El problema de matching óptimo en un grafo

Cristina Urricelqui Olcoz

Trabajo de Fin de Grado en Matemáticas
Universidad de Zaragoza

Director del trabajo: Alfredo García Olaverri
25 de junio de 2021

Prólogo

Un matching en un grafo $G = (V, E)$ es un subconjunto de ejes K de manera que ningún par de ejes en K tenga vértices comunes. El problema que vamos a resolver a lo largo de este trabajo es el de encontrar un matching máximo en un grafo; es decir, un matching con el mayor número de ejes posible. Este problema es uno de los más importantes en la teoría de grafos. Es de gran interés esencialmente por sus aplicaciones en problemas de la vida real como puede ser el problema de asignación de tareas.

Comenzaremos dando unas nociones básicas de teoría de grafos y explicaremos detalladamente en qué consiste el problema del máximo matching a la vez que daremos algunos ejemplos. Además, empezaremos particularizando el problema en el caso de los grafos bipartitos. Hallar el máximo matching en un grafo bipartito equivale a resolver un problema de máximo flujo en una red. La demostración del teorema de Hall aplicando el algoritmo de Ford-Fulkerson nos da el procedimiento a seguir para hallar la solución a este problema.

Más adelante, generalizaremos el problema a grafos de todo tipo. Daremos las condiciones que debe cumplir un grafo para tener un matching perfecto, matching en el que participan todos sus vértices. El teorema de Tutte, análogo al de Hall para grafos en general, dice que tenemos un matching perfecto en un grafo G si y solo si el número de componentes impares que tenemos al quitar un conjunto de vértices S al grafo G es menor o igual que el número de vértices de S. Además, veremos una consecuencia de este teorema para el caso particular de los grafos cúbicos. El teorema de Petersen dice que un grafo cúbico y sin puentes tiene un matching perfecto.

Finalmente, veremos cómo hallar un matching máximo usando el Algoritmo de Edmonds que se basa en la construcción de árboles alternados. La idea es construir el matching máximo iterativamente a partir de uno dado mejorándolo a través de caminos de aumento y contrayendo, a lo largo de este proceso, los ciclos de longitud impar (blossom) en un solo vértice. El proceso termina dando un camino alternado de aumento o un árbol completo.

Estudiaremos también a lo largo del trabajo la complejidad computacional de cada algoritmo, dado que en la práctica estos algoritmos son muy útiles por su implementación y resolución con ayuda de un ordenador, especialmente en el caso de grafos con un elevado número de vértices y ejes.

Summary

In graph theory, a matching in an undirected graph $G = (V, E)$ is a subset of edges K such that any of them have common vertices. In this work, we will solve the problem of finding a maximum matching from a given graph. The first, and most evident interest of this problem comes from its applications. For instance, it gives a solution to the assignement problem. The main purpose of this work resides in the description of the maximum matching problem, solving it first for bipartite graphs and then for general graphs.

The document consists of 4 chapters. We are going to briefly describe each of them.

First of all, we will bring up some basic definitions and concepts of graph theory such as graph, subgraph, adyacent vertex, walk, path, cycle, connected graph, odd component, bipartite graph, tree... Furthermore, we will expand these definitions to directed graphs. In chapter 1, we will also explain the maximum matching problem giving some applications to real life as the assignement problem and the role it played during the Battle of Britain to know which pilots could fly at the same time in the same squadron. Moreover, we introduce the term of computational complexity of an algorithm, which is a measure of how many steps the algorithm will require, in the worst case, for an instance or input of a given size. It is an important concept since computers play an important role to solve these kind of problems in practise, specially for large graphs.

The main body of this work is focused on solving the maximum matching problem for all types of graphs but in chapter 2 we start by giving a solution for bipartite graphs. We will see that finding a matching in a bipartite graph can be treated as a network flow problem. Hall's theorem states that a bipartite graph $G = (V, E)$ with bipartition $V = V_1 + V_2$ contains a matching of size $|V_1|$ if and only if the number of neighbours of S is greater than or equal to the number of vertices of S, $\forall S \subset V_1$. The solution of the maximum matching problem for bipartite graphs is given in the demostration of the Hall's theorem while applying the Ford-Fulkerson algorithm that we already know. To find the maximum matching is equivalent to solve the problem of calculating the maximum flow in a graph for which we have many methods.

In chapter 3, we will see a characterization of maximum matching for any given graph. We will see the conditions which must satisfy a graph to have a perfect matching. A perfect matching is a matching that covers every vertex of the graph. Tutte's theorem is the analogue of Halls' theorem which gives the necessary and sufficient condition for existence of a perfect matching for general graphs. In general graphs, a graph G has a perfect matching if and only if, by deleting any set of vertices S of G, the number of odd components of $G - S$ is less than or equal to the number of vertices of S. An odd component is a component with an odd number of vertices. Furthermore, provided we had a cubic graph, we would see noticeable consequences. In that case, Petersen's theorem says that every bridgeless cubic graph has a perfect matching.

Finally, in chapter 4 we end by exposing the Edmonds' algorithm which is an algorithm for constructing maximum matchings on general graphs. The matching is constructed by iteratively improving an initial empty matching along augmenting alternating paths in the graph. If $K \subset E$ is a matching of

$G = (V,E)$, an alternating path with respect to K is an elementary path of G whose edges are alternatively in K and $\bar{K} = E - K$. We have an augmenting alternating path if it joins two vertices that are not extreme vertices of any edge of the matching. An important theorem says that a matching K is maximum if and only if there exists no augmenting alternating path with respect to K, so we have that to find a maximum matching it is sufficient to know how to find an augmenting alternating path (if one exists) relative to any matching K. To search alternating paths, we introduce the definition of alternating tree.
An alternating tree with root i_0 relative to a matching K is a subgraph of G, $\mathscr{T} = (V_1, E_1)$, which is connected and without a cycle, such that (1) $i_0 \in V_1$ is the only vertex of the tree which is not incident of an edge of the matching; (2) for each $j \in V_1$, the path $L_{E_1}(j)$ of the tree between i_0 and j is an alternating path; (3) the paths which link i_0 to the vertices of degree 1 of the tree have an even number of edges.
The construction of the alternating tree will proceed step by step starting from $V_1 = \{i_0\}$ and $E_1 = \emptyset$. At current stage we try to augment the tree $\mathscr{T} = (V_1, E_1)$ by adding further edges and further vertices. The key new idea of this algorithm is that an odd cycle in the graph (called blossom) is contracted to a single vertex, with the search continuing iteratively in the contracted graph. At the end, we expand the blossoms in an order inverse to their creation to recover the matching induced on the initial graph.
The procedure will terminate either by producing an augmenting alternating path with origin i_0, or by finding an alternating tree which is complete.
Moreover, this work supports the theorem that states that providing a complete alternating tree has been found, no alternating augmenting path containing vertices of the tree can exist.
To search a maximum matching for a general graph, Edmonds' algorithm starts with any matching K. If there is no further unsatured vertex for K, the present matching is already maximum. Otherwise, let i_0 be any unsatured vertex. We construct an alternating tree with root i_0 and we end by giving a complete alternating tree or by finding an augmenting alternating path. In this last case, we obtain a larger matching doing a transfer along that path.
To sum up, the main aim of this chapter is the description of the Edmonds' algorithm by seeing the theory related to alternating paths and the construction of alternating trees.

Índice general

Capítulo 1

Conceptos básicos y planteamiento del problema

En este capítulo explicaremos los conceptos más importantes de la teoría de grafos que van a ser utilizados en el resto de capítulos. Veremos la definición de grafo, subgrafo, camino, cuándo dos grafos son conexos, casos particulares de grafos a los que les damos especial importancia en el trabajo, etc. Además, enunciaremos formalmente el problema del máximo matching y veremos en qué consiste la complejidad computacional de un algoritmo.

1.1. Definiciones

Empezamos dando las definiciones para grafos no dirigidos que es a lo que comúnmente llamamos grafo.

Definición. Un **grafo no dirigido** es un par $G = (V, E)$ donde V es un conjunto finito a cuyos elementos llamaremos vértices y E es un conjunto de pares no ordenados de vértices a cuyos elementos llamaremos ejes. Llamaremos **orden** de un grafo G al número de sus vértices y lo denotaremos por $n = |V|$. Por otro lado, llamaremos **tamaño** de un grafo G al número de sus ejes y lo denotaremos por $m = |E|$.

Denotaremos los vértices por $i, j, ...$ y los ejes por (i, j). En la definición de grafo no dirigido, cuando decimos que los ejes son pares no ordenados, estamos diciendo que el par (i, j) y el par (j, i) son el mismo.

Un **bucle** es un eje de la forma (i, i). Dado un eje (i, j) diremos que une i con j y a los vértices i y j los denominaremos **extremos**. También diremos que i y j son **adyacentes** o **vecinos**, y que el eje (i, j) es **incidente** al vértice i. El **grado** de un vértice es el número de ejes incidentes con él.

Además, solo vamos a trabajar con **grafos simples** que son aquellos que no presentan bucles ni pares repetidos.

Definición. Un **subgrafo** de $G = (V, E)$ es otro grafo $G_1 = (V_1, E_1)$ tal que $V_1 \subseteq V$ y $E_1 \subseteq E$.

Definición. Dado un grafo no dirigido, $G = (V, E)$, un **itinerario** de longitud l desde el vértice i al vértice j es una sucesión de vértices $(i_0, i_1, i_2, ..., i_l)$ tal que $i_0 = i$, $i_l = j$ y $(i_{h-1}, i_h) \in E$ para $h = 1, 2, ..., l$. Un **camino** es un itinerario en el que no aparecen ejes repetidos y diremos que el camino es simple cuando en la secuencia además no aparecen vértices repetidos.

Un **ciclo** es un camino que al menos contiene un eje y donde el vértice inicial y final coinciden; es decir, $i = j$. Si además no hay otros vértices repetidos diremos que el ciclo es simple.

Definición. Un grafo $G = (V,E)$ es **conexo** cuando $\forall i, j \in V$ existe un camino desde i hasta j. Un subgrafo maximal y conexo de G se denomina componente conexa o simplemente **componente** del grafo.

Más adelante hablaremos de **componentes impares** que son aquellas que presentan un número impar de vértices. De igual manera, diremos que una componente es par si presenta un número par de vértices.

Dentro de los grafos no dirigidos tenemos un caso especial que va a ser muy relevante, el de los grafos bipartitos.

Definición. Un grafo no dirigido, $G = (V,E)$, es **bipartito** si el conjunto de vértices V puede dividirse en dos partes no vacías V_1 y V_2 de manera que todos los ejes tienen un extremo en V_1 y otro en V_2. Cuando $|V_1| = n_1$, $|V_2| = n_2$ y en el grafo aparecen todos los ejes de V_1 a V_2, el grafo se suele denotar K_{n_1,n_2} y se dice que es **bipartito completo**.

