Modelos estáticos de localización de instalaciones competitivas. Una aplicación

Salvador Rodríguez Chopo

Trabajo de Fin de Grado en Matemáticas
Universidad de Zaragoza

Directores del trabajo: Herminia I. Calvete Fernández
y José Á. Iranzo Sanz

14 de junio de 2023

Prólogo

La localización de instalaciones es una de las áreas más relevantes dentro de la Investigación Operativa. En este problema se trata de determinar las localizaciones óptimas en las que ubicar un conjunto de instalaciones para conseguir algún objetivo específico, como maximizar ganancias, cubrir toda la demanda o minimizar pérdidas. Para analizar el problema de localización, puede suponerse un mercado monopolista en el que solo opera una empresa con sus instalaciones, o puede suponerse que hay competencia y varias empresas localizan instalaciones y se esfuerzan por alcanzar sus objetivos, obligando a los clientes a elegir entre todas las empresas establecidas según ciertos criterios. Este último caso, en el que hay competencia en el mercado, se denomina localización competitiva de instalaciones y amplía las perspectivas y el alcance analítico del problema. Este es el tema principal abordado en este trabajo, y para ello se han desarrollado tres capítulos, cuyo contenido se introduce brevemente a continuación.

En el primer capítulo se presenta una introducción al problema de localización de instalaciones. Se formulan también los que se puede considerar cuatro modelos clásicos de localización de instalaciones no competitivas: el problema de la p-mediana, el problema del p-centro, el problema de localización con costes fijos y el problema de cubrimiento. Finalmente, se presentan las características básicas del problema de localización competitiva de instalaciones: la naturaleza de la competencia, el número de nuevas instalaciones, la representación espacial y el concepto de distancia, las características de los clientes, la política de precios y la función objetivo.

El segundo capítulo comienza desarrollando el problema de máximo cubrimiento (MAXCOV), punto de partida para los problemas que se desarrollan posteriormente. Se introduce a continuación el primer modelo de localización competiitva, el problema de máxima captación (MAXCAP), desarrollado por Charles ReVelle a mediados de los 70 y que es considerado la base de una variedad de modelos de localización competitiva, algunos de los cuales se desarrollan posteriormente en el capítulo. Este modelo tiene como objetivo maximizar la captación de la demanda por parte de la nueva empresa, que entra a ocupar parte del mercado en el que actualmente se establece otra empresa competidora. Las extensiones que se introducen de este modelo son el MAXRELOC (problema de máxima captación con relocalización) y el UMAXCAP (problema de máxima captación con incertidumbre). El primero de ellos considera en su formulación el cambio de ubicación de algunas de las instalaciones inicialmente situadas por la nueva empresa, mientras que el segundo analiza la cuestión desde un marco de incertidumbre al no conocerse totalmente algunos de los parámetros involucrados. El capítulo termina con una introducción a los modelos con decisión sobre los precios. En la atracción de la instalación entran en juego no solo la distancia a la que la instalación se encuentra sino el precio que ofrece. Se presentan dos variantes de este modelos que extienden los modelos MAXCAP y MAXCOV.

En el tercer capítulo se aplican todos los modelos lineales anteriores (MAXCAP, MAXRELOC, UMAXCAP) a varios escenarios realistas utilizando datos de ubicaciones, clientes y establecimientos en la ciudad de Zaragoza. En primer lugar, se proporciona una explicación detallada sobre el proceso de recopilación de datos, su posterior transformación para su integración en el código de *Python* y la estructura de implementación de los modelos con el software Gurobi. Se realizan después múltiples resoluciones de los modelos indicados variando los diferentes valores de los parámetros para analizar cómo afectan los cambios al resultado. Además de proporcionar una visión general de las simulaciones para cada modelo, se realizan pequeños análisis sobre situaciones específicas de interés. Tanto el código utilizado para resoluciones como buena parte de los mapas que ilustran las soluciones obtenidas se incluyen al final de la memoria.

Summary

The facility location is one of the most relevant areas within Operations Research. It involves determining the optimal locations to place a series of facilities with the aim of achieving specific objectives such as maximizing profits, cover all demand, or minimizing losses. To analyze the location issue, we can consider a monopolistic market in which only one company operates with its facilities, or there can be competition, where multiple companies locate facilities and strive to achieve their objectives, forcing customers to choose among all established companies based on certain criteria.
This last issue, where competition arises in the market, is called competitive facility location and further expands the analytical perspectives and scope of the discipline. This will be the main issue addressed in this work, and for this purpose, three chapters have been developed.

Chapter 1: Competitive facility location

In the first chapter, an introduction to competitive facility location is provided. It begins with a general overview of the location problem. This discipline, which has seen significant development since the 20th century, has its origins in ancient geometric problems such as Fermat's point of a triangle. A publication by Alfred Weber in 1909, applying one of these geometric problems to an industrial context, marked a turning point. From there, several authors started to develop problems and properties that expanded the possibilities for analysis. Twenty years after Weber's publication, Harold Hotelling published a work on a duopoly of ice cream vendors along a street, becoming a precursor to competitive location. Since then, the references have continued to grow and a multitude of issues have been developed on the subject.

The chapter continues by presenting what can be considered four classic models of non-competitive facility location. The first one is the p-median problem, which aims to minimize the total costs of locating p facilities by assigning a single facility to each customer. It is followed by the p-center problem, similar to the previous one but now aiming to minimize the maximum distance between any customer and the assigned facility. The third problem is the fixed-costs location problem, where locating a facility incurs costs and the number of facilities to be located is not fixed. The objective is to minimize the total costs. Finally, the coverage problem is introduced, whose objective is to locate the minimum number of facilities that ensures coverage of all customers.

To conclude, the basic characteristics of the competitive facility location problem are introduced: the nature of competition (problems can be static, dynamic, or sequential depending on the reaction of competing firms), the number of new facilities (single-facility problems or multi-facility problems), spatial representation and the concept of distance (location space can be continuous, network-based, or discrete), customer characteristics (demand elasticity and determinism of their decisions), pricing policies (factory or delivery pricing), and the objective function (pull objectives, also called attraction objectives, or push objectives, also known as repulsion objectives).

Chapter 2: Static competitive location model

The maximum covering problem supposes the origin for the problems that are developed afterwards. We have n possible locations to locate p facilities from which to serve the demand of m customers. The objective of the problem is to maximize the covered demand. Let $I = \{1, \ldots, m\}$ be the set of indices for the customers and $J = \{1, \ldots, n\}$ be the set of indices for the possible locations. We define the binary variables x_j, indicating whether a facility is located at $j \in J$, and w_i, indicating if customer $i \in I$ is covered

by any facility. Let $\widetilde{N}_i$ be the set of locations $j \in J$ capable of covering customer $i \in I$, and let a_i be the demand of that customer. The problem is formulated as follows:

$$\begin{aligned}
&\text{maximize} \quad \sum_{i \in I} a_i w_i \\
&\text{subject to} \\
&\qquad w_i \leqslant \sum_{j \in \widetilde{N}_i} x_j \qquad \forall i \in I \\
&\qquad \sum_{j \in J} x_j = p \\
&\qquad x_j, w_i \in \{0,1\} \qquad \forall i \in I, \forall j \in J
\end{aligned}$$

From this problem, Charles ReVelle developed in the mid-1970s the maximum capture problem, also known as MAXCAP. It is considered the foundation of a series of competitive location models, some of which are developed throughout this chapter. Considering the same notation as in the previous problem, and aiming to maximize the demand capture, we define the binary variables x_j, indicating whether the new company locates a facility at $j \in J$, y_i and z_i, the latter two indicating whether the company captures the entire demand of $i \in I$ or half of it, respectively. The sets N_i and K_i refer to the locations $j \in J$ where the new company would capture the entire demand or half of the demand, respectively, of customer $i \in I$. Thus, the problem is formulated as follows:

$$\begin{aligned}
&\text{maximize} \quad \sum_{i \in I} a_i y_i + \sum_{i \in I} \frac{a_i}{2} z_i \\
&\text{subject to} \\
&\qquad y_i \leqslant \sum_{j \in N_i} x_j \qquad \forall i \in I \\
&\qquad z_i \leqslant \sum_{j \in K_i} x_j \qquad \forall i \in I \\
&\qquad y_i + z_i \leqslant 1 \qquad \forall i \in I \\
&\qquad \sum_{j \in J} x_j = p \\
&\qquad x_j, y_i, z_i \in \{0,1\} \qquad \forall i \in I, \forall j \in J
\end{aligned}$$

The first extension considered is the maximum capture with relocation problem, abbreviated as MAXRELOC. In this problem, the incoming company already has certain facilities located in the market and seeks to locate additional ones. Additionally, it is allowed to change the location of a certain number of them. Once again, the objective is to maximize the demand captured.

Additionally, the problem of maximum capture with uncertainty, referred to as UMAXCAP, is also presented, where not all the problem parameters are considered known. This addresses the case where the demand of the customers and/or the locations of the competition are not entirely known to the incoming company, leading to different scenarios of possibilities. Through two analysis criteria, an optimal solution is obtained, aiming to address the uncertainty issue in the most favorable way for the incoming company.

Finally, competitive facility location problems with price decision are modeled. As customers now consider not only the distance to the facility but also the total cost, the decision maker has the opportunity to incorporate this variable into the problem modeling.

On one hand, the problem of maximum capture with prices (PMAXCAP) is developed as an extension of the MAXCAP model. On the other hand, the problem of maximum coverage with prices (PMAXCOV) is introduced as an extension of the MAXCOV model. Both problems offer a similar perspective on the issue, albeit with different formulations.

Chapter 3: Modeling and simulation with real data

In this final chapter, all the previous linear models (MAXCAP, MAXRELOC, UMAXCAP) are applied to several real-life scenarios using data from locations, customers, and establishments in the city of Zaragoza. Firstly, a detailed explanation is provided on the data collection process and its subsequent transformation for integration into the *Python* code.

Next, the implementation structure of the models in the Gurobi software is described, accompanied by a map representing all the data considered in the simulations. Multiple executions of the MAXCAP, MAXRELOC, and UMAXCAP models are then conducted, varying the parameter values.

In addition to providing an overview of the simulations for each model, small analyses are performed on specific situations of interest. The extensive code used for the tests and all the solution maps are included at the end of the report as an appendix, prior to the bibliography.

Índice general

Capítulo 1

Localización competitiva de instalaciones

En un momento u otro todas las empresas, tanto del sector privado como del público, se enfrentan al problema de la localización de instalaciones. Las industrias deben determinar la ubicación de sus plantas de fabricación y montaje, así como de sus almacenes. Los comercios deben colocar sus tiendas. Las compañías hoteleras deben situar sus alojamientos. Las administraciones públicas deben localizar servicios esenciales como escuelas, vertederos u hospitales, etc.

La capacidad de una compañía para producir y comercializar sus productos de manera efectiva o de una agencia para ofrecer servicios de alta calidad depende, entre otras cosas, de dónde y cómo se ubican sus instalaciones, y de la relación con otros establecimientos semejantes y con sus clientes. A partir de este hecho se ha ido desarrollando la teoría de localización de instalaciones. Esta es la rama de la Investigación Operativa en la que se modelan, formulan y resuelven problemas matemáticos relacionados con dicha localización espacial óptima.

La teoría de localización de instalaciones ha sido desarrollada, en gran medida, en torno a la hipótesis del monopolio espacial: una determinada empresa ubica una instalación capaz de ofrecer un producto o servicio único en un mercado en el que ella es la única protagonista. Sin embargo, en la práctica, este tipo de modelos no se ajustan a un gran número de situaciones reales, de modo que resulta necesario incorporar la competitividad a través de otros factores y agentes. Se dice que un modelo matemático de localización es de instalaciones competitivas cuando incorpora expresamente en su formulación el hecho de que las instalaciones de otras empresas o firmas ya están o estarán presentes en el mercado, de modo que las nuevas instalaciones tienen que competir por la cuota de mercado. Por ello, al formular uno de estos modelos, deben precisarse varias nociones, conceptos e hipótesis relacionados con el problema considerado para que el modelo sea apropiado y refleje sus características.

Los modelos que se presentan en este trabajo han sido principalmente estudiados basándose en las referencias [2, 5, 9, 10, 11, 12, 13, 14, 15, 16, 18]. En este capítulo se ofrece una visión general de los problemas de localización y se introducen los conceptos básicos relacionados con las características de los problemas de localización competitiva.

1.1. Introducción al problema de localización

El análisis de problemas de localización de instalaciones resulta, desde principios del siglo XX, un área de gran importancia dentro de la Investigación Operativa. Lo que se considera generalmente el primer modelo de localización fue propuesto por el economista Alfred Weber en 1909. Pierre de Fermat había propuesto a mediados del siglo XVII el siguiente problema geométrico: hallar el punto del plano que minimiza la suma de sus distancias euclídeas a otros tres puntos dados. En 1790, Thomas Simpson lo generalizó obteniendo el punto que hacía mínima la suma ponderada de dichas distancias. A partir de estas ideas, Weber en su obra en alemán *Über den Standort des Industrien* [20] asoció el problema al contexto industrial, asignando a los vértices distinto peso que simulaba la demanda existente, analizándolo y popularizándolo.

En 1964, Seifollah Hakimi [6] desarrolló y estudió problemas similares sobre un grafo $G = (V,A)$

finito y no dirigido, con $V = \{v_1, ..., v_n\}$ el conjunto de nodos o vértices y A el conjunto de arcos o conexiones. Definió el concepto de *mediana absoluta* como el punto del grafo que minimiza la suma ponderada de las distancias a cada uno de los vértices; es decir, $x_0 \in G$ es una mediana absoluta de G si:

$$\sum_{i=1}^{n} w_i \, d_{v_i,x_0} \leqslant \sum_{i=1}^{n} w_i \, d_{v_i,x} \qquad \forall x \in G$$

donde d_{ij} representa la distancia más corta entre los puntos i y j sobre el grafo y w_i representa el peso o demanda del vértice v_i. En su estudio, Hakimi probó que una mediana absoluta de un grafo se sitúa siempre en uno de sus vértices, discretizando así el problema continuo, y generalizó, además, el problema anterior para hallar p medianas que minimizaran dicha suma de distancias. Desde entonces, el problema de la p-mediana y sus diversas variantes se han convertido en herramientas ampliamente utilizadas para abordar el problema de la localización de instalaciones.

Los problemas de localización, en su forma más general, pueden describirse de la siguiente manera. Un conjunto de clientes distribuidos espacialmente en un área geográfica demandan un cierto producto o servicio. La demanda de los clientes es cubierta por una o varias instalaciones y las instalaciones pueden operar en un marco de cooperación o de competencia dependiendo de los requisitos de los clientes. El proceso de decisión establece cómo y dónde ubicar las instalaciones sobre el territorio para optimizar un cierto objetivo, considerando las necesidades del mercado y las restricciones geográficas.

En el marco de los problemas localización se identifican tres elementos esenciales. En primer lugar, las *instalaciones*, concepto con el que se denota una gran variedad de objetos (almacenes, plantas productivas, escuelas, hospitales, puntos de venta, centros comerciales, edificios públicos, etc.) que serán localizados en un espacio determinado para proporcionar un servicio o producto con el fin de optimizar algún objetivo. Se puede, por ejemplo, tratar de maximizar el beneficio obtenido de dichas instalaciones, disminuir los gastos de transporte hacia otras ubicaciones o maximizar la cuota de clientela potencial que las instalaciones atraen o retienen. En segundo lugar, las *localizaciones*, que se refieren al conjunto de posibles puntos en el espacio disponibles para la ubicación de las instalaciones. El tipo de espacio considerado condiciona la forma de abordar el problema. No resulta igual de complejo considerar un problema de localización en una red sencilla formada por algunos pueblos unidos por carreteras que analizar la ubicación óptima de una antena de televisión en un punto cualquiera de una gran ciudad. Finalmente, el tercer elemento esencial del problema es la *clientela*, nombre dado al conjunto de usuarios que demandan los servicios o productos que las instalaciones, presentes o futuras, van a ofrecer. En general, el término *cliente* se asocia a un individuo o grupo de individuos con una ubicación y un comportamiento únicos e identificables. Los clientes pueden estar situados en lugares geográficos concretos o estar dispersos en áreas del espacio. El término *demanda* abarca una amplia gama de bienes y servicios que pueden ser adquiridos por un cliente específico o por el conjunto total de clientes en un determinado lugar, a fin de satisfacer sus necesidades y deseos. La distribución espacial, las características y el comportamiento de la demanda son fundamentales en la descripción de los modelos de localización. En los modelos considerados en este trabajo no se distingue entre diferentes productos ni se analizan cuestiones relativas a la mezcla de ellos. Por tanto, se asume que la demanda es de un solo producto, aunque pueda representar a una clase de productos. Dado que un cliente posee una ubicación y genera una determinada demanda, se emplea como sinónimo de cliente el término *punto de demanda*. Finalmente, se denomina *mercado* al conjunto de todos los clientes y su demanda, junto con su organización en el espacio.

Harold Hotelling, considerado el precursor del análisis de la localización competitiva, en su trabajo *Stability in Competition* [8] estudió en 1929 las políticas de localización de un duopolio de heladeros. Hotelling definió su modelo de ciudad lineal como aquel en el que dos empresas ofrecen un producto homogéneo a un conjunto de clientes que se distribuyen uniformemente a lo largo de una calle, representada por un segmento. Con ello, describió el conocido como *principio de mínima diferenciación*, que establece que en gran parte de los mercados resulta razonable que los productores ofrezcan productos lo más similares posible, y analizó las estrategias respecto al precio y al emplazamiento en ese mercado lineal acotado cuya demanda se mostraba inelástica al precio, constituyendo así el primer análisis que incorporaba la localización como una variable de elección determinante. En 1968, Michael Teitz [19]

demostró que la extensión a un caso con tres vendedores no conducía a un equilibrio como el presentado por Hotelling. En 1987, Eiselt y Laporte [4] estudiaron la localización de una nueva instalación en un mercado también lineal formado por q vendedores. Para ello, consideraron una distribución uniforme de los puntos de venta existentes y supusieron que la atracción de la clientela sería directamente proporcional al tamaño del punto de venta e inversamente proporcional a la distancia hasta él. En 1983, Hakimi [7] extendió el problema de Hotelling ampliándolo a dos dimensiones y estudiando la ubicación de p nuevos servidores en una red con otros q ya existentes.

En la literatura se han propuestos numerosas variantes tanto en los problemas de localización como de localización competitiva. Las referencias [3, 5, 9, 10] dan una visión general de ambos problemas y presentan numerosas referencias y modelos.

1.2. Modelos clásicos de localización no competitiva

Aunque este trabajo se centra en el estudio de ciertos modelos de localización competitiva, a continuación se formulan cuatro problemas clásicos de localización no competitiva con objeto de presentar las ideas o aspectos básicos de estos problemas [10].

Supondremos que se dispone de n ubicaciones posibles para localizar instalaciones desde las que se ha de atender a m clientes. Sea $I = \{1,\dots,m\}$ el conjunto de índices para los clientes y $J = \{1,\dots,n\}$ el de índices de las posibles ubicaciones. Se definen las siguientes variables:

- $x_j = \begin{cases} 1, & \text{si se localiza una instalación en } j \in J \\ 0, & \text{en caso contrario} \end{cases}$
- $y_{ij} = \begin{cases} 1, & \text{si el cliente } i \in I \text{ se asigna a la instalación de } j \in J \\ 0, & \text{en caso contrario} \end{cases}$

1.2.1. Problema de la p-mediana

Sea $c_{ij} \geqslant 0$ el coste de satisfacer la demanda del cliente $i \in I$ desde la instalación $j \in J$. En el problema de la p-mediana, se trata de localizar p instalaciones con el objetivo de minimizar el coste total, garantizando que cada cliente es atendido desde una única instalación. El modelo de programación entera binaria es:

$$\min \quad \sum_{i\in I}\sum_{j\in j} c_{ij}y_{ij} \tag{1.1a}$$

sujeto a

$$\sum_{j\in J} y_{ij} = 1 \qquad \forall i \in I \tag{1.1b}$$

$$\sum_{j\in J} x_j = p \tag{1.1c}$$

$$y_{ij} \leqslant x_j \qquad \forall i \in I, \forall j \in J \tag{1.1d}$$

$$x_j, y_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{1.1e}$$

La función objetivo (1.1a) representa el coste total. Las restricciones (1.1b) garantizan que cada cliente se asigna a una única instalación, mientras que (1.1c) limita a p el número de instalaciones a localizar. Por otro lado, el conjunto de restricciones (1.1d) garantiza que solo se asignan clientes a instalaciones que se localizan. Finalmente, (1.1e) establecen las condiciones de las variables involucradas.

