Any path on $T_w$ connecting two of its nodes is unique. Let $P$ be a path on $T_w$ whose weight is given by the weight of its lightest edge. The next proposition shows that the weight of $P$ is larger or equal to the weight of any path on $G_w$ between the same nodes.

**Proposition 3.2.** *Let $G_w$ be an edge-weighted connected graph. For any path $P$ on $G_w$, let its weight be defined by the weight of its lightest edge. Then, for every pair of nodes of $G_w$, the path connecting them on a* MaxST *of $G_w$ is a maximum-weight path between such pair.*

*Proof.* Let $T_w$ be a maximum-weight spanning tree (MaxST) of $G_w$. Suppose that $P$ is the only path on $T_w$ connecting nodes $n_i$ and $n_j$ and $e$ is its lightest edge. Consequently, $P$ has weight $w(e)$. Now suppose that path $P^*$ on $G_w$ is a maximum-weight path connecting $n_i$ and $n_j$. Its weight is given by $e^*$, its lightest edge, and so $w(e^*) > w(e)$. Since $P$ is lighter than $P^*$, the edge $e$ is not part of $P^*$. If paths $P$ and $P^*$ are united, then a cycle is created. That cycle contains $e$ which clearly is its lightest edge. But that contradicts the hypothesis that $e$ is an edge of a MaxST of $G_w$. □

The MaxST of $G_w$ found by the previous algorithm can now be used to solve the original problem. The maximum $k \in \mathbb{N}$ such that a $k$-path between two nodes of $G_w$ exists can be calculated as the weight of the only path that exists between those nodes on a MaxST of $G_w$.

**Theorem 3.3.** *Let $G_w = (N, E_w)$ be a planar connected edge-weighted graph with $m$ edges, $|E_w| = m$. Given two nodes $n_p$ and $n_q$ of $G_w$, $\mathrm{MC}_{S,G_w}(P(n_p, n_q))$ and a maximum covered path between $p$ and $q$ can be calculated in $\mathcal{O}(m)$ time.*

*Proof.* Constructing a MaxST of $G_w$ by computing an MST takes $\mathcal{O}(m)$ time using an algorithm by Matsui [11] (since $G_w$ is a planar graph). According to Proposition 3.2, the path connecting $n_p$ and $n_q$ on a MaxST of $G_w$, $T_w = (N, B)$, is a maximum-weight path connecting those nodes on $G_w$. Such path can be found by traversing $T_w$ using using the Depth-First Search [7] technique which takes $\mathcal{O}(|B|)$ time. The path just computed on $T_w$ is a maximum covered path between $p$ and $q$ and its weight is $\mathrm{MC}_{S,G_w}(P(n_p, n_q))$. □

# 4 Maximum Coverage of a Path between Two Points on a Polygonal Region

As it was referred before, the maximum coverage of a polygonal region was previously solved by Huang and Tseng [8]. In this section we solve a related but different problem: given a set $S$ of sensors with transmission range $r$, a polygonal region $R$ and two points $p$ and $q$ on $R$, find the maximum integer $k$ such that a $k$-path connecting $p$ and $q$ exists on $R$. This problem can be solved using a similar idea to the one presented in the last section. In order to do that, there is the need to construct an edge-weighted graph. The arrangement of disks of set $D = \{D(s, r), \forall s \in S\}$ divides the plane in sets of points that are covered by the same disks of $D$. Let $D_R$ be the arrangement of disks of $D$ confined to $R$ since that is the useful part of the arrangement to this problem (see Figure 5). For each face of $D_R$, compute the number of disks that cover it. Next compute the dual graph of $D_R$, denoted by $D'_R$, whose nodes are the faces of $D_R$. There is an edge between two nodes of $D'_R$ if the corresponding primal faces of $D_R$ share an edge. Points $p$ and $q$ act as nodes of $D'_R$ representing the faces that contain them. Since each face of $D_R$ has an associated weight, the dual nodes of $D'_R$ are also weighted. Graph $D'_R$ can be transformed into an edge-weighted graph if each edge is assigned the minimum weight of the nodes it connects (see Figure 6). Now it is possible to compute a MaxST of $D'_R$. Such tree helps to compute the maximum covering of a path $P$ between $p$ and $q$ on $R$, $\mathrm{MC}_{S,R}(P(p, q))$.

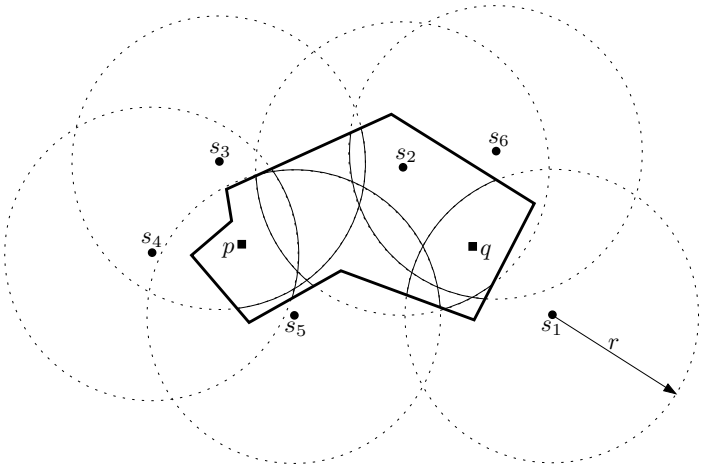



Figure 5: Set of disks of radius $r$ centred at the sensors of $S$. The arrangement defined by set $D$ restricted to region $R$, $D_R$, is shown in a solid trace.

Although the next algorithm does not compute a maximum covered path between $p$ and $q$, it gives a sequence of faces of $D_R$ that can be traversed to form such a $k$-path between $p$ and $q$. In the following, let $i$ and $j$ be faces of $D_R$ and $i'$ and $j'$ the respective dual nodes of $D'_R$. The weight of face $i$ is represented by $w(i)$ and $w(\overline{i'j'})$ denotes the weight of the edge connecting nodes $i'$ and $j'$.

ALGORITHM **Maximum Coverage of a Path on a Region**

INPUT: Set $S$ with range $r$, region $R$ and points $p$ and $q$
OUTPUT: $\mathrm{MC}_{S,R}(P(p,q))$

1. Construct $D_R$, the arrangement of $D$ restricted to $R$.
2. For each face $i$ of $D_R$ do:
   $w(i) \leftarrow |D(s,r), \forall s \in S : i \in D(s,r)|$
3. Construct the graph $D'_R$, dual of $D_R$.
4. For every dual edge $\overline{i'j'} \in D'_R$ do
   $w(\overline{i'j'}) \leftarrow \min\{w(i), w(j)\}$
5. Compute $T$, a MaxST of $D'_R$.
   Find a path $P(p,q)$ on $T$.
6. $\mathrm{MC}_{S,R}(P(p,q))$ is the weight of the lightest edge of $P(p,q)$.

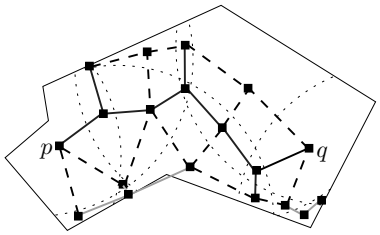



Figure 6: Dual graph $D'_R$ of the arrangement $D_R$, gray edges weight 1, the dashed weight 2 and the ones represented by a heavier black trace weight 3.

In the example in Figure 6, the maximum covering of a path between $p$ and $q$ on $R$ is 3. This means that there is not any other $k$-path covered by $S$ between $p$ and $q$ for $k > 3$.

**Theorem 4.1.** *Let $S$ be a set of $n$ sensors with transmission range $r$. Given two points $p$ and $q$ on a polygonal region $R$ with $m$ edges, $\mathrm{MC}_{S,R}(P(p,q))$ can be calculated in $\mathcal{O}(n(n+m)\log(n+m))$ time.*

*Proof.* Each disk centred at sensors of $S$ may intersect all of the other disks, so the number of intersection points between the disks is $\mathcal{O}(n^2)$. Further, observe that each edge of $R$ can only intersect each disk twice, so at most it intersects $n$ disks. Therefore, $R$ intersects $D$ at most $mn$ times. Consequently, the total number of intersection points in the arrangement of $D$ and $R$ is $\mathcal{O}(n^2+mn)$. The arrangement $D_R$ can be found by sweeping the plane, as long as the disks of $D$ are segmented in a way that each piece is monotone. Such sweep can be done in $\mathcal{O}(n(n+m)\log(n+m))$ time and the weight of each face of $D_R$ can be calculated during the sweep. Since $D_R$ has $\mathcal{O}(n(n+m))$ faces and its dual graph is planar, the edge-weighted dual graph $D'_R$ can be computed in $\mathcal{O}(n(n+m))$ time since that is its number of edges. For the same reason, calculating $\mathrm{MC}_{S,R}(P(p,q))$ using a MaxST of $D'_R$ takes $\mathcal{O}(n(n+m))$ time according to Theorem 3.3. □

# 5 Maximum Coverage of a Path on the Plane

Let $p$ and $q$ be two points on the plane, this last algorithm calculates the maximum $k \in \mathbb{N}$ so that there is a $k$-path between $p$ and $q$, denoted by $\mathrm{MC}_S(P(p,q))$. Since the path between $p$ and $q$ is not restricted to a graph or a polygonal region, this problem can be seen as a generalisation of both of them. The next solution is very similar to the one just presented and so a pseudo-code of the algorithm is omitted. Consider the arrangement of disks of $D$. The weight of each face of the arrangement is given by the number of disks that cover it. When the dual graph $D'$ is computed, each face of $D$ becomes a node of $D'$. Consequently, every dual node has a weight associated. Graph $D'$ is then transformed into an edge-weighted graph by assigning each edge the weight given by the minimum weight of the nodes it connects. In Figure 7 there is an example of arrangement $D$ and its dual $D'$. After computing a maximum-weight spanning tree (MaxST) of $D'$, the weight of the path connecting $p$ and $q$ on the tree is $\mathrm{MC}_S(P(p,q))$. As in the previous section, the algorithm just explained does not compute a maximum covered path between $p$ and $q$, but it gives a sequence of faces of $D$ that can be traversed to form a maximum covered path between $p$ and $q$.