Definición. Un **árbol** es un grafo no dirigido conexo y sin ciclos.

Aunque prácticamente a lo largo de todo el trabajo hablaremos de los grafos no dirigidos, en el problema de máximo flujo es necesario conocer el sentido del flujo. Por ello, introducimos el concepto de grafos dirigidos.

Definición. Un **grafo dirigido** (o red) es un par $G = (V,A)$, donde V es un conjunto finito, a cuyos elementos llamaremos vértices y A es un subconjunto de $V \times V$ a cuyos elementos llamaremos arcos.

Ahora suponemos que cuando $i \neq j$, el par (i, j) y el par (j, i) son distintos. Dado un arco $a_k = (i, j)$ diremos que ese arco va desde i hasta j, que i es el extremo inicial y j el extremo final. Además, diremos que j es vecino "out" de i, y que i es vecino "in" de j.

Dado un grafo dirigido G podemos considerar siempre su versión no dirigida, $\bar{G}$, que es el grafo obtenido sustituyendo arcos por ejes y eliminando repeticiones.

Definición. Dado un grafo dirigido $G = (V,A)$, un **itinerario dirigido** de longitud l desde el vértice i al j es una sucesión de vértices $(i_0, i_1, i_2, ..., i_l)$ tal que $i_0 = i$, $i_l = j$ y $(i_{h-1}, i_h) \in A$ para $h = 1, 2, ..., l$. Un **camino dirigido** es un itinerario dirigido en el que no aparecen arcos repetidos y diremos que el camino dirigido es **simple** cuando en la secuencia además no aparecen vértices repetidos.

Definición. Dado un grafo dirigido $G = (V,A)$, un **camino no dirigido** de longitud l desde el vértice i al j es una sucesión de vértices $(i_0, i_1, i_2, ..., i_l)$ tal que $i_0 = i$, $i_l = j$ y $(i_{h-1}, i_h) \in A$ o $(i_h, i_{h-1}) \in A$ para $h = 1, 2, ..., l$, junto con una especificación de qué arco (el (i_{h-1}, i_h) o el (i_h, i_{h-1}) es usado para "pasar" de i_{h-1} a i_h, y donde además, cada arco puede usarse como mucho una vez.

Cuando se usa el arco (i_{h-1}, i_h) pondremos una flecha entre esos dos vértices en la secuencia que define el camino, $i_{h-1} \rightarrow i_h$, y diremos que ese es un arco hacia adelante en el camino. Igualmente, cuando se usa el arco (i_h, i_{h-1}) pondremos una flecha invertida entre esos dos vértices de la secuencia, $i_{h-1} \leftarrow i_h$, y diremos que ese es un arco hacia atrás en el camino.

Notamos que existe un camino no dirigido de i a j en G si y solo si existe ese camino en $\bar{G}$. Diremos que G es conexo si lo es $\bar{G}$, es decir si existe un camino no dirigido desde cualquier nodo i a cualquier otro nodo j.

1.2. Problema del máximo matching

Veamos primero en qué consiste el concepto de matching, comúnmente empleado el término en inglés, aunque también podríamos hablar de emparejamientos.

Definición. Dado un grafo no dirigido, $G = (V,E)$, un **matching** es un subconjunto $K \subset E$ de manera que ningún par de ejes en K comparte el mismo vértice.

Definición. Dado un grafo no dirigido, $G = (V,E)$, un **matching perfecto** es un matching en el que participan todos los vértices del grafo; es decir, un matching es perfecto si todo vértice del grafo es incidente a un eje del matching.

Observamos que un grafo solo puede tener un matching perfecto si el número de vértices del grafo es un número par.

Definición. Dado un grafo no dirigido, $G = (V,E)$, un **matching máximo** es un matching que contiene el mayor número posible de ejes. El tamaño del máximo matching del grafo lo llamamos el número del matching y lo denotamos por $\upsilon(G)$.

Una vez introducidos estos conceptos básicos, veamos en qué consiste el problema del máximo matching:

PROBLEMA DEL MÁXIMO MATCHING

Sea un grafo no dirigido $G = (V,E)$, el objetivo del problema del máximo matching es encontrar un subconjunto de ejes $E_1 \subseteq E$ de tamaño el máximo posible de manera que E_1 sea un matching. Como cada eje une exactamente dos vértices, esto es equivalente a encontrar un matching que cubra el mayor número de vértices posible.

El problema así explicado parece sencillo pero en realidad no lo es tanto. Primero, veremos métodos para, dado un matching, saber si es máximo o no; es decir, veremos una caracterización del matching óptimo. Más adelante, dado un grafo, trataremos de hallar un matching máximo; siendo este último problema bastante más difícil de resolver. Además empezaremos con grafos bipartitos y luego generalizaremos el problema para todo tipo de grafos.

Veamos algunos ejemplos muy sencillos de la vida real en los que se ha necesitado que el número de estas conexiones sea máximo.

Ejemplo 1. El primer ejemplo que vamos a dar lo ambientamos en la batalla de Inglaterra en la Segunda Guerra Mundial. Durante la batalla de Inglaterra, los escuadrones aéreos estaban formados por dos pilotos. Sin embargo, había pilotos que no podían volar juntos por hablar lenguas distintas u otras diferencias. Lo que se buscaba era encontrar el máximo número de pilotos que podían volar simultáneamente, es decir, necesitaban saber el número de escuadrones que se podían formar y por quiénes estaban formados.

En este caso tenemos un grafo $G = (V,E)$ en el que el conjunto de vértices V representa a los pilotos y el conjunto de ejes E son las conexiones que pueden darse entre ellos. Es decir, entre dos pilotos habrá un eje en el caso de que ambos puedan volar juntos. Al formar esos emparejamientos no puede haber pilotos que estén en varios grupos a la vez, o lo que es lo mismo, distintos ejes no pueden presentar vértices comunes. Así lo que estamos buscando es un matching y, como se trataba de encontrar el máximo número de pilotos que podían volar juntos, no nos vale cualquiera sino uno que sea máximo.

Ejemplo 2. Otro ejemplo es el de asignación de tareas. Supongamos que en una fábrica tenemos p máquinas y q tareas que realizar. No todas las máquinas pueden hacer todas las tareas y, como es lógico, una máquina no puede hacer dos tareas a la vez. El objetivo es tener el mayor número de máquinas trabajando para maximizar los beneficios de la empresa. Por lo tanto, lo que queremos saber es qué tarea asignar a cada máquina para conseguirlo.

En este caso tenemos un grafo bipartito $G = (V,E)$ con $V_1 = 1,..,p$ y $V_2 = 1,..,q$. El conjunto de vértices V_1 representa las máquinas y el conjunto V_2 las tareas a realizar. Además, existe un eje (i, j) entre un elemento de V_1 y un elemento de V_2 si y solo si la máquina i puede realizar el trabajo j. Las máquinas no pueden realizar dos tareas a la vez, es decir, de un vértice de V_1 no pueden salir dos ejes.

Lo mismo ocurre con las tareas, que como es lógico solo se realizan una vez, luego a los vértices de V_2 solo puede llegar un único eje. Así, lo que estamos buscando realmente es un matching máximo entre elementos de V_1 y V_2, o lo que es lo mismo, un matching máximo en un grafo bipartito.

Como podemos ver, este problema puede darse en situaciones muy diversas de diferentes ámbitos por lo que su resolución tiene una gran importancia.

Dibujamos ahora un grafo cualquiera para entender mejor en qué consiste nuestro problema. Sea el grafo G representado en la figura 1.1.

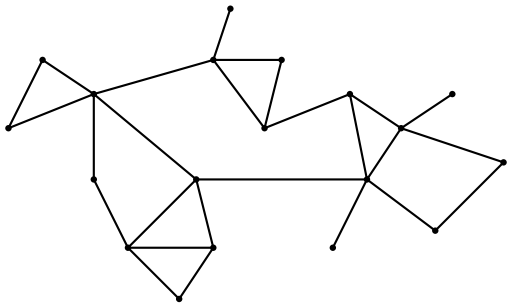


Figura 1.1: Grafo G

Podemos marcar los posibles matchings sin seguir ningún procedimiento hasta hallar, por ejemplo, los representados en color rojo en figura 1.2, en la figura 1.3 y en la figura 1.4.

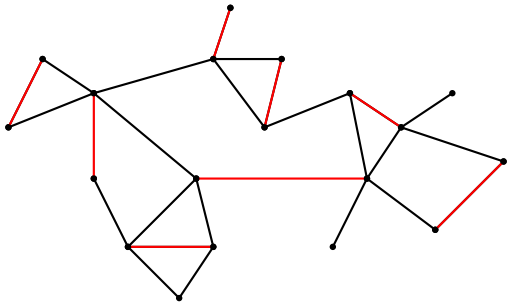


Figura 1.2: Grafo 1

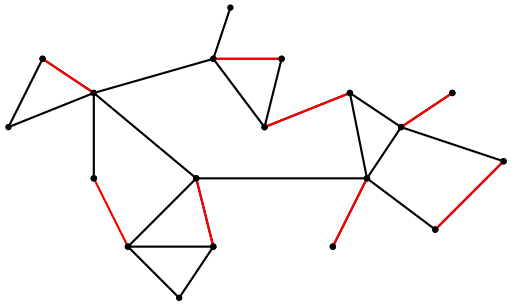


Figura 1.3: Grafo 2

Observamos que en los dos primeros grafos tenemos un matching formado por 8 ejes pero el tercero conseguimos hallar uno con 9 buscando los ejes de forma aleatoria. Probablemente no podamos crear un matching formado por un número mayor de ejes. Pero... ¿cómo podemos asegurar que efectivamente no podemos encontrar uno más grande? ¿qué condiciones tiene que cumplir un grafo para que podamos crear un matching perfecto? Y más importante aún, ¿cómo encontramos uno que sea de ese tamaño

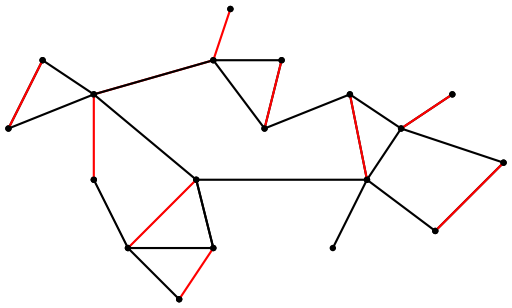


Figura 1.4: Grafo 3

máximo sin marcar los ejes de forma aleatoria como hemos hecho en este caso? Son estas cuestiones las que vamos a resolver a lo largo de este trabajo.

Además, en la práctica nos ayudamos de ordenadores para resolver problemas de este estilo. Es por ello, que introducimos en la siguiente sección el concepto de complejidad computacional de un algoritmo.

