UNIVERSIDAD DE ZARAGOZA

Máster Universitario en Modelización e Investigación Matemática,
Estadística y Computación

Diseño de una herramienta para la valoración del precio de viviendas

Trabajo de Fin de Máster

Autora:
Beatriz A. Muzás Crespo

Directores:
Dr. D. Pedro M. Mateo Collazos
Dr. D. Manuel J. Salvador Figueras

Zaragoza, 2015

Índice general

Índice de figuras

Índice de tablas

Lista de Algoritmos

Agradecimientos

A mis tutores, Manuel y Pedro. A ti, Manuel, por aportar un punto de vista diferente y enriquecedor. Y en especial, gracias Pedro, de todo corazón, sin tu paciencia y tu apoyo este trabajo no hubiera sido posible.

A mi madre, a mi hermana y a Sora, que han soportado mis agobios y, como siempre, me han ayudado en todo lo que han podido. Os quiero mucho, aunque no lo diga con frecuencia.

A mis amigos, y en especial, a Manu, Marta y Eduardo, que estén lejos o estén cerca, siempre están ahí para mí. Y a ti Cris, que has demostrado una fuerza y una valentía enorme en los momentos más difíciles, eres una persona realmente inspiradora y una amiga excepcional.

Introducción

En el mercado inmobiliario es de suma importancia conocer el valor de una vivienda. Los vendedores quieren obtener el mayor beneficio posible en una operación de compra-venta, mientras que los compradores trataran de conseguir pagar lo menos posible. En ambas situaciones se impone conocer el valor real a la hora de poder negociar al alza o la baja. Pero no solo es importante conocer el valor de una vivienda de cara a venderla o a comprarla. Hay múltiples organismos, tanto públicos como privados, que necesitan contar con el valor de mercado de una vivienda y que se podrían ver beneficiados de una herramienta que ayudara a la determinación de estos valores.

Por ejemplo, en el catastro que hay en todos los municipios se encuentra el valor de cada inmueble (valor catastral) y en base a este se liquida un impuesto, llamado Impuesto sobre Bienes Inmuebles (IBI) o coloquialmente "la contribución", que se cobra todos los años a los propietarios de dicho inmueble. La recaudación de este impuesto es de vital importancia tanto para el Estado como para los Entes Locales ya que sus arcas dependen, en gran medida, de su buena gestión. Así mismo, también el Impuesto sobre Trasmisiones Patrimoniales o el Impuesto sobre Sucesiones y Donaciones se determina en base a valores inmobiliarios y estos impuestos van a repercutir en este caso sobre las arcas de las Comunidades Autónomas.

Por otro lado, si se contase con una herramienta que permitiese deducir el valor de cada vivienda, la Administración del Estado vería facilitado su trabajo de inspección de las declaraciones de patrimonio de los obligados al pago de los impuestos ya que sólo deberían realizar una labor comparativa entre el valor declarado y el real.

De igual modo, esto sería útil en Juzgados mercantiles, civiles o penales, para atribuir valor a los inmuebles que van a ser subastados, o

que son objeto de división o indemnización y en Jurados Provinciales de Expropiación para valorar los inmuebles que han sido objeto de un procedimiento administrativo de expropiación. A estos Jurados, compuestos por arquitectos y asesores jurídicos, acuden los expropiados que no están de acuerdo con el importe fijado por la administración expropiante como paso previo a la vía judicial y su misión es fijar una indemnización que no solo sea justa para el que ha sido desposeído de su propiedad sino también para la administración que debe pagar el precio fijado.

Como últimos ejemplos, en el ámbito de lo privado, las compañías de seguros precisan saber el valor de las viviendas aseguradas para hacerse cargo de las indemnizaciones por siniestros tales como incendios, robos o sucesos menos probables como catástrofes. Mientras que las entidades bancarias requieren conocer el precio de los inmuebles sobre los que imponen cargas mejor de cara a recuperar inversiones frente a un posible impago.

Por tanto, el objetivo que se persigue en el presente Trabajo de Fin de Máster es precisamente dar respuesta a todas estas necesidades creando una herramienta para la valoración de precios de viviendas. Para ello se propone utilizar técnicas de algoritmos genéticos y regresión lineal desde un enfoque bayesiano. Estas técnicas van a ser presentadas en el siguiente capítulo junto con una descripción detallada del problema y su propuesta de resolución. Posteriormente, en el Capítulo 3, se expone de manera pormenorizada el algoritmo implementado y en el Capítulo 4 se puede encontrar el desarrollo del experimento y se muestran los resultados obtenidos. Estos, junto con la herramienta desarrollada, serán trasladados a un conjunto de expertos para su posterior revisión y validación. En el Capítulo 5 se propone una mejora del algoritmo detallado en el Capítulo 3 y se presentan los resultados que se obtienen con su uso. Previo a todos estos resultados en el Capítulo 2 se habrá mostrado un estudio descriptivo de los datos sobre transacciones que se utilizan para la construcción del modelo. Se finaliza esta memoria con la presentación de las conclusiones en el Capítulo 6.

Capítulo 1

Antecedentes y propuesta

Una vez justificado en la Introducción el interés del trabajo vamos a plantear en este capítulo de qué modo se va a proceder para construir la herramienta de valoración de viviendas. Para la realización del trabajo se cuenta con la colección de transacciones de viviendas de la Comunicad Autónoma de Aragón entre Enero de 2010 a Mayo de 2014, ambos inclusive. Estos datos han sido proporcionados por el grupo de análisis del mercado inmobiliario (GAMERIN). Para cada una de estas transacciones se ha registrado la información siguiente:

P: Precio total de la vivienda.

S: Superficie de la vivienda.

A: Antigüedad de la vivienda.

C: Índice de calidad de la vivienda.

M: Municipio en el que se encuentra encuadrada.

El precio de cada vivienda proviene del Colegio Nacional de Registradores de la Propiedad y los datos de superficie, antigüedad así como el índice de calidad de la vivienda tienen su origen en el catastro del municipio en el que se encuentra encuadrada.

Con el objeto de realizar la estimación del valor de la vivienda P en función del resto de variables se plantea el modelo de regresión recogido en la Ecuación 1.1.

$$\log P = \beta_{g,0} + \beta_{g,S} \log S + \beta_{g,A} \log A + \beta_{g,C} C + \epsilon_g, \quad g = 1, \ldots, Q,\ \epsilon_g \sim N(0, \sigma_g) \quad (1.1)$$

En este modelo se divide el conjunto de municipios en Q grupos, cada uno de los cuales aparece indexado con el índice $g = 1, \ldots, Q$. Por tanto, dado el conjunto de municipios $\mathbf{M} = \{M_1, \ldots, M_R\}$ el problema ahora se concentra en determinar una partición $\{G_1, \ldots, G_Q\}$ de éste en Q grupos tales que $\mathbf{M} = \cup_{g=1}^{Q} G_g$ y $G_i \cap G_j = \emptyset, \forall i \neq j$.

De forma habitual este tipo de problemas son modelados mediante Regresión Espacio-Temporal, pero en Aragón, que es el territorio al que vamos a circunscribirnos, las transacciones de viviendas están muy dispersas y por este motivo se propone, para este Proyecto, trabajar a nivel de área (municipio) y plantear la construcción de un conjunto de regresiones asociadas a una partición de los municipios. El uso de este modelo de regresión lineal junto con el enfoque a utilizar para su estimación viene fijado por expertos del grupo de análisis del mercado inmobiliario (GAMERIN) en colaboración con Manuel Salvador Figueras [1], y es por ello, que en el presente trabajo se asume que las hipótesis son las adecuadas para aplicarlo.

El enfoque que se ha adoptado en la construcción del modelo de regresión lineal es bayesiano. Así, si llamamos $\boldsymbol{\beta} = (\beta_0, \beta_S, \beta_A, \beta_C)'$ a la matriz que contiene los parámetros de las regresiones de la Ecuación 1.1 y $\tau = \frac{1}{\sigma^2}$ entonces se asume que $\boldsymbol{\beta}|\tau \sim N_4(\mathbf{0}, \tau^{-1}\mathbf{S}_\beta)$, donde $\mathbf{S}_\beta$ es una matriz diagonal 4×4 conocida y $\tau \sim Gamma\left(\frac{n_0}{2}, \frac{n_0 s_0^2}{2}\right)$ con n_0 y s_0^2 conocidas.

Tras plantear el modelo a construir aparece de forma automática una primera restricción a tener en cuenta. Dicha restricción consiste en que, para que pueda realizarse adecuadamente la estimación de los parámetros, cada grupo debe tener un conjunto de municipios tal que el número acumulado de transacciones supere un umbral que se ha fijado en 40 transacciones por grupo, esto es, diez por cada parámetro del modelo.

Los expertos proponen además que para la construcción de los grupos se tengan en cuenta qué municipios son 'vecinos' de otros, tratando de garantizar que los grupos de municipios estén formados por municipios geográficamente vecinos. Para ello proporciona una relación de vecindad de tal forma que a cada municipio $M_i, i = 1, \ldots, R$ se le asocia un con-

[1] A partir de ahora, cuando nos refiramos a los expertos en esta memoria será en referencia a Manuel Salvador Figueras y al personal del grupo GAMERIN.

junto $V(M_i) = \{M_{i,1}, \ldots, M_{i,n_i}\}$ con sus municipios vecinos inmediatos. Aunque inicialmente los expertos querían considerar dicha restricción como una restricción obligada, finalmente se plantea que el algoritmo que se construye sea capaz de incluir dicha restricción con un grado variable de cumplimiento. En el Anexo I se muestra la tabla de vecindades que proporcionaron y con la que se ha trabajado.

El problema que se presenta consiste por tanto en determinar cuánto vale Q, número de grupos, y la composición de los grupos G_i, $i = 1, \ldots, Q$, penalizando en cierta medida el que un grupo contenga infactibilidades, es decir, municipios que no son vecinos.

Dos municipios M y M' de un cierto grupo G serán vecinos si existe una cadena $M_1 = M, \ldots, M_L = M'$ con $M_j \in G, j = 1, \ldots, L$ y tal que $M_i \in V(M_{i-1}), i = 2, \ldots, L$. El grado de infactibilidad de un grupo se define como el número de subgrupos de municipios suyos en los que los municipios son vecinos menos 1.

El siguiente paso es definir el criterio para determinar si una cierta partición de los municipios es mejor o peor que otra. Es decir dado un número Q de grupos y la partición dada $G_1, \ldots, G_Q$ necesitamos asignarle un valor numérico que refleje su calidad y con el cual pueda ser comparada con otra partición.

Así pues, si como se ha comentado anteriormente, se dispone de una muestra de viviendas de tamaño N y para cada una de ellas se tiene la siguiente información: $\{(P_i, S_i, A_i, C_i, M_{R_i}),\ i = 1, \ldots, N\}$ se buscará maximizar el valor de la densidad marginal $[\mathbf{y}|\mathbf{X}, \mathbf{M}_G]$ donde $\mathbf{y} = (y_1, \ldots, y_N)'$ con $y_i = \log P_i$ y $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)'$ con $\mathbf{x}_i' = (1, \log S_i, \log A_i, C_i)$ y $\mathbf{M_G}$ denota el modelo de la Ecuación 1.1 correspondiente a la partición de grupos G. Las funciones necesarias para la realización de la estimación de los parámetros del modelo así como para la obtención del valor de la densidad marginal asociada han sido proporcionadas por el Codirector del trabajo Manuel Salvador Figueras.

En este momento ya tenemos establecido el problema al que nos enfrentamos, qué es lo que se quiere construir y cómo valorar la calidad del modelo construido. El siguiente elemento a tener en cuenta es la construcción de la herramienta de optimización encargada de determinar la mejor división teniendo en cuenta los dos criterios propuestos así como

la restricción sobre el número de transacciones. La herramienta de optimización seleccionada son los Algoritmos Genéticos.

Un algoritmo genético o evolutivo es un método de búsqueda que se basa en la teoría de la evolución biológica de Darwin para la resolución de problemas. De este modo, el origen de un algoritmo genético se fundamenta en la mejor adaptación, técnicamente llamada *fitness* y que en este trabajo será la densidad marginal menos el número de infactibilidades por vecindad multiplicada por una cierta penalización positiva, de los individuos de su población. Para que un algoritmo genético funcione correctamente en primer lugar se ha de garantizar la diversidad en la población. Esto suele hacerse generando los individuos, que se representan por cromosomas, de manera aleatoria. Para poder encontrar el óptimo todas las soluciones factibles deben ser representables.

El algoritmo genético sigue un proceso similar al que sucede en la naturaleza. Hay varios operadores de variación: recombinación o crossover y mutación. El papel de ambos es generar nuevas soluciones candidatas. En el primer caso, se mezcla la información de los padres en su descendiente y en el segundo se producen cambios directamente en los genes de un cromosoma. Posteriormente, se seleccionan los supervivientes atendiendo a criterios de *fitness* o de edad. En nuestro caso particular, debido a nuestro objetivo nos basaremos en el *fitness* definido para seleccionar los individuos que pasaran a la siguiente generación. En la naturaleza este proceso no tendría fin, sin embargo, en un algoritmo genético se establece un criterio de parada atendiendo al número de iteraciones sin haberse producido mejora, alcanzar algún *fitness* conocido, etc. Estas condiciones de finalización se comprueban en cada generación.

Repasando la literatura encontramos que el problema al que nos enfrentamos podría encuadrarse dentro de los problemas de construcción de agrupamientos. Un ejemplo clásico de este tipo de problemas es el conocido como *bin packing*, en el que se dispone de una colección de ítems cada uno de los cuales tiene un cierto tamaño y de una colección de contenedores de unas dimensiones dadas y lo que se desea conocer es si es posible acomodar dichos ítems en los contenedores disponibles e incluso puede interesar también determinar el mínimo número de contenedores que pueden acomodar todos los ítems. Este problema es un problema NP-Duro ya que no existe ningún algoritmo que en tiempo polinomial sea capaz de resolverlo. El problema planteado tiene pues bastante similari-

dad con este problema en el sentido de tener 28 ítems, los 28 municipios en los que se han registrado transacciones, y tener que determinar como agruparlos de forma que en cada grupo tengamos al menos 40 transacciones. En el problema de empaquetamiento, este se realiza en base a una cota superior, el tamaño del contenedor, y en nuestro caso la cota es inferior pues habrá que garantizar una cantidad mínima de transacciones.