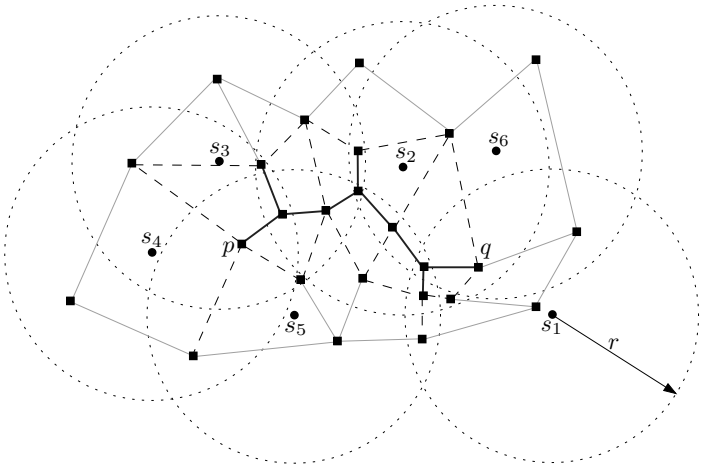



Figure 7: Dual graph of the arrangement of disks $D$, the gray edges weight 1, the dashed weight 2 and the ones represented by a heavier black trace weight 3.

In the example in Figure 7, the maximum coverage of a path between $p$ and $q$ is 3. This means that there is not any other $k$-path covered by $S$ between $p$ and $q$ for $k > 3$.

**Theorem 5.1.** *Let $S$ be a set of $n$ sensors with transmission range $r$. Given two points $p$ and $q$ on the plane, the maximum $k \in \mathbb{N}$ so that a $k$-path between $p$ and $q$ exists can be calculated in $\mathcal{O}(n^2)$ time.*

*Proof.* Since the set $S$ is formed by $n$ sensors on the plane, set $D$ also has $n$ disks. This arrangement of disks can be computed in $\mathcal{O}(n^2)$ time [6] since all disks have the same radius $r$. The weights of the

faces of $D$ are calculated using the information of arrangement $D$. Therefore, the weights of the edges of $D'$ are directly calculated as the minimum weight of their endpoints. Since $D$ may have a quadratic number of faces, $D'$ also has a quadratic number of edges because it is a planar graph. Computing a MaxST of $D'$ and calculating $\mathrm{MC}_S(P(p,q))$ takes $\mathcal{O}(n^2)$ time according to Theorem 3.3. □

# 6 Results

Given a set $S$ of $n$ sensors with transmission range $r$, the problems presented in this paper aimed to compute the maximum $k \in \mathbb{N}$ so that a $k$-path between two points on a region exists. The first two sections study the problem when the region is degenerated into a line segment and a planar graph. Given two points, the maximum coverage of the line segment connecting them is calculated in $\mathcal{O}(n \log n)$ time. In the case of the planar graph $G$ with $m$ edges, the maximum covered path between two arbitrary nodes of $G$ can be found in $\mathcal{O}(m)$ time after a preprocess that runs in $\mathcal{O}(mn \log n)$ time. For two points lying in a polygonal region $R$ with $m$ edges, the algorithm presented in Section 4 computes the maximum coverage of a path connecting both points in $\mathcal{O}(n(n+m)\log(n+m))$ time and outputs a sequence of regions that contain a maximum covered path between the points. Finally, if the region is seen as the whole plane, the maximum coverage of a path between two points can be computed in $\mathcal{O}(n^2)$ time.

# References

[1] M. Abellanas, A.L. Bajuelos, I. Matos, Safe Routes on a Street Graph with Minimum Power Transmission Range, *Proceedings of the* $25^{th}$ *European Workshop on Computational Geometry* (2009), 85–88.

[2] M. Abellanas, A.L. Bajuelos, I. Matos, 2-Covered Paths by a Set of Antennas with Minimum Power Transmission Range, *Information Processing Letters* (to appear). doi: 10.1016/j.ipl.2009.03.016

[3] M. Abellanas, G. Hernández, Optimización de Rutas de Evacuación, *Proceedings of the XII Encuentros de Geometría Computacional* (2007), 273–280.

[4] P. Belleville, P. Bose, J. Czyzowicz, J. Urrutia, J. Zaks, $K$-Guarding Polygons on the Plane, *Proceedings of the* $6^{th}$ *Canadian Conference in Computational Geometry* (1994), 381–386.

[5] A. Boukerche, X. Fei, R.B. Araujo, An Energy Aware Coverage-Preserving Scheme for Wireless Sensor Networks, *Proocedings of the* $2^{nd}$ *ACM international Workshop on Performance Evaluation of Wireless ad hoc, Sensor and Ubiquitous Networks*, ACM, Montreal (2005), 205–213.

[6] B.M. Chazelle, D.T. Lee, On a Circle Placement Problem, *Computing* 36, 1-2 (1986), 1–16.

[7] S.W. Golomb, L.D. Baumert, Backtrack Programming, *Journal of ACM* 12, 4 (1965), 516–524.

[8] Chi-Fu Huang, Yu-Chee Tseng, The Coverage Problem in a Wireless Sensor Network, *Mobile Networks and Applications* 10, 4 (2005), 519–528.

[9] B. Kaučič, B. Žalikm, $K$-guarding of Polyhedral Terrain, *International Journal of Geographical Information Science* 18, 7 (2004), 709–718.

[10] X. Li, P. Wan, O. Frieder, Coverage in Wireless Ad-hoc Sensor Networks, *IEEE Transactions on Computers* 52, 6 (2003), 753–763.

[11] T. Matsui, The Minimum Spanning Tree Problem on a Planar Graph, *Discrete Applied Mathematics* 58, 1 (1995), 91–94.

[12] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M.B. Srivastava, Coverage Problems in Wireless Ad-hoc Sensor Networks, *Proceedings of the* $20^{th}$ *Annual Joint Conference of the IEEE Computer and Communications Societies* 3 (2001), 1380–1387.

[13] D.P. Mehta, M.A. Lopez, L. Lin, Optimal Coverage Paths in Ad-hoc Sensor Networks, *Proceedings of the ICC '03, IEEE International Conference on Communications* 1 (2003), 507–511.

# Caminos de desviación mínima local*

Manuel Abellanas† Gregorio Hernández ‡ Vera Sacristán§

**Resumen**

Un camino que conecta dos puntos $s$ y $t$ en el plano es de *desviación mínima* respecto de un conjunto de puntos $S$ si la mayor de las distancias entre un punto del camino y $S$ es la menor posible entre todos los caminos que conectan $s$ y $t$. En este trabajo estudiamos los caminos de desviación mínima que satisfacen además que todo subcamino es también de desviación mínima, a los que llamamos *caminos de desviación mínima local*.

## 1 Introducción

Consideremos un conjunto $S$ de $n$ puntos del plano. La desviación respecto de $S$ de un camino en el plano que conecta dos puntos $s$ y $t$ es la máxima distancia entre un punto del camino y el conjunto $S$ (entendiendo que la distancia de un punto a un conjunto de puntos es la menor entre el punto y los puntos del conjunto). Nos interesan los caminos que conectan puntos del plano alejándose lo menos posible del conjunto $S$. Esta situación ocurre, por ejemplo, cuando se quiere desplazar un receptor de un punto a otro dentro de una región en la que hay una red de antenas emitiendo una señal. Cuanto más próximos estén los puntos del camino a las antenas, mejor será la señal que reciba el receptor en su desplazamiento. En [1] y [5] se describe cómo obtener caminos de desviación mínima entre dos puntos. En general, existen muchos caminos de desviación mínima entre dos puntos, por lo que tiene interés seleccionar entre todos ellos los que satisfagan otras propiedades, como por ejemplo: longitud mínima, giro total mínimo, exposición mínima o máxima a la señal de las antenas, etc..

Este tipo de problemas aparecen en el diseño y análisis de redes de sensores. En [4] se plantean dos problemas: Entre todos los caminos de desviación mínima, hallar el que se mantiene más alejado de los sitios (*Maximal Breach Path*), que se aplica como test a una red de sensores para medir hasta qué punto es fácil escapar a su vigilancia, y hallar el que se mantiene más próximo a los sitios (*Maximal Support Path*), que se aplica para medir la mejor cobertura. En [3] se demuestra que existe un camino de desviación mínima en el grafo de Gabriel, por lo que se puede calcular localmente. También abordan los autores la minimización de la longitud del camino, pero se restringen al grafo de discos unidad (*Unit Disk Graph*), obteniendo una 5/2-aproximación del camino óptimo en dicho grafo. En [5] se subraya que la solución sobre el grafo de discos unidad no se corresponde con la solución al problema en el plano, se mejora el algoritmo de [4] para el problema *Maximal Breach Path* y se calcula el de mínima longitud. Además se indica, sin detallar, cómo hallar el camino de desviación mínima de mínima longitud.

En este trabajo nos interesamos por los caminos que, además de ser de desviación mínima, son tales que cualquier subcamino suyo hereda dicha propiedad. Es decir, todo subcamino es a su vez un camino de desviación mínima entre sus extremos. A estos caminos los llamamos de desviación mínima local. En el apartado 2 damos las definiciones y algunas propiedades. En el apartado 3 damos dos caracterizaciones: una a partir de los diagramas de Voronoi de alcance limitado (el diagrama de Voronoi de alcance limitado $r$ está formado por las intersecciones de las regiones del diagrama de Voronoi

*Parcialmente subvencionado por los proyectos MEC MTM2008-05043, MEC MTM2006-01267 y DURSI 2005SGR00692

†Universidad Politécnica de Madrid, manuel.abellanas@upm.es

‡Universidad Politécnica de Madrid, gregorio@fi.upm.es

§Universitat Politècnica de Catalunya, Vera.Sacristan@upc.edu

ordinario con los discos de radio $r$ centrados en los sitios. En [2] se construye y se analiza su estructura combinatoria y en [6] los denominan *aberrant Voronoi graphs* y los emplean para resolver de forma local el *Maximal Breach Path*) y otra en función de los máximos locales de la desviación del camino. La segunda caracterización nos permite obtener en la sección 4 un algoritmo de complejidad $O(nk)$ para el problema de decisión para caminos formados por segmentos de recta y arcos de circunferencia concéntricos con los puntos de $S$. Finalmente, en la sección 5 planteamos el problema de optimizar la longitud del camino de desviación mínima local y damos algunas propiedades que esperamos conduzcan a resolver el problema.