1.2.2. Problema del p-centro

Es semejante al problema de la p-mediana, pero ahora el objetivo es minimizar la distancia máxima entre cualquier cliente y su instalación asignada. Sea d_{ij} la menor distancia entre el cliente $i \in I$ y la

ubicación $j \in J$. El problema se formula como:

$$\min \quad z \tag{1.2a}$$

sujeto a

$$\sum_{j \in J} d_{ij} y_{ij} \leqslant z \qquad \forall i \in I \tag{1.2b}$$

$$\text{(1.1b) - (1.1e)} \tag{1.2c}$$

La función objetivo (1.2a) representa la distancia máxima. Las restricciones (1.2b) garantizan que la distancia de cualquier cliente a su instalación asignada no supera el valor de z. Las restricciones (1.2c) han sido explicadas en el modelo de la p-mediana.

1.2.3. Problema de localización con costes fijos

En este modelo no se establece a priori el número de instalaciones que deben localizarse, pero la instalación de cada una de ellas tiene un coste fijo y quiere minimizarse el coste total. Sea $f_j \geqslant 0$ el coste fijo asociado a la localización de una instalación en $j \in J$. El problema se formula como:

$$\min \quad \sum_{j \in J} f_j x_j + \sum_{j \in J} \sum_{i \in I} c_{ij} y_{ij} \tag{1.3a}$$

sujeto a

$$\text{(1.1b), (1.1d), (1.1e)} \tag{1.3b}$$

La función objetivo (1.3a) representa el coste total, compuesto por la suma de los costes fijos de instalación y de los costes de asignación de los clientes. Las restricciones (1.3b) han sido explicadas anteriormente.

1.2.4. Problema de cubrimiento

En este modelo se supone que un cliente solo puede ser atendido por una instalación si esta se encuentra a una distancia menor que un valor fijado D, llamado *distancia de servicio*. El objetivo es localizar el mínimo número de instalaciones que garantiza que todos los clientes pueden ser atendidos, o dicho de otro modo, que todos están cubiertos. Se define el parámetro binario

$$h_{ij} = \begin{cases} 1, & \text{si } d_{ij} \leqslant D \\ 0, & \text{en caso contrario} \end{cases}$$

que indica si el cliente $i \in I$ está cubierto por la instalación $j \in J$ o no. El problema se formula como:

$$\min \quad \sum_{j \in J} x_j \tag{1.4a}$$

sujeto a

$$\sum_{j \in J} h_{ij} x_j \geqslant 1 \qquad \forall i \in I \tag{1.4b}$$

$$x_j \in \{0,1\} \qquad \forall j \in J \tag{1.4c}$$

La función objetivo (1.4a) indica el número de instalaciones que se localizan. Las restricciones (1.4b) garantizan el cubrimiento de los clientes, es decir, cada cliente tiene al menos una instalación a una distancia menor o igual que D. Finalmente, (1.4c) establece las características de las variables del modelo.

1.3. Características básicas del problema de localización competitiva

La aparente simplicidad del concepto de modelo de localización competitiva, donde las nuevas instalaciones compiten por la cuota de mercado contra las ya existentes, oculta varias nociones implícitas y explícitas que deber ser precisadas antes de llegar a una formulación matemática precisa de los modelos considerados. En esta sección, se presenta una descripción general de las características básicas de los modelos de localización competitiva, así como algunas notaciones.

1.3.1. Naturaleza de la competencia

Los modelos se distinguen por la reacción de las empresas competidoras ante la aparición de nuevas empresas, clasificándose en:

- **Estáticos**: la compañía que se incorpora dispone de información completa y anticipada sobre las características y las estrategias del mercado, siendo ambas fijas y no habiendo reacción por parte de la competencia. El problema queda entonces reducido a situar de forma óptima las nuevas instalaciones teniendo en cuenta la demanda que podrán captar con la disposición actual, sin atender al efecto que producirá sobre las demás instalaciones ya existentes en el mercado.

- **Dinámicos**: en este caso se elimina la hipótesis de no reacción de la competencia y se supone que las firmas competidoras deciden simultáneamente sobre la localización de sus instalaciones. La solución suele basarse en encontrar el equilibrio Nash, concepto de la teoría de juegos basado en hallar una estrategia que maximice las ganancias de un jugador dadas las estrategias de los otros. Bajo esta hipótesis, ninguna de las empresas involucrada tiene incentivos para modificar unilateralmente su estrategia de mercado puesto que han elegido la mejor posible en ese momento.

- **Secuenciales**: como en los problemas dinámicos, se prescinde también de la hipótesis de no reacción por parte de la competencia pero se supone una jerarquía en el proceso de toma de decisiones. Se formula, generalmente, como un juego de Stackelberg con dos tipos de jugadores. Los líderes toman la decisión sobre las instalaciones teniendo en cuenta la reacción de sus competidores, que a su vez toman sus decisiones basadas en las decisiones de los líderes.

1.3.2. Número de nuevas instalaciones

Sea p el número de instalaciones a localizar en el espacio de mercado. Se diferencian dos tipos de problemas:

- **Problema de instalación simple** ($p = 1$): se trata de decidir dónde situar una única instalación nueva en el espacio, teniendo en cuenta su efecto sobre las ya existentes.

- **Problema multi-instalaciones** ($p > 1$): se trata de decidir dónde situar más de una planta debiendo considerar, además de su efecto sobre las existentes, la interacción entre ellas, tanto por la distancia como por la asignación de los clientes.

1.3.3. Representación espacial y distancia entre instalaciones

El espacio de localización, que determina el conjunto de métodos empleados para la resolución del modelo, establece el modo en el que el actual mercado y las futuras instalaciones están distribuidos. Puede ser de tres tipos:

- **Continuo**: viene determinado por un sistema cualquiera de coordenadas reales. Se puede considerar un problema unidimensional (sobre una recta), bidimensional (por ejemplo sobre una esfera o sobre un plano) o incluso n-dimensional, con $n > 2$. De entre los anteriores, la localización de instalaciones sobre el plano $\mathbb{R}^2$ es el caso más habitual.

- **Estructura de red**: el espacio de localización se define mediante un grafo. La ubicación de los clientes y la localización de las instalaciones, tanto de las existentes como de las nuevas, puede ser directamente sobre los vértices del grafo o bien en un punto cualquiera de sus arcos. Es por ello que los problemas sobre una red pueden ser considerados tanto continuos como discretos.

- **Discreto**: el mercado se compone por un conjunto finito de puntos de demanda y se establece otro conjunto finito de localizaciones posibles, conocidas de antemano, para las nuevas instalaciones.

Además de la representación espacial del problema, se debe especificar una *distancia* sobre ella, es decir, se ha de precisar cuán separadas están dos instalaciones cualesquiera. En el caso continuo se considera, generalmente, la distancia euclídea (ℓ_2), aunque también se usan otras como la distancia rectangular o de Manhattan (ℓ_1), o directamente cualquier otra norma ℓ_p ($1 \leqslant p \leqslant +\infty$). Para un problema con espacio discreto se ha de proporcionar una matriz de distancias. Para un problema con estructura de red se toma la distancia entre cada par de vértices a través de la noción de camino más corto en un grafo. Por otro lado, los clientes o puntos de demanda no siempre se corresponden con puntos concretos del sistema de coordenadas, sino que a veces son representados por áreas o regiones. Para abordar esta característica puede aplicarse un factor corrector de la distancia. Siguiendo el estudio realizado en [1] en el que consideran regiones circulares (o aproximaciones suyas), se aplica dicha corrección mediante el valor $d' = \sqrt{d^2 + k(r^2 + R^2)}$, donde d representa la distancia entre los centros de los puntos considerados (instalación y cliente), r es el radio de la instalación y R es el radio del área de demanda. Los análisis realizados sugieren que la constante puede tomar el valor $k = \frac{1}{4}$ para casos de distancias grandes entre radios, mientras que puede ser reemplazada por $k = \frac{4}{9}$ en el caso de aproximar distancias cortas.

1.3.4. Características de los clientes

Los clientes pueden situarse en puntos concretos, estar dispersos en una región o estar distribuidos uniformemente en la región. Los aspectos claves a identificar son el comportamiento de los clientes en relación con la demanda y con su capacidad de seleccionar las instalaciones.

En relación con la demanda, esta puede ser:

- **Inelástica**: se conoce y es fija o se muestra poco sensible ante cambios, es decir, las variaciones en los factores que definen los bienes o servicios no repercuten sustancialmente en la demanda de los consumidores. Suele corresponderse con aquellos casos en los que están involucrados productos considerados esenciales para el cliente.

- **Elástica**: varía en función de diversos factores como el precio o la calidad. Dicho de otro modo, las decisiones de los clientes se muestran muy sensibles a los cambios en los bienes. La elasticidad se mide calculando el cociente entre el porcentaje de variación de la cantidad demandada de un producto y el porcentaje de variación de su precio. Si resulta > 1 se considera demanda elástica y si es < 1 será inelástica. Al contrario que para la demanda inelástica, la elástica suele corresponderse con aquellos casos en los que están involucrados productos no esenciales.

En relación con la selección de las instalaciones, esta puede ser:

- **Determinista**: la demanda total de cada cliente es servida por la instalación hacia la que siente más atracción. Es necesario destacar que una elección determinista no especifica qué ocurre cuando un cliente se siente igual de atraído por varias instalaciones, incluyendo las nuevas, de modo que pueden considerarse varias reglas de resolución de empates: dividir la demanda entre todas las plantas hacia las que se siente igualmente atraído, destinar toda la demanda exclusivamente a las nuevas plantas, hacerlo exclusivamente a las plantas de la competencia, etc.

- **Probabilística**: aparece un factor estocástico en el problema porque el cliente reparte su demanda entre las instalaciones de acuerdo con una determinada distribución de probabilidad que es función de las ventajas competitivas que le ofrecen.

Parece razonable considerar que la distancia que separa al cliente de la instalación influye en gran medida en su decisión de elegirla o no. Sin embargo, en muchos casos, no solamente este factor es clave sino que están presentes otros como la variedad o el precio de los productos, el tamaño de la instalación o la atención al cliente. Esa inclinación de la clientela hacia un establecimiento concreto se conoce como *calidad* de la instalación y, para cuantificarla, se considera una *función de atracción* que generalmente se define como $u_{ij}(\alpha_j, d_{ij}) = \alpha_j / f(d_{ij})$, es decir, se supone que la atracción que un cliente i siente hacia una instalación j es directamente proporcional a la calidad de la instalación α_j e inversamente proporcional a una función positiva f de la distancia d_{ij}. Si la atracción solo depende de la distancia, lo que se asume cuando todas las instalaciones son similares, se tiene un problema de localización *uniforme*, mientras que si depende de más factores recibe el nombre de problema *multiforme*.

1.3.5. Política de precios

Los precios fijados para los bienes o servicios constituyen otro componente importante del modelo. Dichos precios pueden establecerse de forma simultánea o secuencial y, generalmente, se utilizan las siguientes políticas:

- **Precio de fábrica**: esta política también se conoce como *política de precios en origen*. En este enfoque, el vendedor establece un precio uniforme para su producto en todo el mercado y es responsabilidad del comprador encargarse del transporte. Es la única política que no tiene en cuenta la diferenciación entre los clientes.
- **Precio de entrega**: a esta estrategia también se le conoce como *política de precios en destino*. El vendedor establece precios distintos para el producto en cada una de las zonas del mercado y se hace cargo del transporte el cliente.

1.3.6. Función objetivo

En la literatura relacionada con la modelización de problemas de localización competitiva de instalaciones, los objetivos suelen clasificarse en dos categorías principales:

- **Objetivo pull**: también llamados *objetivos de atracción*. Se supone que una mayor cercanía, o equivalentemente una menor distancia, de la clientela a las instalaciones producirá un mejor valor de la función objetivo.
- **Objetivo push**: se llaman también *objetivos de repulsión*. Cuando las instalaciones son indeseables para la clientela interesa que estén lejos o que no influyan demasiado.

Algunos ejemplos de objetivos pull pueden ser maximizar la cuota de mercado, minimizar la distancia a las instalaciones, maximizar el beneficio o minimizar el coste de transporte. Por otro lado, maximizar la distancia a las instalaciones, dispersarlas lo suficiente en el espacio u obtener el cubrimiento mínimo posible para un determinado servicio son ejemplos de objetivos de tipo push.

Capítulo 2

Modelos estáticos de localización competitiva

Los modelos estáticos de localización competitiva de instalaciones corresponden a un análisis del mercado a corto plazo al asumir la hipótesis de no reacción por parte de la competencia: suponen que esta tardará mucho tiempo y/o necesitará una cantidad notable de recursos para reaccionar ante la nueva instalación, de modo que esta tendrá tiempo suficiente para cosechar beneficios significativos. Los modelos que se presentan en este capítulo constituyen la base sobre la que se han formulado otros modelos más complejos en la literatura. Supondremos que el espacio de localización de los modelos es de tipo discreto, ya sea porque la distribución espacial de los elementos se realiza únicamente sobre los nodos de una red, o bien porque directamente los conjuntos de datos son discretos, resultando ambos casos equivalentes tanto en formulación como en interpretación. Supondremos también que la demanda de los clientes es inelástica y que la selección de las instalaciones es determinista.

Los modelos formulados en los siguientes apartados pueden considerarse en cierta medida extensiones, que tienen en cuenta la competencia, del modelo de localización no competitiva **problema de máximo cubrimiento** (en adelante **MAXCOV**, por las siglas en inglés de "Maximal Covering Location Problem") [2]. Este problema aborda una cuestión que surge directamente del problema de cubrimiento, descrito en el apartado 1.2.4, cuando los recursos para localizar las instalaciones son insuficientes para lograr cubrir toda la demanda. En este caso, el decisor puede optar por cubrir la mayor cantidad de demanda posible, utilizando todos los medios disponibles. Por tanto, en este modelo se trata de localizar p instalaciones con el objetivo de maximizar la demanda cubierta.

Sea a_i la demanda potencial del cliente $i \in I$. Sea $\widetilde{N}_i$ el conjunto de ubicaciones $j \in J$ que cubren al cliente $i \in I$, es decir, $\widetilde{N}_i = \{j \in J \mid d_{ij} \leqslant D\}$. Además de la variable x_j ya introducida, pero cuya definición se repite para facilitar la comprensión del modelo, se define también w_i como se indica a continuación:

- $x_j = \begin{cases} 1, & \text{si se localiza una instalación en } j \in J \\ 0, & \text{en caso contrario} \end{cases}$
- $w_i = \begin{cases} 1, & \text{si el cliente } i \in I \text{ está cubierto por alguna instalación} \\ 0, & \text{en caso contrario} \end{cases}$

El modelo MAXCOV se formula matemáticamente como:

$$\max \quad \sum_{i \in I} a_i w_i \tag{2.1a}$$

sujeto a

$$w_i \leqslant \sum_{j \in \widetilde{N}_i} x_j \qquad \forall i \in I \tag{2.1b}$$

$$\sum_{j \in J} x_j = p \tag{2.1c}$$

$$x_j, w_i \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{2.1d}$$

La función objetivo (2.1a) representa la captación total de demanda. Las restricciones (2.1b) garantizan que el cliente $i \in I$ solo es cubierto si tiene alguna instalación a una distancia menor o igual que D. Como ya se ha indicado, la restricción (2.1c) fuerza a que haya p instalaciones, mientras que (2.1d) establecen las condiciones de las variables involucradas. Aunque en el modelo anterior la distancia D es igual para todos los clientes, el modelo se generaliza de manera inmediata considerando una D_i distinta para cada cliente $i \in I$.

2.1. Problema de máxima captación (MAXCAP)

El **problema de máxima captación** (en adelante MAXCAP, por las siglas en inglés de "Maximum Capture Problem") es reconocido como el punto de partida de una serie de modelos de localización competitiva discreta. Fue introducido por Charles ReVelle [13] en 1986 y revisado en 1995 en otra publicación junto a Daniel Serra [16]. Se supone un mercado en el que está presente la empresa B, al que se trata de incorporar una nueva empresa A. La empresa A desea conocer dónde ubicar p instalaciones con objeto de maximizar la captación de clientes. Se suponen conocidas las ubicaciones de las instalaciones ya existentes de la empresa B, de las n posibles localizaciones para las nuevas instalaciones y de los m clientes junto con su demanda. Se supone también que los clientes van a la instalación más cercana a su ubicación y que cuando dos instalaciones están equidistantes, el modelo reparte la captación de clientes de manera equitativa entre ellas. El objetivo del modelo MAXCAP es que la empresa A maximice la captación de clientes mediante la instalación de sus p nuevas instalaciones. Notemos que para la formulación de este problema se están asumiendo las siguientes hipótesis:

- La competencia presente en el mercado se conoce y es fija.
- La venta del producto es homogénea entre instalaciones de las distintas empresas (no existen diferencias entre las propias instalaciones).
- La decisión de la clientela se basa únicamente en la distancia recorrida.
- Los costes unitarios del producto ofrecido son iguales en todas las instalaciones del mercado.

A continuación, se introduce la nueva notación necesaria para formular el modelo. Sea S_i la distancia entre el cliente $i \in I$ y la instalación de B más cercana. Si A no sitúa una instalación más cerca, no logrará captar la demanda correspondiente puesto que el cliente seguirá acudiendo a la instalación de B. Sea N_i el conjunto de localizaciones que están más cerca del cliente $i \in I$ que la instalación de B más cercana a él, es decir, $N_i = \{j \in J \mid d_{ij} < S_i\}$. Este conjunto contiene las ubicaciones que le permiten a la empresa A captar la demanda del cliente $i \in I$. Sea K_i el conjunto de localizaciones que equidistan del cliente $i \in I$ y de la instalación de B más cercana a él, es decir, $K_i = \{j \in J \mid d_{ij} = S_i\}$. Este conjunto contiene las ubicaciones en las que la empresa A se reparte equitativamente la demanda con la empresa B.

Para formular el modelo MAXCAP se definen las siguientes variables:

- $x_j = \begin{cases} 1, & \text{si } A \text{ localiza una instalación en } j \in J \\ 0, & \text{en caso contrario} \end{cases}$
- $y_i = \begin{cases} 1, & \text{si } A \text{ capta toda la demanda del cliente } i \in I \\ 0, & \text{en caso contrario} \end{cases}$
- $z_i = \begin{cases} 1, & \text{si la demanda del cliente } i \in I \text{ se reparte entre } A \text{ y } B \\ 0, & \text{en caso contrario} \end{cases}$

El problema se formula matemáticamente como:

$$\max \quad \sum_{i \in I} a_i y_i + \sum_{i \in I} \frac{a_i}{2} z_i \tag{2.2a}$$

sujeto a

$$y_i \leqslant \sum_{j \in N_i} x_j \qquad \forall i \in I \tag{2.2b}$$

$$z_i \leqslant \sum_{j \in K_i} x_j \qquad \forall i \in I \tag{2.2c}$$

$$y_i + z_i \leqslant 1 \qquad \forall i \in I \tag{2.2d}$$

$$\sum_{j \in J} x_j = p \tag{2.2e}$$

$$x_j, y_i, z_i \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{2.2f}$$

La función objetivo (2.2a) define la captación total de demanda de la empresa A con sus p instalaciones. Esta captación resulta de la suma de la demanda captada exclusivamente por A y de la parte correspondiente de la demanda compartida por ambas empresas. Las restricciones (2.2b) garantizan, que si la empresa A no localiza alguna instalación más cerca del cliente $i \in I$ que la instalación de B más cercana, entonces la empresa A no capta su demanda. Por otro lado, si se localiza alguna instalación más cerca, aunque la variable y_i queda libre para tomar el valor 0 o el valor 1, el hecho de que la función objetivo maximice y $a_i \geqslant 0$ garantiza que tomará el valor 1. El conjunto de restricciones (2.2c) tienen una interpretación semejante para el caso de equidistancia entre la instalación de B más cercana al cliente $i \in I$ y la localización de la instalación de A. Las restricciones (2.2d) garantizan los tres estados posibles para la demanda de $i \in I$: captada totalmente por A ($y_i = 1, z_i = 0$), captada totalmente por B ($y_i = 0, z_i = 0$) y captada a medias por A y por B ($y_i = 0, z_i = 1$). La restricción (2.2e) determina el número p de instalaciones que va a localizar A, y (2.2f) establecen las condiciones de las variables involucradas.