Al tratarse de problemas complejos la resolución exacta de estos solo es posible para problemas de dimensiones reducidas. Como alternativa para problemas de mayor dimensión se puede recurrir a los algoritmos heurísticos, y un caso particular de estos son los algoritmos genéticos que acabamos de comentar unos párrafos atrás. En relación con la resolución del problema de agrupamiento mediante algoritmos genéticos se pueden encontrar diversos artículos en la literatura, como por ejemplo, [1, 2, 3] y [4]. Todos ellos tienen en común el hecho de que se basan en un algoritmo genético que no incorpora mucho conocimiento del problema. Posteriormente en [5] se presenta un algoritmo genético específico para problemas de agrupamientos. En él se plantean unos elementos comunes como son la representación, recombinación y mutación a los que después hay que incorporar los elementos específicos del problema particular a resolver. Aparentemente en la literatura este último algoritmo, o alguna de sus variantes, es superior al resto. Sin embargo, este hecho no queda muy claro ya que algunas comparaciones, como por ejemplo [6], se realizan con un algoritmo genético básico, similar al presentado por Holland en [7] o a la versión de Michalewicz en [8], que es un algoritmo introductorio que ha sido superado por muchas otras versiones de algoritmos genéticos. Debido a esto, y por indicación de los Directores, el algoritmo ha sido diseñado casi desde cero sin considerar estas opciones. Se ha partido del material de la asignatura *Algoritmos Bioinspirados y Técnicas de Computación Evolutiva* del máster en *Modelización e Investigación Matemática, Estadística y Computación.*

Por último, en cuanto al software empleado, se ha de señalar que se ha utilizado R [9], CPLEX [10] y Matlab [11].

Capítulo 2

Análisis descriptivo de los datos

Como se ha comentado en el capítulo anterior, se parte de un fichero en el que se incluyen los datos de las transacciones de venta de viviendas realizadas en el área de Aragón desde Enero de 2010 hasta Mayo de 2014. Los datos que se tienen de cada transacción son un identificador de la transacción, el valor por el que se ha vendido la vivienda, su superficie medida en m^2, su antigüedad medida en años, el municipio en el que se encuentra situada y un índice de calidad que toma valores enteros entre 1 y 10 y ha sido otorgado para cada vivienda en función de sus características: tipología de la vivienda, tipo de calle, clase de inmueble, etc.

En primer lugar, se va a realizar un análisis global de los datos por variable. Posteriormente, se estudiará la relación de las variables aportadas por el conjunto de datos entre sí y se hará un análisis municipio a municipio. Para finalizar el capítulo se analizarán los datos en relación a las restricciones impuestas en el modelo, número mínimo de transacciones y relaciones de vecindad.

2.1. Análisis global

En la Figura 2.1 se puede observar que hay una gran concentración de las ventas en un solo municipio, Zaragoza, en el que se producen 516 transacciones de las 927 totales. Esto se debe a que Zaragoza capital acumula aproximadamente la mitad de la población del área total, Aragón. También cabe destacar que hay municipios en los que hay muy pocas

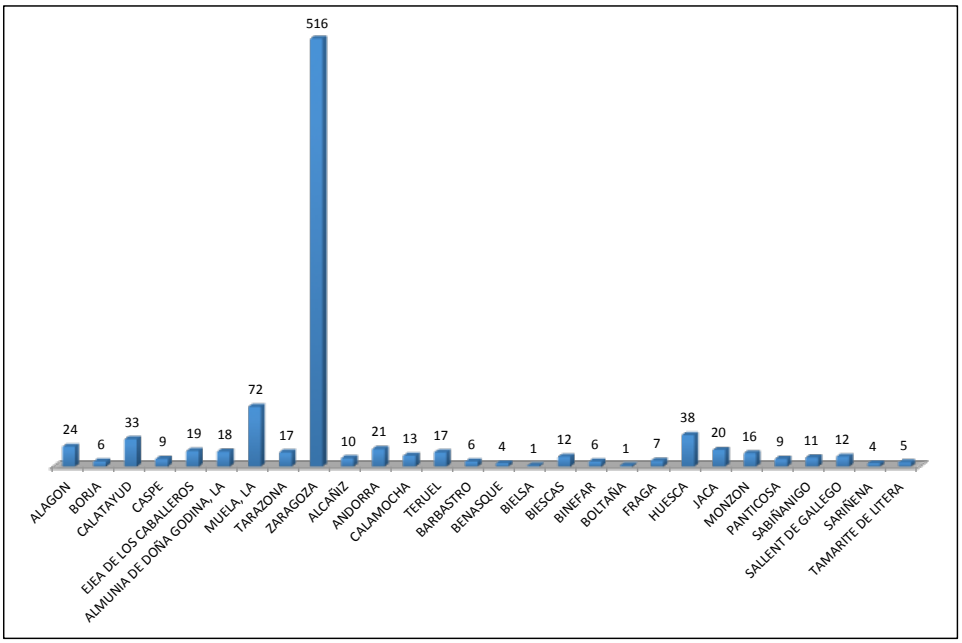

Figura 2.1: Frecuencia de transacciones según municipio

ventas (menos de 5), como Benasque (4), Bielsa (1), Boltaña (1) y Sariñena (4). Observando las vecindades inmediatas entre municipios (ver Anexo I), se evidencia que tanto Benasque como Sariñena podrían agruparse con diferentes municipios haciendo que el número de transacciones del grupo se elevará y se pudiera tratar de lograr la factibilidad de vecindades y de número de transacciones. Sin embargo, Bielsa y Boltaña solo podrían agruparse entre ellos, formando un grupo que contaría con 2 transacciones. Si se unen a otros municipios incrementarán la infactibilidad de vecindades y aportarán muy poco, 2 transacciones, al número de transacciones requeridas, es por esto que ambos municipios, con el consentimiento de los expertos, son eliminados del estudio.

Sobre el precio de venta de las viviendas, el valor mínimo ha sido 13.500€, mientras que el mayor valor ha sido 841.000€. Todos los precios de las viviendas de la muestra son valores factibles de acuerdo a las indicaciones sugeridas por los expertos ya que se encuentran entre $300€/m^2$ y $18.000€/m^2$. En cuanto a su distribución, la variable "Valor de la Vivienda" es asimétrica positiva, con un valor medio igual a 236.273€ y con la mediana situada en 214.000€, tal y como puede apreciarse en la Figura 2.2.

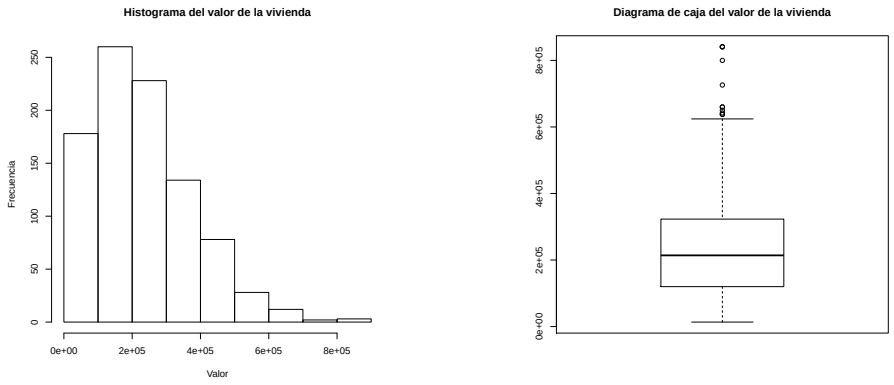

Figura 2.2: Histograma y diagrama de caja de la variable "Valor de la Vivienda".

En cuanto a la variable "Superficie de la vivienda", si se observa su distribución en el histograma y diagrama de caja de la Figura 2.3 se evidencia que la superficie del 50 % de las viviendas se sitúa entre $108m^2$ y $217m^2$. Sin embargo, podemos encontrar viviendas mucho más pequeñas, de $32.46m^2$ y también mucho más grandes, de $616.92m^2$. Esta última se trata de un caso atípico que se separa excesivamente del resto de viviendas, tal y como se puede apreciar en el diagrama de caja de la Figura 2.3 y se eliminará del estudio.

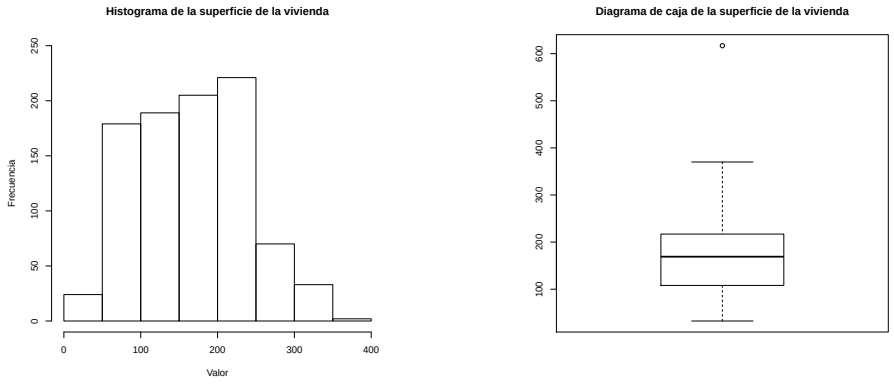

Figura 2.3: Histograma y diagrama de caja de la variable "Superficie de la Vivienda".

En los gráficos de la Figura 2.4 se observa que la variable "Antigüedad

de la vivienda" es muy asimétrica. El 25 % de las viviendas vendidas tienen menos de 4.3 años, el 50 % menos de 10 años, el 75 % menos de 34.4 años y en el 25 % restante se encuentran valores que alcanzan hasta casi los 200 años. En particular aparece una vivienda de 190.5 años que se ha considerado como un caso atípico y se ha eliminado del estudio. El resto de valores indicados como atípicos por el software se mantienen en el estudio.

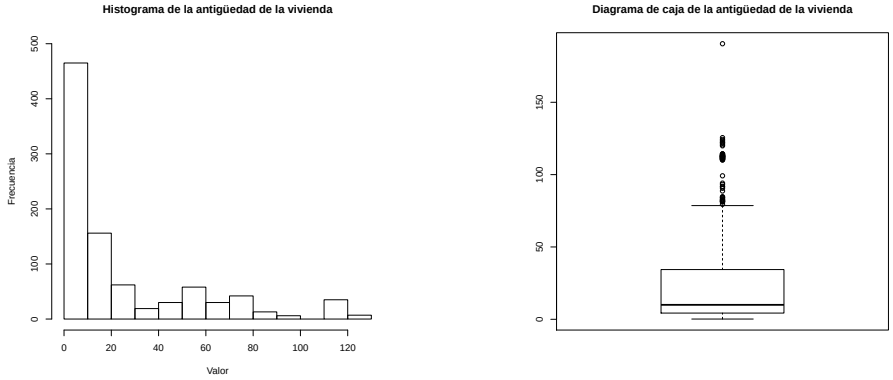

Figura 2.4: Histograma y diagrama de caja de la variable "Antigüedad de la Vivienda".

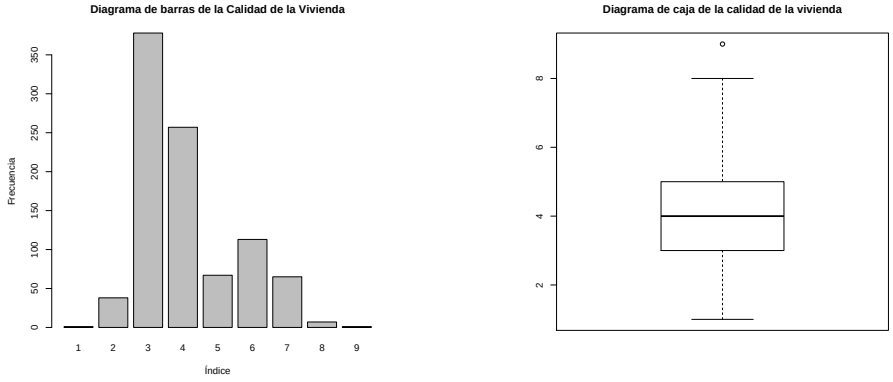

Figura 2.5: Diagrama de barras y diagrama de caja de la variable "Índice de calidad".

Finalmente, en relación a la variable "Índice de calidad", que toma valores de 1 a 10, en donde el 1 corresponde a la mejor calidad y 10 a la

peor, se puede observar que 378 viviendas tienen un valor de la variable "índice de calidad" igual a 3 y 256 un índice de calidad 4, es decir, el 68.5 % de las viviendas tienen un índice de calidad bueno, quedando un 4.2 % de las viviendas con un índice muy bueno o excelente (valor 1 o 2), véase la Figura 2.5.

2.2. Relación entre variables

En la Tabla 2.1 se presentan las correlaciones entre variables y posteriormente se estudiarán en mayor profundidad aquellas variables cuyo coeficiente de correlación sea mayor que 0.6.

	Calidad	Valor	Superficie	Antigüedad
Calidad	1.00000	-0.61419	-0.50383	0.79621
Valor	-0.61419	1.00000	0.66754	-0.47330
Superficie	-0.50383	0.66754	1.00000	-0.40390
Antigüedad	0.79621	-0.47330	-0.40390	1.00000

Tabla 2.1: Tabla de correlaciones entre variables.

- **Valor de la vivienda-Calidad de la vivienda**
  Se observa en la Figura 2.6 que los valores altos del índice de calidad, es decir, de peor calidad, corresponden con viviendas de precio bajo.