# 2 Definiciones y propiedades

**Definición 2.1.** Sean $S = \{s_1, \ldots, s_n\}$ un conjunto de $n$ puntos y $\Gamma$ un camino en el plano. La *desviación* de $\Gamma$ respecto de $S$ es $d(\Gamma, S) = \max_{t \in \Gamma} \min_{i=1\ldots n} d(t, s_i)$. Considerando todos los caminos $\Gamma$ entre dos puntos dados del plano, $x$ e $y$, la desviación de $x$ e $y$ respecto de $S$ es $d(x, y, S) = min_{\Gamma} d(\Gamma, S)$.

**Definición 2.2.** Diremos que $\Gamma$ es de *desviación mínima local* si, para cada par de puntos $x$ e $y$ de $\Gamma$, el camino $\Gamma_{|xy}$ es de desviación mínima, esto es, si $d(\Gamma_{|xy}, S) = d(x, y, S)$.

Empecemos dando algunos ejemplos de caminos de desviación mínima local.

**Lema 2.3.** *Dados dos sitios $s = s_i$ y $t = s_j$, el camino entre $s$ y $t$ contenido en el árbol generador euclideo mínimo de $S$ es de desviación mínima local.*

**Lema 2.4.** *Dados dos puntos $s$ y $t$ pertenecientes, respectivamente, a las regiones de Voronoi de dos sitios $s_i$ y $s_j$, sea $\Gamma_1$ el camino entre $s$ y $t$ formado por el segmento $ss_i$, el segmento $s_jt$ y el camino entre $s_i$ y $s_j$ contenido en el árbol generador euclídeo mínimo de $S$. El camino $\Gamma_1$ es de desviación mínima local.*

**Lema 2.5.** *Dados dos puntos $s$ y $t$ pertenecientes, respectivamente, a las regiones de Voronoi de dos sitios $s_i$ y $s_j$, sea $\Gamma_2$ el camino entre $s$ y $t$ formado por los segmentos que unen los puntos medios de los segmentos de $\Gamma_1$, junto con los segmentos que unen sus extremos a $s$ y a $t$, respectivamente. El camino $\Gamma_2$ es de desviación mínima local.*

**Lema 2.6.** *Todo camino a lo largo del cual la desviación sea una función monótona es de desviación mínima local.*

**Proposición 2.7.** *Todo camino a lo largo del cual la desviación sea una función unimodal con un solo mínimo es de desviación mínima local.*

# 3 Caracterizaciones de la desviación mínima local

## 3.1 Caracterización por diagramas de Voronoi de alcance limitado

**Definición 3.1.** Dado un conjunto $S = \{s_1, \ldots, s_n\}$ de $n$ puntos del plano y un número real $r \geq 0$, el *diagrama de Voronoi de $S$ de alcance $r$*, $Vor(S, r)$, está formado por las regiones:

$$Vor(s_i, r) = \{x \in \mathbb{R}^2 \mid d(x, s_i) \leq d(x, s_j) \forall j \neq i \ \wedge \ d(x, s_i) \leq r\}.$$

**Definición 3.2.** Con la notación anterior, llamaremos *puntos críticos* a los puntos medios de las aristas del árbol generador euclídeo mínimo de $S$. Asimismo, llamaremos *valores críticos* a la mitad de las longitudes de las aristas de dicho árbol, que consideraremos ordenadas de forma creciente, $r_1, \ldots, r_{n-1}$.

**Definición 3.3.** Con la notación anterior, llamaremos $G_i$ a la unión de los círculos con centro en los puntos de $S$ y radio $r_i$, para $i = 1, \ldots, n-1$. Asimismo, $G_0 = \emptyset$ y $G_n = \mathbb{R}^2$. Denotaremos $F_i$ la diferencia $F_i = G_i - G_{i-1}$, donde $i = 1, \ldots, n$.

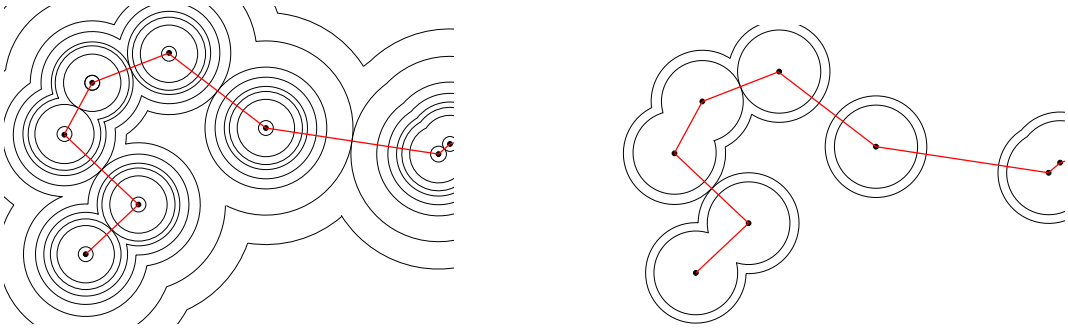


Figura 1: Los conjuntos $G_i$ y $F_i$ en un caso concreto.

La Figura 1 ilustra los conjuntos $G_i$ y $F_i$ de un conjunto de 9 puntos, para todos los valores $i = 1, \ldots, 8$. También se muestra el árbol generador euclídeo mínimo.

**Definición 3.4.** Cada conjunto $G_i$ tiene exactamente una componente conexa, $C_i$, a la que pertenece el punto medio, $m_i$, de la arista $a_i$ de longitud $2r_i$ del árbol generador euclídeo mínimo de $S$. Llamaremos *semicomponentes* de $G_i$ a las dos componentes conexas de $C_i - \{m_i\}$.

Estamos en condiciones de presentar la caracterización de los caminos simples de desviación mínima local entre dos puntos $s$ y $t$, en términos de su comportamiento a lo largo de las regiones $G_i$ y $F_i$.

**Teorema 3.5.** *Un camino $\Gamma$ simple entre dos puntos $s$ y $t$ del plano es de desviación mínima local si, y sólo si,*

1. *La intersección de $\Gamma$ con cada componente conexa de $G_i$ es conexa.*
2. *En la intersección de $\Gamma$ con cada componente conexa del interior de cada $F_i$, la desviación es una función unimodal con un único mínimo, sea o no la intersección continua.*

*Demostración.* La primera condición es necesaria. Supongamos que la intersección de $\Gamma$ con una de las componentes conexas de $G_i$ tuviera dos componentes correspondientes, en cierta parametrización de $\Gamma$, a los intervalos del parámetro $t \in [a, b]$ y $t \in [c, d]$, donde $b < c$. En tal caso, la porción del camino correspondiente a $t \in [b, c]$ pasaría por $F_{i+1}$ y, por lo tanto, el camino entre los puntos $x = \Gamma(b)$ y $y = \Gamma(c)$ no sería de desviación mínima, ya que, tal como se ilustra en la Figura 2, sería posible establecer un camino entre $x$ e $y$ de desviación constante, recorriendo la frontera de $G_i$, mientras que $\Gamma$ conecta $x$ e $y$ con desviación mayor.

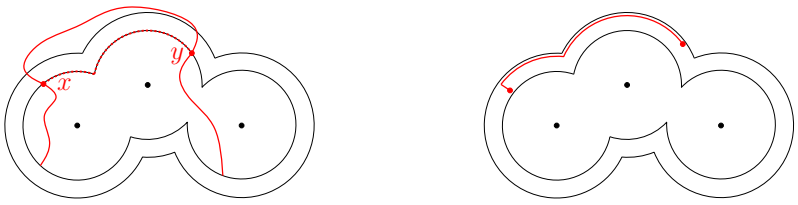



Figura 2: A la izquierda, ejemplo de camino de desviación constante entre $x$ e $y$ en una componente de $G_i$. A la derecha, ejemplo de camino monótono entre dos puntos de una misma componente del interior de una franja $F_i$.

La segunda condición es necesaria. Si $\Gamma$ no cumpliera dicha condición, tendría un máximo estricto. En un entorno de ese máximo, y dentro de la componente de $F_i$, existirían dos puntos del camino, $x$ e $y$, a lados distintos del máximo, con desviación menor a la del máximo. Entre estos dos puntos, el camino no sería de desviación mínima ya que, dada la forma de $F_i$, sería posible establecer un camino alternativo entre ellos, con desviación menor a la desviación a través del máximo (que es la única a lo largo del camino, ya que éste es simple). Se trata del camino siguiente: de los dos puntos, se toma el

que más dista del sitio $s_i$ de la región de Voronoi en que se encuentra, y se avanza de forma equidistante a la frontera de $F_i$ hasta poder acercarse de forma monótona al otro punto, tal como se ilustra en la Figura 2.

Estas condiciones son, además, suficientes. Tomemos dos puntos cualesquiera, $x$ e $y$, en un camino $\Gamma$ que las satisfaga. Se trata de demostrar que $\Gamma_{|xy}$ es de desviación mínima. Sea $i$ el índice más pequeño tal que $x$ e $y$ pertenecen a la misma componente conexa de $G_i$. Hay dos casos.