1.3. Complejidad computacional de algoritmos

Es claro que el problema del máximo matching es más complicado de resolver cuanto mayor es el número de vértices y ejes del grafo. Por ello, es importante saber el número de pasos que han de darse para la resolución del problema y si un ordenador es capaz de resolverlo. Así, es importante introducir el concepto de complejidad computacional.

La complejidad computacional de un algoritmo es una magnitud que mide el número de pasos que requiere el algoritmo en el peor de los casos para una entrada de un tamaño determinado. El número de pasos viene dado como una función de ese tamaño. Debemos saber cómo medir el tamaño de un problema y qué consideramos un paso en el algoritmo.

En un modelo simple de un dispositivo informático, un paso es una de las siguientes operaciones: suma, resta, multiplicación, división y comparación de dos números.

Por lo tanto, si un algoritmo requiere 100 sumas y 220 comparaciones en algún caso, decimos que el algoritmo requiere 320 pasos en ese caso. Para que este número sea significativo, deberíamos expresarlo como una función del tamaño de la entrada correspondiente, pero determinar la función exacta no sería práctico.

Lo que realmente importa es cuánto le cuesta al ordenador resolver el algoritmo (en el peor de los casos) asintóticamente a medida que aumenta el tamaño de una entrada. Así, la función simple del tamaño de la entrada que buscamos se puede aproximar con una cota superior del número máximo de pasos a dar en el peor de los casos. Esta función es la que se denomina complejidad del algoritmo.

Para calcular el tamaño de un grafo $G = (V, E)$ usualmente se utiliza la expresión $n + m$ siendo $n = |V|$ el número de vértices y $m = |E|$ el número de ejes, entonces la complejidad del algoritmo en grafos se suele dar en función de n y m.

Capítulo 2

Matching en grafos bipartitos

Prestamos especial atención a los grafos bipartitos porque la mayoría de las aplicaciones de la teoría de grafos a la vida real se dan en modelos de grafos de este tipo. En este capítulo nos centraremos en el teorema de Hall, que es un caso particular del Teorema de máximo flujo-mínimo corte. Más adelante generalizaremos los resultados obtenidos en este capítulo a grafos de todo tipo.

2.1. Problema de máximo flujo en una red

Definición. Una **red de flujo** es un grafo dirigido $G = (V,A)$ simple, conexo, donde destacamos el vértice s que es el origen y el vértice t que es el final y donde cada arco $(i,j) \in A$ tiene asociado un número no negativo $c(i,j) \geq 0$ llamado capacidad de ese arco.

Nota. Consideramos $c(i,j) = 0$ cuando no existe el arco de va de i a j; es decir, $(i,j) \notin A$.

Definición. Un **flujo** sobre la red de flujo G es una aplicación $f : V \times V \longrightarrow \mathbb{R}$ que cumple las siguientes restricciones:

$$0 \leq f(i,j) \leq c(i,j), \quad \forall (i,j) \in V \times V$$
$$\sum_{j \in V} f(i,j) = \sum_{j \in V} f(j,i), \quad \forall i \in V - \{s,t\}$$

A estas restricciones las llamamos restricción de compatibilidad y de conservación de flujo, respectivamente.
El **valor de un flujo** f se define como

$$v(f) = \sum_{j \in V} f(s,j) - \sum_{j \in V} f(j,t)$$

es decir, el flujo neto que sale de s.

Dada una red de flujo, el problema de máximo flujo consiste en calcular un flujo f con valor $v(f)$ máximo.

Denotaremos por (X,Y) al conjunto de arcos que tienen el extremo inicial en algún vértice $i \in X$ y el final en algún vértice $j \in Y$. Notar que $f(X,Y)$ es la suma de los valores de f en esos arcos. Además, denotaremos por $\bar{X}$ al conjunto complementario de X.

Definición. Dada una red de flujo G, un **corte** separando s de t, es un conjunto de arcos $(X,\bar{X})$ donde $s \in X$ y $t \in \bar{X}$. A la cantidad $c(X,\bar{X})$ le llamaremos **capacidad del corte** $(X,\bar{X})$.

Si encontramos un flujo f y un corte $(X,\bar{X})$ tales que $v(f) = c(X,\bar{X})$ tendremos que ese flujo es máximo y ese corte es mínimo. Además, se tiene que cumplir $f(X,\bar{X}) = c(X,\bar{X})$ y $f(\bar{X},X) = 0$ que equivale a decir que por los arcos de $(X,\bar{X})$ circula un flujo igual a su capacidad y por arcos de $(\bar{X},X)$ no circula flujo.

Definición. Dado un flujo f, los arcos (i,j) que cumplen $f(i,j) = c(i,j)$ decimos que están **saturados**, y los que cumplen $f(i,j) = 0$ decimos que son **nulos**.

Definición. Sea f un flujo definido sobre la red G. Un **camino de aumento** para el flujo f es un camino no dirigido desde s hasta t, tal que los arcos hacia adelante en el camino son no saturados y los arcos hacia atrás son no nulos.

Recordamos también en qué consiste el método BFS, que resuelve el siguiente problema:
Dado un grafo dirigido G y un vértice i, determinar todos los vértices j para los que existe un camino dirigido desde i hasta j.

Este método funciona mediante un proceso de marcado y exploración de vértices, y necesita un vector de marcas, que para cada vértice nos indicará si el vértice está marcado o no, y una lista, que funcionará como una cola, donde guardamos los vértices marcados. De entrada suponemos que conocemos n el número de vértices, (podemos suponer que $V = \{1,2,...,n\}$), las listas "out" de los n vértices, el vértice inicial i, y que disponemos de un vector *marca* con n componentes, inicialmente todos los vértices sin marcar, y un vector $lista$ inicialmente vacío.

Algoritmo BFS

1. (Paso inicial). Marcar el vértice i, es decir asignar un valor a $marca(i)$. Incluirlo en $lista$, que contendrá una lista con los vértices no explorados.

2. (Paso genérico). Mientras $lista$ no esté vacía: Coger el primer vértice k de $lista$ y recorrer su lista "out" de vecinos. A cada vecino de k no marcado, se le pone una marca y se añade al final de $lista$. De esta manera k pasa al estado de explorado, y se quita de $lista$.

3. (Paso final). Ahora $lista$ con los vértices no explorados está vacía. Los nodos j marcados, (los que tienen definido $marca(j)$), son a los que se llega desde el i.

Notamos que cada vértice sólo se explora una vez, y por tanto la complejidad computacional de este algoritmo es de $O(m)$ pasos.

Este algoritmo lo utilizamos para buscar los caminos de aumento al aplicar el Teorema de máximo flujo-mínimo corte como explicamos a continuación.

2.2. Teorema de máximo flujo-mínimo corte

Teorema 2.1 (Teorema de máximo flujo-mínimo corte)**.** *Sea G una red de flujo. El valor máximo de $v(f)$ tomado entre todos los posibles flujos f es igual al mínimo de $c(X,\bar{X})$ tomado entre todos los cortes $(X,\bar{X})$.*

Veamos un algoritmo para saber cómo aplicar este teorema a la hora de hallar el camino de aumento de un grafo G de manera que el valor de su flujo sea máximo.

Algoritmo de Ford-Fulkerson

1. (Paso inicial). Comenzamos con un flujo inicial cualquiera f_0, por ejemplo con el flujo nulo $f_0(i,j) = 0, \forall (i,j)$.

2. (Paso genérico). Utilizamos ahora el método BFS explicado anteriormente para hallar caminos de aumento de la siguiente manera: Si existe camino de aumento para f_l obtenemos otro flujo f_{l+1} de manera que $v(f_{l+1}) > v(f_l)$. Si no existe camino de aumento terminamos. Repetimos el proceso haciendo $l = l+1$ hasta obtener el último flujo que es el óptimo.

El número de iteraciones que realiza el método de Ford-Fulkerson es siempre menor que nm luego tenemos que la complejidad computacional del algoritmo es $O(nm)$.

Este teorema nos permite dar demostraciones sencillas de varios resultados clásicos en grafos. Nos quedamos con la que nos interesa que es la del Teorema de König-Egerváry. Este teorema particulariza la obtención del máximo matching a los grafos bipartitos.

Teorema 2.2 (Teorema de König-Egerváry)**.** *Sea $G = (V,E)$ un grafo bipartito no dirigido. Es decir, $V = S \cup T$, $S \cap T = \emptyset$, $S,T \neq \emptyset$, $E \subseteq S \times T$. Entonces, el máximo número de ejes de E que podemos hallar con extremos disjuntos es igual al número mínimo de vértices que hay que eliminar del grafo para hacer desaparecer todos los ejes.*

Observamos que este teorema da una equivalencia entre el problema de máximo matching y el de hallar una cobertura de vértices mínima en grafos bipartitos.

Para buscar un máximo matching en un grafo bipartito $G = (V,E)$ donde $V_1 + V_2$, aplicamos el algoritmo de Ford-Fulkerson a un grafo $G' = (V',E')$. G' es tal que V' es el conjunto de vértices V añadiéndole los vértices origen s y final t y E' es el conjunto de ejes añadiéndole los ejes (s,j), $\forall j \in V_1$ y (i,t), $\forall i \in V_2$. Además, a los ejes (s,j) y (i,t) les asignamos capacidad 1 y a los ejes (i,j) con $i \in V_1$ y $j \in V_2$ les asignamos capacidad infinita. El matching buscado será el conjunto de ejes (i,j) con $i \in V_1$ y $j \in V_2$ por los que circula flujo. Además, el corte nos dice por qué no se puede mejorar; es decir, nos asegura que este matching es máximo.

Ejemplo 3. Damos un ejemplo ilustrativo para ver la aplicación de este método en un grafo dado. Tenemos el grafo G de la figura 2.1.

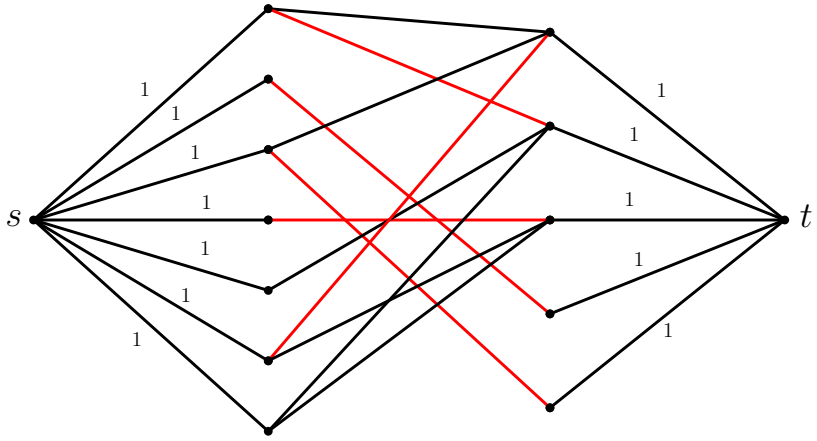


Figura 2.1: Grafo G

Observamos que, en este caso, el matching hallado es el conjunto de ejes representados en color rojo. Todos los ejes que no están marcados con un 1 tienen capacidad infinita.