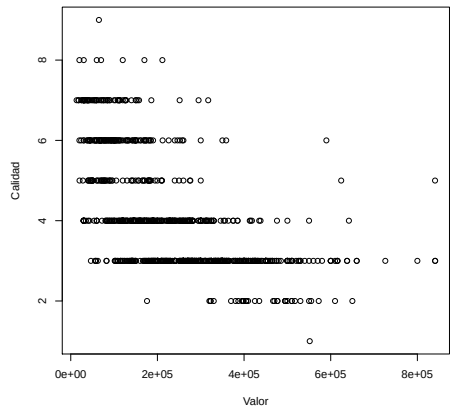

Figura 2.6: Valor de la vivienda frente a Calidad de la vivienda.

- **Antigüedad de la vivienda-Calidad de la vivienda**
  Examinando la Figura 2.7 se observa como el valor del índice de calidad aumenta, o lo que es lo mismo, empeora, con la antigüedad de las viviendas. Así, y como es lógico, las viviendas más nuevas tienden a presentar los mejores índices de calidad.

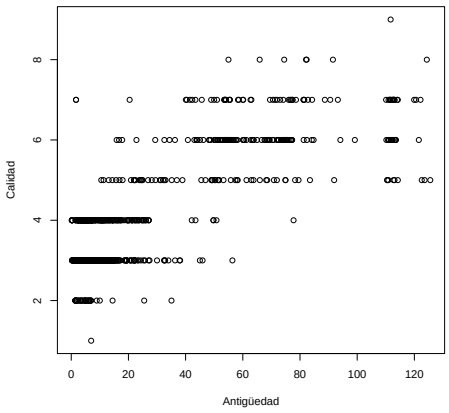

Figura 2.7: Antigüedad de la vivienda frente a Calidad de la vivienda.

- **Valor de la vivienda-Superficie de la vivienda**
  Como es de esperar el valor de la vivienda aparece positivamente correlado con la superficie de esta, es más, la Figura 2.8 podría sugerir una posible relación lineal entre ellas.

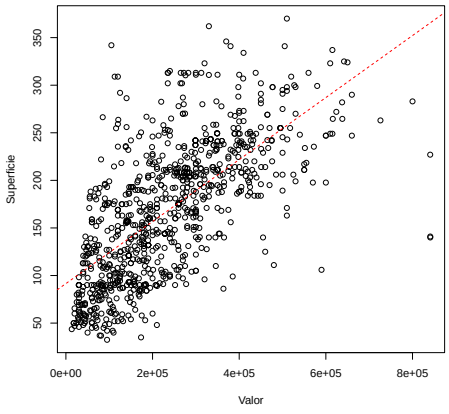

Figura 2.8: Valor de la vivienda frente a Superficie de la vivienda.

2.3. Análisis por municipios

A continuación se presentan los diagramas de caja para cada variable y cada municipio, con el objeto de poner de manifiesto las diferencias entre los diferentes municipios. Se debe recordar que se han eliminado los municipios Bielsa y Boltaña, correspondientes a los números 16 y 19 respectivamente, por escaso número de transacciones incluso al hacer la agrupación de ambos. Las Figuras 2.9, 2.10, 2.11 y 2.12 muestran, los diagramas para las variables "Valor de la vivienda", "Superficie de la vivienda", "Antigüedad de la vivienda" e "Índice de calidad de la vivienda", respectivamente.

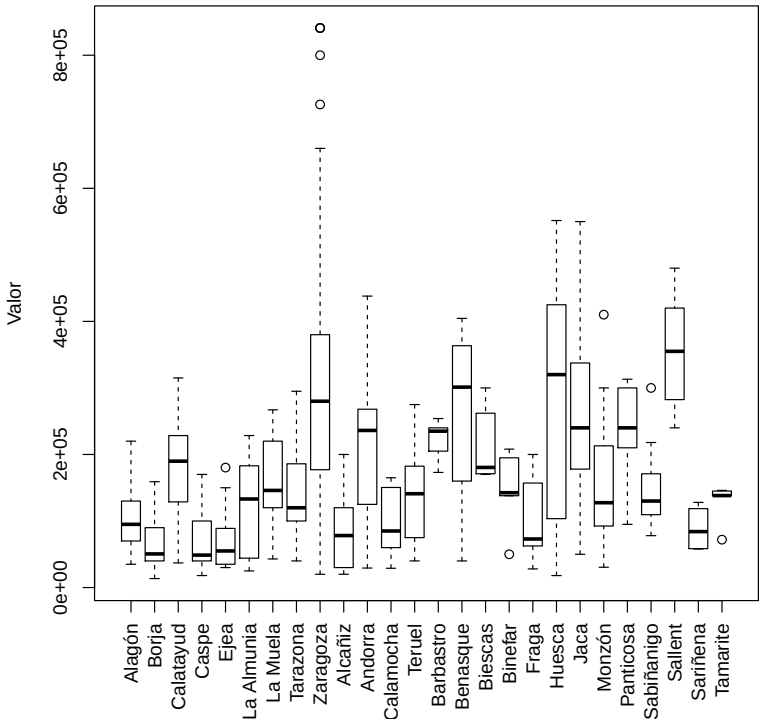

Figura 2.9: Diagrama de caja del "Valor de la vivienda" para cada municipio.

Se analiza ahora en mayor profundidad cada uno de los municipios:

- **Alagón**
  De las 24 transacciones que se realizan 17 son por un valor entre

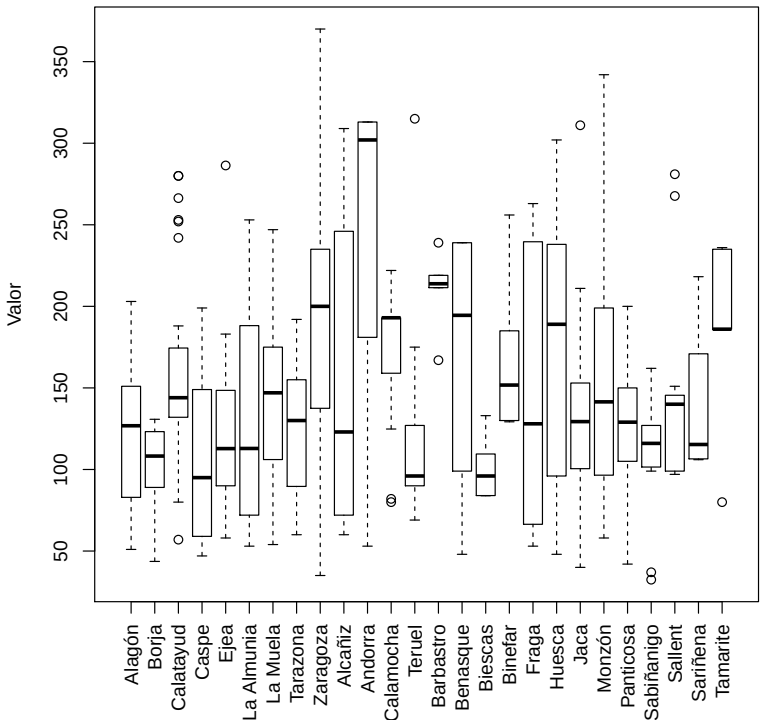

Figura 2.10: Diagrama de caja de la "Superficie de la vivienda" para cada municipio.

50.000€ y 150.000€. Además, en cuanto a la antigüedad de las viviendas, cabe destacar que 8 de las 24 viviendas tienen menos de 20 años, otras 7 tienen entre 100 y 120 años y el resto se sitúan en una edad intermedia (entre 20 y 100 años).

- **Borja**
  En este municipio se venden 6 viviendas. De ellas 5 han costado menos de 100.000€ y también 5 son mayores de $80m^2$. Las viviendas más nuevas que se han vendido tienen entre 20 y 40 años y el índice de calidad de las ventas se sitúa entre 4 y 7.

- **Calatayud**
  El número de transacciones que hay en este municipio es 32. De ellas, 20 viviendas se han vendido por un valor entre 1.150.000€ y 250.000€ y 24 de ellas tenían una superficie entre 100 y $200m^2$. Además 22 de las 32 totales tienen menos de 20 años de antigüedad

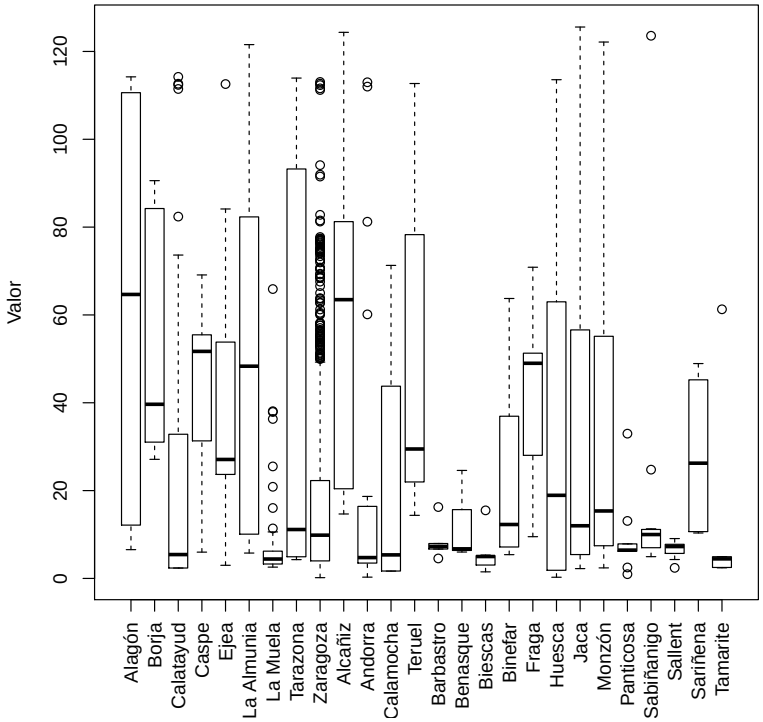

Figura 2.11: Diagrama de caja de la "Antigüedad de la vivienda" para cada municipio.

y 26 tienen un índice de calidad menor o igual que 4.

- **Caspe**
  5 de las 9 ventas que se realizan son por un importe inferior a 50000€ y además, también 5 viviendas tienen entre 50 y 70 años de antigüedad.

- **Ejea de los Caballeros**
  En este municipio se han realizado 19 transacciones. De ellas, 14 son por un importe menor que 100.000€, 14 tienen una superficie entre 50 y 150m^2 y también 14 tienen una antigüedad entre 20 y 60 años.

- **La Almunia de Doña Godina**
  De las 18 transacciones que se realizan, 9 son de viviendas cuya superficie oscila entre 50 y 100m^2. Además, cabe destacar que un

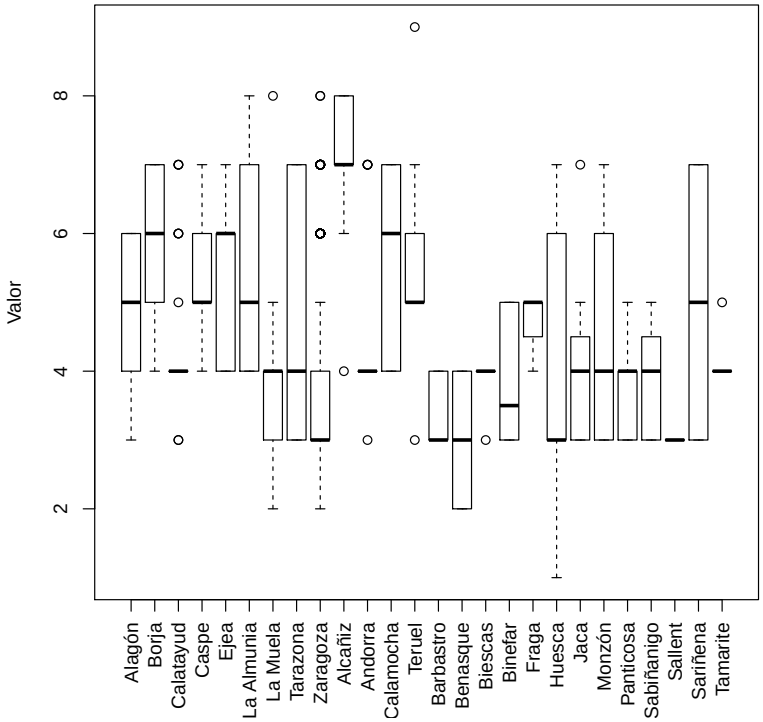

Figura 2.12: Diagrama de caja del "Índice de calidad de la vivienda" para cada municipio.

tercio de las viviendas vendidas tiene una antigüedad inferior a 20 años.

- **La Muela**
  En este municipio se han vendido 72 viviendas. Más de un tercio costaban entre 100.000€ y 150.000€. Además, 31 viviendas tienen una superficie entre 150 y 200m^2. Sin embargo, lo mas destacable es que 61 viviendas tienen menos de 10 años de antigüedad y por tanto también los índices de calidad son más bajos que en otros municipios.

- **Tarazona**
  El número de ventas que se da en este municipio es 17. De ellas, 6 tienen una superficie entre 80 y 100m^2 y 10 tienen menos de 20 años de antigüedad.

- **Zaragoza**
  En este municipio, como ya se ha comentado anteriormente, se acumulan más de la mitad de las transacciones totales, concretamente 516 de 925. De estas 516, la mitad de ellas se han vendido por un valor entre 177.650€ y 380.000€ y 190 tienen una superficie entre 200 y 250m^2. Además, la mitad de las viviendas vendidas tienen menos de 10 años. Este hecho hace que muchos valores aparezcan como outliers en la Figura 2.11.

- **Alcañiz**
  De las 10 transacciones que se realizan 5 son de viviendas con una superficie entre 50 y 100m^2. Además, 8 de las viviendas tienen un índice igual o superior a 7.