Si $x$ e $y$ pertenecen cada uno a una semicomponente distinta de $G_i$, entonces la propiedad 1 garantiza que el camino entre $x$ e $y$ está totalmente contenido en la componente de $G_i$ y también que su desviación máxima se alcanza en el punto medio, $m_i$, de la arista $a_i$, que no es otro que el punto de tangencia entre las dos semicomponentes. Por consiguiente, el camino entre $x$ e $y$ es de desviación mínima, ya que cualquier camino entre $x$ e $y$ que salga de la componente conexa de $G_i$ tiene desviación mayor, y cualquiera que no salga debe pasar por $m_i$.

En caso contrario, $x$ e $y$ pertenecen a la misma componente del interior de $G_i$. Recuérdese que $i$ es el mínimo índice tal que $x$ e $y$ pertenecen a la misma componente conexa de $G_i$. Eso garantiza que al menos uno de los dos pertenece a la franja $F_i$. En el caso de que ambos pertenezcan a $F_i$, consideremos el más alejado del sitio $s_i$ de la región de Voronoi en que se encuentra. Sin pérdida de generalidad, puede suponerse que se trata de $x$. En el caso de que sólo uno pertenezca a $F_i$, sea éste $x$. La propiedad 2 garantiza que el camino $\Gamma$ es unimodal con un único mínimo en $F_i$, de modo que la desviación de $\Gamma_{|xy}$ viene dada por $d(x, s_i)$ y, por consiguiente, es mínima. □

La segunda condición de la caracterización anterior de los caminos de desviación mínima local puede reescribirse del modo siguiente:

**Proposición 3.6.** *Son equivalentes:*

- *2) En la intersección de* $\Gamma$ *con cada componente conexa del interior de cada* $F_i$*, la desviación es una función unimodal con un único mínimo, sea o no la intersección continua.*
- *2') La intersección de* $\Gamma$ *con cada componente conexa del interior de cada* $F_i$ *puede tener una única componente conexa, en cuyo caso es unimodal con un único mínimo, o tener dos componentes conexas, en cuyo caso ambas son monótonas con el mínimo en la frontera entre* $F_i$ *y* $F_{i-1}$*.*

## 3.2 Otras caracterizaciones

También es posible caracterizar los caminos simples de desviación mínima local en términos de sus máximos locales. Para ello se requiere la siguiente

**Definición 3.7.** Un punto $p \in \Gamma$ es un *máximo local estricto* de $\Gamma$ si existe un entorno (a ambos lados) de $p$ en $\Gamma$ en el que todos los puntos tienen desviación estrictamente inferior a la desviación de $p$.

Esta definición se extiende de forma natural a intervalos de puntos del camino.

**Definición 3.8.** Un intervalo (esto es, una porción conexa) $\Gamma'$ de $\Gamma$ es un *máximo local estricto* de $\Gamma$ si todos los puntos de $\Gamma'$ tienen la misma desviación y existe un entorno (a ambos lados) de $\Gamma'$ en el que todos los puntos tienen desviación estrictamente inferior.

De este modo, en ambos casos se tienen puntos a lo largo de $\Gamma$, a un lado y a otro del máximo estricto, con desviación estrictamente inferior. En particular, los extremos del camino no son ni pertenecen a un máximo estricto.

Estamos ya en condiciones de ofrecer una segunda caracterización de los caminos simples de desviación mínima local.

**Teorema 3.9.** *Un camino simple* $\Gamma$ *es de desviación mínima local si, y sólo si, todos sus máximos locales estrictos son o contienen un punto crítico y, en él,* $\Gamma$ *cambia de región de Voronoi.*

*Demostración.* La condición es suficiente. Considérense dos puntos $x$ e $y$ de $\Gamma$. Si entre ellos $\Gamma$ no tiene ningún máximo local estricto, entonces $\Gamma_{|xy}$ es unimodal con un único mínimo y, por lo tanto, de desviación mínima local. En particular, es de desviación mínima.

Si entre ellos hay algún máximo local estricto, consideremos el que corresponde al valor crítico mayor, $r_i$. En este caso, veremos que la desviación del par de puntos $x$ e $y$ es igual a $r_i$. Para ello demostraremos que $x$ e $y$ pertenecen necesariamente a semicomponentes distintas de $G_i$. Obsérvese que, si $x$ e $y$ pertenecieran a componentes distintas de $G_i$, entonces $\Gamma$, para conectar $x$ e $y$, tendría que atravesar la franja $F_{i+1}$ y, por consiguiente, $\Gamma_{|xy}$ alcanzaría un máximo estrictamente mayor que $r_i$, contradiciendo el hecho de que todos los máximos se alcanzan en puntos críticos. Por otro lado, si $x$ e $y$ pertenecieran a una misma semicomponente de $G_i$, y puesto que $\Gamma_{|xy}$ pasa, a través del punto crítico, a la otra semicomponente y es simple, el camino, en su recorrido entre $x$ e $y$, debe volver a entrar en la semicomponente inicial y, para hacerlo, debe pasar por $F_{i+1}$, lo cual, como se ha visto ya, no puede ocurrir. En conclusión, $x$ e $y$ pertenecen a semicomponentes distintas de $G_i$, tal como se quería demostrar.

La condición es necesaria. Si $\Gamma$ tiene un máximo local estricto $p$ que no es o no contiene un punto crítico, o bien si dicho máximo local es un punto crítico pero el camino no cruza por él de una región de Voronoi a otra, entonces $\Gamma$ no es de desviación mínima local, pues podemos utilizar el razonamiento usado en la demostración del Teorema 3.5 para comprobar que la condición segunda es necesaria. □

Para finalizar este apartado de caracterizaciones, observaremos que la desviación mínima local es una propiedad local (valga la aparente redundancia).

**Definición 3.10.** Un camino simple $\Gamma$ es *localmente* de desviación mínima si para cada punto $p \in \Gamma$ y para cada intervalo $I \subset \Gamma$ de desviación constante, existe un número real $\delta > 0$ tal que $\Gamma' = \Gamma \cap B(p, \delta)$ (respectivamente, $\Gamma' = \Gamma \cap B(I, \delta)$) es un camino de desviación mínima entre sus extremos.

**Teorema 3.11.** *Un camino simple es de desviación mínima local si, y sólo si, es localmente de desviación mínima.*

*Demostración.* Obviamente, todo camino de desviación mínima local es localmente de desviación mínima. En cuanto al recíproco, sea $\Gamma$ un camino localmente de desviación mínima, y sean $x$ e $y$ dos puntos de $\Gamma$. Supongamos que el camino $\Gamma_{|xy}$ no fuera de desviación mínima. Entonces tendría un máximo estricto que no coincidiría ni contendría un punto crítico, o bien que lo contendría pero el camino no cruzaría por él de una región de Voronoi a otra. En cualquiera de estos casos, es trivial comprobar que $\Gamma$ no sería localmente de desviación mínima en $p$ (respectivamente, en $I$). □

# 4 El problema de decisión

**Teorema 4.1.** *Sea $\Gamma$ un camino simple formado por segmentos de recta y arcos de circunferencias de radios arbitrarios, centrados en los sitios de $S$ y contenidos en la región de Voronoi correspondiente a su centro, y sea $k$ la complejidad de $\Gamma$. Conocidos $Vor(S)$ y las aristas del árbol generador euclídeo mínimo de $S$, es posible decidir en tiempo $O(nk)$ si $\Gamma$ es o no de desviación mínima local.*

*Demostración.* El algoritmo de decisión comienza localizando el origen del camino en $Vor(S)$, es decir, hallando la región de Voronoi en la que se encuentra. A partir de aquí, se va recorriendo la zona de $\Gamma$, es decir, el conjunto de las regiones de Voronoi atravesadas por $\Gamma$, y se va comprobando, región por región, si $\Gamma$ tiene máximos locales estrictos y si éstos coinciden o incluyen puntos críticos.

El coste de este algoritmo es trivialmente $O(nk)$. □

**Proposición 4.2.** *La intersección de un camino poligonal de complejidad $k$ de desviación mínima local con el 1-esqueleto del diagrama de Voronoi de un conjunto de $n$ puntos tiene complejidad $O(nk)$ y existen ejemplos de complejidad $\Omega(nk)$.*

Figura 3: Camino de desviación mínima local que corta $nk$ veces el 1-esqueleto del diagrama de Voronoi de $S$.

*Demostración.* La cota superior es obvia. La inferior se ilustra en la Figura 3.

□

Así, pues, aunque no podemos asegurar que el problema de decisión tenga complejidad $\Omega(nk)$, sí podemos garantizar que cualquier algoritmo que pretenda resolverlo en tiempo $o(nk)$ deberá ser capaz de responder la cuestión sin explorar todos los puntos de corte del camino con las aristas de Voronoi de $S$.

# 5 Optimización de la longitud del camino

Obsérvese que si los puntos $s$ y $t$ están muy alejados de los sitios de las regiones de Voronoi a las que respectivamente pertenecen, o bien si la arista más larga del camino que conecta en el árbol generador euclídeo mínimo los sitios correspondientes a las regiones de Voronoi que contienen a $s$ y a $t$ (denotada arista crítica entre $s$ y $t$ en [1]) es muy larga, entonces existen caminos no sólo de desviación mínima sino incluso de desviación mínima local entre $s$ y $t$ muy alejados de los sitios, tal como se ilustra en la Figura 4.

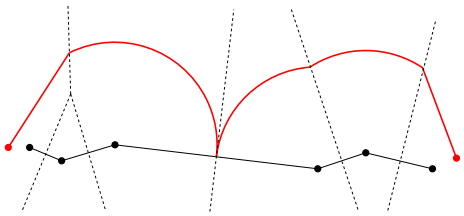


Figura 4: Ejemplo de camino de desviación mínima local que se mantiene muy alejado de los sitios.

Se plantea entonces la cuestión de hallar el camino entre dos puntos $s$ y $t$ que, siendo de desviación mínima local, cumpla algún criterio adicional de optimización, el más natural de los cuales es, posiblemente, el de minimización de la longitud.