Finalmente, veamos el teorema de Hall, que a pesar de tener un enunciado bastante diferente, podemos decir que esencialmente equivale al Teorema de König-Egerváry.

Definición. Sea $E = \{e_1,...,e_m\}$ un conjunto finito, y $S = \{S_1,...,S_n\}$ una familia de subconjuntos de E. Un **sistema de representantes distintos (SRD)** para esa familia, es una lista de n elementos distintos $e_{i_1}, \ldots, e_{i_n}$ de E, cumpliendo que $e_{i_j} \in S_j$ para $j = 1,...,n$.

Teorema 2.3 (Teorema de Hall)**.** *Sea $E = \{e_1,...,e_m\}$, y $S = \{S_1,...,S_n\}$ una familia de subconjuntos de E. La familia S admite un SRD si y solo si cada subfamilia de S formada por k subconjuntos contiene al menos k elementos, para $k = 1,...,n$.*

Capítulo 3

Teoremas de Tutte y Petersen

En este capítulo generalizaremos lo visto en el capítulo anterior. Es decir, daremos una caracterización del matching óptimo en grafos de cualquier tipo. Veremos qué condiciones tiene que cumplir un grafo para tener un matching perfecto. En el siguiente capítulo veremos cómo hallarlo.

3.1. Teorema de Tutte

Sea un grafo $G = (V,E)$ tal que tiene un matching perfecto $K \subset E$. Si le quitamos un conjunto de vértices S a G, entonces tenemos lo siguiente:
En una componente C de $G - S$ habrá un número par de vértices que pertenezcan ambos a C uniendo ejes del matching perfecto K. El resto de vértices de C estarán en ejes de K que unan un vértice de C con uno de S. En particular, para toda componente impar C de $G - S$ hay un eje de K que une un vértice de C con uno de S. Como los ejes de K forman un matching perfecto, el grafo $G - S$ tiene como mucho $|S|$ componentes impares, una para cada vértice de S. Por tanto, es claro que el número de componentes impares de $G - S$ es menor o igual que $|S|$.

Ejemplo 4. Para entender mejor lo descrito en el párrafo anterior ilustramos el grafo G_1 de la figura 3.1. Los ejes del matching son los marcados en rojo luego $K = \{(1,2),(3,4),(5,6),(7,8),(9,10)\}$. Observamos que es perfecto porque incluye a todos los vértices del grafo. Si tomamos $S = \{4,7\}$; es decir, quitamos los nodos 4 y 7 observamos que tenemos tres componentes $C_1 = \{1,2,3\}$, $C_2 = \{5,6\}$ y $C_3 = \{8,9,10\}$ de las cuales solo son dos impares. Como tenemos que el número de componentes impares es 2 y el de vértices que hemos quitado también es 2 observamos que se cumple la condición del Teorema de Tutte.

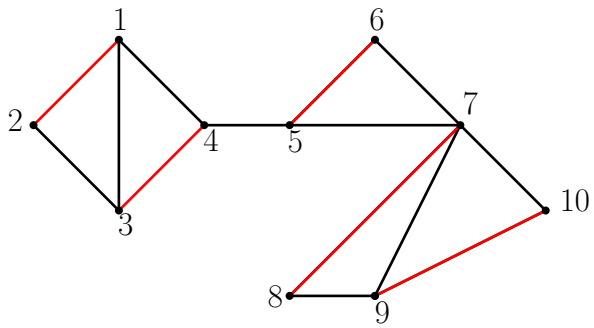


Figura 3.1: Grafo G_1

Ejemplo 5. Damos ahora un ejemplo de un grafo en el que no se cumple esta condición y, por lo tanto, no va a tener un matching perfecto. El grafo G_2 está representado en la figura 3.2. Observamos que

en este caso, si quitamos los vértices 4 y 8 tenemos 4 componentes impares que son las siguientes: $C_1 = \{1,2,3\}$, $C_2 = \{5,6,7\}$, $C_3 = \{9\}$ y $C_4 = \{10,11,12\}$. Como el número de componentes impares que se forman al quitar 2 vértices no es menor o igual que 2, no podemos formar un matching perfecto en este grafo.

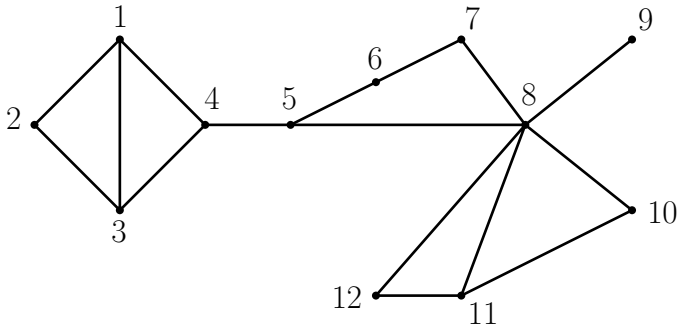


Figura 3.2: Grafo G_2

En 1947 Tutte demostró que el hecho de que el número de componentes impares de $G-S$ sea menor o igual que $|S|$ es condición suficiente para que exista un matching perfecto para el grafo G. A continuación, damos el teorema que enuncia este hecho formalmente y su demostración.

Nota. Denotaremos por $q(G)$ el número de componentes impares de un grafo G. Además, si tenemos componentes C_i de un grafo G denotaremos por $V(C_i)$ el conjunto de vértices y $E(C_i)$ el conjunto de ejes de esta componente en particular.

Teorema 3.1 (Teorema de Tutte)**.** *Sea un grafo $G = (V,E)$, entonces G tiene un matching perfecto* $\Leftrightarrow$ $q(G-S) \leq |S|$, $\forall S \subset V$.

Demostración. $\Rightarrow$) Ya hemos visto que esta implicación es cierta.
$\Leftarrow$) Veamos ahora que $q(G-S) \leq |S|$, $\forall S \subset V$ implica que G tiene un matching perfecto.
Notamos que si $S = \emptyset$ entonces tenemos $q(G-S) \leq 0$ luego tiene que ser $q(G-S) = 0$ por lo que todas las componentes han de ser pares, o lo que es lo mismo, G tiene un número par de vértices.
Veamos por inducción sobre el número de vértices del grafo G que G tiene un matching perfecto.
Como el número de vértices de G es par, empezamos el proceso de inducción tomando $|V| = 2$ y suponemos que se verifica $q(G-S) \leq |S|$. Observamos que G tiene que tener un solo eje, luego tiene un matching perfecto formado por ese eje.
Supongamos que G tiene un número n par de vértices y que el teorema es cierto para grafos de orden menor que n.
Dado $v \in V$, $G-v$ tiene un número impar de vértices luego $q(G-v) \geq 1$. Como por hipótesis teníamos que $q(G-v) \leq 1$, entonces $|S| = 1$. Esto demuestra que existen subconjuntos no vacíos S donde se alcanza la igualdad en la fórmula del enunciado. Sea $S_0 \subset V$ el que tiene un mayor número de vértices. Sean $C_1, C_2, ..., C_m$, donde $m = |S_0| \geq 1$, las componentes impares de $G - S_0$ y $D_1, D_2, ..., D_k$ las componentes pares. Notar que:

(i) Cada D_i tiene un matching perfecto, veámoslo:
Si $S \subset V(D_i)$, entonces tenemos

$$q(V - S_0) + q(D_i - S) = q(V - S_0 \cup S) \leq |S_0 \cup S| = |S_0| + |S|$$

luego $q(D_i - S) \leq |S|$. Por la hipótesis de inducción, D_i tendrá un matching perfecto.

(ii) Si $c \in C_i$ entonces $C_i - c$ tiene un matching perfecto, veámoslo:
Procedemos por reducción al absurdo luego supongamos que $C_i - c$ no tiene un matching perfecto.
Por hipótesis de inducción, existe un subconjunto $S \subset V(C_i) - \{c\}$ tal que $q(C_i - \{c\} \cup S) > |S|$.
Ahora bien, los vértices de C_i (cantidad impar) son la suma de :

- los vértices de las componentes impares de $C_i - \{c\} \cup S$
- los vértices de las componentes pares de $C_i - \{c\} \cup S$ (cantidad par)
- los vértices de $\{c\} \cup S$

Por tanto, si $q(C_i - \{c\} \cup S)$ es un número impar, $|S|+1$ debe ser par, y si $q(C_i - \{c\} \cup S)$ es un número par, $|S|+1$ debe ser impar.
Teníamos que $q(C_i - \{c\} \cup S) > |S|$ luego $q(C_i - \{c\} \cup S) \geq |S|+1$, y por las cuestiones de paridad que acabamos de comentar, es claro que $q(C_i - \{c\} \cup S) \geq |S|+2$.
Pero entonces,

$$|S_0 \cup \{c\} \cup S| = |S_0|+1+|S| \geq q(G - S_0 \cup \{c\} \cup S) = q(G-S_0) - 1 + q(C_i - \{c\} \cup S) \geq |S_0|+1+|S|$$

Esto implica que las desigualdades deben ser igualdades y $S' = S_0 \cup \{c\} \cup S$ verifica $q(G-S') \leq |S'|$. Llegamos a contradicción con que S_0 es el subconjunto con el mayor número de vértices que verifica la desigualdad. Por tanto, $C_i - c$ tiene un matching perfecto.

(iii) El grafo G tiene un matching formado por m ejes de la forma $(s_i, c_i), s_i \in S_0, c_i \in C_i$ con $i = 1,2,...,m$ y donde C_i son las distintas componentes impares de $G - S_0$. Veámoslo:
Consideramos un grafo bipartito con $V_1 = C_1, C_2, ..., C_m$ y $V_2 = S_0$ que cumple que entre C_i y $s \in S_0$ existe una conexión si y solo si G contiene un eje que va de s a C_i.
Lo que queremos probar es que este grafo bipartito tiene un matching perfecto y para ello utilizaremos el Teorema de Hall visto en el capítulo anterior.
Sea V_1' un subconjunto de V_1 y V_2' el subconjunto de sus imágenes, entonces tenemos $V_2' \subset S_0$. Por hipótesis, $q(G - V_2') \leq |V_2'|$. Pero tenemos $q(G - V_2') \geq |V_1'|$ ya que al quitarle a G los vértices de V_2' al menos las componentes de V_1' quedan separadas. Por tanto, $|V_1'| \leq |V_2'|$.
Al verificarse las condiciones del Teorema de Hall, podemos asegurar que para cualquier V_1' debe existir un matching perfecto uniendo V_1' con V_2'.