- **Andorra**
  En este municipio se han vendido 21 viviendas. De ellas, 12 han costado entre 200.000€ y 300.000€, 11 tienen una superficie entre 300 y 350m^2 y 17 tienen una antigüedad inferior a 20 años. Los índices de calidad, por tanto, son bajos: 17 viviendas tienen un índice de calidad igual o inferior a 4.

- **Calamocha**
  El número de ventas que se da en este municipio es 13. Cabe destacar que 9 de estas viviendas tienen una superficie entre 150 y 200m^2 y 7 tienen menos de 10 años de antigüedad.

- **Teruel**
  En este municipio se han realizado 17 transacciones. 15 de ellas tienen una superficie entre 50 y 150m^2 y 9 tiene una antigüedad igual o inferior a 40 años.

- **Basbastro**
  La superficie de las 5 viviendas que se vendieron en este municipio oscila entre 160 y 240m^2, pero más concretamente 3 de ellas tenian una superficie entre 200 y 220m^2. En cuanto al valor de la vivienda 3 de las 5 viviendas vendidas costaban entre 200.000€ y 250.000€. Además todas las viviendas tienen menos de 50 años y un índice de calidad entre 3 y 4.

- **Benasque**
  En este municipio ha habido tan solo 4 ventas y todas ellas presentan diferentes características. Cabe destacar que 3 de las 4 viviendas

tienen entre 5 y 10 años de antigüedad, la cuarta tiene 24 años y medio aproximadamente.

- **Biescas**
  En este municipio se han realizado 12 ventas, y la mitad de ellas han sido por un valor entre 160.000€ y 180.000€. Es destacable que 7 de las 12 viviendas tenían menos 5 de años de antigüedad, otras 4 no superaban los 10 años y la última tenía 15 años y medio aproximadamente.

- **Binéfar**
  De las 6 transacciones que se han producido en este municipio 3 de ellas han sido por un valor entre 100.000€ y 150.000€. Además, 5 de las viviendas tenían una superficie entre 100 y $200m^2$ y 4 una antigüedad menor a 20 años.

- **Fraga**
  En este municipio ha habido 7 transacciones de diferentes características. Cabe destacar que 4 de los 7 municipios tienen entre 40 y 60 años de antigüedad.

- **Huesca**
  El número de ventas que se da en este municipio es de 38. Estas ventas son por diferentes valores y de distintas superficies. Sin embargo, se evidencia la gran cantidad de viviendas nuevas ya que 20 viviendas tienen menos de 20 años, y la calidad de las viviendas porque 21 tienen un índice de calidad igual o menor que 3.

- **Jaca**
  En este municipio se han producido 20 transacciones, 13 de las cuales han sido por un valor entre 100.000€ y 300.000€. Además, la mitad de dichas transacciones eran de viviendas con una superficie entre 100 y $150m^2$ y más de la mitad, 11, eran viviendas con una antigüedad inferior a 20 años.

- **Monzón**
  Hay viviendas de diferentes precios y de distintas superficies, sin embargo, es destacable que 9 de las 16 viviendas vendidas tienen menos de 20 años.

- **Panticosa**
  En este municipio ha habido 9 ventas. De ellas, 5 eran viviendas con una superficie entre 100 y $150m^2$ y además, 2 viviendas tenían

menos de 5 años de antigüedad y también 5 viviendas tenían entre 5 y 10 años de antigüedad, es decir, se observa que las viviendas son bastante nuevas.

- **Sabiñanigo**
  De las 11 transacciones que se han realizado en este municipio 7 de ellas han sido por un valor entre 50.000€ y 150.000€. En cuanto a la superficie, cabe destacas que dos de las viviendas vendidas tenían una superficie inferior a $50m^2$ mientras que más de la mitad de ellas se sitúan entre los 100 y $150m^2$. Respecto a la antigüedad, 9 de las viviendas tenían menos de 20 años, una entre 20 y 40 y la restante 123.578 años.

- **Sallent de Gállego**
  En este municipio se han producido 12 transacciones. La superficie de 9 de las viviendas oscila entre 100 y $150m^2$ mientras que los valores son muy diferentes. Cabe destacar que la antigüedad mínima que se encuentra es 2 y la máxima 12 habiendo 6 viviendas con 7 años de antigüedad. Todas las viviendas tienen índice de calidad 3.

- **Sariñena**
  En este municipio ha habido tan solo 4 ventas y todas ellas presentan diferentes características. Cabe destacar que 3 de las 4 viviendas tienen tienen una superficie entre 100 y $150m^2$ y que 2 de las viviendas han costado entre 40.000€ y 60.000€ mientras que las otras dos han costado entre 100.000€ y 140.000€.

- **Tamarite de la Litera**
  En este municipio se han realizado tan solo 5 transacciones, de las cuales 4 han sido por un valor entre 120.000€ y 160.000€. Además, 4 viviendas tienen una superficie entre 150 y $250m^2$ y también 4 viviendas tienen una antigüedad inferior a 20 años.

2.4. Estructura de los datos en cuanto al número de transacciones y vecindad

Tras el análisis descriptivo previo en el que se han eliminado dos municipios (Bielsa y Boltaña) y dos transacciones (una de Calatayud y otra de Barbastro) se realiza un pequeño análisis en relación con las dos restricciones consideradas en el modelo, el número mínimo de transacciones

y la vecindad.

Por un lado, debido a que la evaluación se realizará mediante una colección de regresiones lineales Bayesianas, es necesario que en cada una de ellas se tengan al menos 40 transacciones (al menos 10 por cada parámetro a estimar). Un simple cálculo directo nos proporciona una primera estimación del número máximo de grupos que se podrían obtener resultando ser 23 ya que disponemos de datos de 923 transacciones y queremos unirlas en grupos de como mínimo 40 transacciones. Podemos afinar un poco más este cálculo ya que, como se ha comentado con anterioridad, hay un municipio (Zaragoza) que presenta un número muy elevado de transacciones (516). Por tanto, si tenemos en cuenta que la asignación a los grupos debe realizarse a nivel de municipio con todas sus transacciones e incluimos Zaragoza en un grupo propio, el resto, 407 transacciones podrían generar en el mejor de los casos otros 10 grupos más, acotando de esta manera el número máximo de grupos a 11. Asumiendo que, como se acaba de comentar, los municipios deben asignarse completos a los grupos, este número tal vez tampoco sea alcanzable. Para determinar el número máximo de grupos se ha planteado el problema de programación lineal entera binaria que se describe a continuación:

$$
\begin{aligned}
&\text{máx } Z = \textstyle\sum_{i=1}^{11} Y_i \\
&\text{sujeto a:} \\
&\qquad \begin{aligned}
&\textstyle\sum_{j=1}^{26} Trs_j \cdot X_{ij} \geq 40 \cdot Y_i && i = 1, \ldots, 11 \\
&\textstyle\sum_{j=1}^{26} X_{ij} \leq 26 \cdot Y_i && i = 1, \ldots, 11 \\
&\textstyle\sum_{i=1}^{11} X_{ij} = 1 && j = 1, \ldots, 26 \\
&X_{ij}, Y_i \in \{0, 1\}, \forall i, j
\end{aligned}
\end{aligned}
\tag{2.1}
$$

donde Trs_j corresponde al número de transacciones del municipio j-ésimo. Las variables X_{ij} toman valor 1 cuando el municipio j se asigna al grupo i y 0 en caso contrario y de igual modo, las variables Y_i toman valor 1 cuando algún municipio se ha asignado al grupo i y 0 en caso contrario. El primer grupo de restricciones obliga a que si se asignan municipios al grupo i-ésimo el número total de transacciones del grupo sea al menos 40. El segundo grupo de restricciones fuerza a que si se asignan municipios a un grupo i, la variable correspondiente Y_i tome valor 1. Por último, el tercer bloque de restricciones garantiza que cada municipio sólo pueda asignarse a un grupo.

La solución óptima[1] del problema anterior proporciona un número máximo de grupos igual a 10. En la Tabla 2.2 se muestra una de las posibles soluciones de 10 grupos.

Grupo 1	LA MUELA (72) Total transacciones 72
Grupo 2	ZARAGOZA (516) Total transacciones 516
Grupo 3	CASPE (9) CALAMOCHA (13) JACA (20) Total transacciones 42
Grupo 4	BORJA (6) ALCAÑIZ (10) PANTICOSA (9) SABIÑANIGO (11) TAMARITE DE LITERA (5) Total transacciones 41
Grupo 5	ALAGON (24) MONZON (16) Total transacciones 40
Grupo 6	ANDORRA (21) FRAGA (7) SALLENT DE GALLEGO (12) Total transacciones 40
Grupo 7	EJEA DE LOS CABALLEROS (19) TERUEL (17) BARBASTRO (5) Total transacciones 41
Grupo 8	CALATAYUD (32) BIESCAS (12) Total transacciones 44
Grupo 9	BENASQUE (4) HUESCA (38) SARIÑENA (4) Total transacciones 46
Grupo 10	LA ALMUNIA (18) TARAZONA (17) BINEFAR (6) Total transacciones 41

Tabla 2.2: Número máximo de grupos verificando el número mínimo de transacciones.

Por otro lado, en relación a la estructura de vecindades entre municipios, la matriz que se muestra en la Tabla 6.1 del Anexo I contiene los vecinos inmediatos de los municipios. Se puede observar como Alagón es vecino inmediato del municipio 2, Borja, y este por otro lado tiene como vecinos inmediatos a Alagón y además a los municipios 5 y 8, Ejea de los Caballeros y Tarazona. Con esta relación de vecindad inmediata se ha planteado una relación de vecindad más general en la que dos municipios de un cierto grupo serán vecinos si se pueden unir por una secuencia de vecinos inmediatos que pertenecen al grupo. Esta es precisamente la idea

[1]El problema se ha resuelto mediante el software CPLEX [10]

de existencia de una cadena a la que se hace referencia en el Capítulo 1. Por ejemplo, si en el grupo están Alagón, Borja y Ejea de los Caballeros, entonces Alagón y Ejea de los Caballeros serán municipios vecinos, ya que Alagón y Borja lo son y Borja y Ejea también.

A la hora de diseñar el modelo de regresiones se tendrá en cuenta en menor o mayor medida la relación de vecindad pero en ningún momento se planteará su cumplimiento de forma estricta. Sin embargo, se obligará a que se cumpla el número mínimo de transacciones en un grupo ya que en otro caso no se podría realizar la estimación de los parámetros adecuadamente.

Para tener una idea de cómo afecta la estructura de vecindades a la creación de los grupos (independientemente del número de transacciones) se plantea la determinación del número mínimo de grupos que se pueden construir de forma que todos los municipios de cada grupo sean vecinos.

Para ello recurrimos a la teoría de grafos y construimos un grafo cuyos vértices corresponden a los nodos y cuyos arcos corresponden a las relaciones de vecindad inmediata, es decir $G = (N, A)$ donde $N = \{M_1, M_2, \ldots, M_{26}\}$ corresponde a los 26 municipios y $A = \{(i, j)|M_i \text{ y } M_j \text{ son vecinos inmediatos}\}$. A dicho grafo se le aplica un algoritmo para la determinación del número de componentes conexas, en particular, se le aplica una versión básica del algoritmo de Prim [12], asumiendo costos 1 para todos los arcos y partiendo de un nodo para obtener una componente conexa. Si se han incluido todos los nodos en la componente ya estaría finalizado. En caso contrario, se toman los nodos no incluidos y se repite el proceso lo que nos proporciona la segunda componente, y así se continua de manera sucesiva hasta incluir todos los nodos.

El resultado se muestra en la Tabla 2.3. El número mínimo de grupos verificando las vecindades es 6 y además la solución verifica los requerimientos del número mínimo de transacciones. Los 6 grupos podría ampliarse hasta 8 grupos verificando vecindades e incluso número de transacciones sin dificultad. Los grupos 4 y 5 podrían dividirse de forma que los subgrupos resultantes fueran factibles. Se podría hacer un grupo 4_1 con La Muela y otro 4_2 con Zaragoza. El Grupo 5, por su parte, podría dividirse, por ejemplo, en un grupo 5_1 conteniendo Huesca y Sariñena y sumando 42 transacciones y otro grupo 5_2 que contuviera

Grupo 1	ALAGON (24) BORJA (6) TARAZONA (17) EJEA DE LOS CABALLEROS (19) Total transacciones 66
Grupo 2	CALATAYUD (32) LA ALMUNIA (18) Total transacciones 50
Grupo 3	CASPE (9) ALCAÑIZ (10) ANDORRA (21) CALAMOCHA (13) TERUEL (17) Total transacciones 70
Grupo 4	LA MUELA (72) ZARAGOZA (516) Total transacciones 588
Grupo 5	BARBASTRO (5) BENASQUE (4) BINEFAR (6) FRAGA (7) HUESCA (38) MONZON (16) SARIÑENA (4) TAMARITE DE LITERA (5) Total transacciones 85
Grupo 6	BIESCAS (12) JACA (20) PANTICOSA (9) SABIÑANIGO (11) SALLENT DE GALLEGO (12) Total transacciones 64

Tabla 2.3: Número mínimo de grupos verificando las vecindades entre municipios.

el resto de municipios: Barbastro, Benasque, Binefar, Fraga, Monzón y Tamarite de la Litera los cuales sumarían 43 transacciones. Mas allá de estos 8 grupos y por debajo de 6 grupos la estructura de vecindades se incumplirá en algún grupo.