**Proposición 5.1.** *El camino de desviación mínima local más corto entre dos puntos de una misma región de Voronoi es rectilíneo.*

*Demostración.* Dentro de una misma región de Voronoi, todo segmento es unimodal con un único mínimo y, por consiguiente, es de desviación mínima local. □

Sin embargo, eso no significa que el camino de desviación mínima local de longitud mínima entre dos puntos situados en regiones de Voronoi distintas sea siempre rectilíneo, ni siquiera poligonal, como se verá más adelante.

**Proposición 5.2.** *Sean $s_i$ y $s_j$ dos sitios adyacentes. Sea $p$ un punto de la arista de Voronoi definida por $s_i$ y $s_j$, distinto del punto medio de los dos sitios. Un camino de desviación mínima local puede pasar por $p$ en línea recta si, y sólo si, su pendiente se encuentra entre las de las rectas tangentes en $p$ a las circunferencias con centro en $s_i$ y $s_j$ que pasan por $p$. Si, en cambio, $p$ es el punto medio de $s_i$ y $s_j$, no hay restricciones sobre la pendiente de la recta.*

La situación se ilustra en la Figura 5.

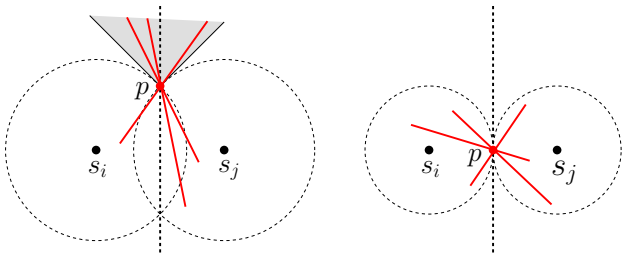



Figura 5: Pendientes permitidas en un camino de desviación mínima local rectilíneo al cruzar una arista de Voronoi.

La tangencia indicada en la proposición anterior se describe mejor en términos de la parábola de foco $s_i$ y cuya directriz es la recta que pasa por $s_j$ y es perpendicular al segmento $s_i s_j$. Denotamos por $par(s_i, s_j)$ esta parábola.

**Lema 5.3.** *Sean $s_i$ y $s_j$ dos sitios adyacentes, y $p$ un punto de su arista de Voronoi, distinto del punto crítico. Una recta $r$ que pase por $p$ es tangente en $p$ a la circunferencia que pasa por $p$ y tiene centro en $s_i$ si, y sólo si, $r$ es tangente a la parábola $par(s_i, s_j)$.*

Con esta herramienta podemos describir cómo son los codos de un camino poligonal simple de desviación mínima local y longitud mínima. El siguiente teorema, que se ilustra en la Figura 6, se presenta sin demostración.

**Teorema 5.4.** *Sea $\Gamma$ un camino poligonal simple. Si $\Gamma$ es de longitud mínima de entre los de desviación mínima local, entonces todos sus vértices se encuentran sobre aristas de Voronoi y, o bien son puntos críticos, o bien son puntos $p$ de la arista de Voronoi de sitios contiguos, $s_i$ y $s_j$, tales que:*

1. *El segmento $p^- p$ es tangente a la parábola $par(s_i, s_j)$ y no contiene el punto de tangencia de su recta de soporte con dicha parábola.*
2. *El segmento $pp^+$ corta al interior del círculo que pasa por $p$ y está centrado en $s_j$, y el giro $p^- pp^+$ aleja el camino $\Gamma$ de $par(s_i, s_j)$.*

*Los puntos $p^-$ y $p^+$ son los puntos de intersección de $\Gamma$ con la frontera de las regiones de Voronoi de $s_i$ y $s_j$, respectivamente, o bien los puntos inicial o final de $\Gamma$, si éstos se encuentran en el interior de dichas regiones. Lo mismo puede darse intercambiando los papeles de $s_i$ y $s_j$.*

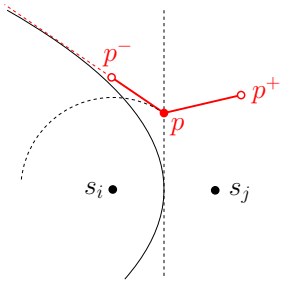



Figura 6: Codo permitido en un camino poligonal de desviación mínima local y longitud mínima.

**Observación 5.5.** El camino de desviación mínima local más corto entre dos puntos puede no ser poligonal.

En la Figura 7 se muestra un ejemplo con dos sitios $s_1$ y $s_2$. Elegimos $s \in Vor(s_1)$ y $t \in Vor(s_2)$ situados de tal forma que 1) cada uno de ellos se encuentra en el interior de la región convexa delimitada por la parábola correspondiente a su sitio, 2) ambos están suficientemente alejados de sus sitios

respectivos y 3) ambos se encuentran a un mismo lado de la recta que une los dos sitios. Cualquier camino entre $s$ y $t$ debe cortar la mediatriz de $s_1s_2$ en algún punto $p$. Dada la posición relativa de $s$ y $t$ respecto de las parábolas $par(s_1, s_2)$ y $par(s_2, s_1)$, los segmentos $sp$ y $pt$ inciden en $p$ por el interior de los círculos que pasan por $p$ y tienen centros en $s_1$ y $s_2$, de forma que el único camino poligonal entre $s$ y $t$ que es de desviación mínima local es el que pasa por el punto medio de $s_1$ y $s_2$. Sin embargo, es posible construir caminos de desviación mínima local más cortos entre $s$ y $t$. En particular, el camino indicado de color rojo en la Figura 7, que se obtiene por concatenación del segmento $sq$, el arco de circunferencia $qp$ y el segmento $pt$, donde $sq$ es tangente a la circunferencia.

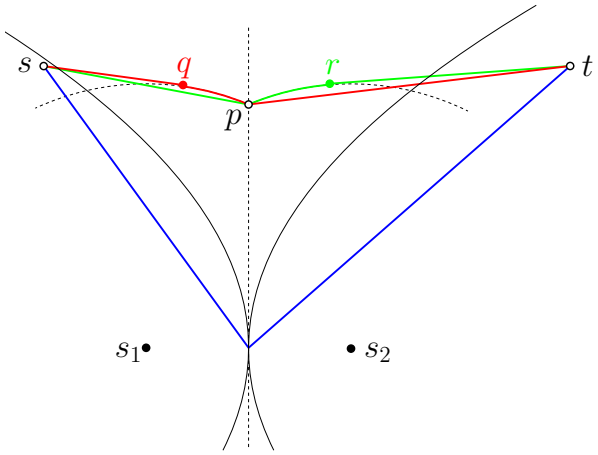



Figura 7: El camino de desviación mínima local más corto entre $s$ y $t$ no es poligonal, ya que los caminos $sqpt$ y $sprt$ son de desviación mínima.

## Referencias

[1] M. Abellanas, G. Hernández. Optimización de rutas de evacuación, *Actas XII Encuentros de Geometría Computacional, EGC'07*, pp. 273-280, 2007.

[2] M. Abellanas, G. Hernández, J. L. Moreno, S. Ordóñez, V. Sacristán. DVALon: una herramienta para diagramas de Voronoi y grafos de proximidad de alcance limitado, presentado a *XIII Encuentros de Geometría Computacional, EGC'09*.

[3] X.-Y. Li, P.-J. Wan, O. Frieder. Coverage in Wireless Ad Hoc Sensor Networks, *IEEE Transactions on Computers*, 52, 6, pp. 753-763, 2003.

[4] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava. Coverage Problems in Wireless Ad-hoc Sensor Networks, *Proc. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2001*, pp. 1380-1387, 2001

[5] D. P. Mehta, M. A. Lopez, L. Lin. Optimal Coverage Paths in Ad-hoc Sensor Networks, *Proc. IEEE International Conference on Communications, ICC'03*, pp. 507- 511, 2003.

[6] H. Xu, L. Huang, Y. Wan, K. Lu. Localized Algorithm for Coverage in Wireless Sensor Networks, *Proc. IEEE Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05)*, pp. 750-754, 2005.

# *DVALon*: una herramienta para diagramas de Voronoi y grafos de proximidad de alcance limitado*

Manuel Abellanas† Gregorio Hernández‡ José Luis Moreno§
Sergio Ordóñez ¶ Vera Sacristán‖

**Resumen**

Presentamos un estudio de los diagramas de Voronoi de alcance limitado de un conjunto finito de puntos del plano y *DVALon*, una herramienta para la manipulación de dichos diagramas, los grafos de proximidad de alcance limitado y su aplicación a los caminos de desviación mínima y de separación máxima.

## 1 Introducción

Recientemente, los diagramas de Voronoi y, más en general, los grafos de proximidad están siendo objeto del interés de comunidades científicas tan dispares como las surgidas alrededor del análisis y el desarrollo de redes de sensores inalámbricos [8] o del estudio de la evolución de los bosques [1].

En ingeniería forestal se emplea el concepto de región potencialmente disponible de un árbol (*APA: Area Potentially Available*) para estudiar la influencia entre los árboles vecinos en el desarrollo de un bosque. La primera referencia de la que hay constancia es del ingeniero alemán Koenig y se remonta a 1864. Existen varias descripciones de dicha región. Smith en 1987 [11] hace un repaso de las primeras aproximaciones y llega a la conclusión de que los diagramas de Voronoi y los algoritmos que proporciona la Geometría Computacional son los instrumentos adecuados para su modelización. En este caso, las regiones de influencia son necesariamente limitadas por la capacidad de crecimiento de cada especie vegetal. En [10] se emplean las propiedades de los diagramas de Voronoi obtenidos como resultado de la expansión de círculos para estudiar las islas verdes que se crean en los incendios forestales (zonas rodeadas de fuego).

Sean éstas *ad hoc* o fijas, las redes de sensores tienen aplicaciones en la monitorización y el control de procesos industriales, el seguimiento de parámetros medioambientales, el control del tráfico, la automatización y la vigilancia domésticas, de edificios públicos y de oficinas, los cuidados sanitarios y sociales, la detección precoz de incendios, el seguimiento de movimientos sísmicos, las redes de radares y sónares, la detección de partículas químicas o nucleares en el aire, el control de robots móviles y, por descontado, las antenas de telefonía móvil.