Resumiendo, tenemos que si G verifica los apartados $(i), (ii)$ y (iii), en los cuales se asegura que en cada parte existe un matching perfecto; G tendrá un matching perfecto y será el formado por los ejes de los matchings perfectos de las componentes pares de $G - S_0$, los ejes del matching formado por (s_i, c_i), con $s_i \in S_0$ y $c_0 \in q(G - S_0)$, y los ejes de los matchings de cada subconjunto $C_i - c_i$. □

3.2. Teorema de Petersen

Una vez demostrado el Teorema de Tutte, enunciamos el de Petersen que podemos ver como una consecuencia del primero.

Antes que nada, recordamos que un **grafo cúbico** es aquel en el que todos sus vértices tienen grado 3.

Teorema 3.2 (Teorema de Petersen)**.** *Un grafo cúbico y sin puentes tiene un matching perfecto.*

Demostración. Para probar que todo grafo cúbico y sin puentes tiene un matching perfecto basta probar que tal grafo satisface la condición del Teorema de Tutte. Sea $G = (V, E)$ un grafo cúbico y sin puentes. Sea $S \subset V$ y consideramos una componente impar C de $G - S$.
Como G es cúbico, la suma de los grados de todos los vértices de C es un número impar. Separamos esta suma en dos partes siendo una de ellas el número de ejes que unen únicamente vértices de C y la otra el número de ejes que unen vértices de C con vértices de S.
Como el número de ejes que unen vértices de C es un número par tenemos que el número de ejes que conectan S con C es impar. Además, es mayor que 3 ya que no hay puentes en nuestro grafo G.
Entonces, el número de ejes entre S y $G - S$ es mayor que $3q(G - S)$ y, como G es cúbico, este número es a su vez menor que $3|S|$. Por lo tanto, $q(G - S) \leq |S|$ que es lo que estábamos buscando. □

Capítulo 4

Algoritmo de Edmonds

En este capítulo veremos un algoritmo cuyo resultado es la obtención de un matching máximo para un grafo G cualquiera. Se trata del Algoritmo de Edmonds que fue desarrollado por Jack Edmonds y publicado en 1965.

Primero daremos una idea sencilla que resume en qué consiste este algoritmo. El máximo matching se construye iterativamente a partir de uno dado mejorándolo a través de m caminos de aumento mientras exista al menos uno. La idea esencial del algoritmo es que un ciclo de longitud impar (también denominado por el término en inglés blossom) se contrae en un solo vértice para luego continuar la búsqueda de m caminos de aumento en el grafo resultante. La idea de contraer los ciclos de longitud impar se debe a que si no se hiciera, el algoritmo podría encontrar falsos caminos de aumento al entrar en alguno de estos ciclos.

Antes de enunciar formalmente el algoritmo de Edmonds, veremos la teoría necesaria para poder entenderlo. En particular, veremos qué son los caminos alternados y daremos un algoritmo para hallar un árbol alternado, ya que es una parte fundamental en la búsqueda del matching máximo en un grafo cualquiera.

4.1. Caminos alternados

Definición. Sea un grafo $G = (V,E)$ decimos que un vértice $i \in V$ es **saturado** por el matching $K \subset E$ si existe un eje de K incidente con i. En caso contrario diremos que es **no saturado** o **aislado**.

Definición. Sea $K \subset E$ un matching de $G = (V,E)$, llamamos **camino alternado** a un camino simple de G cuyos ejes están alternadamente en K y en $\bar{K} = E - K$. Un **camino alternado de aumento** es un camino alternado que une dos vértices no saturados.

Notamos que, en un grafo $G = (V,E)$, si a lo largo de todo el camino alternado de aumento cambiamos los ejes pertenecientes al matching K por ejes de $\bar{K} = E - K$ entonces obtenemos un matching de orden $|K| + 1$. A esta operación de cambiar los ejes de K por los de $\bar{K}$ a lo largo de un camino alternado de aumento L la llamaremos transferencia sobre L.

Ejemplo 6. En el grafo G que dibujamos a continuación, tenemos un matching K formado por los ejes $(2,3)$ y $(4,6)$ y representado en rojo.
El camino $L = \{1,2,3,5\}$ es un camino de aumento cuyos vértices inicial y final son no saturados; es decir, es un camino alternado de aumento. En el dibujo L está representado por una línea discontinua. Entonces, obtenemos un nuevo matching $K' = \{(1,2),(3,5),(4,6)\}$ que tiene, efectivamente, un eje más que K.

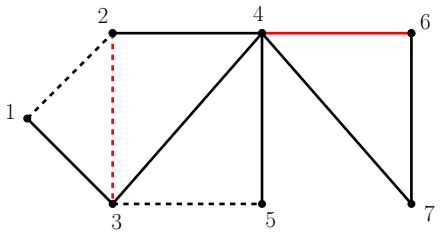


Figura 4.1: Grafo ejemplo 6

Lema 4.1. *Sea $G = (V,E)$ un grafo simple y sean K y K' dos matchings distintos de G. Entonces las componentes conexas del subgrafo $G_1 = (V,E_1)$ donde $E_1 = K\Delta K' = (K - K') \cup (K' - K)$ son de tres tipos:*

Tipo 1 : un punto aislado

Tipo 2 : un ciclo simple par cuyos ejes pertenecen alternativamente a K y K'

Tipo 3 : un camino simple cuyos ejes pertenecen alternativamente a K y K'

Demostración. Notamos que el grado de cualquier vértice de G_1 es como máximo 2. De hecho, puede haber como máximo un eje de K y un eje de K' incidente con cada vértice $i \in V$. De esto se sigue que las componentes sean del tipo 1, 2 o 3 vistos en el enunciado. □

Teorema 4.2. *Un matching K es máximo si y solo si no existe un camino alternado de aumento con respecto a K.*

Demostración. $\Rightarrow$) Veámoslo por reducción al absurdo. Sea K un matching máximo. Supongamos que existe un camino alternado de aumento L, luego podemos construir el matching $K' = (K \cup L) - (K \cap L)$ de orden $|K'| = |K| + 1$. Contradicción con que K sea máximo luego no existe tal camino alternado de aumento.
$\Leftarrow$) Sea K un matching de forma que no hay un camino alternado de aumento respecto de K. Sea K' un matching máximo, por $(\Rightarrow)$ sabemos que no puede haber un camino alternado de aumento tampoco respecto de K'. Así, el subgrafo $G_1 = (V, K\Delta K')$ no tiene componentes impares del tipo 3. Por tanto, $|K - K'| = |K' - K|$ luego $|K| = |K'|$ de donde deducimos que K es un matching máximo. □

Luego este teorema nos dice que para encontrar un matching máximo, basta con encontrar un camino alternado de aumento (si existe) a partir de algún matching $K \subset E$.

Notar que es claro que un matching $K \subset E$ que tiene solo un vértice no saturado es máximo.

Corolario 4.3. *Sea K un matching máximo de $G = (V,E)$. Entonces, todo matching máximo K' de G se puede obtener de K gracias a una secuencia de caminos alternados disjuntos dos a dos y que son o ciclos simples alternados pares o caminos alternados pares.*

Demostración. Consideramos el subgrafo $G_1 = (V, K\Delta K')$. Observamos que no tiene componentes impares del Tipo 3 (lema 4.1) porque esto significaría que para K o K' existiría un camino alternado de aumento y esto contradice que K y K' sean máximos. Entonces basta con realizar una trasferencia sobre cada componente del tipo 2 o sobre cada componente del tipo 3. □

Veamos algún ejemplo de por qué los ciclos impares dificultan la obtención de caminos alternados de aumento.

Ejemplo 7. En el ejemplo que ilustramos en la figura 4.2 tenemos el matching K representado en rojo. En la búsqueda de un camino alternado de aumento partimos del vértice no saturado 1. Es claro que existe un camino alternado de aumento que va del vértice 1 al vértice 8 (ambos no saturados). Sin embargo, cuando llegamos al vértice 3, si en vez de continuar por el vértice 4 lo hacemos por el

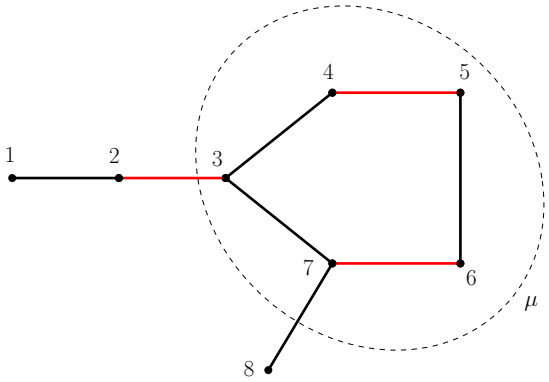


Figura 4.2: Grafo ejemplo 7

7 no logramos encontrarlo. Es decir, dependiendo del sentido en el que recorremos el ciclo impar μ encontramos el camino alternado de aumento o no.

Así, hemos visto el gran problema que supone el hecho de que haya ciclos impares en nuestro grafo a la hora de buscar caminos alternados de aumento.

Definición. Llamamos **grafo reducido** G/V_1 al obtenido de $G=(V,E)$ reduciendo a un punto un conjunto de vértices $V_1 \subset V$. Es el grafo cuyo conjunto de vértices es $V-V_1+\{v_1\}$ (el conjunto de vértices V_1 es reemplazado por un solo vértice $\{v_1\}$) y cuyos ejes son de la forma : (i,j) si $i \in V-V_1, j \in V-V_1$ y $(i,j) \in E$; y (i,v_1) si $i \in V-V_1$ y existe un $j \in V_1$ tal que $(i,j) \in E$.

Lema 4.4. *Sea $V_1 \subset V$ el conjunto de vértices de un ciclo simple impar μ con $|V_1| = 2k+1$. Si K_1 es un matching de G/V_1, entonces existe un matching máximo K_μ de μ tal que $K = K_1 \cup K_\mu$ es un matching de G.*

Demostración. Como K_1 es un matching de G/V_1, hay como mucho un eje de K_1 incidente con el vértice v_1 de G/V_1. Si ningún eje de K_1 es incidente con v_1, ya lo tenemos pues cualquier matching de μ de orden k será el que estamos buscando. Si hay un único eje de K_1 es incidente con v_1, tenemos que este eje es incidente en G con un vértice de μ. Llamamos a este vértice v_0 y elegimos para K_μ el único matching máximo de μ tal que v_0 es no saturado. □

Lema 4.5. *Sea K un matching de G tal que existen al menos dos vértices no saturados y sea $\mu=(V_1,E_1)$ un ciclo simple impar $(|V_1|=2k+1)$ tal que $K_\mu = K \cap E_1$ $(|K_\mu|=k)$ es un matching máximo de μ.*
Sea j_0 el vértice no saturado de μ respecto a K_μ y supongamos que j_0 está unido a un vértice no saturado i_0 (respecto a K) por un camino alternado par L_0 (posiblemente de longitud 0).
Entonces K es un matching máximo de G si y solo si $K_1 = K - K_\mu$ es un matching máximo de G/V_1.