Capítulo 3

Algoritmo Genético Grupo a Grupo

Este capítulo se dedica a presentar el algoritmo genético diseñado para determinar la división de los municipios en un número de grupos, Q, fijado previamente. El algoritmo tratará de determinar la mejor división atendiendo a dos criterios que se ponderaran adecuadamente. Por un lado, procurará maximizar la densidad marginal asociada a las Q regresiones y por otro lado, intentará minimizar el número de infactibilidades de acuerdo a la relación de vecindad (restricción 'soft'). Para reducir este problema bi-objetivo a un contexto uni-objetivo, se planteará una función objetivo que ponderará el logro de cada uno de los objetivos. El capítulo constará de las siguientes secciones: En primer lugar se mostrará la representación utilizada para las soluciones junto con la estructura de las poblaciones. A continuación se presenta como se realiza la evaluación de las soluciones. En una tercera sección se desarrolla el esquema general del algoritmo que posteriormente es detallado en las secciones cuarta a sexta, con los operadores de selección, recombinación y mutación, respectivamente.

3.1. Individuos, representación de cromosomas y población

La representación de los individuos es el primer problema a abordar para la utilización de los algoritmos evolutivos, y es también de suma importancia porque de ella dependerán los resultados que se obtengan.

En este caso en particular, se ha decidido utilizar una representación vectorial donde las primeras 26 posiciones son utilizadas para indicar el número de grupo al que pertenece el municipio cuyo número es la posición en el vector. Es decir, si en la quinta posición del vector hay un 1, el municipio 5 pertenece al primer grupo.

Como se vio en el capítulo anterior, el número máximo de grupos atendiendo a la restricción sobre el número de transacciones es 10 y entre 6 y 8 grupos teniendo en cuenta las vecindades. Por tanto, si sólo interesasen las vecindades y el número de transacciones se deberían buscar soluciones entre 6 y 8 grupos. Sin embargo, la densidad marginal, en general, tiende a aumentar conforme disminuye el número de grupos ya que penaliza la falta de parsimonia del modelo, por lo que dependiendo de los intereses del experto podría resultar interesante buscar soluciones que aún no siendo factibles para la relación de vecindad presentasen un buenos valores para la densidad marginal. Por este motivo se amplía la búsqueda a valores inferiores a 6 para el número de grupos y se fija, por tanto, un rango de variación para este entre 2 y 8, incluyendo ambos valores. Teniendo en cuenta este hecho, las siguientes 8 posiciones del vector están reservadas para indicar el número de transacciones que hay en cada grupo. Se reserva espacio para un máximo de 8 grupos, en el caso de que hubiera menos grupos no todas las posiciones serían utilizadas.

Por último, la representación escogida contará con una componente más, la número 35, en la que se indicará el número de grupos en los que se dividen los municipios.

La población estará formada por una colección de cromosomas con la estructura que se comenta en los párrafos anteriores y su generación se muestra en el proceso del Algoritmo 3.1. Esta generación de la población se realiza de manera aleatoria para un número de grupos dado. En primer lugar, se realiza un sorteo para cada asignar cada municipio a un grupo, tras lo cual se aplica el Algoritmo 3.2 que se encarga de modificar, si es necesario, el cromosoma actual para que verifique el requerimiento del número de transacciones mínimas por grupo. A continuación el Algoritmo 3.3 se encarga de reordenar la numeración de los grupos con el objeto de identificar soluciones equivalentes pero con ordenaciones diferentes de los grupos. Por último, se evalúa la función objetivo que permite el establecimiento del *fitness* de la solución.

Algoritmo 3.1: Creación de la población

Data: Tamaño de la población, número de grupos a considerar.
Result: Una población del tamaño y grupos indicados y su correspondiente fitness.

1 Se crea el vector de fitness;
2 Se crea la matriz para almacenar la población;
3 **for** *i=1* **to** *tamaño de la población* **do**
4 Generar un vector de números aleatorios entre 1 y número de grupos de longitud 26 (con reemplazamiento) en el que aparezcan todos los grupos;
5 Guardar la asignación de municipios a grupos en componentes 1 a 26;
6 Guardar número de transacciones en componentes 27 a 27+número de grupos;
7 Guardar número de grupos en componente 35;
8 Aplicar algoritmo *recupera* (Algoritmo 3.2);
9 Reordenar grupos, algoritmo *copia_ordenada* (Algoritmo 3.3);
10 Evaluar el fitness del cromosoma;
11 **end**

En la creación de la población inicial y en diversos puntos del algoritmo genético será necesario modificar una solución para que cumpla con el requerimiento del número mínimo de transacciones. Esta labor se realiza mediante el Algoritmo 3.2. Este, dado un cromosoma, va tomando aleatoriamente los grupos definidos por este que no tienen suficientes transacciones y trata de arreglarlos incorporando municipios de otros grupos a los que les sobran transacciones. Si esto no es posible y han pasado varios intentos (10), el algoritmo permite arreglar el grupo con algún municipio de otro grupo aunque ello suponga estropear la factibilidad de otro grupo (el algoritmo considera posteriormente dicho grupo para ser arreglado).

Por último, se comenta también en esta sección el Algoritmo 3.3, encargado de seleccionar entre todas las posibles formas de representar una misma asignación de municipios a grupos tan solo una de ellas. Por ejemplo, podría suceder que al dividir los municipios en dos grupos una solución asignara los primeros 13 municipios al grupo 1 y los otros 13 al 2, y que otra solución hiciera exactamente lo contrario, o sea, los 13

Algoritmo 3.2: Recupera

Data: Cromosoma.
Result: Cromosoma reconstruido.

```
1  aux ← Grupos con menos de 40 transacciones;
2  Se desordena aleatoriamente aux;
3  contador ← 0;
4  while aux ≠ ∅ do
5      i ← aux(1);
6      while |Grupo i| < 40 do
7          contador ← contador+1;
8          m ← municipio aleatorio de un grupo j ≠ i;
9          if |Grupo j| − transacciones(m) ≥ 40 then
10             contador ← 0;
11             Cambiar municipio m de grupo j a grupo i;
12         end
13         else
14             if contador>10 then
15                 Cambiar municipio m de grupo j a grupo i;
16             end
17         end
18     end
19     aux ← Grupos con menos de 40 transacciones;
20     Se desordena aleatoriamente aux;
21     contador ← 0;
22 end
```

primeros al grupo 2 y los restantes al grupo 1. Ambas soluciones son idénticas aunque las representaciones son diferentes. El algoritmo propuesto unifica las representaciones seleccionando la primera de ellas.

El funcionamiento del algoritmo consiste en asignar el grupo a los municipios por orden de primera aparición, es decir, al primer municipio se le asignará grupo 1 y todos los que sean del mismo grupo que el primer municipio pasan a ser también del grupo 1 mientras que el grupo que originalmente era el 1 pasa a ser del grupo que originalmente era el municipio 1. Se sigue examinando los municipios y cuando aparezca un municipio distinto del 1 a este se le asigna el grupo 2. A su vez, todos los municipios que pertenecen a dicho grupo pasan a formar parte del gru-

po 2 y los que originalmente estaban en el grupo 2 estarán en el grupo que se acaba de modificar. Este proceso se repite hasta llegar al último municipio.

Algoritmo 3.3: Copia ordenada

Data: Cromosoma
Result: Cromosoma ordenado

```
1  ultimo ← 1;
2  nGrupos ← Número de grupos en los que se divide;
3  conversion ← vector de tamaño nGrupos con 0 en todas sus
   componentes;
4  Municipio 1 a grupo 1;
5  conversion(grupo de municipio 1) ← 1;
6  Intercambia tamaños entre grupo 1 y grupo de municipio 1;
7  for j=2 to 26 do
8      if conversion(grupo de municipio j)>0 then
9          Municipio j a grupo indicado en conversion(grupo de
           municipio j);
10     else
11         ultimo ← ultimo + 1;
12         conversion(grupo de municipio j) ← ultimo;
13         Intercambia tamaños entre grupo de municipio j y grupo
           ultimo;
14         Municipio j a grupo ultimo;
15     end
16 end
```

3.2. Evaluación de las soluciones

El siguiente punto a tener en cuenta de cara al diseño del algoritmo es determinar cómo se evalúan las soluciones con objeto de establecer su *fitness*. El algoritmo maximizará el valor de la densidad marginal menos el número de infactibilidades multiplicado por un cierto peso positivo. De esta forma, asignando valores más o menos grandes a dicho peso se consigue ir desde un modelo en el que sólo se tiene en cuenta la densidad marginal a otro en el que se prime únicamente que los grupos construidos respeten la relación de vecindad, pasando por un conjunto infinito de posibilidades en los que se gradúa la importancia entre ambos objetivos.

El valor de la densidad marginal se obtiene tras realizar el ajuste de la regresión y para determinar el grado de infactibilidad de la solución se toman uno a uno los grupos en los que se dividen los municipios, a cada uno de ellos se aplica el Algoritmo 3.4 y se acumula el resultado devuelto por este. Es decir, el valor de la infactibilidad del cromosoma será la suma de las infactibilidad de cada uno de los trozos.

Algoritmo 3.4: Infactibilidad cadena

Data: Subcadena de un cromosoma con municipios de un grupo, nMunicipios en cadena
Result: Número de infatibilidades de la subcadena

```
1  componenteActual ←nAgrupados ←0;
2  componentes ←pendientes ← zeros(nMunicipios,1);
3  while nAgrupados<nMunicipios do
4  |  componenteActual ← componenteActual+1;
5  |  i ←1;
6  |  while componentes(i)≠0 do i=i+1 pendientes(1) ← i;
7  |  componentes(i) ← componenteActual;
8  |  nPendientes ← 1;
9  |  while nPendientes>0 do
10 |  |  municipioActual ← pendientes(nPendientes);
11 |  |  nPendientes ← nPendientes-1;
12 |  |  nAgrupados ← nAgrupados+1;
13 |  |  for i=1 to nCiudades do
14 |  |  |  if i≠municipioActual ∧ municipioActual es vecino de i
   |  |  |  ∧ componentes(i)=0 then
15 |  |  |  |  componentes(i) ←componenteActual;
16 |  |  |  |  nPendientes ← nPendientes+1;
17 |  |  |  |  pendientes(nPendientes) ←i;
18 |  |  |  end
19 |  |  end
20 |  end
21 end
22 número de infactibilidades ← componenteActual-1;
```

El Algoritmo 3.4 aplica el método de Prim [12] para determinar el número mínimo de componentes conexas que se pueden formar con los municipios del grupo atendiendo a la relación de vecindad y devuelve

dicho número menos 1, es decir, si todos los municipios son vecinos devuelve 0.

Finalmente los dos valores obtenidos se agregan en la forma:

Densidad Marginal − Penalización × Número de infactibilidades.

Un valor de la penalización igual a cero conduce a una optimización en la que sólo interesa el valor de la densidad marginal y un valor elevado de esta conduce a una búsqueda en la que el objetivo principal es cumplir las vecindades.

3.3. Estructura del Algoritmo

En esta sección se presenta la estructura general del algoritmo propuesto y cuyos elementos particulares se describirán detalladamente en la secciones siguientes.

En el Algoritmo 3.5 tenemos los distintos elementos que lo componen. En primer lugar se genera la población inicial con ayuda del Algoritmo 3.1. Posteriormente y mientras no se supere un número máximo de iteraciones fijado, $numMaxIteraciones$, ni en la mitad de ellas haya habido mejora alguna, se procede a realizar una iteración del algoritmo.

En cada iteración se realiza la selección de padres (selección proporcional al *fitness*). Si no se ha superado el primer 90 % de las iteraciones máximas fijadas estos padres se recombinan mediante el uso del proceso mostrado en la Sección 3.5. Tras recombinar las soluciones, a los elementos obtenidos se les aplica el Algoritmo 3.2, encargado de recuperar la factibilidad en cuanto al número mínimo de transacciones por grupo. Después de esto, las soluciones obtenidas son mutadas con el Algoritmo 3.8. En el caso de que ya se haya superado el 90 % de las iteraciones máximas, solo se realiza el proceso de mutación sobre los padres y además con una probabilidad de mutación igual a la mitad de la fijada en el caso general. Tras ambos casos, para asegurarnos de que se cumple la restricción sobre el mínimo número de transacciones, se aplica nuevamente el Algoritmo 3.2.

Posteriormente, después de la evolución de la población calculamos el *fitness* de los nuevos individuos y aplicamos el Algoritmo 3.3 para garantizar la unicidad de las representaciones, es decir, para evitar que existan

individuos iguales con representaciones distintas.

Algoritmo 3.5: Esquema del algoritmo genético

Data: Semilla, tamaño de la población, probabilidad de mutación, probabilidad de recombinación, número máximo de iteraciones, número de grupos, alfa y penalización.

```
1  Creación y evaluación de la población inicial (Algoritmo 3.1);
2  sinMejora ←numIteraciones ← 0;
3  mejorSolucion ←-999999999;
4  while numIteraciones<numMaxIteraciones ∧
   sinMejora<0.5∗numMaxIteraciones do
5  |  numIteraciones ← numIteraciones+1;
6  |  Selección de padres (Proporcional al fitness, Sección 3.4);
7  |  if numIteraciones<0.9∗numMaxIteraciones then
8  |  |  Recombinación (Sección 3.5);
9  |  |  Recupera soluciones (Algoritmo recupera);
10 |  |  Mutación (Algoritmo 3.8) ;
11 |  else
12 |  |  Mutación de los padres seleccionados (Algoritmo 3.8);
13 |  end
14 |  Recupera soluciones (Algoritmo recupera);
15 |  Calculo del fitness para soluciones evolucionadas;
16 |  Reordenación de grupos (Algoritmo 3.3);
17 |  Selección de supervivientes (Algoritmo 3.6);
18 |  if número de supervivientes<TamañoPoblacion then
19 |  |  Completa población (Algoritmo 3.1);
20 |  end
21 |  if fitness máximo>mejorSolucion then
22 |  |  sinMejora ← 0;
23 |  |  mejorSolucion ←fitness máximo;
24 |  else
25 |  |  sinMejora ← sinMejora+1;
26 |  end
27 end
```

Tras haber realizado la fase de variación, se ejecuta el proceso de selección de supervivientes, Algoritmo 3.6 y si no se han obtenido suficientes individuos se procederá a completar la población mediante la generación de individuos aleatorios utilizando nuevamente el Algoritmo 3.1. En último lugar, en la parte final de la iteración se comprueba si se ha mejorado

la mejor solución existente con objeto de aplicar correctamente el criterio de parada del algoritmo.