Algoritmo 1 Pseudocódigo de AMACA

Input: p: Número de instalaciones a localizar por A
Output: *ins_A*: Distribución final de las p instalaciones de A

```
1:  ins_A ← Localizacion_Inicial(p, J)
2:  Mejor_Captacion ← Captacion(ins_A, ins_B)
3:  repeat
4:      Mejor_Iteracion ← false
5:      for i in ins_A do
6:          for j in J do
7:              Nueva_Configuracion ← ins_A
8:              Nueva_Configuracion[i] ← j
9:              Nueva_Captacion ← Captacion(Nueva_Configuracion, ins_B)
10:             if Nueva_Captacion > Mejor_Captacion then
11:                 Mejor_Iteracion ← true
12:                 Mejor_Captacion ← Nueva_Captacion
13:                 ins_A ← Nueva_Configuracion
14: until Mejor_Iteracion = false
```

Aunque las técnicas de resolución del problema han evolucionado mucho desde la publicación de Serra y ReVelle [16], a modo ilustrativo se presenta a continuación el algoritmo heurístico que se propuso en dicha publicación para resolver el problema, denominado **AMACA**, por sus siglas en inglés de "Approximate Maximum Capture Algorithm". Se construye inicialmente una solución factible, es decir, una localización de las p instalaciones, que puede ser obtenida con cualquier método y se calcula su captación total. A continuación, en cada iteración del algoritmo se busca mejorar la captación de la distribución actual, analizando las consecuencias de cambiar de ubicación las instalaciones ya localizadas. Para ello se seleccionan en algún orden las instalaciones. Se toma la primera de ellas y se evalúa qué sucede si se mueve a alguna localización actualmente vacía. Si la captación de clientes mejora, la instalación es relocalizada en dicha instalación y se pasa a considerar la siguiente instalación. Si no es posible relocalizar aquella instalación porque se empeora la captación, se pasa a considerar la siguiente

instalación, La iteración termina cuando han sido recorridas todas las instalaciones. Si la captación ha mejorado respecto a la actual, el algoritmo itera de nuevo con la nueva localización de las instalaciones. En otro caso, el algoritmo termina y la solución proporcionada es la distribución disponible en la última iteración. Notemos que en cada iteración el valor de la función objetivo mejora ya que solo se permiten relocalizaciones cuando se mejora la captación actual. En Algoritmo 1 se presenta el pseudocódigo del algoritmo AMACA.

2.2. Problema de máxima captación con relocalización (MAXRELOC)

El **problema de máxima captación con relocalización** (en adelante MAXRELOC, por sus siglas en inglés de "Maximum Capture Problem Including Relocation") [14, 15] es una extensión del modelo MAXCAP en el que la empresa A tiene actualmente ubicadas s instalaciones y se permite tanto la localización de nuevas instalaciones como el cambio de ubicación de las ya existentes. Con objeto de mejorar la demanda actualmente captada, la empresa A contempla la relocalización de r de las s instalaciones, así como la localización de p nuevas. Igual que para el modelo MAXCAP, se supone que ambas empresas tienen una demanda estructuralmente similar y, por tanto, el cliente es indiferente cuando elige entre ellas, utilizando como criterio la cercanía de la instalación. Como en ese modelo, cuando las instalaciones están equidistantes, la demanda se reparte equitativamente entre ambas empresas. Sea $J_A \subset J$ el conjunto de localizaciones ocupadas actualmente por instalaciones de la empresa A. El modelo se formula así:

$$\max \quad \sum_{i\in I} a_i y_i + \sum_{i\in I} \frac{a_i}{2} z_i \tag{2.3a}$$

sujeto a

$$y_i \leqslant \sum_{j\in N_i} x_j \qquad \forall i \in I \tag{2.3b}$$

$$z_i \leqslant \sum_{j\in K_i} x_j \qquad \forall i \in I \tag{2.3c}$$

$$y_i + z_i \leqslant 1 \qquad \forall i \in I \tag{2.3d}$$

$$\sum_{j\in J} x_j = s + p \tag{2.3e}$$

$$\sum_{j\in J_A} x_j = s - r \tag{2.3f}$$

$$x_j, y_i, z_i \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{2.3g}$$

En la función objetivo (2.3a) se maximiza la demanda neta captada. Los conjuntos de restricciones (2.3b)-(2.3d) han sido explicados en la formulación del modelo MAXCAP. La restricción (2.3e) fija el número total de instalaciones de A localizadas como la suma de las s relocalizadas y las p nuevas. Por otro lado, la restricción (2.3f) establece el número de instalaciones de A que no pueden ser relocalizadas, es decir, que deben permanecer en su localización original. Finalmente, las restricciones (2.3g) establecen las condiciones de las variables involucradas.

2.3. Problema de máxima captación con incertidumbre (UMAXCAP)

El **problema de máxima captación con incertidumbre** (en adelante UMAXCAP, por sus siglas en inglés de "Maximum Capture Problem with Uncertainty") [16, 17] es otra extensión del modelo MAXCAP en la que se considera que algunos de los parámetros involucrados no son completamente conocidos. En este caso, la localización de p instalaciones por parte de la empresa A para maximizar la captación de demanda en un mercado donde ya opera la competencia se aborda desde un marco de incertidumbre, puesto que existen varios escenarios futuros posibles como, por ejemplo, que la demanda puede tomar diferentes valores según las características del mercado o que las localizaciones de la empresa competidora pueden ser distintas. Se proponen dos criterios diferentes para formular el modelo. En

el primero, el objetivo es maximizar la mínima captación de demanda sobre todos los escenarios; en el segundo, se minimiza la máxima diferencia entre la captación de demanda que se obtiene y la que podría haberse logrado. La dificultad reside en que el alcance de la captación no solo depende de la ubicación de las instalaciones de la empresa A, sino también del escenario particular que se presenta.

Sea E el conjunto de índices que representan los posibles escenarios. Sea a_{ie} la demanda potencial del cliente $i \in I$ en el escenario $e \in E$ y sea S_{ie} la distancia entre el cliente $i \in I$ y la instalación de B más cercana en el escenario $e \in E$. Sean N_{ie} y K_{ie} los conjuntos análogos a N_i, K_i, respectivamente, asociados al escenario $e \in E$, es decir, $N_{ie} = \{j \in J \mid d_{ij} < S_{ie}\}$ y $K_{ie} = \{j \in J \mid d_{ij} = S_{ie}\}$. Se definen estas variables:

- $y_{ie} = \begin{cases} 1, & \text{si } A \text{ capta toda la demanda del cliente } i \in I \text{ en el escenario } e \in E \\ 0, & \text{en caso contrario} \end{cases}$
- $z_{ie} = \begin{cases} 1, & \text{si la demanda del cliente } i \in I \text{ se reparte entre } A \text{ y } B \text{ en el escenario } e \in E \\ 0, & \text{en caso contrario} \end{cases}$

Notemos que, si solo difiere la demanda en cada uno de los escenarios, pueden utilizarse directamente en los modelos siguientes los conjuntos N_i y K_i, así como las variables y_i y z_i definidas en el modelo MAXCAP. Por otro lado, si solo se modifican las localizaciones de B, entonces puede utilizarse directamente la demanda a_i del modelo MAXCAP.

2.3.1. Criterio de la máxima mínima captación (Maxmin criterion)

Con este modelo se trata de determinar la ubicación de las instalaciones con objeto de maximizar la demanda mínima captada. La formulación del modelo es:

$$\max \quad g \tag{2.4a}$$

sujeto a

$$\sum_{i \in I} a_{ie} y_{ie} + \sum_{i \in I} \frac{a_{ie}}{2} z_{ie} \geqslant g \qquad \forall e \in E \tag{2.4b}$$

$$y_{ie} \leqslant \sum_{j \in N_{ie}} x_j \qquad \forall i \in I, \forall e \in E \tag{2.4c}$$

$$z_{ie} \leqslant \sum_{j \in K_{ie}} x_j \qquad \forall i \in I, \forall e \in E \tag{2.4d}$$

$$y_{ie} + z_{ie} \leqslant 1 \qquad \forall i \in I, \forall e \in E \tag{2.4e}$$

$$\sum_{j \in J} x_j = p \tag{2.4f}$$

$$g \geqslant 0 \tag{2.4g}$$

$$x_j, y_{ie}, z_{ie} \in \{0,1\} \qquad \forall i \in I, \forall j \in J, \forall e \in E \tag{2.4h}$$

La función objetivo (2.4a) representa la captación mínima, de acuerdo con las restricciones (2.4b), que garantizan que la demanda captada por A en el escenario $e \in E$ es al menos igual a g. Las restricciones (2.4c)-(2.4e) son análogas a (2.2b)-(2.2d), respectivamente, considerando ahora los escenarios. La restricción (2.4f) fuerza a que el número de instalaciones a localizar por la empresa A sea p en cualquiera de los escenarios. Las restricciones (2.4g) y (2.4h) establecen las condiciones de las variables involucradas.

2.3.2. Criterio del mínimo arrepentimiento (Regret criterion)

Una vez que la empresa ha decidido dónde localizar las instalaciones, como consecuencia del escenario que finalmente se presente, tendrá una determinada captación de la demanda. Sin embargo, si hubiera sabido cuál iba a ser el escenario, podría haber decidido dónde localizar las instalaciones y, en general, esta captación hubiera sido mayor. El objetivo de este modelo es minimizar esa sensación de arrepentimiento por no haber elegido las mejores localizaciones de acuerdo con el escenario que al final

se presenta. Este modelo trata de encontrar una estrategia de localización robusta frente a los escenarios posibles, minimizando la mayor de las desviaciones de la optimalidad. Para ello, en primer lugar se resuelven, para cada escenario, el correspondiente modelo MAXCAP. Sea Z_e el valor óptimo de su función objetivo bajo el escenario $e \in E$. El modelo UMAXCAP, con el criterio del mínimo arrepentimiento tiene la siguiente formulación:

$$\min \quad U \tag{2.5a}$$

sujeto a

$$Z_e - \sum_{i \in I} a_{ie} y_{ie} - \sum_{i \in I} \frac{a_{ie}}{2} z_{ie} \leqslant U \qquad \forall e \in E \tag{2.5b}$$

$$U \geqslant 0 \tag{2.5c}$$

$$(2.4c) - (2.4f), (2.4h) \tag{2.5d}$$

La función objetivo (2.5a), en combinación con las restricciones (2.5b), representa que el arrepentimiento es al menos la diferencia entre la captación óptima que puede lograrse para cada escenario $e \in E$ y la captación lograda con las localizaciones elegidas. La restricción (2.5c) indica cómo ha de ser la variable U. El resto de las restricciones han sido ya explicadas en el modelo anterior.

2.4. Localización competitiva y precios

Los modelos presentados en los apartados anteriores suponen que la atracción que una instalación ejerce sobre el cliente depende exclusivamente de la distancia a la que se encuentra: una instalación captura un cliente si no hay otra instalación más cerca de él. En los modelos siguientes, la empresa A busca competir por el mercado actualmente ocupado por la empresa B, utilizando la ubicación de las instalaciones y el precio del producto como estrategias para maximizar las ganancias. Supondremos que el cliente elige la instalación más barata, teniendo en cuenta que el precio utilizado para la comparación es una función del precio de fábrica que fija la empresa y de la distancia que el cliente debe recorrer para llegar a la instalación. Estos modelos son notablemente más complejos, en particular, porque son modelos de programación no lineal entera mixta.

2.4.1. Problema de máxima captación con precios (PMAXCAP)

En el **problema de máxima captación con precios** (en adelante PMAXCAP, por sus siglas en inglés de "Maximum Capture Problem With Prices") [18], que también extiende el modelo MAXCAP, se asumen las siguientes hipótesis:

- La competencia presente en el mercado se conoce y es fija.
- La venta del producto es homogénea entre instalaciones de las distintas empresas (no existen diferencias entre las propias instalaciones).
- Ambas empresas tienen una política de precios de fábrica.
- La decisión de la clientela se basa en el precio total que incluye el coste de desplazamiento y el coste del producto.

Sea v^H el precio fijado por la empresa H, con $H \in \{A, B\}$. Sea β el coste unitario de transporte asumido por la clientela. Sea S_i^H la distancia entre el cliente $i \in I$ y la instalación más cercana de la empresa H, con $H \in \{A, B\}$. El precio total del producto para el cliente $i \in I$ se calcula como $v^H + \beta S_i^H$, $H \in \{A, B\}$. La empresa A capta la demanda a_i del cliente $i \in I$ si se verifica que $v^A + \beta S_i^A < v^B + \beta S_i^B$, es decir, el precio total al que se enfrenta un cliente en una instalación de A es mejor que en una instalación de B. En otro caso, el cliente $i \in I$ continúa fiel a la empresa B. Se supone, además, que cada localización $j \in J$ tiene asociados unos costes fijos f_j de instalación y unos costes variables φ por unidad de producto

fabricada. El objetivo de la empresa A es determinar dónde deben ubicarse las instalaciones y que precio de fábrica debe fijarse para maximizar el beneficio. El problema de máxima captación con precios se formula matemáticamente como:

$$\max \quad (v^A - \varphi)\sum_{i\in I} a_i y_i - \sum_{j\in J} f_j x_j \tag{2.6a}$$

sujeto a

$$y_i \leqslant \sum_{j\in N_i(v^A)} x_j \qquad \forall i \in I \tag{2.6b}$$

$$v^A \geqslant 0 \tag{2.6c}$$

$$x_j, y_i \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{2.6d}$$

donde $N_i(v^A) = \{j \in J \mid v^A + \beta d_{ij} < v^B + \beta S_i^B\}$ depende no solo de las distancias a las instalaciones más cercanas actualmente sino también del precio. La función objetivo (2.6a), que es no lineal, representa el beneficio total como la suma de los ingresos por las ventas $v^A \sum_{i\in I} a_i y_i$ menos los costes totales $\varphi \sum_{i\in I} a_i y_i + \sum_{j\in J} f_j x_j$. El conjunto de restricciones (2.6b) son semejantes a las restricciones (2.2b) del modelo MAXCAP. No obstante, a diferencia de estas, en las restricciones (2.6b) el conjunto $N_i(v^A)$ no es conocido a priori ya que depende del precio de fábrica, que es una variable. Las restricciones (2.6c) y (2.6d) establecen las condiciones de las variables involucradas. Dada la complejidad del modelo, en [18] se propone para su resolución un algoritmo heurístico que combina un procedimiento 1-opt para obtener ubicaciones con un método de búsqueda para encontrar precios.

2.4.2. Problema de máximo cubrimiento con precios (PMAXCOV)

El **problema de máximo cubrimiento con precios** (en adelante PMAXCOV, por sus siglas en inglés de "Maximal Covering Location Problem With Prices Decision") [12] extiende el modelo MAXCOV incorporando en la modelización la decisión sobre el precio de fábrica del producto. En el modelo, un cliente es cubierto por una instalación específica siempre que esta esté situada suficientemente cerca (dentro de la distancia establecida) y el precio total sea el menor. Las hipótesis consideradas en el modelo corresponden a las descritas para el PMAXCAP. El objetivo de la empresa A es determinar dónde localizar p instalaciones y qué precio de fábrica debe fijarse para maximizar los ingresos.

Las variables w_i introducidas en el modelo MAXCOV se referirán ahora a que el cliente $i \in I$ está cubierto por alguna instalación de la empresa A. En el modelo presentado en [12], que será el que se formule en este apartado, se supone que todas las instalaciones tienen la misma estructura de costes y por tanto no es necesario introducir los costes fijos y variables de producción considerados en el modelo PMAXCAP; no obstante, la generalización a este caso sería inmediata. Finalmente, en este modelo, se supone que los clientes en caso de empate se inclinarán por la novedad, es decir, elegirán la empresa A. Se define el *precio crítico*, denotado por v_{ij}, $i \in I, j \in J$, como

$$v_{ij} = v^B + \beta S_i^B - \beta d_{ij} = v^B + \beta(S_i^B - d_{ij})$$

Notemos que si el precio que fija la empresa A verifica que $v^A \leqslant v_{ij}$ entonces el cliente $i \in I$ preferirá la instalación $j \in J$, es decir, es una cota sobre el precio v^A que no debe superarse si la instalación de $j \in J$ va a cubrir al cliente $i \in I$. En consecuencia, el modelo PMAXCOV se formula como:

$$\max \quad \sum_{i\in I} a_i v^A w_i \tag{2.7a}$$

sujeto a

$$w_i \leqslant \sum_{j\in \widetilde{N}_i(v^A)} x_j \qquad \forall i \in I \tag{2.7b}$$

$$\sum_{j\in J} x_j = p \tag{2.7c}$$

$$v^A \geqslant 0 \tag{2.7d}$$
$$x_j, w_i \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{2.7e}$$

donde los conjuntos $\widetilde{N}_i(v^A) = \{j \in J \mid v^A \leqslant v_{ij},\ d_{ij} \leqslant D\}$ dependen también del precio. La función objetivo (2.7a), que es no lineal, representa los ingresos totales de la empresa A. Las restricciones (2.7b) garantizan el cubrimiento de los clientes siempre que se cumplan las condiciones de precio total y distancia. La restricción (2.7c) garantiza que se localizan p instalaciones y las restricciones (2.7d) y (2.7e) establecen las condiciones de las variables. Se puede añadir una restricción adicional $v^A \geqslant v_{min}$ que garantice que el precio de fábrica cubre al menos los costes de la empresa, para evitar que la empresa opere con pérdidas.

Algoritmo 2 Algoritmo de enumeración completa para PMAXCOV

Input: P: Lista de precios v_{ij} ordenada de forma creciente
Output: $\overline{\Pi}^*$: Mejor valor de ingresos totales para A

```
1: Π̄* ← 0
2: for v in P do
3:     CP*(v) ← MAXCOV(v)
4:     Π*(v) ← v · CP*(v)
5:     if Π*(v) > Π̄* then
6:         Π̄* ← Π*(v)
```

En [12] se proponen dos algoritmos exactos de tipo enumerativo para resolver el problema. El primero de ellos, del que se presenta el pseudocódigo en Algoritmo 2 está basado en las siguientes propiedades que tienen en cuenta que si se fija el precio v^A el problema que resulta es un MAXCOV. Dado un precio v, sea $CP(v)$ el problema de máximo cubrimiento en el que un cliente es cubierto por la instalación $j \in J$ si esta está dentro del límite de distancia y tiene un menor coste total. Sea $CP^*(v)$ el valor óptimo de su función objetivo.

Propiedad 1. *Si $v < v'$, entonces $N_i(v') \subset N_i(v)$ para todo $i \in I$.*

Demostración. De la definición de los conjuntos $N_i(v)$ se sigue que si $j \in N_i(v')$ entonces $v_{ij} \geqslant v' \geqslant v$. Por tanto, $j \in N_i(v)$. □

Propiedad 2. *Si $v < v'$, entonces cualquier solución factible de $CP(v')$ es una solución factible de $CP(v)$.*

Demostración. La única diferencia entre los problemas $CP(v)$ y $CP(v')$ está en la definición de los conjuntos $N_i(v)$ y $N_i(v')$. Por la Propiedad 1 se tiene que $\sum_{j \in N_i(v)} x_j \geqslant \sum_{j \in N_i(v')} x_j$, por tanto siempre que $\sum_{j \in N_i(v')} x_j \geqslant y_i$ se tendrá también que $\sum_{j \in N_i(v)} x_j \geqslant y_i$. □

Propiedad 3. *Si $v < v'$, entonces $CP^*(v) \geqslant CP^*(v')$.*

Demostración. La expresión de la función objetivo de los $CP(v)$ y $CP(v')$ es igual. Por la Propiedad 2 la solución óptima para $CP(v')$, cuyo valor de la función objetivo es $CP^*(v')$, es una solución factible para $CP(v)$. Por tanto, el valor óptimo de este problema $CP^*(v) \geqslant CP^*(v')$. □

Sea P el conjunto de todos los precios críticos v_{ij} definido como $P = \{v_{ij} \mid i \in I, j \in J\}$, que se supone ordenado de forma creciente. La siguiente propiedad asegura que, si la nueva empresa A es capaz de cubrir una cierta cantidad de demanda con un precio v', entonces al menos la misma cantidad de demanda se cubrirá con un precio $v < v'$.