Demostración. $\Rightarrow$) Es claro por el lema 4.4.
$\Leftarrow$) Lo probamos por reducción al absurdo. Supongamos que K_1 es un matching máximo de G/V_1 pero que K no es un matching máximo de G. Consideramos los matchings K' y K_1' obtenidos a partir de K y K_1 haciendo transferencias sobre L_0. Como L_0 es par, $|K|=|K'|$; y por tanto, K' tampoco es un matching máximo de G. Por otro lado, K_1' es un matching de G/V_1 y es máximo ya que $|K_1'|=|K_1|$. El vértice i_0 que era no saturado para K, es saturado para K'. Por otro lado, el vértice j_0 ha pasado a ser no saturado para K' lo que implica que el vértice μ es no saturado para K_1' en el grafo reducido G/V_1. Como K' no es máximo en G, existe un camino alternado de aumento L en G que une dos vértices no saturados en K': a y b. El camino L se tiene que encontrar con μ ya que si no sería camino alternado de aumento para K_1', lo cual es imposible. Además, por lo menos uno de los vértices a o b es distinto de j_0. Supongamos $a \neq j_0$. Sea L_a el trozo del camino L que va desde a hasta el primer vértice de μ. El eje de L_a incidente con μ no puede pertenecer a K' (de hecho, ningún eje incidente con μ pertenece a K') y, por lo tanto, tampoco a K_1'. En G/V_1, el camino alternado L_a que une los vértices no saturados para K_1': a y μ, es un camino de aumento. Esto contradice que K_1' sea máximo luego ya lo tenemos. □

4.2. Buscar caminos alternados a partir de un origen dado. Árboles alternados.

Sea $K \subset E$ un matching de $G = (V, E)$ y sea i_0 un vértice no saturado. Sabemos que si i_0 es el único vértice no saturado entonces K es máximo. En el caso general de tener más de un vértice no saturado, tenemos que encontrar un método que nos permita o encontrar un camino alternado de aumento entre i_0 y otro vértice no saturado o afirmar que tal camino no existe.

Definición. Un **árbol alternado** con raíz i_0 respecto a un matching K es un subgrafo de G, $\mathscr{T} = (V_1, E_1)$, conexo y sin ciclos que cumple:

i) $i_0 \in V_1$ es el único vértice no saturado del árbol

ii) para cada $j \in V_1$ el camino $L_{E_1}(j)$ del árbol entre i_0 y j es un camino alternado

iii) los caminos que unen i_0 con los vértices de grado 1 en el árbol tienen un número par de ejes

Además, decimos que los vértices de G que pertenecen a V_1 son vértices **etiquetados** y; por tanto, los que pertenecen a $V - V_1$ los denominamos **no etiquetados**.

Definición. Dado un grafo G, decimos que el árbol alternado es **completo** si contiene a todos los vértices para los que existe un camino alternado desde i_0.

Ejemplo 8. Representamos un árbol alternado $\mathscr{T} = (V_1, E_1)$ en el que los ejes del matching K están en color rojo.

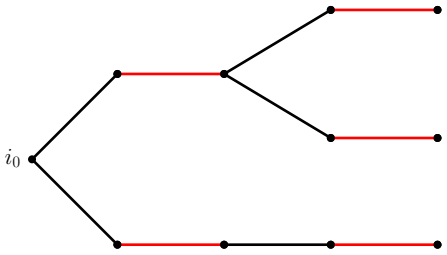


Figura 4.3: Árbol alternado

Sea $j \in V_1$ un vértice etiquetado. Si el camino $L_{E_1}(j)$ tiene un número par de ejes, decimos que el vértice j es **par**. Por el contrario, si el número de ejes de $L_{E_1}(j)$ es impar, diremos que el vértice j es **impar**. Si en un árbol alternado vamos desde un vértice j hasta la raíz i_0 entonces los vértices que nos vayamos encontrando serán alternativamente pares e impares.
Cabe notar que en un árbol alternado, todos los vértices de grado 1 son vértices pares y los vértices impares tienen que tener grado 2. El primer caso es trivial y el segundo se debe a que, por la propia definición de árbol alternado, un vértice impar siempre es alcanzado por ejes no pertenecientes al matching y de él salen ejes del matching. Además, no puede ser alcanzado por más de un eje y no se verificaría la definición de matching si de él saliese más de un eje del matching.

Una vez tenemos clara la definición de árbol alternado veamos un algoritmo para construirlos a partir de un grafo dado:

Algoritmo para construir árboles alternados

Denotamos por $V_1^0 \subset V_1$ al subconjunto de vértices etiquetados pares. La construcción del árbol alternado la haremos paso a paso empezando por $V_1 = V_1^0 = \{i_0\}$, $E_1 = \emptyset$. Al principio, la construcción la hacemos basándonos en el grafo inicial G pero, tras un número de pasos, nos basaremos en otro grafo G' que se obtiene a partir de G reduciendo algunos subconjuntos de vértices. En cada paso, $\mathscr{T} = (V_1, E_1)$ induce un árbol alternado en G'. El procedimiento es el siguiente:

1) Empezamos con $V_1 = V_1^0 = \{i_0\}$, $E_1 = \emptyset$.

2) En la etapa correspondiente, tenemos el árbol alternado $\mathscr{T} = (V_1, E_1)$. El objetivo es aumentarlo añadiendo ejes y vértices adicionales. Hay cuatro posibles casos:

Caso 1: Existe un vértice par $i \in V_1^0$ y un eje $u = (i, j)$, con j no saturado y no etiquetado $(j \notin V_1)$. Entonces, el camino $L' = L_{E_1}(i) + \{u\}$ es un camino alternado de aumento de G' e induce en G un camino alternado de aumento L que une los dos vértices no saturados i_0 y j. Es decir, haciendo una trasferencia sobre L' a la vez que expandimos los posibles blossoms creados en iteraciones anteriores en el caso 3 obtenemos un matching mejor que el inicial. Notar que si $G = G'$ entonces $L = L'$. Fin del procedimiento tras haber hallado un matching mejor. (Exit A)

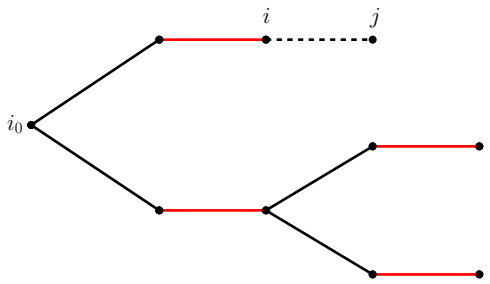


Figura 4.4: Grafo caso 1

Caso 2: Existe un vértice par $i \in V_1^0$ y un eje $u = (i, j)$ con j saturado y no etiquetado $(j \notin V_1)$. Sea $u' = (j, k)$ el eje del matching incidente con j. Entonces, añadimos al árbol que tenemos los ejes u y u' y los vértices j (impar etiquetado) y k (par etiquetado).

$$E_1 \leftarrow E_1 + \{u\} + \{u'\}$$
$$V_1 \leftarrow V_1 + \{j\} + \{k\}$$
$$V_1^0 \leftarrow V_1^0 + \{k\}$$

Volvemos a 2).

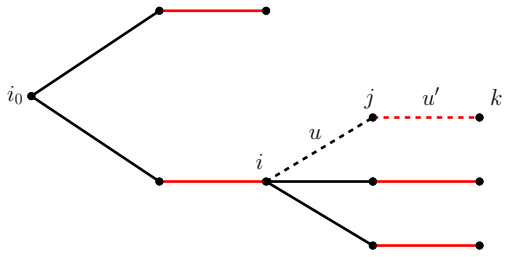


Figura 4.5: Grafo caso 2

Caso 3: Existe un vértice par $i \in V_1^0$ y un eje $u = (i, j)$ con j también par y etiquetado $(j \in V_1^0)$. El eje u hace que se forme un único ciclo μ con los ejes de $\mathscr{T}$. Notamos que el número de vértices de μ tiene que ser impar (el orden es un número impar). Es claro que los caminos $L_{E_1}(i)$ y $L_{E_1}(j)$, de orden par, que tienen por lo menos el vértice i_0 en común se encuentran en algún vértice r (posiblemente $r = i_0$). Tenemos que r, que tiene por lo menos grado 3 a no ser que $r = i_0$, tiene que ser etiquetado par. Los subcaminos L_i y L_j de $L_{E_1}(i)$ y $L_{E_1}(j)$, que van desde r hasta i y desde r hasta j respectivamente, son pares y $\mu = L_i + L_j + \{u\}$ es de orden impar.

El matching $K_\mu = K \cap \mu$ es un matching máximo de μ y r, el único vértice no saturado del ciclo para K_μ, está unido con i_0 por un camino alternado par $L_{E_1}(r)$. El ciclo μ decimos que

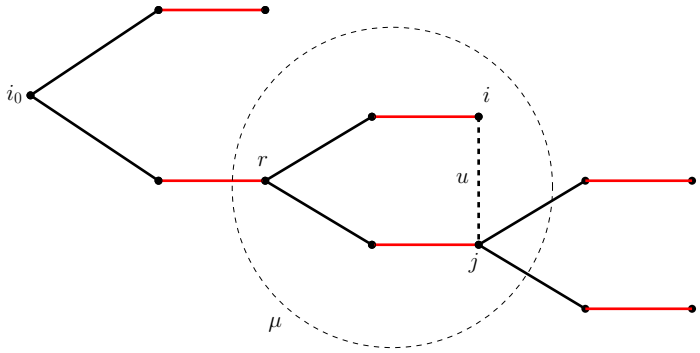


Figura 4.6: Grafo caso 3 inicial

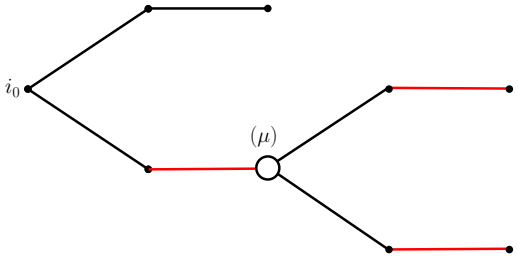


Figura 4.7: Grafo caso 3 reducido

es un **blossom** de K y el vértice r una raíz del blossom. Sea V_μ el conjunto de vértices de μ y G'/μ el grafo reducido de G' respecto de V_μ. El vértice correspondiente a μ en G'/μ lo denotamos por (μ). Aplicando el lema 4.5, reemplazamos G' por G'/μ, $\mathscr{T}$ por $\mathscr{T}/\mu$ y volvemos a 2).

Caso 4: Si no se da ninguno de los casos anteriores, no existen vértices que puedan ser etiquetados y no podemos crear ningún blossom. Fin del procedimiento. (Exit C)

El procedimiento descrito siempre termina ya sea dando un camino alternado de aumento con origen i_0 (Exit A) o encontrando un árbol alternado (Exit C).