3.4. Operadores de selección

El algoritmo presenta dos operadores de selección: por un lado, el proceso de selección de padres para reproducción y por otro lado, el proceso de selección de supervivientes.

En el primer proceso de selección, la selección de padres, se seleccionan los individuos de la población para cruzarse y producir descendientes, que constituirán, una vez mutados, la siguiente generación de individuos que competirán con sus padres.

El proceso de selección utilizado para este fin es el denominado "proporcional al *fitness*", que es uno de los más utilizados en la literatura[1]. En él, cada individuo tiene una probabilidad de ser seleccionado como padre reproductivo proporcional al valor de su función *fitness*, de forma que los individuos con mejor *fitness*, tienen una mayor probabilidad de ser seleccionados.

Por ejemplo, si f_i representa el *fitness* del individuo i-ésimo, la probabilidad de selección, p_i, como padre se establecería de acuerdo a la Ecuación 3.1.

$$p_i = \frac{f_i}{\sum_{j=1}^{tamañoPoblacion} f_j} \tag{3.1}$$

El siguiente operador de selección es el que corresponde a la obtención de la siguiente población de individuos tras la finalización de una iteración del algoritmo y se muestra en el Algoritmo 3.6. Este une la población actual junto con los nuevos individuos evolucionados y elimina aquellas soluciones que estén repetidas. Tras ello, si el número de individuos es mayor que el tamaño de la población se seleccionan los $\alpha\,\%$ mejores individuos de forma elitista y el $(100\text{-}\alpha)\,\%$ restante se selecciona de forma aleatoria entre los individuos no seleccionados previamente. Si el número de soluciones distintas no alcanza el tamaño de población deseado se devuelven todos los individuos y posteriormente se completa

[1]Puede consultarse por ejemplo en [8, pp.34-35]

la población con individuos construidos de forma aleatoria (de la misma forma en la que se construye la población inicial) hasta que ésta tenga el tamaño establecido.

Algoritmo 3.6: Seleción Supervivientes

Data: Población y población evolucionada con sus correspondientes fitness, tamaño de la población y α
Result: Población seleccionada y su correspondiente fitness

```
1 Unir población y población evolucionada;
2 Unir fitness y fitness de los evolucionados;
3 Eliminar individuos repetidos;
4 if Número Individuos > tamaño de población then
5 |   Seleccionar las α % mejores soluciones;
6 |   Completar la población con individuos seleccionados
  |   aleatoriamente de los no incluidos previamente;
7 else
8 |   Seleccionar todos los individuos;
9 end
```

3.5. Recombinación

El uso de un operador de cruce o crossover es una de las principales razones de la eficiencia de los algoritmos genéticos ya que nos permite recombinar algunos esquemas y progresar rápidamente hacia las regiones óptimas del espacio de búsqueda. En particular, el crossover produce nuevos individuos combinando la información contenida en los cromosomas padre. Habitualmente cada pareja de cromosomas padre genera dos hijos. El crossover utilizado en el algoritmo propuesto es el denominado de punto de corte que consiste en seleccionar una posición aleatoria del cromosoma (punto de corte) y tomar la cabecera de uno de los padres (desde la primera posición hasta la de corte) y la cola (desde el punto siguiente al del corte hasta el final) del otro padre para construir con ello uno de los hijos, mientras que el otro hijo será construido tomando la cabecera y cola que no ha usado el primer hijo. Una vez creados los hijos se calcula el número de transacciones resultantes en cada uno de los grupos de cada uno de los hijos y se almacena.

Algoritmo 3.7: Similaridad

Data: Cromosoma a y Cromosoma b
Result: Cromosoma a con grupos renumerados

```
1  Se crea el vector clista de longitud numGrupos^2;
2  índices ← (b(k)-1)∗numGrupos+a(k), ∀k = 1, ..., 26;
3  for i=1 to nMunicipios do
4  |  clista(indices(i)) ← clista(indices(i))+1;
5  end
6  i ← índices de clista según el orden decreciente de sus valores;
7  columnas ←filas ← vector de longitud numGrupos con valor −1;
8  h ←j ← 1;
9  while h≤numGrupos do
10 |  pa ← 1+(resto de (i(j)−1)/numGrupos);
11 |  pb ← techo de i(j)/numGrupos;
12 |  if filas(pa)=−1 y columnas(pb)=−1 then
13 |  |  filas(pa) ← pb;
14 |  |  columnas(pb) ← 1;
15 |  |  h ← h+1;
16 |  end
17 |  j ← j+1;
18 end
19 for j=1 to numGrupos do
20 |  a(índices de los municipios del grupo j) ← filas(j);
21 |  a(26+j) ← a(26+filas(j));
22 end
```

Debido a la sencillez del operador no se incluye su esquema[2], sin embargo sí que se incluye el Algoritmo 3.7. Este algoritmo se aplica a cada par de padres justo antes de realizar el crossover y su intención es que los grupos más parecidos de cada padre tengan la misma numeración, de forma que al transmitir la información de los padres a los hijos por división del cromosoma la información sea lo más coherente posible. Así, la idea del algoritmo es comparar todos los grupos de un cromosoma con los del otro y mirar qué pareja de grupos es la que tiene mayor número de municipios en común. Se vuelve a numerar el grupo correspondiente del primer cromosoma para que sea igual que la del segundo. A continuación se repite el proceso con el resto de grupos no considerados hasta

[2]Puede consultarse por ejemplo en [8, pp.39-41]

que se hayan examinado todos los grupos de los cromosomas. Una vez realizadas las reasignaciones de grupos se procede a realizar el cruce de los individuos.

Tras crear los nuevos individuos recombinados y previo a la aplicación del operador de mutación, en el flujo general del algoritmo genético se procederá a aplicar el Algoritmo 3.2 para "arreglar" aquellos cromosomas que presenten grupos con menos transacciones de las necesarias.

3.6. Mutación

El operador de mutación se aplica a cada hijo de manera individual y consiste en la alteración aleatoria (normalmente con probabilidad pequeña) de cada gen del cromosoma. Si bien en principio puede pensarse que el operador de cruce es más importante que el operador de mutación ya que proporciona una exploración rápida del espacio de búsqueda, éste ultimo asegura que ningún punto del espacio de búsqueda tenga probabilidad cero de ser examinado, y es de capital importancia para asegurar la convergencia de los algoritmos genéticos.

Algoritmo 3.8: Mutación

Data: Población y *pm*, probabilidad de mutación
Result: Población mutada

```
1 for i=1 to tamaño de la población do
2 |   A ← Conjunto de municipios de individuo i seleccionados, con
  |   probabilidad pm, para mutar;
3 |   for j ∈ A do
4 |   |   Sortear nuevo grupo distinto para municipio j;
5 |   |   Asignar nuevo grupo a j y actualizar transacciones de
  |   |   grupo inicial y final;
6 |   end
7 end
```

En la mutación se consideran uno a uno los individuos obtenidos tras la recombinación. Para cada uno de ellos se recorren los municipios y se sortea, con una cierta probabilidad previamente fijada, si el municipio cambia de grupo o no. Si es así, se sortea un nuevo grupo y se actualizan los totales de transacciones de los grupos.

Al igual que en el caso del operador de recombinación o crossover, tras la mutación de los individuos en el flujo general del algoritmo genético se procederá a aplicar el Algoritmo 3.2 para garantizar el mínimo número de transacciones en cada uno de los grupos en los que se integren los municipios.

Capítulo 4

Desarrollo del experimento y resultados

El objetivo principal de este capítulo es mostrar el comportamiento del algoritmo desarrollado al ser aplicado bajo ciertos escenarios: distintas penalizaciones y número de grupos. En ningún momento se trata de determinar la mejor solución para el problema de valoración de la vivienda, ya que este cometido corresponde al grupo de expertos inmobiliarios a los que se les traslada toda la información obtenida. Ellos serán los que finalmente determinen como utilizar la herramienta fijando el escenario que consideren más adecuado en vista de los resultados que se les facilitan.

Los escenarios se definen en función del número de grupos y de la importancia de las relaciones de vecindad frente a la densidad marginal (o viceversa). Se consideran, como ya se comentó en capítulos anteriores, un número de grupos que varía desde 2 hasta 8 grupos, de uno en uno, y se fijan valores de penalización que van desde 0 hasta 5, en particular

$$\text{Penalización} = \{0, 0.1, 0.25, 0.5, 0.75, 1.0, 2.0, 5.0\}$$

El valor de 0 corresponde a no considerar el requerimiento de vecindad y el valor de 5 corresponde a primar la vecindad absolutamente sobre la calidad de la regresión. Cualquier valor mayor que 5 mostraría una estructura de resultados análoga a la de penalización 5. El resto de valores son distintas propuestas de compromiso entre ambos extremos.
Las penalizaciones se incorporan a la función de evaluación en la forma:

$$\text{Densidad Marginal} - \text{Penalización} \times \text{Número de infactibilidades.}$$

Los parámetros que se han fijado para la ejecución del algoritmo genético bajo cada uno de los 56 escenarios son los que se enumeran a continuación:

- ◇ Número de repeticiones. Se fija en 10 y cada una de ellas parte de una semilla diferente. Estas semillas han sido establecidas dentro del programa para que pueda ser replicado cuantas veces se quiera pero fueron generadas aleatoriamente inicialmente. El algoritmo se aplicará 10 veces a cada escenario y se seleccionará la repetición que haya proporcionado un mejor valor de la función objetivo.

- ◇ Tamaño de la población. Se ha establecido que el número de individuos que contendrá la población será 200. Es un número suficiente para garantizar la diversidad de la población y poder encontrar la solución al problema planteado.

- ◇ Número de iteraciones. 500 será el número máximo de iteraciones en las que el algoritmo deberá encontrar una solución. Sin embargo, se ha verificado que no alcanza este número de iteraciones en ninguna de las ejecuciones, es decir, encuentra la solución antes.

- ◇ Probabilidad de mutación. De manera habitual suele tomarse como probabilidad de mutación uno entre el número de genes del cromosoma. En nuestro caso, se tomaría $\frac{1}{26}$ debido a que tan solo las 26 primeras componentes del cromosoma juegan un papel a nivel diferenciación del individuo mientras las 9 restantes nos proporcionan información que nos facilita la implementación del algoritmo. Sin embargo, se ha decidido doblar esa probabilidad de mutación de cara a proporcionar una mayor variabilidad de modo que finalmente la probabilidad de mutación utilizada ha sido $\frac{2}{26}$.

- ◇ Probabilidad de recombinación. Se fija una probabilidad de 0.8. Se suelen escoger probabilidades altas de crossover ya que este operador al mezclar información de los padres en un descendiente permite progresar de manera rápida hacia regiones óptimas del espacio de búsqueda.

- ◇ El parámetro α usado en la selección de supervivientes para determinar el porcentaje de elitismo se fija a 0.75.

Cabe señalar que la elección de dichos parámetros se ha realizado de manera empírica observando como se comportaba el algoritmo. A

continuación, en la Tabla 4.1 se pueden ver los resultados obtenidos tras llevar a cabo el experimento descrito. Se presenta la mejor solución entre las 10 repeticiones si bien, en casi la practica totalidad de los casos, en las 10 repeticiones se daba exactamente la misma solución, lo que denota que el algoritmo es robusto. Además, en cuanto a coste computacional[1], la ejecución de cada escenario le cuesta en media aproximadamente 1 minuto así que se puede afirmar que el coste es mínimo.

Penal.	Gr.	Fitness	Dens. marginal	N. Infac.	Penal.	Gr.	Fitness	Dens. marginal	N. Infac.
0	2	-1300.23	-1300.23	11	0.75	2	-1304.50	-1301.50	4
0	3	-1303.63	-1303.63	11	0.75	3	-1307.24	-1304.99	3
0	4	-1307.03	-1307.03	11	0.75	4	-1310.15	-1308.65	2
0	5	-1310.45	-1310.45	11	0.75	5	-1313.06	-1312.31	1
0	6	-1314.04	-1314.04	10	0.75	6	-1316.17	-1316.17	0
0	7	-1317.59	-1317.59	11	0.75	7	-1319.61	-1319.61	0
0	8	-1321.25	-1321.25	11	0.75	8	-1323.55	-1322.80	1
0.1	2	-1301.17	-1300.27	9	1	2	-1305.50	-1301.50	4
0.1	3	-1304.57	-1303.67	9	1	3	-1307.99	-1304.99	3
0.1	4	-1307.98	-1307.08	9	1	4	-1310.65	-1308.65	2
0.1	5	-1311.45	-1310.45	1	1	5	-1313.31	-1312.31	1
0.1	6	-1315.03	-1314.03	1	1	6	-1316.17	-1316.17	0
0.1	7	-1318.55	-1317.65	9	1	7	-1319.61	-1319.61	0
0.1	8	-1322.18	-1321.48	7	1	8	-1323.58	-1323.58	0
0.25	2	-1302.29	-1300.79	6	2	2	-1309.50	-1301.50	4
0.25	3	-1305.67	-1304.42	5	2	3	-1310.99	-1304.99	3
0.25	4	-1309.00	-1307.50	6	2	4	-1312.65	-1308.65	2
0.25	5	-1312.40	-1311.15	5	2	5	-1314.31	-1312.31	1
0.25	6	-1315.79	-1314.79	4	2	6	-1316.17	-1316.17	0
0.25	7	-1319.22	-1318.22	4	2	7	-1319.61	-1319.61	0
0.25	8	-1322.84	-1322.09	3	2	8	-1323.58	-1323.58	0
0.5	2	-1303.50	-1301.50	4	5	2	-1321.50	-1301.50	4
0.5	3	-1306.49	-1304.99	3	5	3	-1319.99	-1304.99	3
0.5	4	-1309.65	-1308.65	2	5	4	-1318.65	-1308.65	2
0.5	5	-1312.81	-1312.31	1	5	5	-1317.31	-1312.31	1
0.5	6	-1316.17	-1316.17	0	5	6	-1316.17	-1316.17	0
0.5	7	-1319.61	-1319.61	0	5	7	-1319.61	-1319.61	0
0.5	8	-1323.30	-1322.80	1	5	8	-1323.58	-1323.58	0

Tabla 4.1: Resultados del experimento

[1]Algoritmo ejecutado en un ordenador con las siguientes especificaciones: procesador de 2.2 GHz Intel Core i7 4GB RAM. Matlab versión R2014b.