Propiedad 4. *Si $v < v'$ y $[v, v') \cap P = \emptyset$, entonces $CP^*(v) = CP^*(v')$.*

Demostración. Cada $v_{ij} \in P$ es, por hipótesis, menor que v o mayor o igual que v'. Por tanto, $N_i(v) = N_i(v')$ para todo $i \in I$ y, en consecuencia, los problemas $CP(v)$ y $CP(v')$ son idénticos. □

Teniendo en cuenta la expresión de la función objetivo del problema PMAXCOV, se concluye que siempre es preferible el precio v' al precio v si $v < v'$. Por tanto, seleccionar v' como el precio crítico más bajo $v_{ij} \in P$ mayor que v muestra que podemos encontrar un precio óptimo (que maximiza los ingresos) mediante la enumeración completa de todos los precios críticos. El procedimiento comienza calculando todos los precios v_{ij} mayores o iguales que v_{min} ordenándolos de forma creciente. A continuación, para cada precio v_{ij} se resuelve el problema MAXCOV correspondiente y se actualiza el mejor valor de ingresos totales si es necesario. El algoritmo termina cuando el conjunto P ha sido analizado exhaustivamente. En consecuencia, las cuatro propiedades anteriores permiten identificar un método finito (debido a la cardinalidad finita del conjunto de precios críticos: $|P| = |I \times J| = |I| \cdot |J| < \infty$), para resolver el problema.

Capítulo 3

Modelado y simulación con datos reales

El objetivo de este capítulo es presentar una aplicación de los modelos lineales desarrollados en el capítulo anterior (MAXCAP, MAXRELOC y UMAXCAP) a un caso real en la ciudad de Zaragoza, en el que una nueva empresa desea posicionarse estratégicamente en el mercado compitiendo con una cadena de cierto tipo de establecimiento comercial ya presente en varios puntos de la capital aragonesa. En esta sección se van a resolver los problemas usando el motor de optimización matemática Gurobi.

Para ello, es necesario conocer los clientes existentes, el número de instalaciones que se desea localizar y sus ubicaciones posibles, la demanda de los clientes y las distancias entre los clientes y tanto las instalaciones actuales como las posibles localizaciones para la nueva empresa. En este caso, se ha analizado la localización de p tiendas que pueden ser ubicadas en algunos de los 25 locales disponibles seleccionados. Los clientes considerados corresponden con las 491 secciones censales presentes en la ciudad y la demanda asociada a cada uno se define como la población presente en la sección.

3.1. Obtención de los datos

En primer lugar, se han recopilado los datos de las secciones censales. La información necesaria se ha obtenido a través de la página web del Ayuntamiento de Zaragoza dentro del área "Datos Abiertos: Secciones Censales". Es posible descargar un fichero de tipo *.json* que almacena el código identificativo, las coordenadas UTM y la población de cada una de las secciones censales de la ciudad. Este archivo puede integrarse fácilmente en una hoja *Excel* y sus datos corresponden a la versión más reciente disponible del año 2019. Las coordenadas UTM han sido transformadas a coordenadas geográficas mediante una plantilla de *Excel* que puede ser descargada desde el siguiente link: `https://acolita.com/wp-content/uploads/2017/08/conversor_coordenadas.xlsx`.

En lo que respecta a las localizaciones de las instalaciones presentes en el mercado, se han seleccionado 9 establecimientos de cierto tipo, distribuidos estratégicamente en la ciudad. Sus coordenadas geográficas han sido extraídas manualmente mediante la herramienta *Google Maps* a la que se puede acceder en `https://www.google.es/maps`.

Por otro lado, todas las ubicaciones posibles para las nuevas instalaciones se han obtenido a través de varios portales inmobiliarios online. Se han buscado 25 locales comerciales disponibles con características similares a los de la cadena establecida, tratando de cubrir la mayoría de barrios del casco urbano. También las coordenadas geográficas se han extraído con *Google Maps*.

Una vez recopilada la información anterior en una hoja *Excel* y tratada adecuadamente, se extraen los archivos *.csv* necesarios para su posterior importación en la hoja de trabajo de *Google Colab*, donde se escribe el código *Python* para definir y resolver los modelos matemáticos con el paquete Gurobi.

Finalmente se debe calcular la matriz de distancias necesaria para los modelos. Para ello se ha hecho uso de *OpenRouteService*, una herramienta online de enrutamiento y análisis de información geoespacial basada en los datos proporcionados por el servidor cartográfico *OpenStreetMaps* y desarrollado por el "Heidelberg Institute for Geoinformation Technology" (HeiGIT). Mediante la instalación de una API (interfaz de programación de aplicaciones), podemos integrar esta funcionalidad en la hoja de trabajo

para realizar cálculos directamente sobre nuestros datos. Las limitaciones que presenta la versión gratuita utilizada, hacen que se deba dividir la matriz de distancias de dimensión 34×491 (correspondiente a las ubicaciones, tanto las disponibles como las ocupadas actualmente, en las filas y a los clientes en las columnas) en 5 submatrices de menor dimensión (cubriendo cada parte de las ubicaciones, de manera que así todas ellas se pueden calcular). Posteriormente, se incorporan todas juntas en otra hoja de *Excel* y se crea otro archivo *.csv* con la matriz de distancias definitiva (que también debe ser importada en *Google Colab*). Para generar las matrices en *OpenRouteService* debe indicarse el origen de los datos (los archivos *.csv* importados previamente) y el tipo de cálculo (la distancia a través de la ruta más corta a pie, en metros, entre todos los posibles pares de coordenadas).

3.2. Implementación de los modelos

Para determinar la solución óptima de los modelos considerados se ha utilizado Gurobi, tal y como se ha mencionado antes, un software que se integra en *Python* y que permite resolver eficazmente problemas expresados como modelos de programación lineal. La versión empleada ha sido Gurobi 10.0.1. Antes de ejecutar los cálculos, se han importado todos los conjuntos de datos necesarios y se ha calculado la matriz de distancias que, por otro lado, ha sido también importada en un fichero. En la figura 3.1 se muestran, sobre un mapa de la ciudad de Zaragoza, con un círculo azul la localización de los clientes, en rojo las instalaciones pertenecientes a la cadena ya instalada y en verde las localizaciones que puede ocupar la nueva empresa.

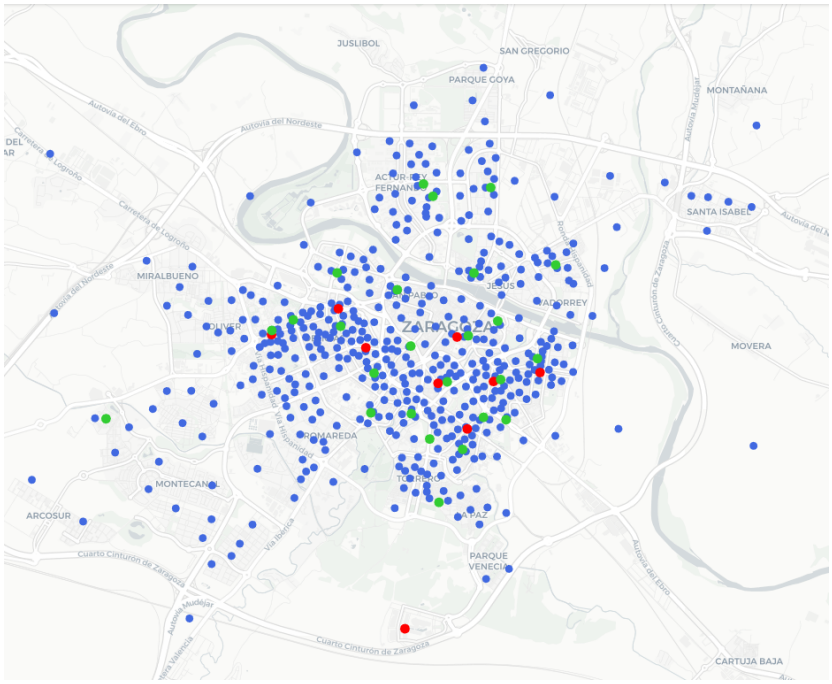

Figura 3.1: Clientes (●), instalaciones de la competencia (●) y ubicaciones disponibles (●)

Para cada modelo se han definido todos los parámetros y conjuntos involucrados en los modelos, siguiendo la estructura de *Python* y haciendo uso de toda la información cargada en el servidor. Finalmente, se han añadido la función objetivo, las variables y las restricciones de los modelos, según las instrucciones de la herramienta Gurobi. Para la representación de todas las soluciones sobre el mapa de Zaragoza se han añadido también unas líneas de código. Las diferentes ejecuciones se han llevado a cabo utilizado un ordenador personal, equipado con un procesador AMD Ryzen™ 3 2200U de 2 núcleos a 2.50 GHz y 8.00 GB de memoria RAM, aunque, en este caso, al utilizar *Google Colab*, los algoritmos se ejecutan utilizando los recursos que proporciona dicha plataforma. Antes de lanzar cada ejecución, se han modificado en el propio código los valores de los parámetros con objeto de poder evaluar y comparar los

resultados obtenidos. En el Apéndice A se adjuntan los mapas correspondientes a los resultados de todos los problemas resueltos que no han sido incluidos en el capítulo actual y el código *Python* empleado para las simulaciones.

3.3. Evaluación de resultados del modelo MAXCAP

El objetivo del modelo MAXCAP es maximizar la captación de clientes para la empresa entrante. Por ello, se ha resuelto el modelo para distintos valores del número de nuevas instalaciones que se desean localizar, $p \in \{1,2,5,9,15\}$, con el fin de analizar cómo afecta dicho valor a la captación obtenida. En la tabla 3.1 se presentan los principales resultados de la simulación. En la primera columna se da el valor de p. La segunda columna muestra el valor óptimo de la función objetivo, es decir, la cantidad de demanda captada con la distribución óptima de instalaciones. La tercera columna muestra el porcentaje de población captado frente al total empadronado en 2019 que son 716576 habitantes. Finalmente, en la cuarta columna se incluye el tiempo empleado por Gurobi, en segundos, para ejecutar los cálculos y hallar la solución óptima.

p	**F. Objetivo**	**% Población**	**Tiempo**
1	158327	22.1	0.19
2	225648	31.5	0.11
5	380094	53.0	0.12
9	497115	69.4	0.09
15	566449	79.0	0.07

Tabla 3.1: Resumen de resultados para el MAXCAP

Analizando los resultados anteriores puede observarse que la relación entre abrir nuevas instalaciones y captar población no resulta lineal. Para $p = 1$ se alcanza más de una quinta parte del total. Añadiendo otra instalación se aumenta la captación en un 42,6 % y llegando hasta $p = 5$ superamos la mitad de la población, lo que supone un aumento del 68,4 % respecto al caso anterior. Resulta evidente que el valor de la función objetivo debe aumentar según se localizan más instalaciones. Sin embargo, pasar de 5 a 9 instalaciones incrementa la demanda captada en menos 120000 personas, lo que supone un $24,2\,\%$ menos que para el paso de 2 a 5 instalaciones. Este hecho se acentúa aún más cuando se pasa de $p = 9$ a $p = 15$: se localizarían 6 instalaciones más y tan solo se incrementa la población captada en un $13,9\,\%$. Se observa, por tanto, que a partir de cierto valor de p la empresa puede considerar razonable no abrir más instalaciones ya que el incremento en el número de clientes captados es pequeño. A continuación se realiza una interpretación más exhaustiva de los resultados obtenidos considerando $p = 2$ y $p = 9$.

3.3.1. Caso $p = 2$

Una de las instalaciones se localiza en el barrio del Arrabal. Resulta razonable que se ubique en la margen izquierda del Ebro en la que se encuentran también otros barrios como Santa Isabel, Actur, La Jota, Parque Goya, San Gregorio y El Picarral porque en ellos se congrega una notable cantidad de demanda y, sin embargo, no hay locales de la competencia, luego la captación está garantizada. La segunda instalación se localiza en el corazón del barrio de La Romareda. A través de ella se consigue captar gran parte de su población además del barrio de Casablanca, parte de Torrero y la mitad este de Valdespartera. Estas localizaciones quedan representadas sobre el mapa en la figura 3.2, con la mismas características de la figura anterior, pero cambiando el color de los clientes captados por la nueva empresa.

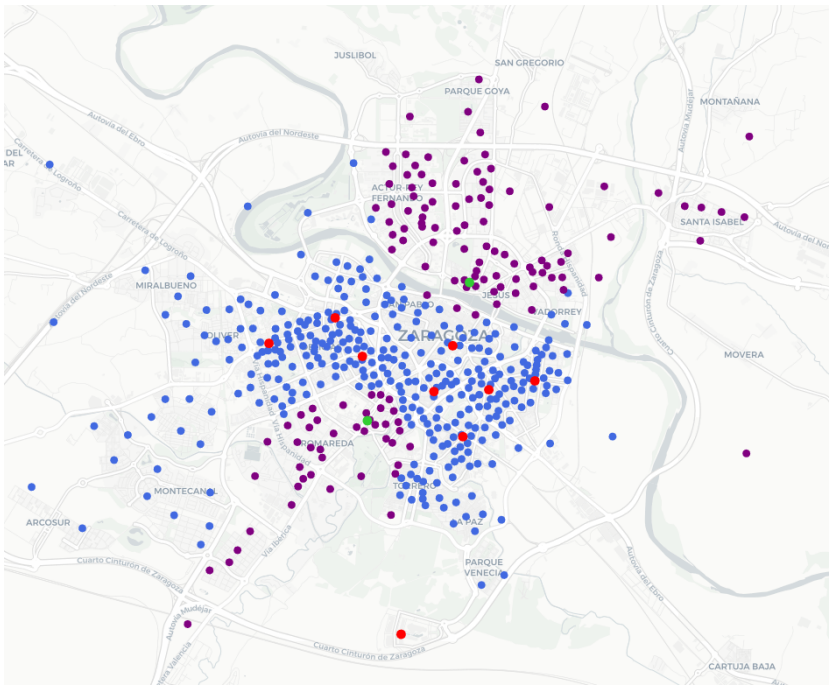

Figura 3.2: MAXCAP, $p = 2$: distribución óptima de instalaciones (●) y clientes captados (●)

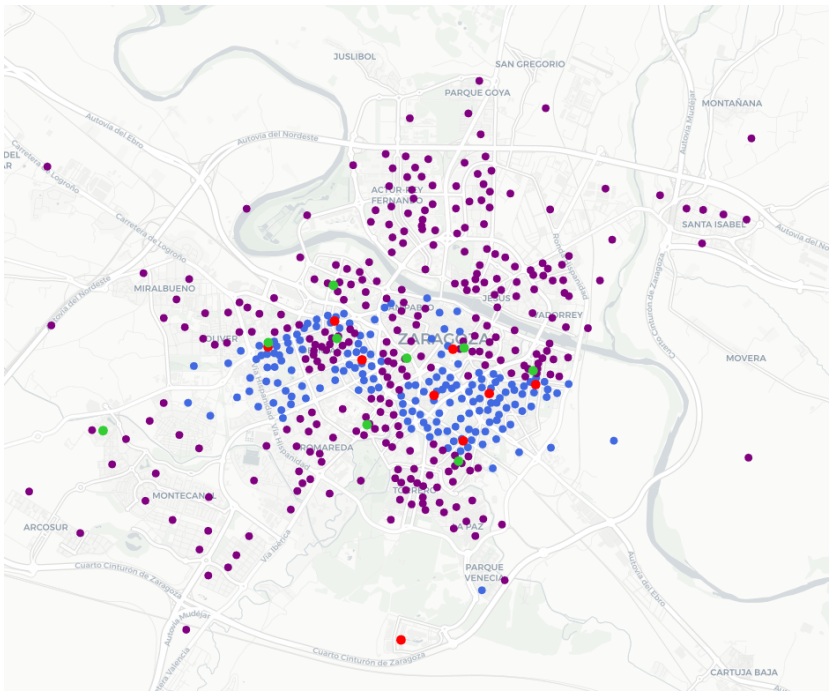

Figura 3.3: MAXCAP, $p = 9$: distribución óptima de instalaciones (●) y clientes captados (●)

3.3.2. Caso $p = 9$

Con este valor de p la nueva empresa iguala en número de instalaciones a la cadena competidora. Algunas de las instalaciones que se localizan para valores más pequeños de p dejan de resultar atractivas porque su demanda pasa a ser captada por otras nuevas que abarcan a más clientes, de modo que la margen izquierda vuelve a quedarse sin establecimientos de ambas cadenas. A la instalación localizada para $p = 2$ en La Romareda se añaden otras dos en el barrio de Delicias, el más poblado de la capital, una

en San José, una en Las Fuentes, otra en La Almozara, dos en el Centro y una última en el joven barrio de Rosales del Canal. Además de la población de estos barrios se consigue también captar la de otros como Torrero, La Paz, Miralbueno, Oliver, Montecanal, Arcosur o La Magdalena. La nueva empresa capta más de la mitad de la población de la ciudad con el mismo número de instalaciones que su competidora. Además, algunas de estas nuevas instalaciones están ubicadas en diferentes barrios, lo que le permite convertirse en una referencia dentro de cada uno de ellos. En la figura 3.3 se representa la distribución de las 9 ubicaciones seleccionadas y los clientes captados por cada empresa.

3.4. Evaluación de resultados del modelo MAXRELOC

El objetivo del modelo MAXRELOC es maximizar la captación de clientes, pero suponiendo ahora que la nueva empresa ya tiene s instalaciones en el mercado localizadas en las ubicaciones del conjunto $J_A \subset J$, de las que quiere relocalizar $r \leqslant s$ y, además, quiere localizar p nuevas instalaciones. El problema se ha resuelto considerando diferentes valores para todos los parámetros anteriores, de manera que $(s,p,r) \in \{(2,3,1),(5,4,2),(9,3,4),(12,1,6)\}$. En la tabla 3.2 se recogen los datos más relevantes de la simulación siguiendo la estructura de la tabla 3.1. Se incorporan la primera y la tercera columna para indicar los valores de s y r, respectivamente.

s	p	r	**F. Objetivo**	**% Población**	**Tiempo**
2	3	1	347 673	48.5	0.11
5	4	2	489 621	68.3	0.12
9	3	4	535 298	74.7	0.09
12	1	6	548 612	76.6	0.08

Tabla 3.2: Resumen de resultados para el MAXRELOC

Analizar cómo afectan los parámetros s, p y r al tipo de solución obtenida requeriría de un diseño del experimento más elaborado, lejos del estudio realizado en el que simplemente se pretende tener una visión inicial. En los dos primeros casos concretos estudiados, con un total de 5 y 9 instalaciones de la nueva compañía, la población captada se queda ligeramente por debajo de la alcanzada para el modelo MAXCAP con esos valores de p, aunque no supone una diferencia significativa. Pese a que existe la opción de relocalizar algunas de las instalaciones localizadas al principio, otras no pueden cambiar de ubicación y, además, no se encuentran en la ubicación óptima, lo que no permite a la nueva empresa captar la máxima demanda posible. Cuando las instalaciones de la nueva empresa son 12 y 13 la población captada es muy similar, a pesar de que en el primer caso se localiza una instalación menos. Notemos que el modelo MAXRELOC tiende a reubicar el mayor número de instalaciones posibles en las ubicaciones óptimas de acuerdo con el modelo MAXCAP. De hecho, esto sucede en los cuatro casos considerados. Esto es consecuencia de que el modelo MAXRELOC es una versión del MAXCAP restringida en ese aspecto puesto no se permite que algunas de las instalaciones puedan moverse a una localización mejor. No obstante, aumenta las posibilidades del análisis al poder ajustarlo a la situación real que tenga una empresa ya establecida. A continuación se realiza una interpretación más detallada de los resultados obtenidos para $(s,p,r) = (5,4,2)$ y $(s,p,r) = (12,1,6)$.