En ninguno de estos casos sería realista asumir que los sensores sean capaces de detectar calor, presión, sonido, luz, campos electromagnéticos, vibración o lo que quiera que sea su especialización, a distancias arbitrariamente grandes, como tampoco que sean capaces de comunicarse independientemente de la distancia a que se encuentren los unos de los otros.

*Parcialmente subvencionado por los proyectos MEC MTM2008-05043, MEC MTM2006-01267 y DURSI 2005SGR00692

†Universidad Politécnica de Madrid, manuel.abellanas@upm.es
‡Universidad Politécnica de Madrid, gregorio@fi.upm.es
§Universitat Politècnica de Catalunya, jlmd86@gmail.com
¶Universitat Politècnica de Catalunya, sarchio86@gmail.com
‖Universitat Politècnica de Catalunya, vera.sacristan@upc.edu

Es en estos contextos donde cobra interés el diagrama de Voronoi de alcance limitado: una descomposición del plano en regiones, cada una de ellas más cercana a un sensor -a un árbol- que a cualquier otro, pero limitadas por cierta distancia a éste. Por ejemplo, en [5] se propone un método de eliminación de redundancia sin pérdida de cobertura en redes de sensores basado en el reconocimiento de los sensores cuya región de Voronoi de alcance limitado coincide con su región de Voronoi ordinaria, y se caracterizan los sensores que intervienen en la frontera del diagrama de Voronoi de alcance limitado. En el mencionado trabajo no se propone una terminología específica, pero en [12] se bautiza el diagrama de Voronoi de alcance limitado con el nombre de *Aberrant Voronoi Graph*.

Del mismo modo, tienen interés y aplicación los grafos de proximidad de alcance limitado: triangulación de Delaunay, grafo de Gabriel, grafo de vecinos relativos, árbol generador mínimo, etc. Estos grafos han sido introducidos independientemente en [6] y [7], donde reciben los nombres de *Unit Delaunay Triangulation*, *Constrained Gabriel Graph* y *Constrained Relative Neighborhood Graph*.

En particular, los diagramas de Voronoi de alcance limitado y sus grafos de proximidad asociados tienen aplicación directa al estudio de caminos de desviación mínima y de separación máxima, tal como se expone en [2, 4]. La importancia de estos caminos deriva de su uso como herramientas de medida de la calidad de una red de sensores para cubrir un cierto territorio o *área de interés* [9].

En este trabajo presentamos un somero estudio de los diagramas de Voronoi de alcance limitado (Apartado 2) y *DVALon*, una herramienta para la manipulación de dichos diagramas, los grafos de proximidad de alcance limitado más relevantes y su aplicación a los caminos de desviación mínima y de separación máxima (Apartado 3).

# 2 Definiciones y propiedades

## 2.1 Definiciones

Dado un conjunto $S = \{s_1, ..., s_n\}$ de $n$ puntos del plano y un número real $r \geq 0$, la *región de Voronoi de alcance* $r$ de un punto $s_i \in S$ es:

$$Vor(s_i, r) = \{x \in \mathbb{R}^2 \mid d(x, s_i) \leq d(x, s_j) \forall j \neq i \ \wedge \ d(x, s_i) \leq r\}.$$

La región $Vor(s_i, r)$ es, pues, la intersección de la región de Voronoi ordinaria $Vor(s_i)$ con el círculo de centro $s_i$ y radio $r$.

El *diagrama de Voronoi de alcance* $r$ del conjunto $S$, $Vor(S, r)$, es la descomposición del plano en las regiones $Vor(s_i, r)$. Si $r = 0$, el diagrama coincide con $S$. Si $r = +\infty$, coincide con el diagrama de Voronoi ordinario, $Vor(S)$. En el primer caso, consta de $n$ componentes conexas, en el último de una sola. Si $r \neq +\infty$, $Vor(S, r)$ puede constar de una o más componentes conexas, cada una de las cuales con posibles agujeros. La Figura 1 ilustra un ejemplo.

## 2.2 Estructura combinatoria de los diagramas

Pese a la continuidad con la que el valor del alcance puede variar a lo largo de la semirrecta real positiva, la estructura combinatoria de $Vor(S, r)$ se mantiene constante a lo largo de ciertos intervalos y sólo experimenta cambios en determinados valores de $r \in [0, +\infty)$ en los que se produce la fusión de componentes conexas, la aparición o desaparición de agujeros en éstas, etc. Estos cambios combinatorios, que también llamamos eventos, son de cuatro tipos, que se describen a continuación.

**Lema 2.1.** *Cuando $r$ alcanza la mitad de la longitud de una arista $s_i s_j$ del árbol generador mínimo euclídeo de $S$, $EMST(S)$, se conectan dos de las componentes conexas de $Vor(S, r)$.*

*Demostración.* Es inmediato ver que cuando $r$ alcanza la mitad de la distancia entre dos puntos $s_i$ y $s_j$ sus regiones $Vor(s_i, r)$ y $Vor(s_j, r)$ quedan conectadas entre sí, por lo que lo único que queda por demostrar es que $s_i$ y $s_j$ pertenecen a componentes conexas distintas antes de este evento. Supongamos

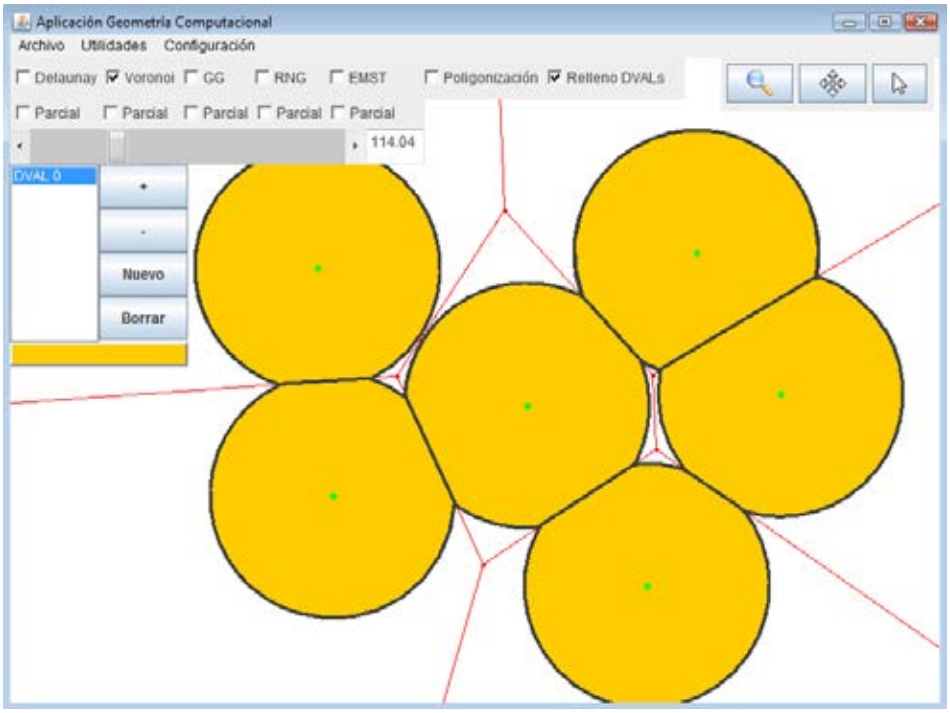



Figura 1: Ejemplo de $Vor(S, r)$ con sus distintas regiones $Vor(s_i, r)$.

que $s_i$ y $s_j$ se encuentran en la misma componente conexa. En tal caso, tendría que existir un camino entre $s_i$ y $s_j$ formado únicamente por aristas de tamaño menor a $2r$ y, si esto fuera cierto, la arista $s_i s_j$ nunca habría sido elegida para formar parte de $EMST(S)$ por formar un ciclo. Por lo tanto, si la arista $s_i s_j$ pertenece a $EMST(S)$, significa que $s_i$ y $s_j$ pertenecen a componentes conexas distintas en $Vor(S, r')$, para todo $r' < r$. □

**Lema 2.2.** *Cuando $r$ alcanza la mitad de la longitud de una arista $s_i s_j$ de Gabriel que no pertenece al árbol generador mínimo euclídeo de $S$, se produce un agujero en el diagrama $Vor(S, r)$.*

*Demostración.* El hecho de que la arista $s_i s_j$ no pertenezca a $EMST(S)$ implica que $s_i$ y $s_j$ pertenecen a una misma componente conexa de $Vor(S, r)$ para algún valor de $r$ menor que la mitad de la longitud de la arista $s_i s_j$, porque existe un camino en $EMST(S)$ que los une, con aristas de longitud estrictamente inferior a $2r$. Por otra parte, en hecho de que la arista $s_i s_j$ sea de Gabriel implica que corta a su arista dual de Voronoi, de modo que los circuncentros de los dos triángulos adyacentes a la arista de Delaunay $s_i s_j$ están uno a cada lado de ésta (son los dos extremos de la arista de Voronoi). Cuando $r$ alcanza la mitad de la longitud de la arista $s_i s_j$, $Vor(S, r)$ contiene tanto la arista $s_i s_j$ como el camino entre $s_i$ y $s_j$ que pasa por aristas de $EMST(S)$ formando un ciclo, pero no contiene un punto interior como es el circuncentro del triángulo que queda rodeado por el ciclo, y que está en el interior de éste, pero a una distancia mayor que $r$ de $s_i$, $s_j$ y de cualquier otro punto de la nube (por ser la triangulación de Delaunay). Resulta así que, al cerrar la arista $s_i s_j$ un ciclo en $Del(S, r)$, queda un conjunto de puntos interiores al ciclo y que no pertenecen a $Vor(S, r)$, es decir, se produce un agujero en $Vor(S, r)$. □

**Lema 2.3.** *Cuando $r$ alcanza el radio de la circunferencia circunscrita a un triángulo de Delaunay cuyo centro es interior al triángulo, es decir, a un triángulo de Delaunay acutángulo, desaparece uno de los agujeros de $Vor(S, r)$ y aparece un nuevo vértice de Voronoi.*