Teorema 4.6. *El árbol completo obtenido al implementar el algoritmo para construir árboles alternados contiene todos los vértices para los que existe camino alternado desde la raíz del árbol i_0; es decir, es completo. Además, si existe camino alternado de aumento desde i_0 hasta un vértice i, entonces el algoritmo lo encuentra.*

Demostración. Supongamos que el vértice i no pertenece al árbol con raíz i_0, luego i es no etiquetado. Como existe un camino alternado entre i_0 e i, el vértice saturado que precede a i en el camino será etiquetado por haber sido etiquetado el vértice anterior; y así, sucesivamente hasta llegar al vértice del que partimos (i_0). Por tanto, al realizar una nueva iteración del algoritmo tendremos que marcar el vértice i como etiquetado, lo cual contradice la hipótesis inicial.
Además, si el vértice i es no saturado y no etiquetado, por lo que acabamos de ver, el vértice anterior pertenecerá al árbol. Realizando una nueva iteración del algoritmo nos encontramos ante el caso 1) e i tendría que ser etiquetado por lo que llegamos a contradicción. □

Ejemplo 9. Aplicamos el algoritmo para construir árboles alternados al grafo G que ilustramos en la figura 4.8. Tenemos el matching $K = \{(2,4),(5,7),(9,10),(11,12)\}$ y comenzamos a construir el árbol partiendo de $i_0 = 1$. Observamos que como el vértice 4 es saturado y no etiquetado estamos en el caso 2) del algoritmo y añadimos al árbol los ejes $u = (1,2)$ y $u' = (2,4)$. De la misma manera, aplicando lo enunciado el caso 2) cuatro veces, obtenemos el árbol representado en la figura 4.8 en línea discontinua.

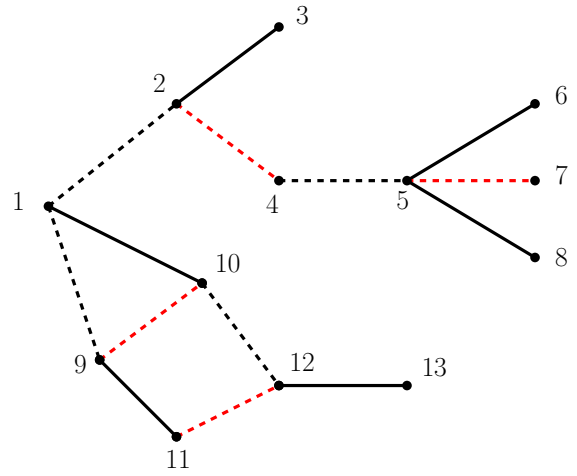


Figura 4.8: Grafo G

Observamos que no podemos seguir añadiendo ejes al árbol pero partiendo de nuevo del vértice 1 (par) tenemos que el vértice 10 es par y etiquetado. Así, por el caso 3) el eje $(1,10)$ forma un blossom $\mu_1 = \{1,9,10\}$ representado en la figura 4.9.

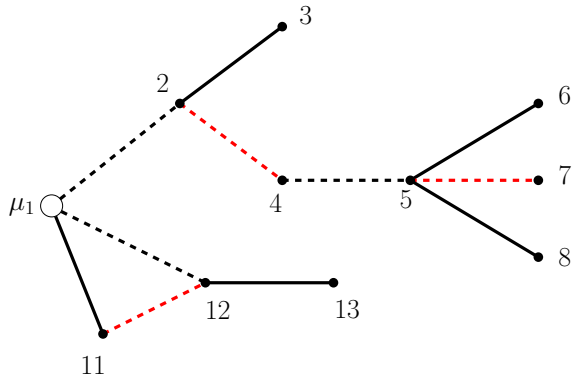


Figura 4.9: Grafo G con el primer blossom μ_1

De nuevo, repetimos esto último teniendo en cuenta que partimos del vértice par μ_1 y que el vértice 11 es también par y etiquetado. Luego el eje $(\mu_1, 11)$ forma un nuevo blossom $\mu_2 = \{\mu_1, 11, 12\}$. Lo representamos en la figura 4.10.

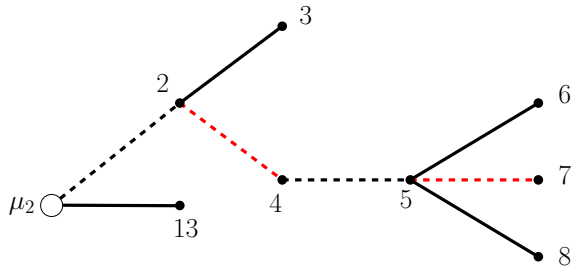


Figura 4.10: Grafo G con el segundo blossom μ_2

Ahora bien, realizando una nueva iteración del algoritmo, vemos que el vértice μ_2 es par y el vértice 13 es no saturado y no etiquetado. Así pues, estamos en el caso 1) del algoritmo y realizando una transferencia sobre el camino alternado L, que va desde el vértice 1 hasta el vértice 13 y está representado en gris y rojo en la figura 4.11, aumentamos en una unidad el número de ejes del matching.

Además, es claro que a partir de 1 no podemos continuar el proceso ya que no existen más vértices

que puedan ser etiquetados y no podemos crear ningún blossom.

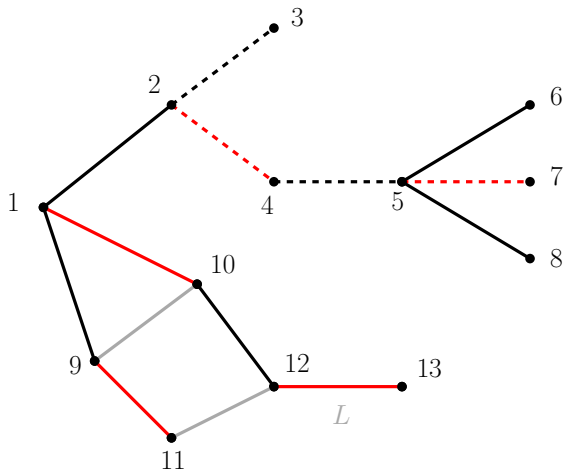


Figura 4.11: Grafo G final

Cabe destacar que, como tomando como raíz del nuevo árbol $i_0 = 3$ no podemos alcanzar ninguno de los vértices no saturados (vértices 6 y 8), no existe ningún camino alternado de aumento que comience en 3. Observamos que si desde 3 no podemos construir ese camino alternado de aumento, entre 6 y 8 tampoco va a existir. Por lo tanto, como veremos más adelante con el algoritmo de Edmonds, tenemos que el matching $K' = \{(1,10),(9,11),(12,13),(2,4),(5,7)\}$ es máximo para el grafo G.

Veamos ahora un resultado que nos dice que cuando encontramos un árbol completo no puede existir un camino alternado de aumento conteniendo los vértices etiquetados.

Teorema 4.7. *Sea $V_1 \subset V$ el conjunto máximo de vértices etiquetados y $V_2 = V - V_1$ el conjunto de vértices no etiquetados de G. Denotamos por $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ los subgrafos de G generados por los vértices V_1 y V_2, respectivamente. Sea $K_1 = K \cap E_1$ el conjunto de ejes del matching en G_1 y $K_2 = K \cap E_2$ el conjunto de ejes del matching en G_2. Entonces K es un matching máximo de G si y solo si K_2 es un matching máximo de G_2.*

Demostración. Sabemos que no existe un eje $u = (i_1, i_2)$ de K tal que $i_1 \in V_1$, $i_2 \in V_2$ ya que si existiese i_1 sería etiquetado impar y i_2 no etiquetado por lo que no completaríamos el proceso de etiquetado y llegaríamos a contradicción. Así, tenemos $K = K_1 \cap K_2$.
$\Rightarrow$) Si K_2 no es un matching máximo de G_2 entonces es claro que K no es máximo en G.
$\Leftarrow$) Veamos que si K_2 es máximo en G_2 no puede existir un camino alternado de aumento para K en G: Supongamos que existe tal camino L_0 uniendo los dos vértices no saturados j_0 y k_0. No podemos tener $i_0 = j_0$ porque entonces j_0 hubiese sido un vértice etiquetado. Por el mismo motivo, no podemos tener $i_0 = k_0$. Por lo tanto, j_0 y k_0 pertenecen a V_2. El camino L_0 está formado por vértices de V_1 ya que sino K_2 no sería máximo en G_2. Sea l un vértice de L_0. Como l está etiquetado, existe un camino alternado L_1 (par o impar) entre i_0 y l. Sea l' el primer vértice de L_0 que encontramos al recorrer L_1 desde i_0. Representamos el camino alternado que va de i a j como $(i - j)$. Luego se puede probar fácilmente que l' tiene que ser etiquetado impar y que divide L_0 en dos subcaminos alternados $(l' - j_0)$ y $(l' - k_0)$ los cuales uno es par y el otro impar. Si $(l' - j_0)$ es par entonces j_0 tiene que ser etiquetado y en caso contrario el que tiene que ser etiquetado es k_0. Esto contradice el hecho de que el proceso de etiquetado se haya completado. Por tanto, el matching K tiene que ser máximo. □

Para que quede más clara la demostración, damos un ejemplo en la figura 4.12 en el que observamos que si j_0 no está etiquetado tenemos un camino alternado entre i_0 y j_0 representado con una línea discontinua, lo que contradice que el proceso de búsqueda de caminos alternados se haya completado.

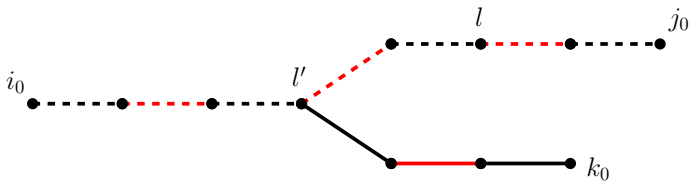


Figura 4.12: Ejemplo Teorema 4.7

4.3. Algoritmo de Edmonds

Una vez vistos todos estos resultados podemos dar el Algoritmo de Edmonds, cuyo objetivo es buscar el máximo matching a partir de un grafo dado.

Algoritmo de Edmonds

a) Empezamos con un matching cualquiera $K = K^0$ (por ejemplo $K^0 = \emptyset$).

b) Si no hay más vértices aislados para K, FIN: el matching es máximo. En caso contrario, sea i_0 un vértice aislado cualquiera. Construimos un árbol alternado con raíz i_0.

c) Cuando se termina el proceso solo hay dos opciones:

   1) Exit C: El árbol alternado es completo. Si no hay más vértices aislados no etiquetados, FIN. El matching actual es máximo. En caso contrario, volvemos a b) y construimos un nuevo árbol alternado en el subgrafo inducido por los vértices no etiquetados.