3.4.1. Caso $(s,p,r) = (5,4,2)$

Inicialmente la empresa poseía instalaciones en los barrios del Actur (1), San José (1), El Gancho (1) y La Romareda (2). Unos cambios en el modelo de negocio han obligado a tomar decisiones a la empresa. Dado que han analizado el modelo MAXCAP con 9 localizaciones, saben que las suyas actuales no son las mejores. Consideran abrir 4 nuevas instalaciones pero, sin embargo, solo están dispuestos a reubicar 2 de las 5 actuales. Los resultados indican que se ha de relocalizar la instalación del Actur y la de La Romareda que está situada junto al Parque Pignatelli. Por otro lado, las nuevas localizaciones elegidas abarcan gran cantidad de población con barrios como Delicias, La Magdalena, Las Fuentes, Vadorrey,

Oliver, Miralbueno, Rosales del Canal, etc. Las localizaciones se muestran en la figura 3.4. El círculo naranja representa la localización original y el verde la ubicación óptima. Cuando el círculo tiene ambos colores es una ubicación original que no requiere relocalización y por tanto en la solución óptima no ha cambiado de sitio. Con esta reestructuración la cadena consigue quedarse solo 7 500 habitantes por debajo de la captación óptima para el modelo MAXCAP.

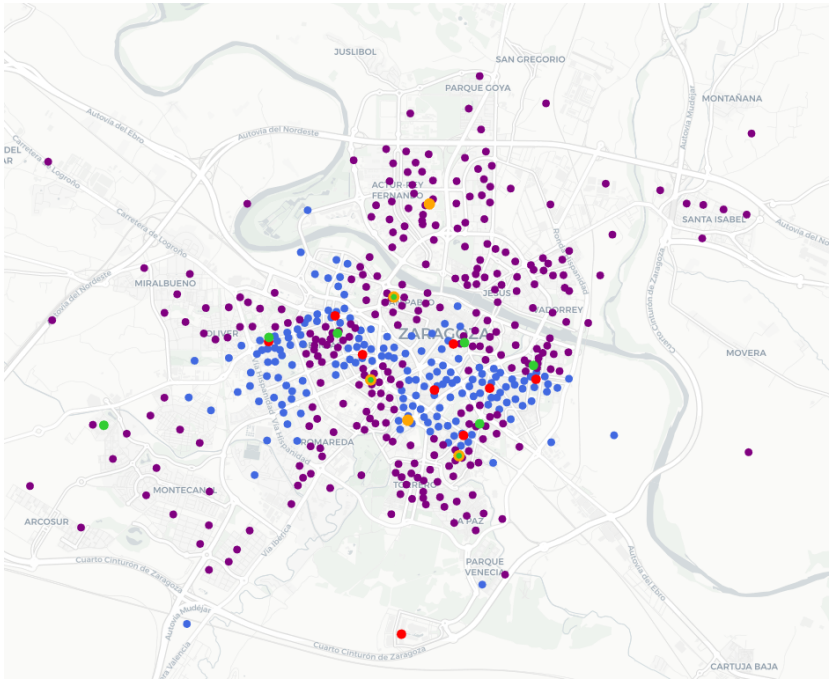

Figura 3.4: MAXRELOC, $(s,p,r) = (5,4,2)$: ubicaciones originales (●), distribución óptima de las instalaciones (●) y clientes captados (●)

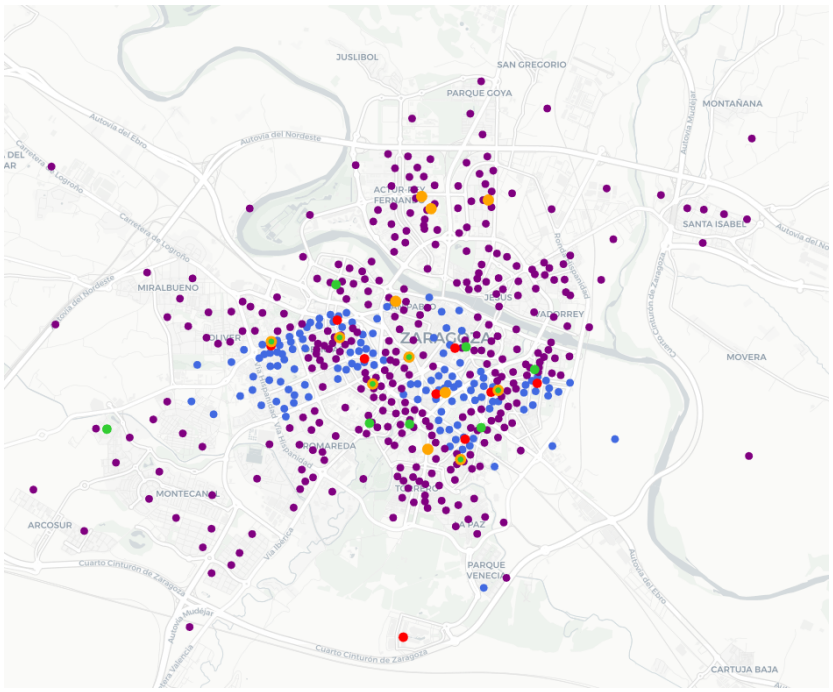

Figura 3.5: MAXRELOC, $(s,p,r) = (12,1,6)$: ubicaciones originales (●), distribución óptima de instalaciones (●) y clientes captados (●)

3.4.2. Caso $(s, p, r) = (12, 1, 6)$

Suponemos que la empresa se ha hecho con instalaciones de otra compañía y ahora tiene 12 instalaciones. Sin embargo, desde la dirección consideran que las ubicaciones no son óptimas y que podrían captar más población, por lo que se plantean relocalizar 6 de ellas. Por otra parte, en sus planes de expansión se incluye también una nueva instalación. Tras resolver el modelo se observan cambios significativos. En el Actur y en El Picarral se pierden los 3 establecimientos que había y también se reubican algunas instalaciones de la zona Centro y uno en la zona La Romareda-San José. La captación alcanza un 76.6 % de la población de la ciudad gracias a la localización de instalaciones en barrios como Las Fuentes, La Almozara, Romareda o Rosales del Canal, este último un importante foco para atraer a la población de todo el Distrito Sur. La distribución resultante pueden verse en el mapa de la figura 3.5. Se observa que con 13 instalaciones se obtiene solo un 3.1 % menos de habitantes que para el caso $p = 15$ en el modelo MAXCAP, lo que nuevamente pone de manifiesto el pequeño aumento de captación que se produce entre diferentes casos con muchas instalaciones.

3.5. Evaluación de resultados del modelo UMAXCAP

El objetivo del modelo UMAXCAP es conocer la ubicación óptima para las p instalaciones de la nueva empresa desde un marco de incertidumbre, ya que no conoce con exactitud la demanda de los clientes y las ubicaciones de la empresa competidora. En las pruebas se han considerado los siguientes tres escenarios:

- $e = 1$: Sólo cambia la demanda potencial. Algunos barrios experimentan un fuerte crecimiento poblacional. Es el caso de Arcosur, Rosales del Canal, Miralbueno, Parque Venecia y Parque Goya. Por otro lado, algunos de los más envejecidos sufren el efecto contrario. Es lo que ocurre con El Arrabal, Las Fuentes y El Gancho.

- $e = 2$: Sólo se modifican las instalaciones de la competencia. Algunos rumores sobre las movimientos estratégicos de la cadena ya instalada en la ciudad hacen pensar que la instalación ubicada actualmente en el Centro Comercial Puerto Venecia pasaría a situarse en el Centro Comercial Grancasa, mientras que la que hoy se ubica en Las Fuentes se espera que cambie su dirección a la principal arteria del barrio de Torrero.

- $e = 3$: Varían tanto la demanda como las instalaciones de la competencia. Ahora se consideran de manera conjunta las variaciones descritas para $e = 1$ y $e = 2$.

Dado que se han generado diferentes escenarios posibles, ha sido necesario obtener los nuevos datos (con ficheros, tanto de clientes con sus demandas como de localizaciones de las instalaciones de la competencia, y con otra matriz de distancias) e importarlos en *Google Colab*, además de definir todos los conjuntos y variables necesarios para integrar correctamente el modelo en el código *Python*. La tabla 3.3 presenta los resultados obtenidos al resolver el modelo UMAXCAP con los tres escenarios anteriores, para distintos valores de p y los dos criterios estudiados, maxmin y mínimo arrepentimiento (regret).

	Maxmin criterion		Regret criterion	
p	F. Objetivo	Tiempo	F. Objetivo	Tiempo
1	96 121	0.09	10 084	0.10
2	161 625	0.15	11 943	0.12
5	313 001	0.12	28 038	0.40
9	440 450	0.07	21 006	0.17
15	530 105	0.09	9 248	0.07

Tabla 3.3: Resumen de resultados para el UMAXCAP

Para el criterio maxmin, se observa en la tabla 3.3 que el aumento de la captación no es lineal respecto al aumento en el número p de nuevas instalaciones localizadas. Para valores pequeños, las cuotas de mercado van aumentando significativamente, mientras que para valores más altos las diferencias resultan poco relevantes. Respecto al criterio del mínimo arrepentimiento, tampoco parece existir relación entre los valores de p y el arrepentimiento (diferencia entre la captación óptima que puede lograrse para cada escenario $e \in E$ y la captación lograda con las localizaciones elegidas) en la captación de demanda. El valor más alto se da para $p = 5$, mientras que el menor corresponde a $p = 15$, dándose una diferencia del 67 % entre ambos valores.

A continuación, se estudia con más detalle el caso $p = 9$. Para el criterio de la máxima mínima captación, las ubicaciones elegidas se distribuyen por gran parte de la ciudad de Zaragoza. Se localizan dos instalaciones en el barrio de Delicias, una en el núcleo del barrio y otra cerca de Los Enlaces, y otras dos en el Centro de la ciudad, una muy cerca del barrio de La Almozara y la otra casi en La Magdalena. Se localiza una única instalación en la margen izquierda del Ebro, Las Fuentes, La Romareda, Torrero y Rosales del Canal. La figura 3.6 muestra las localizaciones seleccionadas.

Para el criterio del mínimo arrepentimiento varias de las instalaciones se ubican en localizaciones diferentes. Las que el criterio anterior situaba en el Centro, ahora las aleja de él, llevándolas directamente a los barrios contiguos: La Almozara y La Magdalena. Además, la instalación de La Magdalena capta también la población de Las Fuentes, porque la instalación que el criterio maxmin situó en este barrio ya no se contempla. La instalación de Torrero tampoco se considera, ya que resulta más conveniente emplazarla en San José. Con este criterio sí se mantienen las instalaciones en el Actur, La Romareda, Rosales del Canal y las dos de Delicias que se localizaban con el criterio maxmin. La última instalación se localiza cerca de la Puerta del Carmen, para captar la demanda del Centro de la ciudad. Las localizaciones seleccionas se muestran en la figura 3.7.

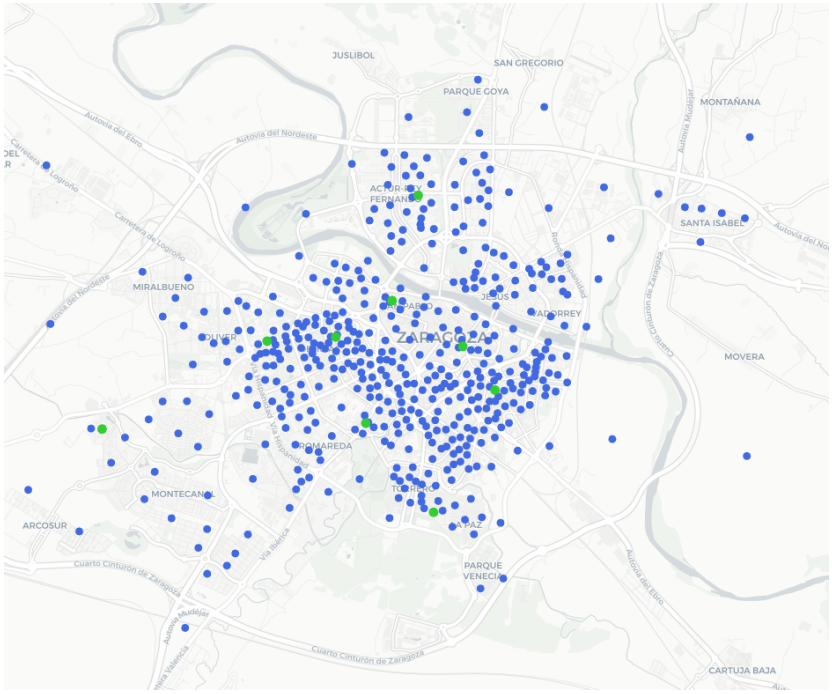

Figura 3.6: UMAXCAP, criterio maxmin, $p = 9$: distribución óptima de instalaciones (●) y clientes (●)

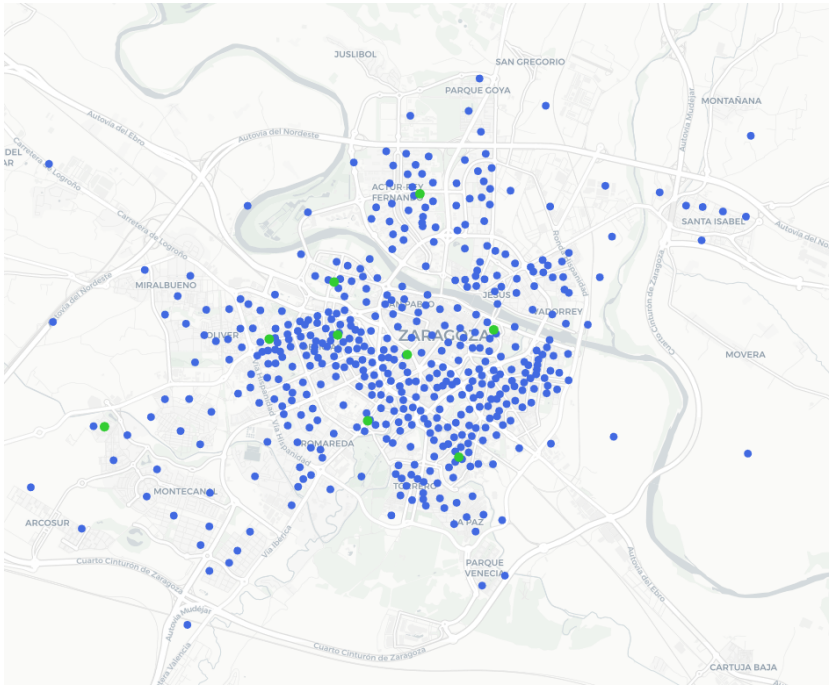

Figura 3.7: UMAXCAP, criterio regret, $p = 9$: distribución óptima de instalaciones (●) y clientes (●)

3.6. Conclusiones del estudio

El estudio realizado ha permitido analizar, desde diferentes perspectivas, algunos problemas sobre localización de instalaciones competitivas en la ciudad de Zaragoza. Se han considerado dos cadenas comerciales, una ya presente en el mercado y otra nueva. Se han utilizado las secciones censales de la ciudad como clientes, para la empresa competidora se ha considerado cierto tipo de establecimientos comerciales y las localizaciones disponibles se han tomado de diferentes portales inmobiliarios.

Se han aplicado los modelos estáticos lineales presentados en el segundo capítulo para obtener las ubicaciones óptimas que permitieran a la nueva empresa maximizar su captación de demanda para distintos valores de los parámetros. Aunque con el estudio experimental realizado no pueden establecerse conclusiones generales, sí que permiten concluir, por ejemplo, que aunque aumentar el número de instalaciones aumenta la demanda captada, el porcentaje de incremento se reduce considerablemente a partir de un cierto valor de p.

Por otro lado, la relocalización de instalaciones y la incertidumbre en los parámetros han sido dos características consideradas que dotan al estudio de una mayor perspectiva y rigor, permitiendo ajustar más a la realidad los modelos simulados. La nueva empresa podría utilizar la información obtenida para tomar decisiones sobre su entrada en el mercado, el propósito principal de la teoría de localización de instalaciones.

Apéndice A

Mapas de resultados y código Python

A.1. Resultados del modelo MAXCAP

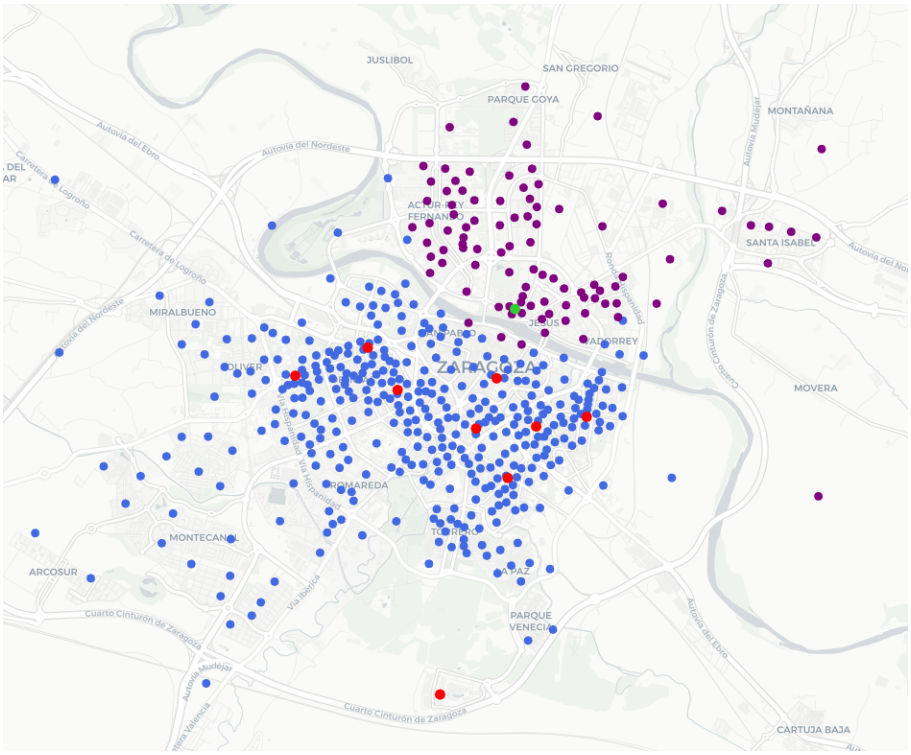

Figura A.1: MAXCAP, $p = 1$: distribución óptima de instalaciones (●) y clientes captados (●)