*Demostración.* El radio de la circunferencia circunscrita al triángulo es mayor que la mitad del lado mayor de éste. Cuando $r$ alcanza la mitad de la longitud del mayor de los lados, los tres lados del triángulo quedan contenidos en $Vor(S, r)$ y se crea un agujero que consiste en cierta porción del interior del triángulo. El circuncentro del triángulo es uno de los puntos incluidos en el agujero, por ser interior al triángulo pero estar a una distancia mayor que $r$ de sus tres vértices. Cuando $r$ alcanza la longitud

del radio de la circunferencia circunscrita, todo el interior del triángulo se encuentra a distancia menor o igual que $r$ de alguno de sus tres vértices y, por tanto, todo el triángulo está contenido en $Vor(S,r)$ y desaparece el agujero. Allí donde estaba el circuncentro aparece un vértice de Voronoi, por ser este punto equidistante a los tres vértices del triángulo. □

**Lema 2.4.** *Cuando $r$ alcanza el radio de la circunferencia circunscrita a un triángulo de Delaunay cuyo centro se encuentra en la frontera o en el exterior del triángulo, es decir, a un triángulo de Delaunay rectángulo u obtusángulo, desaparece un arco de circunferencia en la frontera de $Vor(S,r)$ y aparece un nuevo vértice de Voronoi.*

*Demostración.* En el caso obtusángulo, cuando $r$ alcanza la longitud de la mitad del mayor de los lados del triángulo de Delaunay, los tres vértices delimitan tres arcos de circunferencia en la frontera de $Vor(S,r)$. Al incrementar el alcance $r$ en $Vor(S,r)$, el arco correspondiente al vértice que forma el ángulo obtuso se va haciendo más pequeño en la porción que se acerca al circuncentro del triángulo. Cuando $r$ finalmente alcanza el radio de la circunferencia circunscrita al triángulo, el arco desaparece justamente en el punto equidistante a los tres vértices del triángulo (el circuncentro) y, a partir de ese momento, cualquier incremento de $r$ supone la inclusión de nuevos punto en el $Vor(S,r)$ que están más cerca de alguno de los otros dos vértices del triángulo que del vértice obtuso. Allí donde estaba el circuncentro aparece un vértice de Voronoi, por ser este punto equidistante a los tres vértices del triángulo. El razonamiento anterior se adapta de forma inmediata al caso de un triángulo rectángulo, para el cual el punto medio del mayor de los lados coincide con el circuncentro. □

En conclusión, hemos obtenido el siguiente resultado:

**Proposición 2.5.** *Dado un conjunto $S = \{s_1, \ldots, s_n\}$ de $n$ puntos del plano, el número de diagramas de Voronoi de alcance limitado, $Vor(S,r)$, combinatoriamente distintos es $m = O(n)$ y los valores del alcance $r_1, \ldots, r_m$ en que se producen los cambios combinatorios en la estructura de $Vor(S,r)$ son los descritos en los lemas anteriores.*

## 2.3 Construcción de $Vor(S,r)$

En este apartado se estudia la forma de construir $Vor(S,r)$ para un radio $r$ dado y de obtener $Vor(S,r_i)$ para todos los radios $r_i$ que determinan algún cambio combinatorio en $Vor(S,r)$.

**Proposición 2.6.** *Para cualquier valor $r \geq 0$ y cualquier conjunto $S$ de $n$ puntos del plano, es posible calcular el diagrama de Voronoi de alcance $r$, $Vor(S,r)$, a partir del diagrama de Voronoi ordinario, $Vor(S)$, en tiempo $O(n)$.*

*Demostración.* Teniendo ya calculado $Vor(S)$, el cálculo de cada región $Vor(s_i,r)$ se puede realizar cortando la propia región $Vor(s_i)$ con el círculo $B(s_i,r)$, cosa que puede hacerse en tiempo proporcional a la complejidad de la región $Vor(s_i)$, dando lugar a un algoritmo de coste acumulado $O(n)$. □

**Proposición 2.7.** *El problema de obtener todos los diagramas $Vor(S,r_i)$ combinatoriamente distintos de un conjunto $S$ de $n$ puntos dados puede resolverse en tiempo $O(n \log n)$.*

*Demostración.* Se construyen la triangulación de Delaunay $Del(S)$ y el diagrama de Voronoi $Vor(S)$ en tiempo $O(n \log n)$ y se extraen los grafos $EMST(S)$ y $GG(S)$ en tiempo $O(n)$. A partir de aquí se pueden obtener los valores de todos los alcances $r_1, ..., r_m$ que producen cambios combinatorios en $Vor(S,r)$. Obsérvese que $m = O(n)$ y que los valores $r_1, ..., r_m$ se pueden obtener en tiempo $O(n)$ recorriendo las estructuras $Del(S)$, $Vor(S)$, $EMST(S)$ y $GG(S)$. Se ordenan estos valores en tiempo $O(n \log n)$ y a cada uno de ellos se le asigna el cambio combinatorio que le corresponde. Cada uno de estos cambios se puede insertar sucesivamente a partir del anterior en tiempo $O(\log n)$, de modo que el conjunto de todos ellos se construye en tiempo $O(n \log n)$. □

# 3 Descripción de la aplicación

*DVALon* es una aplicación cuyo objetivo es facilitar el estudio y la manipulación de diagramas de Voronoi de alcance limitado y su aplicación a los caminos de desviación mínima. La aplicación permite, dada una nube de puntos, obtener, visualizar y almacenar la información de los diagramas de Voronoi de alcance limitado y de diversos grafos de proximidad del mencionado conjunto de puntos. En los apartados siguientes se ofrece una breve explicación de las funcionalidades más relevantes de la aplicación y de su funcionamiento.

## 3.1 Funcionalidades

Para comenzar con el uso de la aplicación, es necesario introducir una nube de puntos. Para ello, el usuario dispone de tres opciones. La más intuitiva es introducir puntos con el ratón mediante *clics* sobre la pantalla. La segunda opción es introducir el conjunto de puntos mediante un fichero que contenga las coordenadas de los mismos, lo que permite al usuario precisar la posición exacta de éstos. La tercera posibilidad es la de generar el número deseado de puntos distribuidos aleatoriamente en la pantalla de visualización.

Una vez introducido el conjunto de puntos $S = \{s_1, ..., s_n\}$, el usuario puede visualizar distintos grafos de proximidad entre los que se encuentran los diagramas de Voronoi de alcance limitado calculados a partir de $S$.

Inicialmente la aplicación parte de un único $Vor(S, r)$ con alcance igual a cero. El usuario dispone de tres mecanismos para modificar el alcance: una barra deslizante que permite modificar el valor de $r$ de manera continua, un campo en el que se puede introducir un valor determinado del alcance, y un botón con el que ir moviéndose de manera discreta entre los alcances que producen un cambio combinatorio en $Vor(S, r)$.

La aplicación permite, además, crear tantos diagramas de Voronoi de alcance limitado como el usuario desee, así como borrarlos, con lo que es posible visualizar simultáneamente varios diagramas con alcances distintos. Todos los diagramas aparecen de un color distinto en la pantalla y en orden creciente de alcance en una lista. Gracias a esta lista, el usuario puede seleccionar el diagrama que quiera y modificar únicamente el diagrama seleccionado. En la Figura 2 puede verse cómo la aplicación permite visualizar simultáneamente varios diagramas de Voronoi de alcance limitado.

Además de los diagramas de Voronoi de alcance limitado, la aplicación ofrece la posibilidad de visualizar los siguientes grafos de proximidad: la triangulación de Delaunay del conjunto de puntos, $Del(S)$, su grafo de Gabriel, $GG(S)$, su grafo de los vecinos relativos, $RNG(S)$, y su árbol generador mínimo euclídeo, $EMST(S)$. La Figura 3 muestra algunos de estos grafos calculados por la aplicación.

Dado que el interés de este proyecto es trabajar con diagramas de alcance limitado, la aplicación permite visualizar estos grafos no sólo en su versión ordinaria, sino también de manera parcial, es decir, limitados por el alcance de un $Vor(S, r)$ determinado. Esto puede hacerse seleccionando la opción de visualización parcial del grafo (o grafos) deseado, de forma que lo veremos limitado por el alcance del $Vor(S, r)$ seleccionado.

A la hora de guardar un proyecto, la aplicación genera también un fichero de texto con toda la información descrita anteriormente. De esta manera, el usuario, una vez ha guardado su proyecto, dispone también de un fichero de texto con los datos concretos de éste. Asimismo, el usuario siempre puede abrir proyectos ya guardados y recuperarlos tal y como los tenía en el momento en que fueron almacenados.