   2) Exit A: Hemos encontrado un camino alternado de aumento entre i_0 (o el vértice de G' que contiene a i_0) y otro vértice aislado de G'. Realizamos en G' una transferencia sobre este camino alternado y reconstruimos el matching inducido en el grafo inicial G. El matching K' verifica $|K'| = |K| + 1$. Tomamos como nuevo K el matching K', eliminamos todas las etiquetas del árbol alternado y volvemos a b).

Una vez finalizado el proceso, es importante saber cómo expandir los blossoms creados por el algoritmo; es decir, cómo reajustar los ejes del matching al hacer la expansión. Se trata de reemplazar en el grafo reducido G', los vértices (μ) por los ciclos impares μ a partir de los cuales fueron creados y, más adelante, cambiar el máximo matching de μ, K_μ, por otro matching máximo compatible con el matching de G'. Hay que tener en cuenta que la expansión de varios blossoms consecutivos se hace en orden inverso al que fueron creados.

Notamos que, en el algoritmo, cuando encontramos un primer blossom μ_1 continuamos el proceso a partir del grafo reducido $G_1 = G/\mu_1$. Al vértice (μ_1) obtenido al reducir μ_1 lo llamaremos **pseudo-vértice**. Cuando encontramos sucesivamente los blossoms $\mu_1, ..., \mu_p$ entonces el grafo reducido G' al que llegamos, es el grafo $G_p = G/\mu_1/.../\mu_p$ con pseudo-vértices $\mu_1, ..., \mu_p$.

A continuación, describimos el proceso de expansión del pseudo-vértice k (μ_k) que consiste en dos pasos:

1) Comprobar la compatibilidad del matching en G' con el matching del blossom y determinar el extremo final k_1 del eje del matching de G' incidente con el blossom μ_k.

2) Si el matching que teníamos marcado inicialmente en el blossom es compatible lo dejamos como está. En caso contrario, cambiamos los ejes del matching por los ejes que no pertenecían al matching en el camino impar que va desde el vértice k_1 hasta la antigua raíz del blossom μ_k; es decir, hacemos una transferencia sobre el camino impar que va desde k_1 hasta μ_k.

Veamos ahora un ejemplo en el que aplicamos el algoritmo de Edmonds y mostramos de una manera más clara cómo es la expansión de los pseudo-vértices.

Ejemplo 10. Partimos del grafo G dibujado en la figura 4.13 en el que el matching $K = \{(2,3),(4,5),(6,7)\}$ está representado en rojo.

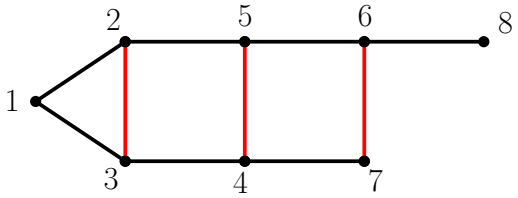


Figura 4.13: Matching inicial en G

Buscamos un camino alternado de aumento construyendo un árbol alternado con raíz $i_0 = 1$. El árbol alternado está representado con una línea discontinua en la figura 4.14.

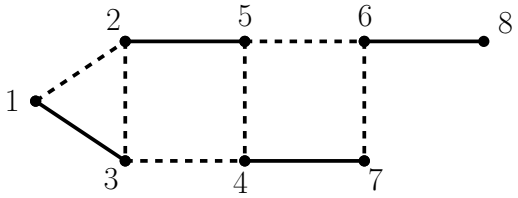


Figura 4.14: Árbol alternado en G

Observamos que los vértices pares son $1,3,5$ y 7 y los impares $2,4$ y 6.
El eje $(1,3)$ forma el blossom $\mu_1 = \{1,2,3\}$ luego obtenemos el grafo G_1 representado en la figura 4.15 con el árbol $\mathscr{T}_1 = \mathscr{T}/\mu_1$ marcado con una línea discontinua.

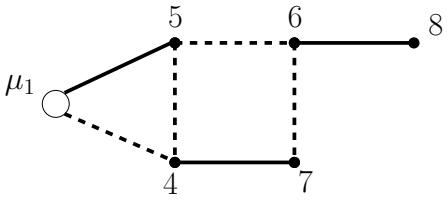


Figura 4.15: Grafo G_1

Observando el grafo G_1, tenemos que el eje $(\mu_1,5)$ forma el blossom $\mu_2 = \{\mu_1,4,5\}$. Así, llegamos de nuevo a un grafo G_2 (Figura 4.16) en el que representamos el árbol $\mathscr{T}_2 = \mathscr{T}_1/\mu_2$ con una línea discontinua.

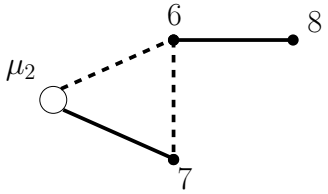


Figura 4.16: Grafo G_2

De la misma manera, continuamos formando un último blossom $\mu_3 = \{\mu_2,6,7\}$ a partir del eje $(\mu_2,7)$ y llegamos al grafo G_3 (Figura 4.17). El árbol alternado $\mathscr{T}_3 = \mathscr{T}_2/\mu_3$ está formado por un solo vértice μ_3.

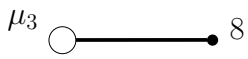
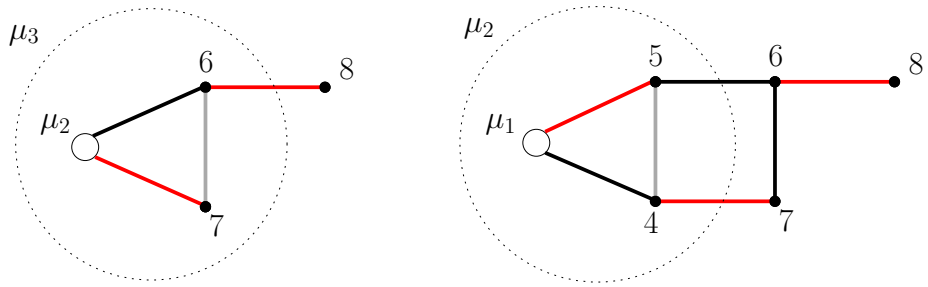


Figura 4.17: Grafo G_3

Ahora bien, observamos que el grafo G_3 está formado por un único eje. El nuevo matching se construye marcando ese único eje y expandiendo los blossoms μ_3, μ_2 y μ_1 sucesivamente a la vez que vamos reajustando los ejes que pertenecen al matching teniendo en cuenta el primero que hemos marcado. Veámoslo gráficamente teniendo en cuenta que representaremos el nuevo matching en color rojo y que dejaremos el matching inicial K en color gris para que sea más evidente lo que estamos modificando.



Finalmente, tras expandir el último blossom μ_1 obtenemos el siguiente matching máximo:

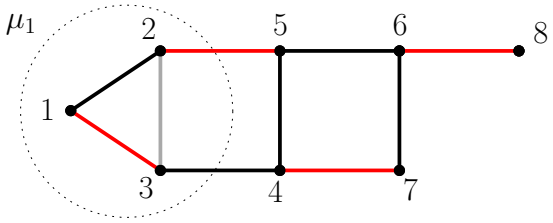


Figura 4.19: Grafo G'

Por último, analizamos la complejidad computacional del algoritmo de Edmonds.

Teorema 4.8. *La complejidad computacional del algoritmo de Edmonds es* $O(n^4)$.

Demostración. En el algoritmo de Edmonds tenemos dos posibles salidas: la salida A (hemos hallado un camino alternado de aumento) y la salida C (el árbol alternado es completo).
El proceso de búsqueda de caminos de aumento se realiza como mucho $n/2$ veces. Una vez hallado un camino alternado de aumento en el grafo reducido G', recuperar el camino alternado de aumento en el grafo original G cuesta $O(m)$. Finalmente, cambiar los ejes de K por los de $\bar{K} = E - K$ a lo largo del camino cuesta $O(n)$. En total, el proceso de aumentar el número de ejes del matching mediante caminos alternados de aumento supone un coste $O(nm)$, lo cual es $O(n^3)$ porque el número de ejes m es $O(n^2)$.
Por otro lado, cada vez que hallamos un árbol alternado completo, en los pasos siguientes ya no tenemos que analizar los vértices pertenecientes a este árbol. Por tanto, este proceso lo podemos realizar como mucho $O(n)$ veces. Eliminar los vértices del árbol y los ejes correspondientes supone un coste $O(n+m)$. Además, el número de salidas A más el número de salidas C es a lo sumo $O(n)$.
Determinamos ahora la complejidad de hallar un árbol completo. Si no formásemos blossoms a lo largo del proceso, encontrar el árbol tendría un coste de $O(m)$. Sin embargo, sí que podemos formar blossoms en cada paso y como mucho pueden ser $O(n)$. En cada contracción hay que crear nuevos ejes (con coste $O(m)$) y, por tanto, el proceso de formar el árbol puede costar $O(nm)$. Como el número máximo de veces que se podía construir un árbol es de orden $O(n)$, la complejidad total del algoritmo es $O(n^2m) = O(n^4)$. □

Es importante recalcar que en el proceso de contracción de grafos no es necesario guardar una copia de cada grafo reducido. Puede guardarse una estructura de orden $O(m)$ con toda la información necesaria para hallar árboles en los grafos reducidos de manera que el número de operaciones pase a ser de orden $O(nm) = O(n^3)$. Esto se explica de forma muy detallada, por ejemplo, en la referencia [3].
Se pueden hacer algunas otras mejoras pero lo que nos interesa es que la complejidad computacional es de orden polinomial y es asequible la implementación de este algoritmo en la práctica.

Bibliografía

[1] A. GARCÍA, *Teoría de grafos*, Universidad de Zaragoza.

[2] BOLLOBÁS, BÉLA, *Modern graph theory*, Springer Science & Business Media, 2013.

[3] M. GONDRAN, M. MINOUX, *Graphs and Algorithms*, Editions Eyrolles, Paris, 1979.

[4] DIESTEL, REINHARD AND SCHRIJVER, ALEXANDER AND SEYMOUR, PAUL, *Graph theory*, Oberwolfach Reports, 2010.

[5] JACK EDMONDS, *Paths, trees, and flowers*, Canadian Journal of mathematics, Cambridge University Press, 1965, disponible en `https://math.nist.gov/~JBernal/p_t_f.pdf`.

[6] LESLIE HALL, *Computational Complexity*, disponible en `http://www.esi2.us.es/~mbilbao/complexi.htm`.

[7] THOMAS F. HOFMANN, *Discrete Algorihtms*, Technical University of Munich, disponible en `https://algorithms.discrete.ma.tum.de/matching/`.

[8] WEST, DOUGLAS BRENT AND OTHERS, *Introduction to graph theory*, Prentice hall Upper Saddle River, 2001.