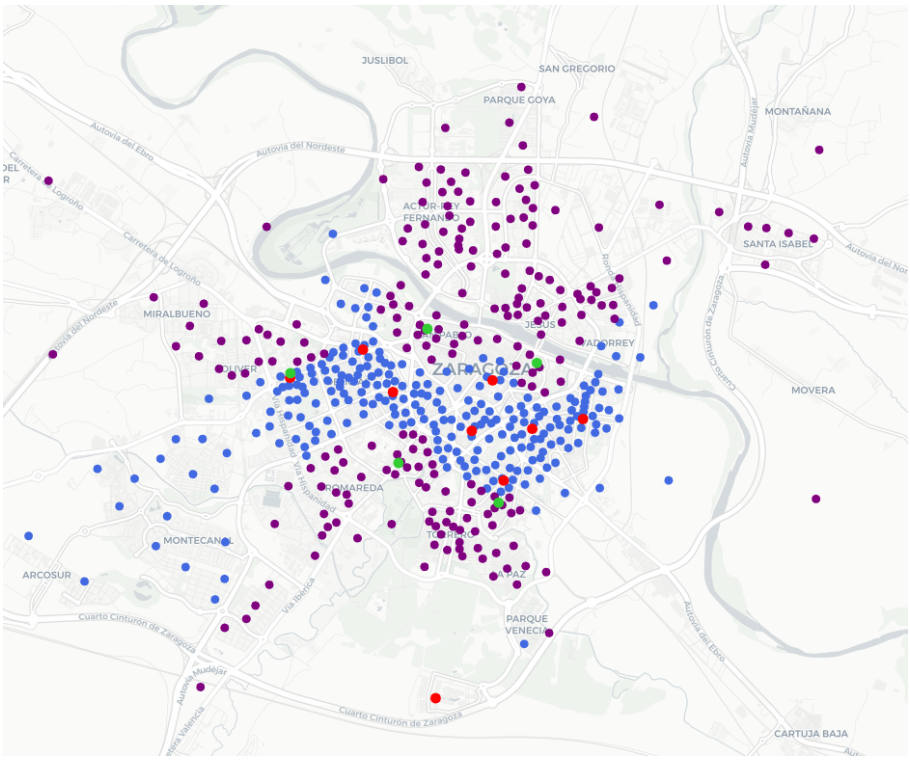

Figura A.2: MAXCAP, $p = 5$: distribución óptima de instalaciones (●) y clientes captados (●)

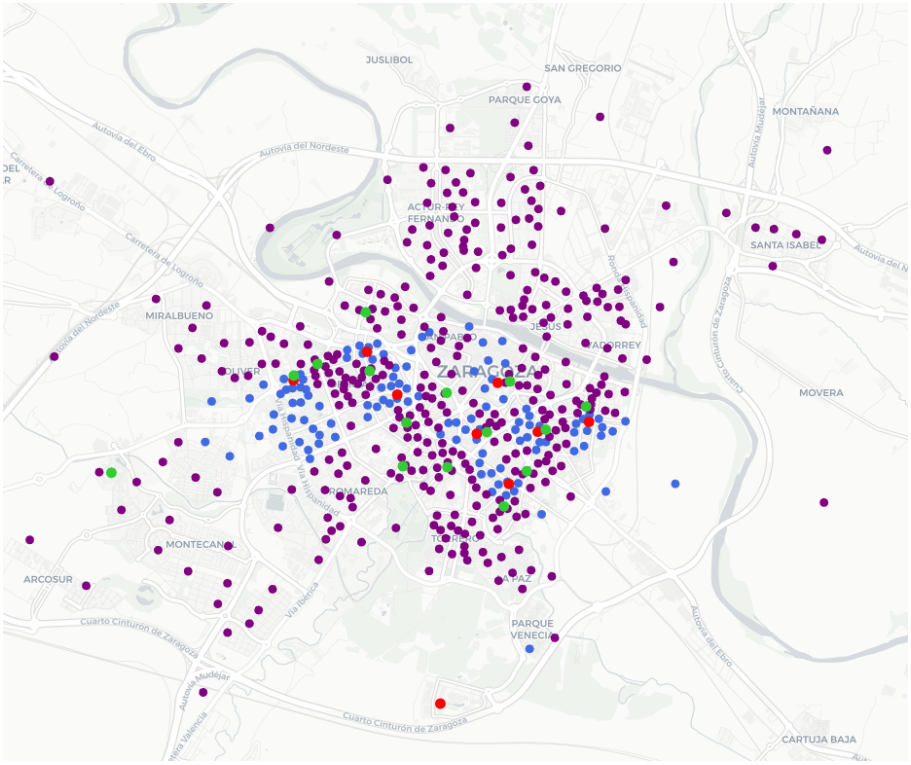

Figura A.3: MAXCAP, $p = 15$: distribución óptima de instalaciones (●) y clientes captados (●)

A.2. Resultados del modelo MAXRELOC

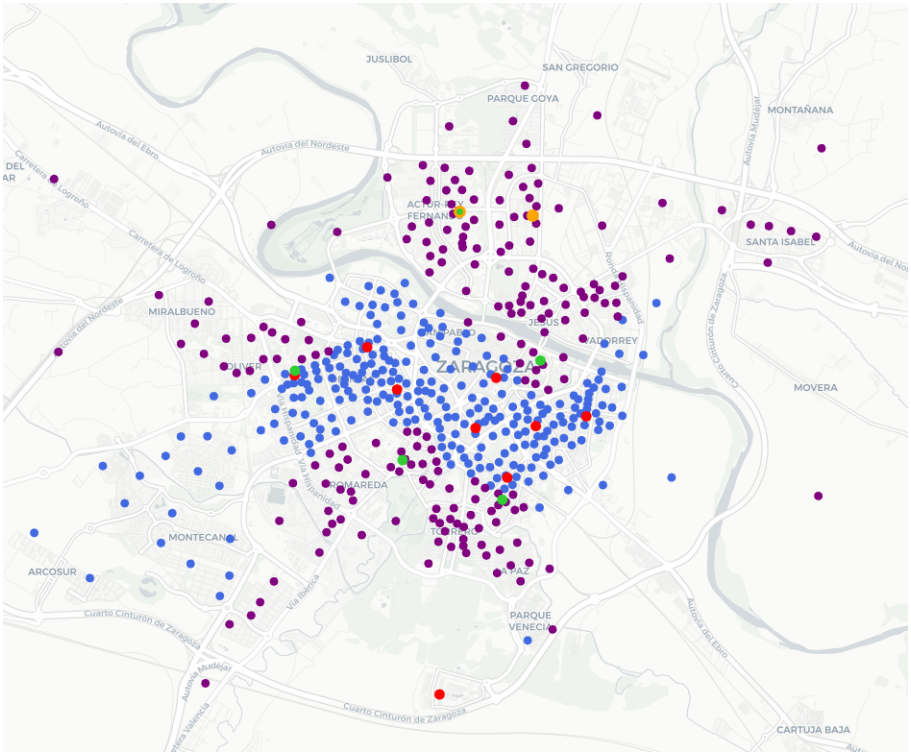

Figura A.4: MAXRELOC, $(s, p, r) = (2, 3, 1)$: ubicaciones originales (●), distribución óptima de instalaciones (●) y clientes captados (●)

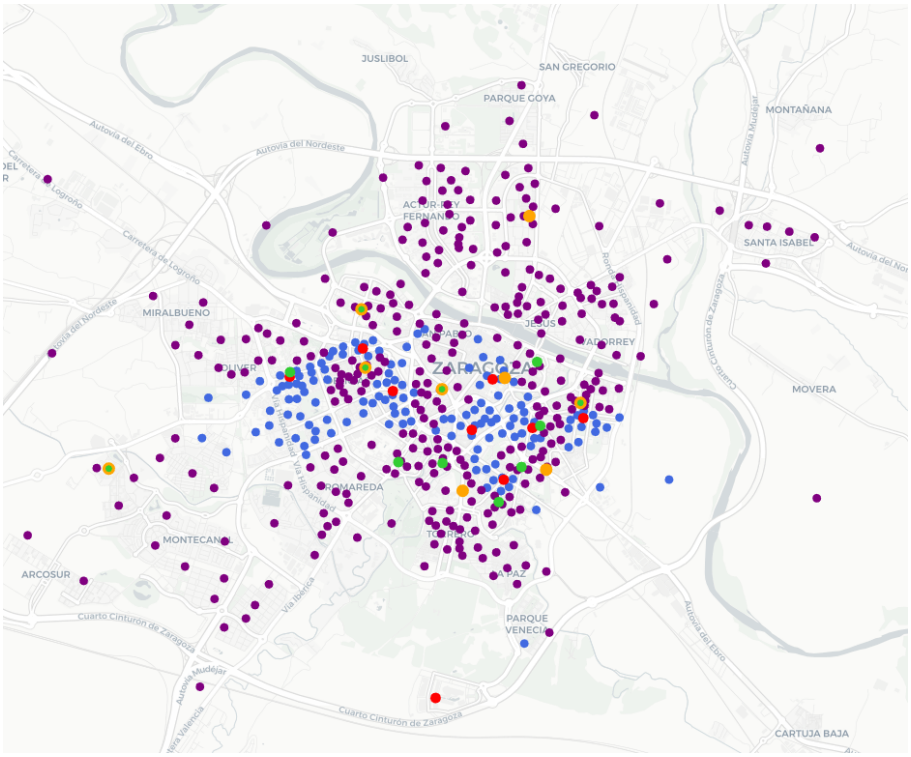

Figura A.5: MAXRELOC, $(s, p, r) = (9, 3, 4)$: ubicaciones originales (●), distribución óptima de instalaciones (●) y clientes captados (●)

A.3. Resultados del modelo UMAXCAP

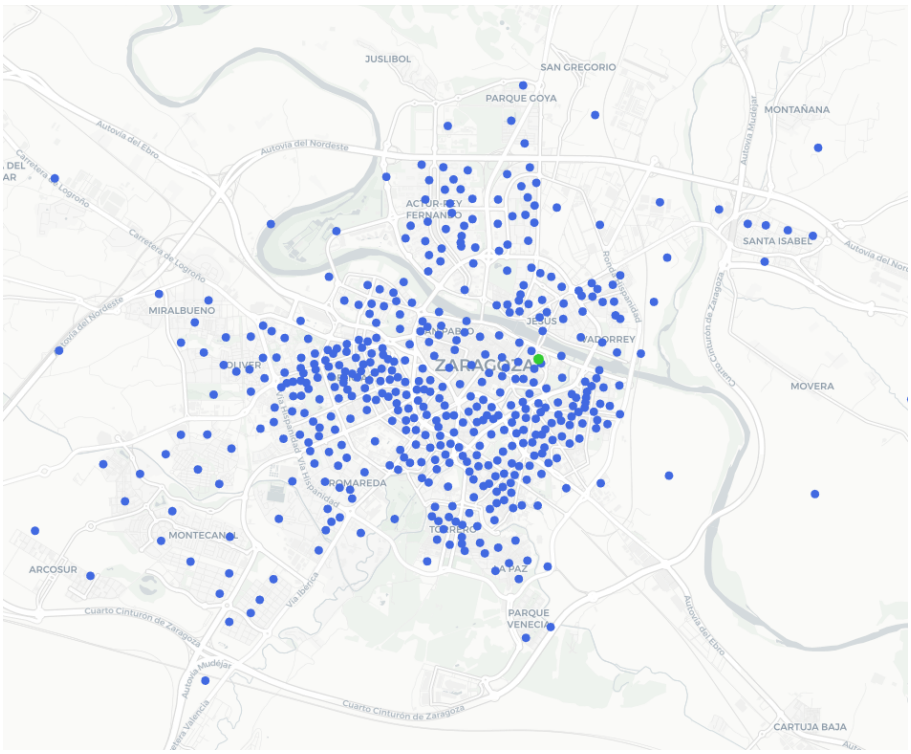

Figura A.6: UMAXCAP, criterio maxmin, $p = 1$: distribución óptima de instalaciones (●) y clientes (●)

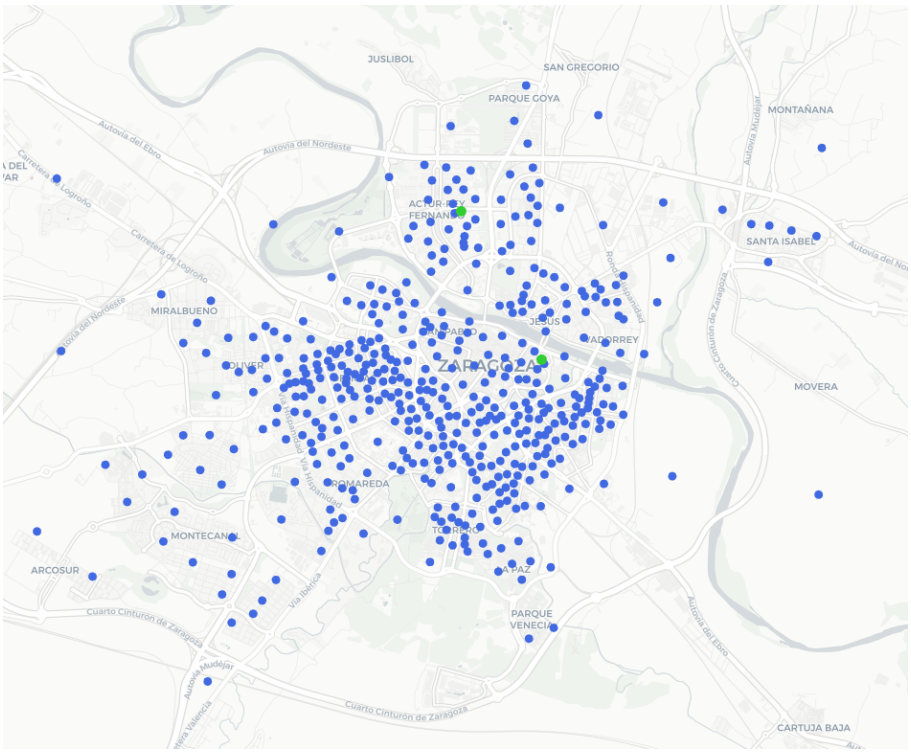

Figura A.7: UMAXCAP, criterio maxmin, $p = 2$: distribución óptima de instalaciones (●) y clientes (●)

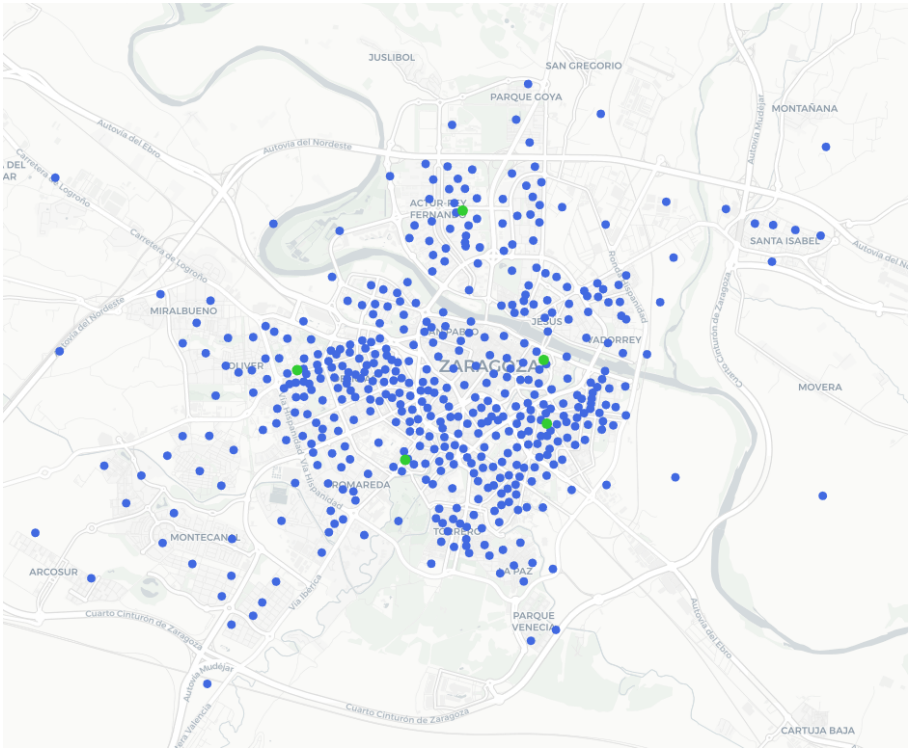

Figura A.8: UMAXCAP, criterio maxmin, $p = 5$: distribución óptima de instalaciones (●) y clientes (●)

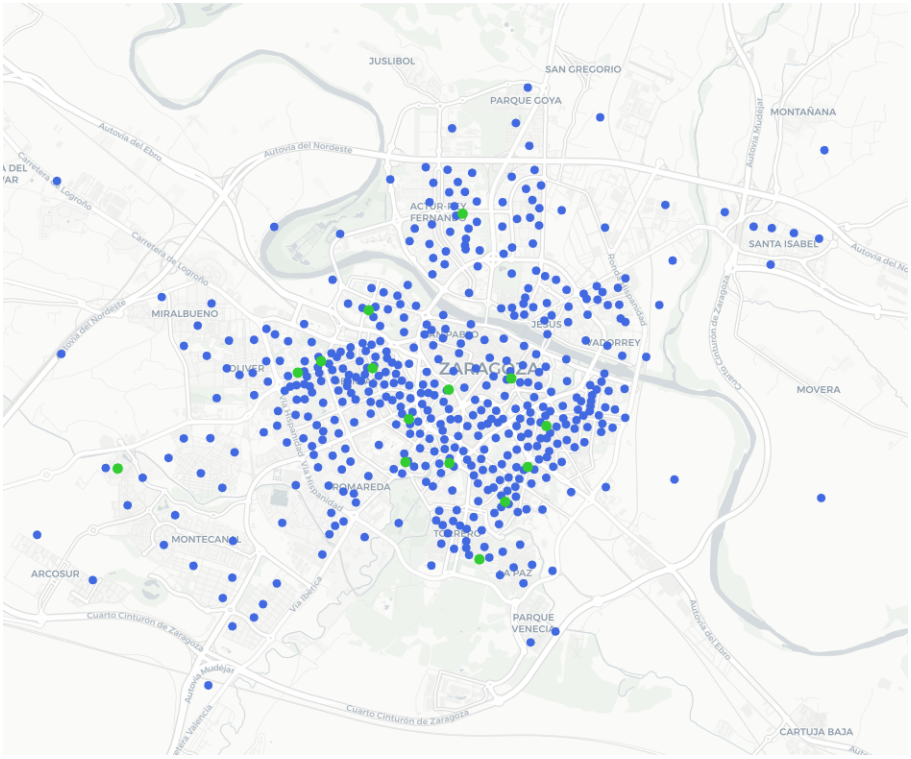

Figura A.9: UMAXCAP, criterio maxmin, $p = 15$: distribución óptima de instalaciones (●) y clientes (●)

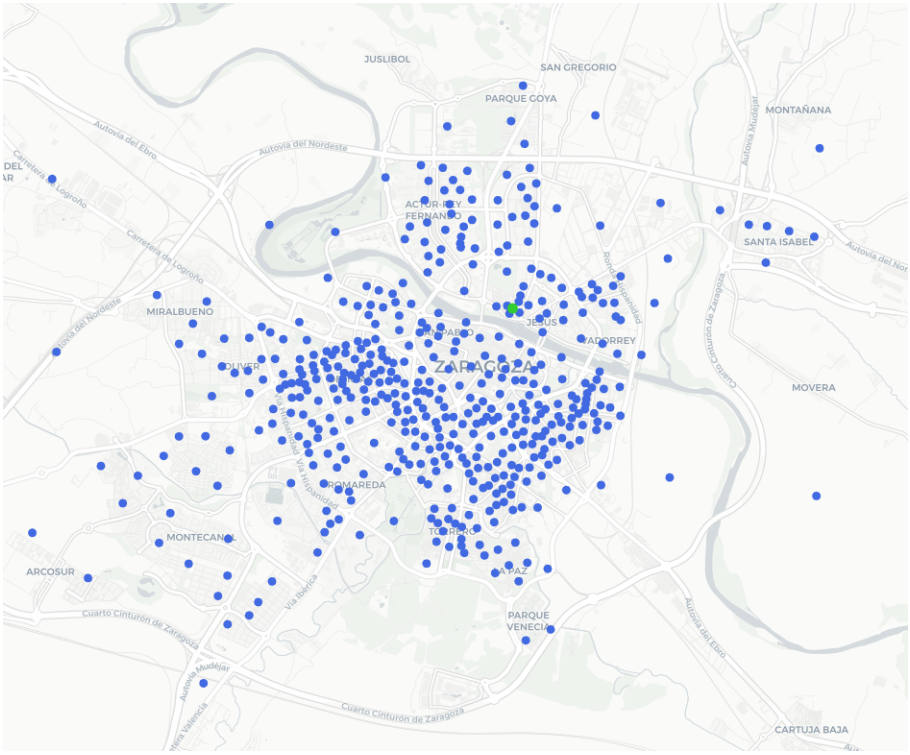

Figura A.10: UMAXCAP, criterio regret, $p = 1$: distribución óptima de instalaciones (●) y clientes (●)