Con el objeto de poder visualizar con mayor claridad estos resultados se presentan las gráficas del *fitness*, Figura 4.1, de la densidad marginal, Figura 4.2 y del número de infactibilidades, Figura 4.3.

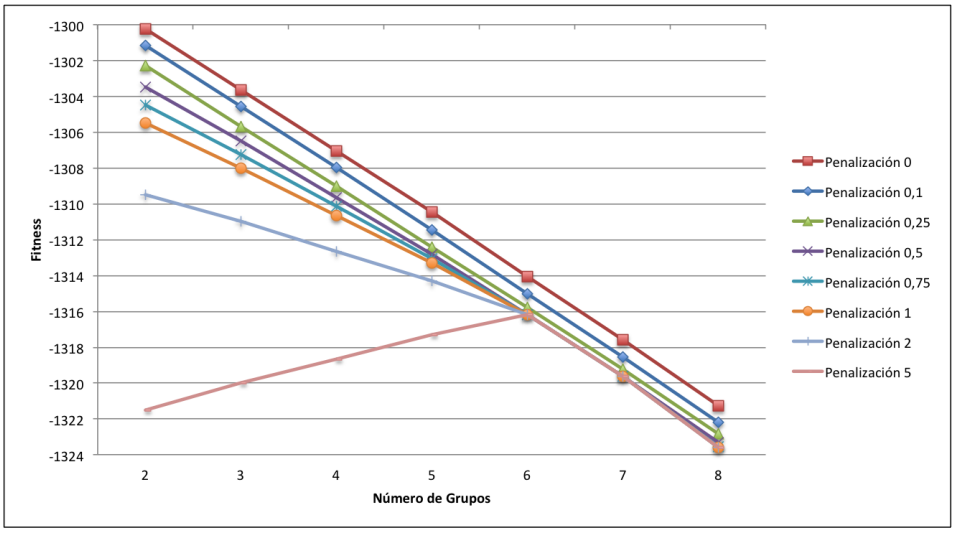

Figura 4.1: Gráfica del *fitness* en función del número de grupos según diferentes penalizaciones.

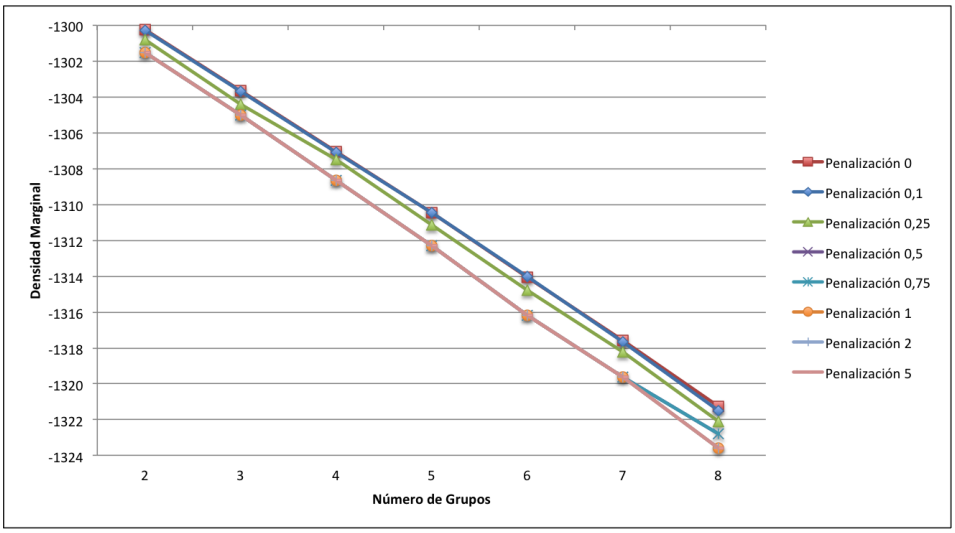

Figura 4.2: Gráficas de la densidad marginal en función del número de grupos según diferentes penalizaciones.

En la gráfica 4.1, se muestra como varía el valor del *fitness* en función

del número de grupos en los que se integran los municipios para cada uno de los posibles valores de penalización considerados. Puede observarse en dicha gráfica como la densidad marginal prima de forma importante la parsimonia del modelo, es decir, que el número de grupos sea pequeño frente a la penalización de las infactibilidades. Dicho comportamiento se mantiene para valores de penalización desde 0 hasta 2 y únicamente desaparece cuando la penalización se hace igual a 5.

Si examinamos la Figura 4.2 se puede observar la componente del *fitness* correspondiente a la densidad marginal, en la cual se ve cómo para todas las penalizaciones la solución con el mejor *fitness* corresponde a sólo dos grupos. Además se tiene que para las penalizaciones 0 y 0.1, el modelo obtenido para los distintos números de grupos alcanza prácticamente el mismo valor de la densidad marginal, sin embargo las soluciones son bastante distintas como puede comprobarse al mirar el número de infactibilidades en la Figura 4.3, en donde para la penalización 0 toma valores iguales a 11 en todos los casos menos uno, y para 0.1 toma valores que van desde 7 a 10. Es decir, se pone de manifiesto el hecho de estar favoreciendo la búsqueda de soluciones que respeten las relaciones de vecindad.

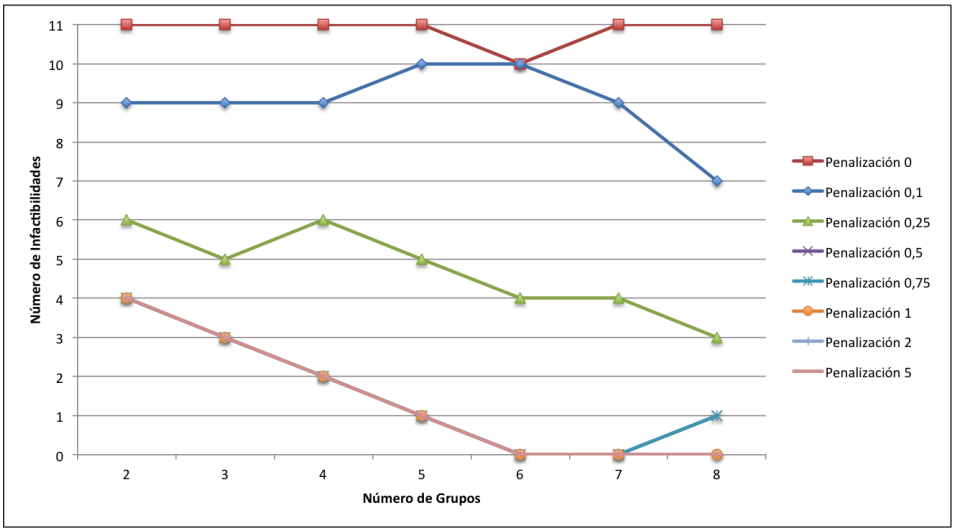

Figura 4.3: Gráficas del número de infactibilidades en función del número de grupos según diferentes penalizaciones.

Tras la penalización 0 y 0.1, la penalización 0.25 presenta otro com-

portamiento destacable: su valor de la densidad marginal ha empeorado respecto a 0 y 0.1 y su número de infactibilidades en la zona central de la Figura 4.3 toma valores entre 4 y 6. A partir de ahí, al considerar penalizaciones de valor 0.5 o superior, el valor de la densidad marginal y número de infactibilidades coincide para todos los números de grupos excepto para 8. En este caso, las penalizaciones 0.5 y 0.75 presentan una mejor densidad marginal -1322.80 y una infactibilidad igual a 1 mientras que para las penalizaciones iguales a 1, 2 y 5 la densidad marginal empeora en 0.78 unidades pero la solución cumple los criterios de vecindad.

En resumen, se puede decir que con penalizaciones inferiores o iguales a 0.25 influye más en la solución la calidad del modelo en términos de densidad marginal que las relaciones de vecindad. Para la penalización igual a 0.5 estamos ante una solución de equilibrio en la que se permite empeorar un poco la densidad marginal y en contrapartida se mejora el cumplimiento de las relaciones de vecindad. A partir de 0.5 las soluciones priman el cumplimiento de las relaciones de vecindad sobre la densidad marginal, obteniéndose soluciones que verifican las vecindades al considerar 6, 7 y 8, grupos, y soluciones en las que se va empeorando la vecindad de unidad en unidad (empeoramiento mínimo posible) al disminuir el número de grupos.

Para finalizar la sección se mostrará en más detalle las dos soluciones mejores correspondientes a los escenarios más extremos: sin penalización (con dos grupos) y con penalización máxima (con 6 grupos). Pueden observarse en la parte superior e inferior de la Figura 4.4, respectivamente. La primera de ellas construye un grupo con 127 transacciones (rojo) y 796 en el otro (verde). La segunda construye 6 grupos con 66 transacciones en el primer grupo, 50 en el segundo, 70 en el tercero, 588 en el cuarto, 85 en el quinto y 64 en el sexto.

Los modelos de regresión construidos se muestran en las Ecuaciones 4.1 y 4.2, para la solución de dos grupos y seis grupos, respectivamente.

$$\begin{aligned} \log P = 0.61654 + 2.17336 \log S + 0.22894 \log A + 0.09974 C + \epsilon_1, \\ \epsilon_1 \sim N(0, \sigma_1) \\ \log P = 0.48396 + 2.06903 \log S + 0.08169 \log A + 0.44879 C + \epsilon_2, \\ \epsilon_2 \sim N(0, \sigma_2) \end{aligned} \tag{4.1}$$

$$
\begin{aligned}
\log P &= 0.37412 + 1.96015 \log S + 0.45573 \log A + 0.49970 C + \epsilon_1, \\
&\qquad \epsilon_1 \sim N(0, \sigma_1) \\
\log P &= 0.35638 + 1.99369 \log S + 0.08947 \log A + 0.63217 C + \epsilon_2, \\
&\qquad \epsilon_2 \sim N(0, \sigma_2) \\
\log P &= 0.39491 + 2.16654 \log S + 0.25228 \log A + 0.11646 C + \epsilon_3, \\
&\qquad \epsilon_3 \sim N(0, \sigma_3) \\
\log P &= 0.33557 + 1.82962 \log S + 0.58620 \log A + 0.68629 C + \epsilon_4, \\
&\qquad \epsilon_4 \sim N(0, \sigma_4) \\
\log P &= 0.55222 + 1.92780 \log S - 0.26661 \log A + 0.80105 C + \epsilon_5, \\
&\qquad \epsilon_5 \sim N(0, \sigma_5) \\
\log P &= 0.32908 + 1.72565 \log S + 0.10299 \log A + 0.98723 C + \epsilon_6, \\
&\qquad \epsilon_6 \sim N(0, \sigma_6)
\end{aligned}
\tag{4.2}
$$

En último lugar, se quiere señalar que, en opinión de la autora, las soluciones obtenidas son razonables. En el primer caso se divide en dos grupos que agrupan las soluciones en función de la similaridad de las transacciones: municipios caros del Pirineo con municipios aragoneses importantes. Todos ellos tienen en común un precio superior de viviendas, frente a municipios más baratos. En el segundo caso, sólo se prima la vecindad y el algoritmo nos conduce a una de las 4 soluciones factibles construidas en la Sección 2.4, en particular a la de menor valor de la densidad marginal, que coincide con la de menor número de grupos de las cuatro.

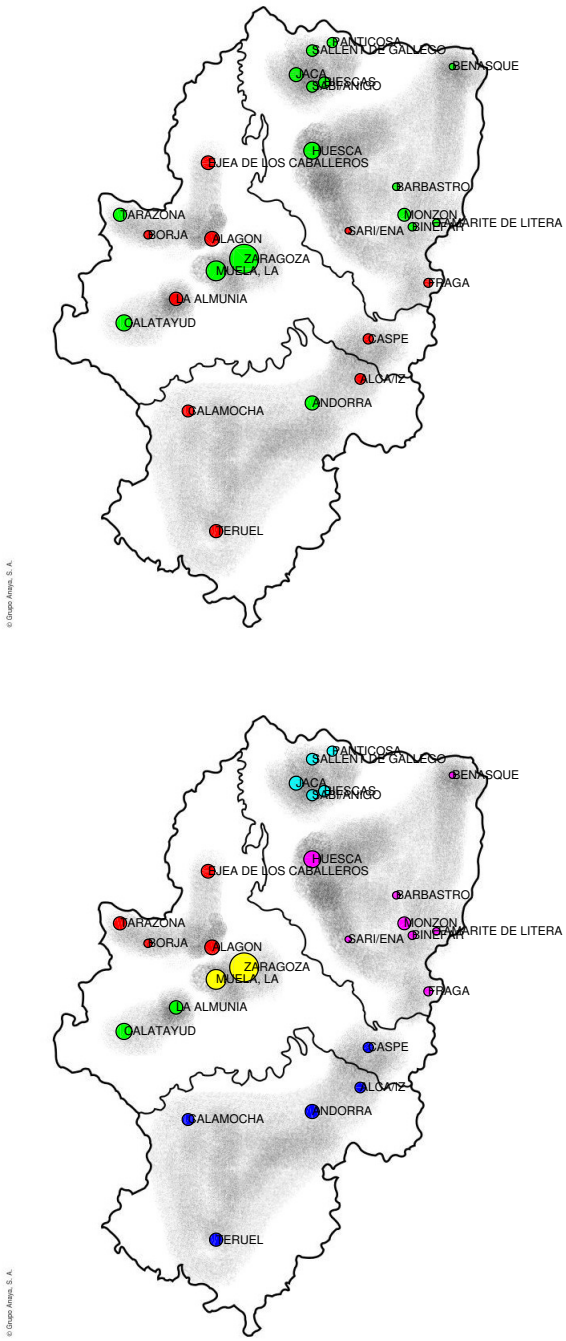

Figura 4.4: Mapa de Aragón para 2 grupos sin penalización (arriba) y 6 grupos con penalización máxima (abajo).