## 3.2 Funcionamiento

Como ya se ha visto, para que la aplicación comience a funcionar es necesario introducir una nube de puntos. Una vez introducidos como mínimo dos puntos, el programa calcula todos los grafos de

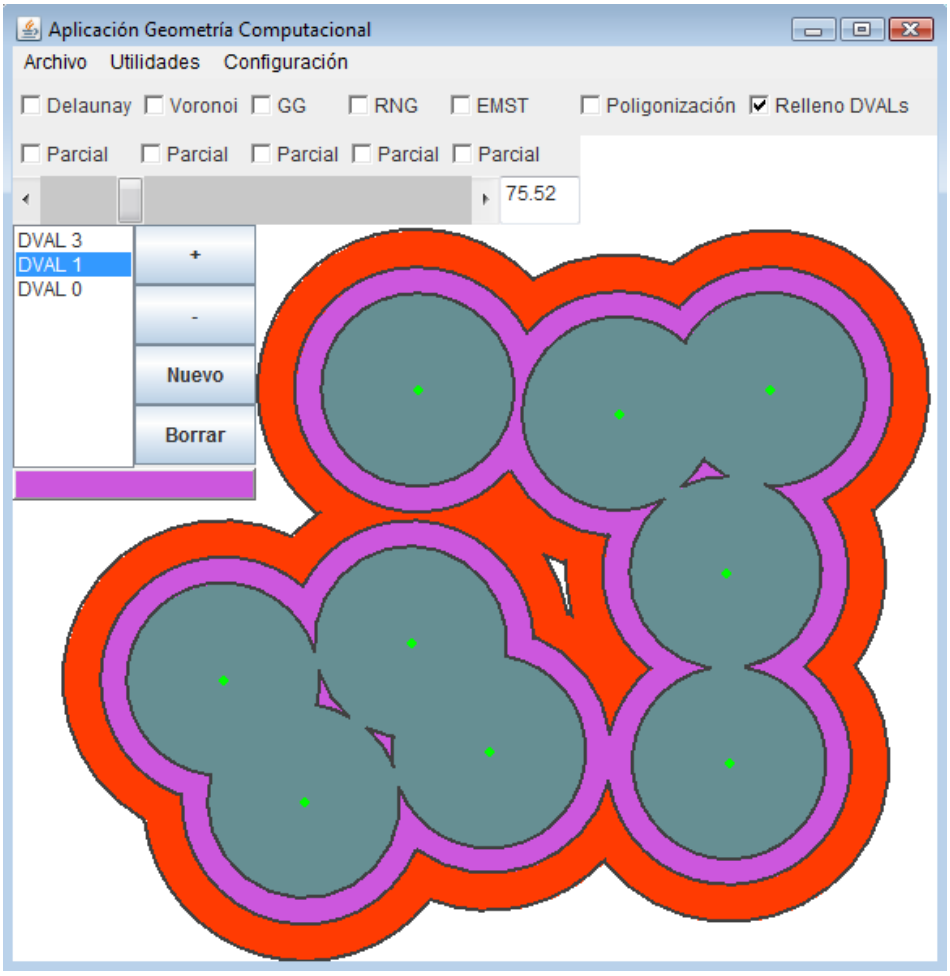



Figura 2: Ejemplo de situación con varios diagramas de Voronoi de alcance limitado.

proximidad descritos anteriormente. Al introducir un nuevo punto, la aplicación recalcula localmente la triangulación de los puntos y su diagrama de Voronoi, es decir, sólo recalcula los elementos de ambos que resultan afectados directamente por la inclusión del nuevo punto. El caso de modificación de un punto es análogo al de inclusión de un nuevo punto a efectos del recálculo de elementos. En cambio, al eliminar un punto la aplicación recalcula de nuevo todos los grafos de proximidad.

Los grafos de Gabriel, de vecinos relativos y al árbol generador mínimo, se recalculan siempre en función del grafo ya calculado en el que están contenidos. Es decir, el grafo de Gabriel se calcula a partir de la triangulación de Delaunay, el grafo de vecinos relativos a partir del grafo de Gabriel, y el árbol generador mínimo a partir del grafo de vecinos relativos.

Tal como se desprende de la descripción de las funcionalidades, la aplicación también calcula, clasifica y almacena los cuatro tipos de eventos, correspondientes a los cambios combinatorios que se producen en los diagramas $Vor(S, r)$ descritos en el apartado 2, para la nube dada de puntos.

Para su fácil uso, la aplicación almacena la triangulación de Delaunay y el diagrama de Voronoi en sendas estructuras DCEL (Doubly-Connected Edge List) y los grafos de Gabriel, de vecinos relativos y el árbol generador mínimo en listas de adyacencias. Esto permite acceder de manera rápida a la información de estos grafos y facilita que funciones de la aplicación que se ejecutan frecuentemente, como por ejemplo el dibujado de todos los elementos calculados, sean lo más eficientes posible. En cuanto a las regiones formadas por los diagramas de Voronoi de alcance limitado, las fronteras y agujeros de sus componentes conexas están descritos como una lista de segmentos y arcos. Los elementos de las fronteras están ordenados en sentido antihorario, mientras que los de los agujeros lo están en sentido horario.

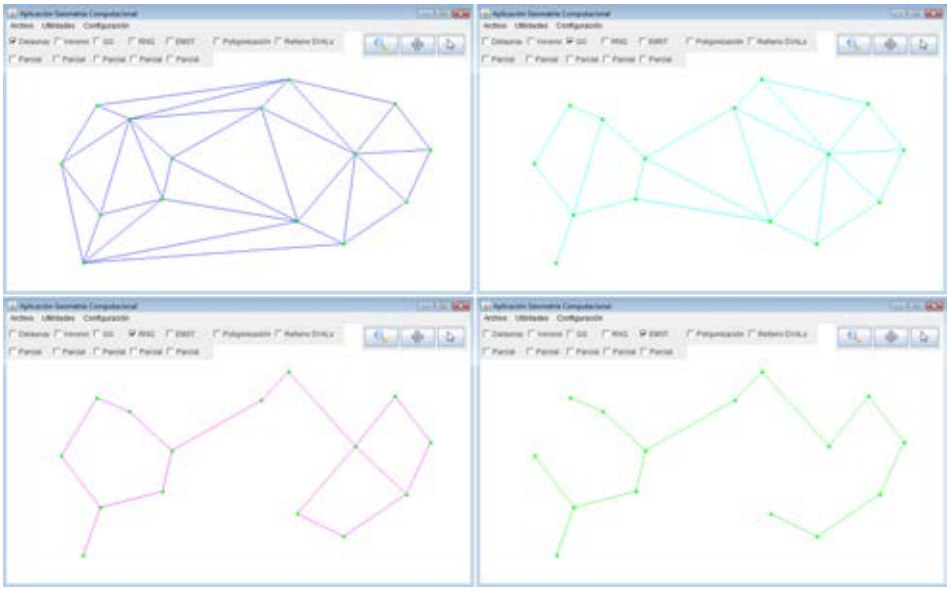


Figura 3: Ejemplo de algunos de los grafos de proximidad calculados.

El fichero de texto generado al guardar un proyecto contiene, en primer lugar, las DCEL correspondientes a la triangulación de Delaunay y al diagrama de Voronoi de la nube de puntos. A continuación, aparece toda la información relativa a los distintos $Vor(S, r)$ creados por el usuario, ordenados siempre por valor creciente del alcance. Esta información consta del radio del diagrama, el número de componentes conexas y, para cada una de ellas, los puntos de la nube que contiene y las descripciones de su frontera exterior, de sus agujeros y de su poligonización. También aparece en este archivo el número de eventos (cambios combinatorios) que se producen en el diagrama de Voronoi de alcance limitado de la nube de puntos y, para cada evento, su tipo y el valor del radio $r$ que lo produce. Para terminar, el fichero guarda la información relativa a los grafos de Gabriel y de vecinos relativos y al árbol generador mínimo de la nube. En concreto, para cada uno de los tres encontramos sus vértices, los vértices adyacentes a éstos y, para cada una de estas aristas, su longitud al cuadrado.

### 3.3 Ampliación en curso

La segunda etapa de este proyecto, que en el momento de escribir este texto se encuentra en fase de desarrollo, consiste en implementar funcionalidades destinadas al estudio y la optimización de caminos de desviación mínima y de separación máxima [2, 9, 4], basadas en diagramas de Voronoi de alcance limitado. Para ello, se prevé implementar algoritmos de decisión para detectar si un camino dado es o no de desviación mínima, de desviación mínima local, de separación máxima, algoritmos para el cálculo y comparación de la longitud entre caminos, etc.

Para obtener información actualizada sobre *DVALon*, consúltese la página [3].

## Referencias

[1] B. Abellanas, M. Abellanas, C. Vilas. VOREST: Modelización de bosques mediante diagramas de Voronoi. *Actas de los XII Encuentros de Geometría Computacional (EGC'07)*, pp. 249–256, 2007.

[2] M. Abellanas, G. Hernández. Optimización de rutas de evacuación, *Actas de los XII Encuentros de Geometría Computacional (EGC'07)*, pp. 273–280, 2007.

[3] M. Abellanas, G. Hernández, J. L. Moreno, S. Ordóñez, V. Sacristán. DVALon. `http://www-ma2.upc.es/~geoc/DVALon/`.

[4] M. Abellanas, G. Hernández, V. Sacristán. Caminos de desviación mínima local, presentado a los *XIII Encuentros de Geometría Computacional (EGC'09)*, 2009.

[5] B. Cărbunar, A. Grama, J. Vitek, O. Cărbunar. Redundancy and Coverage Detection in Sensor Networks, *ACM Transactions on Sensor Networks*, Vol. 2, N. 1, pp. 94–128, 2006.

[6] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, A. Zhu. Geometric Spanner for Routing in Mobile Networks, *Proc. 2nd ACM International Symposium on Mobile and Ad Hoc Networking and Computing (MobiHoc'01)*, pp. 45–55, 2001.

[7] X.-Y. Li, G. Calinescu, P.-J. Wan, Y. Wan. Localized Delaunay Triangulation with Application in Ad Hoc Wireless Networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, N. 10, pp. 1035–1047, 2003.

[8] S. Meguerdichian, F. Koushanfar, M. Poktonjak, M. B. Srivastava. Coverage Problems in Wireless Ad-hoc Sensor Networks, *Proc. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFORCOM 2001)*, Vol. 3, pp. 1380–1387, 2001.

[9] S. Megerian, F. Koushanfar, M. Potkonjak, M. B. Srivastava. Worst and Best-Case Coverage in Sensor Networks, *IEEE Transactions on Mobile Computing*, Vol. 4, N. 1, pp. 84–92, 2005.

[10] W. L. Roque, H. Choset. The Green Island Formation in Forest Fire Modeling with Voronoi Diagrams, *Proc. 3rd CGC Workshop on Computational Geometry*, 1998.

[11] W. R. Smith. Area Potentially Available to a Tree: A Research Tool, *Proc. 19th Southern Forest Tree Improvement Conference, SFTIC*, pp. 22–29, 1987.

[12] H. Xu, L. Huang, Y. Wan, K. Lu. Localized Algorithm for Coverage in Wireless Sensor Networks, *Proc. IEEE Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05)*, pp. 750–754, 2005.

# Índice de autores


ISBN 978-84-9277-41-1