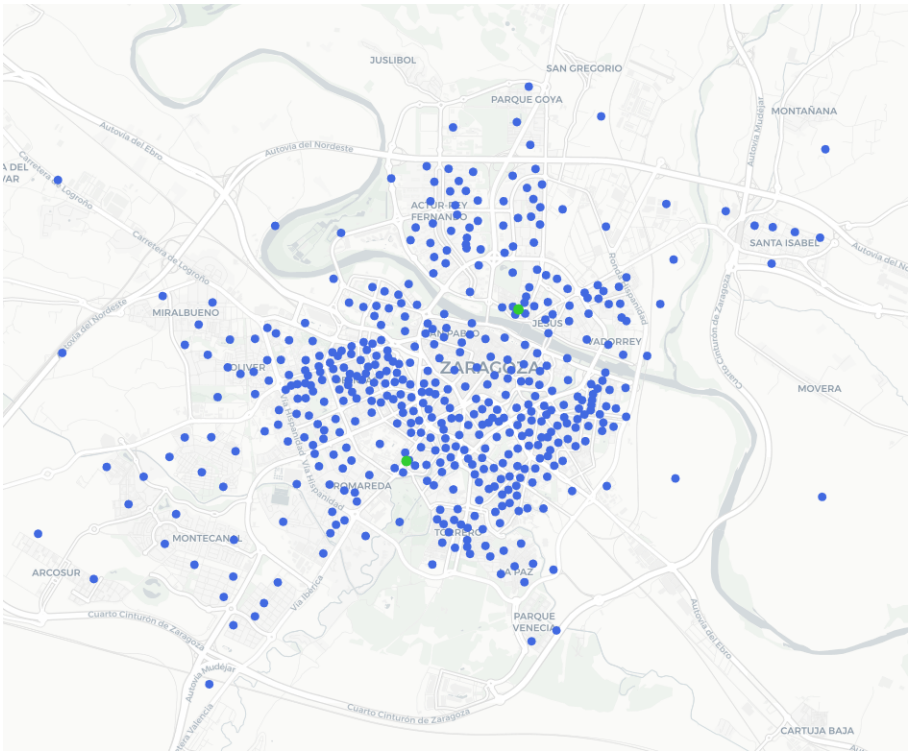

Figura A.11: UMAXCAP, criterio regret, $p = 2$: distribución óptima de instalaciones (●) y clientes (●)

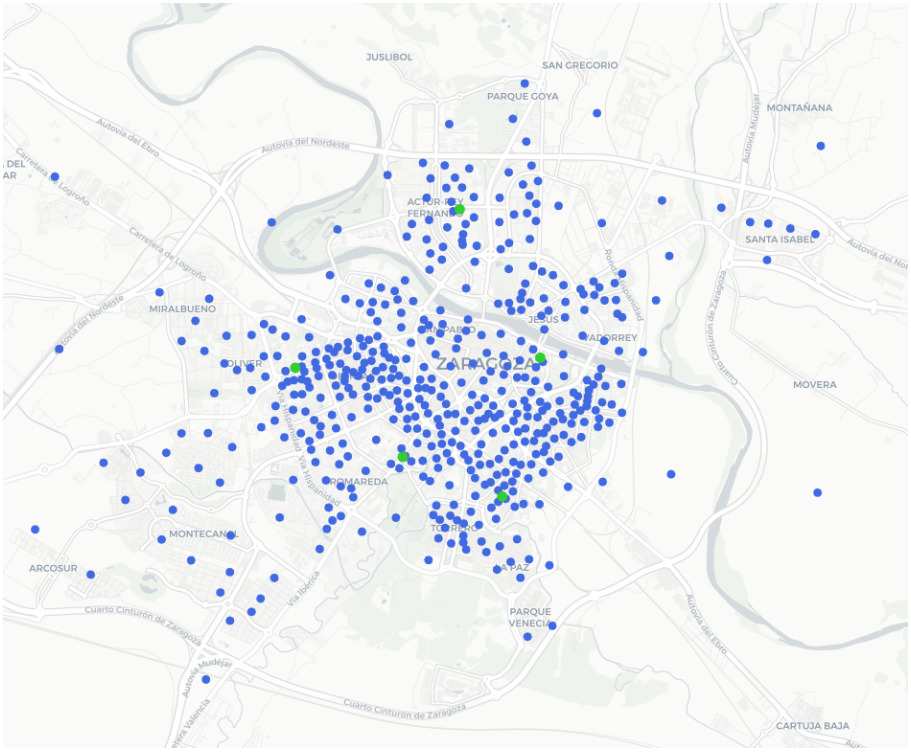

Figura A.12: UMAXCAP, criterio regret, $p = 5$: distribución óptima de instalaciones (●) y clientes (●)

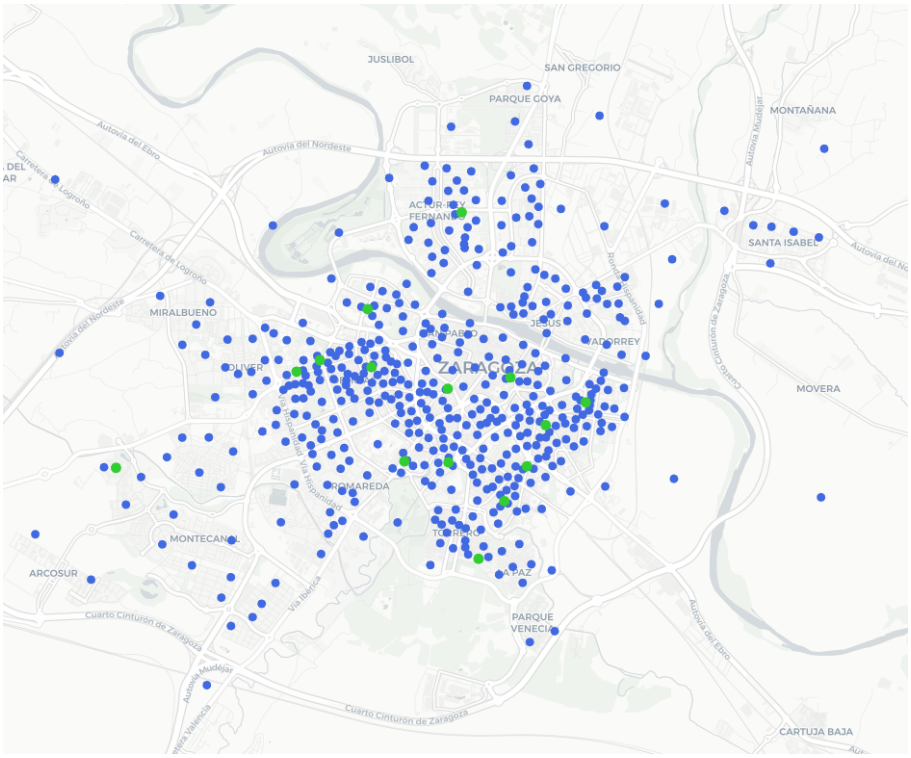

Figura A.13: UMAXCAP, criterio regret, $p = 15$: distribución óptima de instalaciones (●) y clientes (●)

A.4. Código Python

Obtención de los datos

```
nI = 491         #Nº secciones censales
nJ = 25          #Nº locales disponibles
I = range(nI)    #Conjunto de índices para secciones (clientes)
J = range(nJ)    #Conjunto de índices para locales disponibles (posibles ubicaciones)
```

```
from google.colab import files    #Importación de datos en ficheros .csv
uploaded = files.upload()
```

```
import csv
SECCIONES = [line for line in csv.reader(open('SECCIONES.csv'))]   #Carga de secciones censales
SECCIONES.pop(0)
CADENA = [line for line in csv.reader(open('CADENA.csv'))]         #Carga de instalaciones de la competencia
CADENA.pop(0)
LOCALES = [line for line in csv.reader(open('LOCALES.csv'))]       #Carga de ubicaciones disponibles
LOCALES.pop(0)
```

```
import numpy as np    #Importación del paquete NumPy
coordenadas_secciones = [ [float(row[3]),float(row[2])] for row in SECCIONES]   #Lat y long de secciones censales
coordenadas_locales = [ [float(row[2]),float(row[1])] for row in LOCALES]       #Lat y long de ubicaciones disponibles
coordenadas_cadena = [ [float(row[2]),float(row[1])] for row in CADENA]         #Lat y long de instalaciones presentes

#Lat y lon de todas (Secciones, Disponibles y Presentes)
coordenadas = np.concatenate((coordenadas_secciones,coordenadas_locales,coordenadas_cadena)).tolist()

#La fila 0 es la primera sección censal y la fila 490 la última
#La fila 491 es el primer local disponible y la fila 515 es el último
#La fila 516 es la primera instalación presente y la fila 524 es la última
```

```
!pip install openrouteservice         #Instalación y Carga de OpenRouteService
import openrouteservice as ors
client = ors.Client(key='CLAVE PERSONAL')
```

```
#Cálculo de las 5 submatrices de distancias

matriz1 = client.distance_matrix(
    locations=coordenadas,
    sources = [i for i in range(491,498)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")

matriz2 = client.distance_matrix(
    locations=coordenadas,
    sources = [i for i in range(498,505)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")

matriz3 = client.distance_matrix(
    locations=coordenadas,
    sources = [i for i in range(505,512)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")

matriz4 = client.distance_matrix(
    locations=coordenadas,
    sources = [i for i in range(512,519)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")

matriz5 = client.distance_matrix(
    locations=coordenadas,
    sources = [i for i in range(519,525)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")
```

```
for fila1 in matriz1['distances']:
  print(fila1)
for fila2 in matriz2['distances']:
  print(fila2)
for fila3 in matriz3['distances']:     #Escritura de las 5 submatrices de distancias
  print(fila3)
for fila4 in matriz4['distances']:
  print(fila4)
for fila5 in matriz5['distances']:
  print(fila5)
```

```
from google.colab import files      #Importación de matriz de distancias en fichero .csv
uploaded = files.upload()
```

```
#Carga de la matriz de distancias
DISTANCIAS = (np.array([line for line in csv.reader(open('DISTANCIAS.csv'))])).astype(np.float)
```

```
coord_secc = [ [float(row[2]),float(row[3])] for row in SECCIONES]
coord_inst = [ [float(row[1]),float(row[2])] for row in CADENA]
coord_ubic = [ [float(row[1]),float(row[2])] for row in LOCALES]
```

```
#Creación de mapa con datos iniciales
import folium   #Importación del paquete Folium
mapa_inicial = folium.Figure(width=1000, height=800)
map1 = folium.Map(location=[41.65,-0.89], zoom_start = 13, max_bounds = True).add_to(mapa_inicial)
folium.TileLayer('cartodbpositron').add_to(map1)

#Puntos azules de clientes
for i in I:
  folium.Circle(coord_secc[i], radius=40, color="royalblue", fill=True, opacity=1, fill_opacity=1).add_to(map1)

#Puntos rojos de instalaciones actuales
for i in range(0,9):
  folium.Circle(coord_inst[i], radius=60, color="red", fill=True, opacity=1, fill_opacity=1).add_to(map1)
```

```
#Puntos verdes de ubicaciones disponibles
for i in range(0,25):
  folium.Circle(coord_ubic[i], radius=60, color="limegreen", fill=True, opacity=1, fill_opacity=1).add_to(map1)
mapa_inicial
```

Problema de máxima captación (MAXCAP)

Implementación del modelo

```
!pip install gurobipy         #Instalación y carga de la herramienta Gurobi
from gurobipy import *
env = Env(params = {"WLSACCESSID" : 'CLAVE PERSONAL', "WLSSECRET" : 'CLAVE PERSONAL', "LICENSEID" : CLAVE PERSONAL})
```

```
modelo = Model("MAXCAP")  #Definición del modelo
```

```
p = 1    #Número de nuevas instalaciones a localizar (modificar para cada simulación)

a = [int(row[1]) for row in SECCIONES]  #Demanda de los clientes
S = np.min(DISTANCIAS[25:34],axis=0)    #Instalación actual más cercana a cliente
d = np.transpose(DISTANCIAS)            #Distancias entre cliente i y ubicación j
```

```
#Definición y cálculo de los conjuntos N_i
N = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]<S[i]:
            N[i]=np.append(N[i],j)

#Definición y cálculo de los conjuntos K_i
K = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]==S[i]:
            K[i]=np.append(K[i],j)
```

```
#Definición de las variables del modelo
x = modelo.addVars(J, vtype="B", name = "x")
y = modelo.addVars(I, vtype="B", name = "y")
z = modelo.addVars(I, vtype="B", name = "z")
```

```
#Definición de la función objetivo
modelo.setObjective(quicksum( a[i]*y[i] for i in I) + quicksum( (a[i]/2)*z[i] for i in I), sense=GRB.MAXIMIZE)
```

```
#Definición de los conjuntos de restricciones del modelo
r1 = modelo.addConstrs((y[i] <= quicksum (x[j] for j in N[i])) for i in I)
r2 = modelo.addConstrs((z[i] <= quicksum (x[j] for j in K[i])) for i in I)
r3 = modelo.addConstrs(y[i]+z[i]<=1 for i in I)
r4 = modelo.addConstr(quicksum(x[j] for j in J) == p)
```

Evaluación de resultados

```
modelo.optimize()  #Ejecución de Gurobi para calcular la solución óptima
```

```
print("Valor óptimo de la función objetivo =",round(modelo.ObjVal))
```

```
modelo.Runtime      #Tiempo total de ejecución de Gurobi
```

```
print("Soluciones:\n")
print("Conjunto de ubicaciones donde se localizarán las nuevas instalaciones (variables x_j>0):")
solucion_x_j = np.empty((0,1),dtype = int)
coord_solucion_x_j = np.empty((0, 2))
for j in range (0,nJ):
      if x[j].X>0:
           solucion_x_j = np.append(solucion_x_j,j)
           coord_solucion_x_j = np.append(coord_solucion_x_j,[coord_ubic[j]],axis= 0)
print(solucion_x_j,"\n")
print(coord_solucion_x_j,"\n")

print("Conjunto de clientes que pasan a ser captados por las nuevas instalaciones (variables y_i>0):")
solucion_y_i = np.empty((0,1),dtype = int)
coord_solucion_y_i = np.empty((0, 2))
for i in range (0,nI):
      if y[i].X>0:
          solucion_y_i = np.append(solucion_y_i,i)
          coord_secc[i]
          coord_solucion_y_i = np.append(coord_solucion_y_i,[coord_secc[i]],axis=0)
print(solucion_y_i,"\n")
print(coord_solucion_y_i,"\n")

print("Conjunto de clientes que pasan a ser repartidos entre instalaciones de ambas empresas (variables z_i>0):")
solucion_z_i = np.empty((0,1),dtype = int)
coord_solucion_z_i = np.empty((0, 2))
for i in range (0,nI):
      if z[i].X>0:
          solucion_z_i = np.append(solucion_z_i,i)
          coord_secc[i]
          coord_solucion_z_i = np.append(coord_solucion_z_i,[coord_secc[i]],axis=0)
print(solucion_z_i,"\n")
print(coord_solucion_z_i,"\n")
```

```
#Creación de mapa con resultados obtenidos
mapa_resultados = folium.Figure(width=1000, height=800)
map2 = folium.Map(location=[41.65,-0.89], zoom_start = 13, max_bounds = True).add_to(mapa_resultados)
folium.TileLayer('cartodbpositron').add_to(map2)

#Puntos azules de clientes que no cambian
for i in I:
  folium.Circle(coord_secc[i], radius=40, color="royalblue", fill=True, opacity=1, fill_opacity=1).add_to(map2)

#Puntos morados de clientes que cambian totalmente
for i in range(0,np.shape(coord_solucion_y_i)[0]):
  folium.Circle(coord_solucion_y_i[i], radius=40, color="purple", fill=True, opacity=1, fill_opacity=1).add_to(map2)
```

```
#Puntos negros de clientes que se reparten
for i in range(0,np.shape(coord_solucion_z_i)[0]):
  folium.Circle(coord_solucion_z_i[i], radius=40, color="black", fill=True, opacity=1, fill_opacity=1).add_to(map2)

#Puntos rojos de instalaciones actuales
for i in range(0,9):
  folium.Circle(coord_inst[i], radius=60, color="red", fill=True, opacity=1, fill_opacity=1).add_to(map2)

#Puntos verdes de localizaciones elegidas
for i in range(0,np.shape(coord_solucion_x_j)[0]):
  folium.Circle(coord_solucion_x_j[i], radius=60, color="limegreen", fill=True, opacity=1, fill_opacity=1).add_to(map2)

mapa_resultados
```

Problema de máxima captación con relocalización (MAXRELOC)

Implementación del modelo

```
modelo = Model("MAXRELOC")  #Definición del modelo
```

```
s = 2                    #Número de instalaciones ya presentes (modificar para cada simulación)
p = 3                    #Número de nuevas instalaciones a localizar (modificar para cada simulación)
r = 1                    #Número de instalaciones a relocalizar (modificar para cada simulación)
J_A = [3,4]              #Conjunto J_A de índices J ocupados inicialmente por A (modificar para cada simulación)
```

```
#Definición de las variables del modelo
x = modelo.addVars(J, vtype="B", name = "x")
y = modelo.addVars(I, vtype="B", name = "y")
z = modelo.addVars(I, vtype="B", name = "z")
```

```
#Definición de la función objetivo
modelo.setObjective(quicksum( a[i]*y[i] for i in I) + quicksum( (a[i]/2)*z[i] for i in I), sense=GRB.MAXIMIZE)
```

```
#Definición de los conjuntos de restricciones del modelo
r1 = modelo.addConstrs((y[i] <= quicksum (x[j] for j in N[i])) for i in I)
r2 = modelo.addConstrs((z[i] <= quicksum (x[j] for j in K[i])) for i in I)
r3 = modelo.addConstrs(y[i]+z[i]<=1 for i in I)
r4 = modelo.addConstr(quicksum(x[j] for j in J) == s+p)
r5 = modelo.addConstr(quicksum(x[j] for j in J_A) == s-r)
```