Capítulo 5

Algoritmo Genético Global

Este algoritmo es una generalización del anterior presentado en el Capítulo 3 con el objetivo de que el propio algoritmo sea capaz de determinar tanto el número de grupos en los que dividir los municipios como de determinar la división con objeto de maximizar el valor de la función objetivo.

Para lograr tal objetivo se han de modificar de manera adecuada algunos de los procesos que integraban el algoritmo genético inicial (Algoritmo 3.5). En este capitulo se mostrarán únicamente aquellos elementos del algoritmo original que se han modificado.

En cuanto al esquema general del algoritmo se mantiene la estructura mostrada en la versión original salvo por la inclusión de un operador de mutación del número de grupos que aparece justo antes de la recombinación de los padres: línea 9 del Algoritmo 5.1.

El capítulo consta de dos secciones, en la primera de ellas se presentarán los nuevos elementos y los elementos modificados de la primera versión y en la segunda se mostrará su funcionamiento, en un experimento similar al del capítulo anterior.

Algoritmo 5.1: Genético Global

Data: Semilla, tamaño de la población, probabilidad de mutación,probabilidad de mutación de grupos, probabilidad de recombinación, número máximo de iteraciones, alfa y penalización.

```
1  Creación de población inicial (Algoritmo 3.1);
2  sinMejora ← numIteraciones ← 0;
3  mejorSolucion ←-999999999;
4  while numIteraciones<numMaxIteraciones ∧
   sinMejora<0.25∗numMaxIteraciones do
5  |  numIteraciones ← numIteraciones+1;
6  |  Selección de padres(Proporcional al fitness, Sección 3.4);
7  |  if numIteraciones<0.9∗numMaxIteraciones then
8  |  |  Mutación del número de grupos;
9  |  |  Recupera soluciones (Algoritmo 3.2);
10 |  |  Recombinación (Sección 3.5);
11 |  |  Recupera soluciones (Algoritmo 3.2);
12 |  |  Mutación (Algoritmo 3.8);
13 |  else
14 |  |  Mutación de los padres seleccionados (Algoritmo 3.8);
15 |  end
16 |  Recupera soluciones (Algoritmo 3.2);
17 |  Cálculo del fitness para soluciones evolucionadas;
18 |  Reordenación de grupos (Algoritmo 3.3);
19 |  Selección de supervivientes (Algoritmo 3.6);
20 |  if Número de supervivientes < Tamaño de poblacion then
21 |  |  Completa población (Algoritmo 3.1);
22 |  end
23 |  if fitness máximo> mejorSolucion then
24 |  |  sinMejora ← 0;
25 |  |  mejorSolucion ←fitness máximo;
26 |  else
27 |  |  sinMejora ← sinMejora+1;
28 |  end
29 end
```

5.1. Modificaciones respecto al Algoritmo Genético Grupo a Grupo

Siguiendo el mismo orden que en Algoritmo 3.5, en primer lugar se crea la población inicial y se evalúa. La única diferencia en este caso es que para cada nuevo individuo se genera un valor aleatorio entre 2 y 8 para asignar el número de grupos en los que se repartirán los municipios. Una vez asignado el proceso para la construcción es idéntico al mostrado en el Algoritmo 3.1.

Algoritmo 5.2: Mutación del número de grupos.

Data: Población de padres, *pmg*, probabilidad mutación de grupos
Result: Padres con número de grupos mutados

```
1  for cada individuo ∈ Población de padres do
2      a = valor aleatorio ∈ (0, 1);
3      if a <= pmg then
4          Seleccionar dos grupos al azar y juntarlos en uno;
5          Diminuir en 1 el número de grupos y actualizar contador de
           transacciones;
6      end
7      if a > 1 − pmg then
8          Seleccionar un grupo aleatoriamente;
9          Incrementar en 1 el número de grupos;
10         Con probabilidad 0.5 mantener los municipios en el grupo
           inicial o asignar al nuevo;
11         Actualizar contadores de transacciones;
12     end
13     else
14         Mantener el padre como estaba;
15     end
16 end
```

Una vez creada la población inicial y ya dentro del proceso iterativo del algoritmo (líneas 6 a 29) se produce la selección de padres para reproducirse de forma idéntica al algoritmo original. A continuación se produce la mutación del número de grupos. El Algoritmo 5.2 muestra el proceso que se realiza. A cada uno de los individuos de la población de padres se le sortea si se mantiene con el mismo número de grupos

(probabilidad pmg), si aumenta en 1 el número de grupos (probabilidad pmg) o si disminuye este en 1 (probabilidad $2 \cdot pmg$).

Algoritmo 5.3: Función de similaridad modificada.

Data: Cromosoma a y cromosoma b
Result: Cromosoma a con grupos renumerados

conteo← matriz de ceros de dimensión $numGruposA \times numGruposB$;
for $i = 1$**to** $nMunicipios$ **do**
  $conteo(a(i), b(i)) \leftarrow conteo(a(i), b(i)) + 1$;
end
$clista \leftarrow$ matriz $conteo$ organizada en un vector ;
Ordenar decrecientemente $clista$;
$filas \leftarrow$ vector de tamaño $numGruposA$ con -1;
$columnas \leftarrow$vector de tamaño $numGruposB$ con -1;
$j \leftarrow h \leftarrow 1$;
while $h <= min(numGruposA, numGruposB)$ **do**
  $pa \leftarrow 1 + mod(i(j) - 1, numGruposA)$;
  $pb \leftarrow 1 + fix((i(j) - 1)/numGruposA)$;
  if $filas(pa) == -1 \&\& columnas(pb) == -1$ **then**
    $filas(pa) \leftarrow pb$;
    $columnas(pb) \leftarrow 1$;
    $h \leftarrow h + 1$;
  end
  j=j+1;
end
$aa \leftarrow a$;
for $j = 1$ **to** $numGruposA$ **do**
  $a(aa(1 : nMunicipios) == j) \leftarrow filas(j)$;
  $a(nMunicipios + j) \leftarrow aa(nMunicipios + filas(j))$;
end

En el caso de mantenerse el número de grupos no se le hace nada al individuo. Si se decide disminuir el número de grupos se seleccionan aleatoriamente dos grupos del individuo y se unen. En caso de decidir aumentar el número de grupos se selecciona aleatoriamente un grupo de los actuales y se divide en dos, para lo que se recorren todos los municipios que pertenecen al grupo seleccionado y con probabilidad 0.5 se

mantienen en dicho grupo o se cambian al grupo nuevo. Al acabar se actualiza el número de transacciones de ambos grupos.

Una vez realizado el proceso de mutación del número de grupos se procede a realizar la recombinación de los padres de igual forma a como se realizó inicialmente pero con una función de similaridad modificada para que trabaje con individuos en los que el número de grupos puede ser distinto. La nueva función de similaridad Algoritmo 5.3 requiere que el primer individuo tenga un número de grupos menor o igual que el segundo ya que trata de identificar y renumerar los grupos del primero que más se parecen a los del segundo. Salvo por ese detalle el funcionamiento es equivalente al de la original.

Tras la recombinación y la recuperación del número de transacciones se procede a la mutación de los individuos para la que se utiliza la misma función que en el Capítulo 3. En el resto del algoritmo se mantienen los mismos procesos y elementos que los presentados originalmente.

5.2. Desarrollo del experimento y resultados

A continuación se muestra el funcionamiento del nuevo algoritmo, planteando un experimento equivalente al del Capítulo 4 pero en el que no se fija el número de grupos, ya que los fijará la propia evolución del algoritmo y en el que el número de iteraciones se ha elevado a 3000. Además, en este algoritmo, se cuenta con un parámetro nuevo: la probabilidad de mutación de grupos, que se toma como 0.25, tanto para unir grupos como para dividirlos.

El resto de parámetros del algoritmo no se especifican puesto que son los mismos que los tomados para el Algoritmo Genético Grupo a Grupo. Llevando a cabo la prueba se ha constatado que el algoritmo es robusto ya que en el 100 % de los casos para distinta inicialización se obtiene la misma solución. Su coste computacional[1], además, es bajo ya que la ejecución de cada escenario le cuesta en media aproximadamente 3 minutos. A continuación, en la Tabla 5.1 se pueden ver los resultados

[1] Algoritmo ejecutado en un ordenador con las siguientes especificaciones: procesador de 2.2 GHz Intel Core i7 4GB RAM. Matlab versión R2014b.

obtenidos tras llevar a cabo el experimento descrito.

Penal.	Gr.	Fitness	Dens. marginal	N. Infac.	Penal.	Gr.	Fitness	Dens. marginal	N. Infac.
0	2	-1300.23	-1300.23	11	0.75	2	-1304.50	-1301.50	4
0.1	2	-1301.17	-1300.27	9	1	2	-1305.50	-1301.50	4
0.25	2	-1302.29	-1300.79	6	2	2	-1309.50	-1301.50	4
0.5	2	-1303.50	-1301.50	4	5	6	-1316.17	-1316.17	0

Tabla 5.1: Resultados del experimento

Si comparamos estos resultados con los del algoritmo previo mostrados en la Tabla 4.1, se observa que estos resultados son un reflejo de la mejor situación posible en términos de número de grupos para cada penalización, que era precisamente lo que se buscaba y por otro lado lo que cabía esperar.

Capítulo 6

Conclusiones

En esta memoria se ha desarrollado una herramienta para resolver el problema de la valoración del precio de las viviendas. Dicho problema tiene una gran repercusión como ha quedado de manifiesto en la introducción donde se han señalado numerosas situaciones en las que dicha herramienta sería de gran utilidad.

La herramienta construida se basa en el uso de Algoritmos Genéticos y modelos de regresión lineal. Los resultados que se han obtenido con este planteamiento y que se muestran en los capítulos 4 y 5, demuestran la robustez de dicha herramienta ya que, independientemente de la inicialización, prácticamente en el 100 % de los casos para el Algoritmo Grupo a Grupo y en el 100 % de las ocasiones en el Algoritmo Global se obtiene la misma solución. Además, el costo computacional es mínimo: una ejecución de un escenario, en media, tarda aproximadamente un minuto en el caso del Algoritmo Grupo a Grupo y tres minutos en el Algoritmo Global.

La implementación del modelo para su uso por parte de los expertos es sencilla, ya que bastaría con una hoja excel programada con los parámetros de la regresión según el modelo escogido.

Por último, cabe destacar que, tal y como se ha construido la herramienta, sería fácilmente modificable para considerar restricciones adicionales o modificaciones de la función objetivo por lo que este trabajo queda abierto a posibles futuros desarrollos.

Anexo I

	Municipio nº																										
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1 ALAGON	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2 BORJA	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3 CALATAYUD	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 CASPE	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5 EJEA CABALLEROS	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 LA ALMUNIA	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7 LA MUELA	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8 TARAZONA	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9 ZARAGOZA	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10 ALCAÑIZ	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11 ANDORRA	0	0	0	1	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
12 CALAMOCHA	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13 TERUEL	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14 BARBASTRO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0
15 BENASQUE	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
16 BIESCAS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
17 BINEFAR	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1
18 FRAGA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
19 HUESCA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
20 JACA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0
21 MONZON	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1
22 PANTICOSA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0
23 SABIÑANIGO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0
24 SALLENT GALLEGO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
25 SARIÑENA	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
26 TAMARITE LITERA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0

Tabla 6.1: Tabla de vecindades inmediatas.

Se debe recordar que la relación de vecindad es una relación simétrica y un municipio es vecino de sí mismo. Sin embargo la matriz que aquí se

presenta es una matriz de cara a la implementación de algoritmos y si se pusieran unos en la diagonal principal entraría en bucles infinitos.

Bibliografía

[1] D.R. Jones and M.A. Beltramo. Solving partitioning problems with genetic algorithms. In *Proc. Fourth Int. Conf. Genet. Algorithm*, pages 442–9, 1991.

[2] Gregor Von Laszewski. Intelligent structural operators for the k-way graph partitioning problem. In *4th Int. Conf. Genet. Algorithms*, pages 45–52, Plenum, 1991. Morgan-Kaufman.

[3] Raf van Driessche and Robert Piessens. Load Balancing with Genetic Algorithms. In *Parallel Probl. solving from Nat. 2*, pages 341–350, 1992.

[4] H. Ding, A.A. El-Keib, and R. Smith. Optimal clustering of power networks using genetic algorithms. *Electr. Power Syst. Res.*, 30(3):209–214, 1994.

[5] Emanuel Falkenauer. A hybrid grouping genetic algorithm for bin packing. *J. Heuristics*, 2(1):5–30, 1996.

[6] Evelyn C. Brown and Robert T. Sumichrast. Evaluating performance advantages of grouping genetic algorithms. *Eng. Appl. Artif. Intell.*, 18:1–12, 2005.

[7] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

[8] T. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, third edn. edition, 1996.

[9] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[10] IBM. *Ilog Cplex*. IBM, Armonk, New York, 2015.

[11] Matlab. *Release 2012b*. The MathWorks Inc., Natick, Massachusetts, 2013.

[12] R.C. Prim. Shortest Connection Networks And Some Generalizations. *Bell Syst. Tech. J.*, 36:1389–1401, 1957.