Evaluación de resultados

```
modelo.optimize()  #Ejecución de Gurobi para calcular solución óptima
```

```
print("Valor óptimo de la función objetivo =",round(modelo.ObjVal))
```

```
modelo.Runtime      #Tiempo total de ejecución de Gurobi
```

```
print("Soluciones:\n")

print("Conjunto de ubicaciones donde se relocalizarán las r y se localizarán las p instalaciones (variables x_j>0):")
solucion_x_j = np.empty((0,1),dtype = int)
solucion_x_j
coord_solucion_x_j = np.empty((0, 2))
for j in range (0,nJ):
      if x[j].X>0:
           solucion_x_j = np.append(solucion_x_j,j)
           coord_solucion_x_j = np.append(coord_solucion_x_j,[coord_ubic[j]],axis= 0)
print(solucion_x_j.tolist(),"\n")
print(coord_solucion_x_j,"\n")

print("Conjunto de ubicaciones iniciales para la empresa entrante:")
ubi_iniciales = np.empty((0, 2))
for j in J_A:
           ubi_iniciales = np.append(ubi_iniciales,[coord_ubic[j]],axis= 0)
print(J_A,"\n")
print(ubi_iniciales,"\n")

print("Conjunto de ubicaciones iniciales no relocalizadas:")
ubi_no_reloc = np.empty((0,1),dtype = int)
coord_no_reloc = np.empty((0,2))
for j in J_A:
  if j in solucion_x_j:
           ubi_no_reloc = np.append(ubi_no_reloc,j)
           coord_no_reloc = np.append(coord_no_reloc,[coord_ubic[j]],axis= 0)
print(ubi_no_reloc,"\n")
print(coord_no_reloc,"\n")

print("Conjunto de clientes captados por la nueva empresa (variables y_i>0):")
solucion_y_i = np.empty((0,1),dtype = int)
coord_solucion_y_i = np.empty((0, 2))
for i in range (0,nI):
      if y[i].X>0:
          solucion_y_i = np.append(solucion_y_i,i)
          coord_secc[i]
          coord_solucion_y_i = np.append(coord_solucion_y_i,[coord_secc[i]],axis=0)
print(solucion_y_i,"\n")
print(coord_solucion_y_i,"\n")

print("Conjunto de clientes repartidos entre instalaciones de ambas empresas (variables z_i>0):")
solucion_z_i = np.empty((0,1),dtype = int)
coord_solucion_z_i = np.empty((0, 2))
for i in range (0,nI):
      if z[i].X>0:
          solucion_z_i = np.append(solucion_z_i,i)
          coord_secc[i]
          coord_solucion_z_i = np.append(coord_solucion_z_i,[coord_secc[i]],axis=0)
print(solucion_z_i,"\n")
print(coord_solucion_z_i,"\n")
```

```
#Creación de mapa con resultados obtenidos
mapa_resultados = folium.Figure(width=1000, height=800)
map2 = folium.Map(location=[41.65,-0.89], zoom_start = 13, max_bounds = True).add_to(mapa_resultados)
folium.TileLayer('cartodbpositron').add_to(map2)
```

```
#Puntos azules: clientes que no cambian
for i in I:
  folium.Circle(coord_secc[i], radius=40, color="royalblue", fill=True, opacity=1, fill_opacity=1).add_to(map2)

#Puntos morados: clientes que cambian totalmente
for i in range(0,np.shape(coord_solucion_y_i)[0]):
  folium.Circle(coord_solucion_y_i[i], radius=40, color="purple", fill=True, opacity=1, fill_opacity=1).add_to(map2)

#Puntos negros: clientes que se reparten
for i in range(0,np.shape(coord_solucion_z_i)[0]):
  folium.Circle(coord_solucion_z_i[i], radius=40, color="black", fill=True, opacity=1, fill_opacity=1).add_to(map2)

#Puntos rojos: instalaciones de la competencia
for i in range(0,9):
  folium.Circle(coord_inst[i], radius=60, color="red", fill=True, opacity=1, fill_opacity=1).add_to(map2)

#Puntos verdes: localizaciones elegidas y relocalizadas
for i in range(0,np.shape(coord_solucion_x_j)[0]):
  folium.Circle(coord_solucion_x_j[i], radius=60, color="limegreen", fill=True, opacity=1, fill_opacity=1).add_to(map2)

#Puntos naranjas: instalaciones situadas inicialmente
for i in range(0,np.shape(ubi_iniciales)[0]):
  folium.Circle(ubi_iniciales[i], radius=70, color="orange", fill=True, opacity=1, fill_opacity=1).add_to(map2)

#Puntos verdes pequeños: instalaciones no relocalizadas
for i in range(0,np.shape(coord_no_reloc)[0]):
  folium.Circle(coord_no_reloc[i], radius=35, color="limegreen", fill=True, opacity=1, fill_opacity=1).add_to(map2)

mapa_resultados
```

Problema de máxima captación con incertidumbre (UMAXCAP)

Obtención de los datos

```
from google.colab import files    #Importación de nuevos datos en ficheros .csv
uploaded = files.upload()
```

Escenario 1: Demanda Modificada, Competencia Estática

```
#Carga de secciones censales para los escenarios 1 y 3
import csv
SECCIONES_e1 = [line for line in csv.reader(open('SECCIONES1.csv'))]
SECCIONES_e1.pop(0)
```

Escenario 2: Demanda Estática, Competencia Modificada

```
#Carga de instalaciones de la competencia para e=2,3
import csv
CADENA_e2 = [line for line in csv.reader(open('CADENA2.csv'))]
CADENA_e2.pop(0)
```

```
#Coordenadas de instalaciones de la competencia para e=2,3
coordenadas_cadena_e2 = [ [float(row[2]),float(row[1])] for row in CADENA_e2]

#Coordenadas agrupadas para e=2,3 (Secciones, Ubicaciones y Competencia)
coordenadas_esc2 = np.concatenate((coordenadas_secciones,coordenadas_locales,coordenadas_cadena_e2)).tolist()

#La fila 0 es la primera sección censal y la fila 490 la última
#La fila 491 es el primer local disponible y la fila 515 es el último
#La fila 516 es la primera instalación presente y la fila 524 es la última
```

```
#Cálculo de las nuevas 5 submatrices de distancias

matriz1 = client.distance_matrix(
    locations=coordenadas_esc2,
    sources = [i for i in range(491,498)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")

matriz2 = client.distance_matrix(
    locations=coordenadas_esc2,
    sources = [i for i in range(498,505)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")

matriz3 = client.distance_matrix(
    locations=coordenadas_esc2,
    sources = [i for i in range(505,512)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")

matriz4 = client.distance_matrix(
    locations=coordenadas_esc2,
    sources = [i for i in range(512,519)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")

matriz5 = client.distance_matrix(
    locations=coordenadas_esc2,
    sources = [i for i in range(519,525)],
    destinations = [i for i in range(0,nI)],
    profile='foot-walking',
    metrics=['distance'],
    units ="m")
```

```
for fila1 in matriz1['distances']:
  print(fila1)
for fila2 in matriz2['distances']:
  print(fila2)
for fila3 in matriz3['distances']:      #Escritura de las 5 submatrices de distancias
  print(fila3)
```

```
for fila4 in matriz4['distances']:
  print(fila4)
for fila5 in matriz5['distances']:
  print(fila5)
```

```
#Importación de matriz de distancias en fichero .csv
from google.colab import files
uploaded = files.upload()
```

```
#Carga de la matriz de distancias para e=2,3
DISTANCIAS2 = (np.array([line for line in csv.reader(open('DISTANCIAS2.csv'))])).astype(np.float)

#Coordenadas de instalaciones de la competencia con e=2,3 para mapa de resultados
coord_inst2 = [ [float(row[1]),float(row[2])] for row in CADENA_e2]
```

```
#Creación de mapa con datos iniciales
import folium
mapa_inicial = folium.Figure(width=1000, height=800)
map1 = folium.Map(location=[41.65,-0.89], zoom_start = 13, max_bounds = True).add_to(mapa_inicial)
folium.TileLayer('cartodbpositron').add_to(map1)

#Puntos azules de clientes
for i in I:
  folium.Circle(coord_secc[i], radius=40, color="royalblue", fill=True, opacity=1, fill_opacity=1).add_to(map1)

#Puntos rojos de instalaciones actuales
for i in range(0,9):
  folium.Circle(coord_inst2[i], radius=60, color="red", fill=True, opacity=1, fill_opacity=1).add_to(map1)

#Puntos verdes de ubicaciones disponibles
for i in range(0,25):
  folium.Circle(coord_ubic[i], radius=60, color="limegreen", fill=True, opacity=1, fill_opacity=1).add_to(map1)
mapa_inicial
```

Escenario 3: Demanda Modificada, Competencia Modificada

```
#Se toman las secciones del escenario 1 y las instalaciones de la competencia del escenario 2.
```

Implementación y evaluación

```
E=[0,1,2] #Definición del conjunto de índices para los escenarios e=1,2,3 respectivamente
```

```
modelo = Model("UMAXCAP", env = env)  #Definición del modelo
```

```
p = 15                    #Número de nuevas instalaciones a localizar (modificar para cada simulación)

a13 = np.array([int(row[1]) for row in SECCIONES_e1])   #Demanda para e=1,3
a2 = np.array([int(row[1]) for row in SECCIONES])       #Demanda para e=2
a = np.c_[np.c_[np.c_[a13, a2], a13]]                   #Matriz de demandas a_ie

S1= (np.min(DISTANCIAS[25:34],axis=0)).tolist()      #Distancias mínimas S para e=1
S23 = (np.min(DISTANCIAS2[25:34],axis=0)).tolist()   #Distancias mínimas S para e=2,3
S = [S1,S23]                                         #Matriz de distancias S_ei (índices intercambiados)

d = np.transpose(DISTANCIAS)                          #Redefinición de la distancia entre el cliente i y la ubicación j
```

```
#Definición y cálculo de los conjuntos N_i para e=1
N1 = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]<S[0][i]:
            N1[i]=np.append(N1[i],j)

#Definición y cálculo de los conjuntos N_i para e=2,3
N23 = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]<S[1][i]:
            N23[i]=np.append(N23[i],j)

#Matriz de conjuntos N_ei (índices intercambiados)
N = [N1,N23,N23]
```

```
#Definición y cálculo de los conjuntos K_i para e=1
K1 = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]==S[0][i]:
            K1[i]=np.append(K1[i],j)

#Definición y cálculo de los conjuntos K_i para e=2,3
K23 = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]==S[1][i]:
            K23[i]=np.append(K23[i],j)

#Matriz de conjuntos N_ei (índices intercambiados)
K = [K1,K23,K23]
```

Maxmin criterion

```
modelo = Model("UMAXCAP", env = env)  #Definición del modelo
```

```
#Definición de las variables del modelo
x = modelo.addVars(J, vtype="B", name = "x")
y = modelo.addVars([(e,i) for i in I for e in E], vtype="B", name = "y")
z = modelo.addVars([(e,i) for i in I for e in E], vtype="B", name = "z")
g = modelo.addVar(vtype="C", name = "g")
```

```
#Definición de la función objetivo
modelo.setObjective(g, GRB.MAXIMIZE)
```

```
#Definición de los conjuntos de restricciones del modelo
r1 = modelo.addConstrs(quicksum(a[i][e] * y[e,i] for i in I) + quicksum((1/2) * a[i][e] * z[e,i] for i in I) >= g for e in E)
r2 = modelo.addConstrs(y[e,i] <= quicksum(x[j] for j in N[e][i]) for i in I for e in E)
r3 = modelo.addConstrs(z[e,i] <= quicksum(x[j] for j in K[e][i]) for i in I for e in E)
```

```
r4 = modelo.addConstrs(y[e,i]+z[e,i]<=1 for e in E for i in I)
r5 = modelo.addConstr(quicksum(x[j] for j in J) == p)
```

```
modelo.optimize()         #Ejecución de Gurobi para calcular la solución óptima
```

```
print("Valor óptimo de la función objetivo =",round(modelo.ObjVal))
```

```
modelo.Runtime      #Tiempo total de ejecución de Gurobi
```

```
print("Soluciones:\n")
print("Conjunto de ubicaciones donde se localizarán las nuevas instalaciones (variables x_j>0):")
solucion_x_j = np.empty((0,1),dtype = int)
coord_solucion_x_j = np.empty((0, 2))
for j in range (0,nJ):
      if x[j].X>0:
           solucion_x_j = np.append(solucion_x_j,j)
           coord_solucion_x_j = np.append(coord_solucion_x_j,[coord_ubic[j]],axis= 0)
print(solucion_x_j,"\n")
print(coord_solucion_x_j,"\n")
```

```
#Creación de mapa con resultados obtenidos
mapa_umaxcap = folium.Figure(width=1000, height=800)
map_umaxcap = folium.Map(location=[41.65,-0.89], zoom_start = 13, max_bounds = True).add_to(mapa_umaxcap)
folium.TileLayer('cartodbpositron').add_to(map_umaxcap)

#Puntos azules de clientes
for i in I:
  folium.Circle(coord_secc[i], radius=40, color="royalblue", fill=True, opacity=1, fill_opacity=1).add_to(map_umaxcap)

#Puntos verdes de localizaciones elegidas
for i in range(0,np.shape(coord_solucion_x_j)[0]):
  folium.Circle(coord_solucion_x_j[i], radius=60, color="limegreen", fill=True, opacity=1, fill_opacity=1).add_to(map_umaxcap)

mapa_umaxcap
```

Regret criterion

```
p=15 #(modificar para cada simulación)

#Definición del conjunto Z_e$

#Cálculo de Z_1 (valor objetivo del MAXCAP para el escenario e=1)

#Definición de inputs
a_max1 = [int(row[1]) for row in SECCIONES_e1]
S_max1 = np.min(DISTANCIAS[25:34],axis=0)
d_max1 = np.transpose(DISTANCIAS)
N = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d_max1[i][j]<S_max1[i]:
            N[i]=np.append(N[i],j)
K = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d_max1[i][j]==S_max1[i]:
            K[i]=np.append(K[i],j)

#Definición del modelo
modelo_max1 = Model("MAXCAP", env = env)
x = modelo_max1.addVars(J, vtype="B", name = "x")
y = modelo_max1.addVars(I, vtype="B", name = "y")
z = modelo_max1.addVars(I, vtype="B", name = "z")
modelo_max1.setObjective(quicksum( a_max1[i]*y[i] for i in I) + quicksum( (a_max1[i]/2)*z[i] for i in I), sense=GRB.MAXIMIZE)
r1 = modelo_max1.addConstrs((y[i] <= quicksum (x[j] for j in N[i])) for i in I)
r2 = modelo_max1.addConstrs((z[i] <= quicksum (x[j] for j in K[i])) for i in I)
r3 = modelo_max1.addConstrs(y[i]+z[i]<=1 for i in I)
r4 = modelo_max1.addConstr(quicksum(x[j] for j in J) == p)

#Ejecución del modelo
modelo_max1.optimize()


#Cálculo de Z_2 (valor objetivo del MAXCAP para el escenario e=2)

#Definición de inputs
a_max2 = [int(row[1]) for row in SECCIONES]
S_max2 = np.min(DISTANCIAS2[25:34],axis=0)
d_max2 = np.transpose(DISTANCIAS2)
N = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d_max2[i][j]<S_max2[i]:
            N[i]=np.append(N[i],j)
K = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d_max2[i][j]==S_max2[i]:
            K[i]=np.append(K[i],j)

#Definición del modelo
modelo_max2 = Model("MAXCAP", env = env)
x = modelo_max2.addVars(J, vtype="B", name = "x")
y = modelo_max2.addVars(I, vtype="B", name = "y")
z = modelo_max2.addVars(I, vtype="B", name = "z")
modelo_max2.setObjective(quicksum( a_max2[i]*y[i] for i in I) + quicksum( (a_max2[i]/2)*z[i] for i in I), sense=GRB.MAXIMIZE)
r1 = modelo_max2.addConstrs((y[i] <= quicksum (x[j] for j in N[i])) for i in I)
r2 = modelo_max2.addConstrs((z[i] <= quicksum (x[j] for j in K[i])) for i in I)
r3 = modelo_max2.addConstrs(y[i]+z[i]<=1 for i in I)
r4 = modelo_max2.addConstr(quicksum(x[j] for j in J) == p)

#Ejecución del modelo
modelo_max2.optimize()


#Cálculo de Z_3 (valor objetivo del MAXCAP para el escenario e=2)

#Definición de inputs
a_max3 = [int(row[1]) for row in SECCIONES_e1]
S_max3 = np.min(DISTANCIAS2[25:34],axis=0)
d_max3 = np.transpose(DISTANCIAS2)
N = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d_max3[i][j]<S_max3[i]:
            N[i]=np.append(N[i],j)
```

```
K = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d_max3[i][j]==S_max3[i]:
            K[i]=np.append(K[i],j)

#Definición del modelo
modelo_max3 = Model("MAXCAP", env = env)
x = modelo_max3.addVars(J, vtype="B", name = "x")
y = modelo_max3.addVars(I, vtype="B", name = "y")
z = modelo_max3.addVars(I, vtype="B", name = "z")
modelo_max3.setObjective(quicksum( a_max3[i]*y[i] for i in I) + quicksum( (a_max3[i]/2)*z[i] for i in I), sense=GRB.MAXIMIZE)
r1 = modelo_max3.addConstrs((y[i] <= quicksum (x[j] for j in N[i])) for i in I)
r2 = modelo_max3.addConstrs((z[i] <= quicksum (x[j] for j in K[i])) for i in I)
r3 = modelo_max3.addConstrs(y[i]+z[i]<=1 for i in I)
r4 = modelo_max3.addConstr(quicksum(x[j] for j in J) == p)

#Ejecución del modelo
modelo_max3.optimize()

Z_umaxcap=np.array([round(modelo_max1.ObjVal),round(modelo_max2.ObjVal),round(modelo_max3.ObjVal)])
print("\n\nConjunto de valores óptimos del MAXCAP para cada escenario=",Z_umaxcap)
```

Se define ahora el modelo del propio criterio

```
#Redefinición y cálculo de los conjuntos N_i para e=1
N1 = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]<S[0][i]:
            N1[i]=np.append(N1[i],j)

#Redefinición y cálculo de los conjuntos N_i para e=2,3
N23 = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]<S[1][i]:
            N23[i]=np.append(N23[i],j)

#Matriz de conjuntos N_ei (índices intercambiados)
N = [N1,N23,N23]
```

```
#Redefinición y cálculo de los conjuntos K_i para e=1
K1 = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]==S[0][i]:
            K1[i]=np.append(K1[i],j)

#Redefinición y cálculo de los conjuntos K_i para e=2,3
K23 = [np.array([]) for i in range (0,nI)]
for i in range (0,nI):
    for j in range (0,nJ):
        if d[i][j]==S[1][i]:
            K23[i]=np.append(K23[i],j)

#Matriz de conjuntos K_ei (índices intercambiados)
K = [K1,K23,K23]
```

```
modelo = Model("UMAXCAP", env = env)  #Definición del modelo
```

```
#Definición de las variables del modelo UMAXCAP
x = modelo.addVars(J, vtype="B", name = "x")
y = modelo.addVars([(e,i) for i in I for e in E], vtype="B", name = "y")
z = modelo.addVars([(e,i) for i in I for e in E], vtype="B", name = "z")
U = modelo.addVar(vtype="C", name = "U")
```

```
#Definición de la función objetivo
modelo.setObjective(U, GRB.MINIMIZE)
```

```
#Definición de los conjuntos de restricciones del modelo
r1 = modelo.addConstrs(Z_umaxcap[e] - quicksum(a[i][e] * y[e,i] for i in I) - quicksum((1/2) * a[i][e] * z[e,i] for i in I) <= U for e in E)
r2 = modelo.addConstrs(y[e,i] <= quicksum(x[j] for j in N[e][i]) for i in I for e in E)
r3 = modelo.addConstrs(z[e,i] <= quicksum(x[j] for j in K[e][i]) for i in I for e in E)
r4 = modelo.addConstrs(y[e,i]+z[e,i]<=1 for e in E for i in I)
r5 = modelo.addConstr(quicksum(x[j] for j in J) == p)
```

```
modelo.optimize()         #Ejecución de Gurobi para calcular la solución óptima
```

```
print("Valor óptimo de la función objetivo =",round(modelo.ObjVal))
```

```
modelo.Runtime      #Tiempo total de ejecución de Gurobi
```

```
print("Soluciones:\n")
print("Conjunto de ubicaciones donde se localizarán las nuevas instalaciones (variables x_j>0):")
solucion_x_j = np.empty((0,1),dtype = int)
coord_solucion_x_j = np.empty((0, 2))
for j in range (0,nJ):
      if x[j].X>0:
           solucion_x_j = np.append(solucion_x_j,j)
           coord_solucion_x_j = np.append(coord_solucion_x_j,[coord_ubic[j]],axis= 0)
print(solucion_x_j,"\n")
print(coord_solucion_x_j,"\n")
```

```
#Creación de mapa con resultados obtenidos
mapa_umaxcap = folium.Figure(width=1000, height=800)
map_umaxcap = folium.Map(location=[41.65,-0.89], zoom_start = 13, max_bounds = True).add_to(mapa_umaxcap)
folium.TileLayer('cartodbpositron').add_to(map_umaxcap)

#Puntos azules de clientes
for i in I:
  folium.Circle(coord_secc[i], radius=40, color="royalblue", fill=True, opacity=1, fill_opacity=1).add_to(map_umaxcap)

#Puntos verdes de localizaciones elegidas
for i in range(0,np.shape(coord_solucion_x_j)[0]):
  folium.Circle(coord_solucion_x_j[i], radius=60, color="limegreen", fill=True, opacity=1, fill_opacity=1).add_to(map_umaxcap)

mapa_umaxcap
```

Bibliografía

[1] O. BERMAN, T. DREZNER Y D. KRASS, Modeling Competitive Facility Location Problems. New Approaches and Results, *Decision Technologies and Applications, INFORMS* (2009), 156-181.

[2] R. CHURCH Y C. REVELLE, The Maximal Covering Location Problem, *Papers in Regional Science* **1** (32) (1974), 101-118.

[3] J. CURRENT, M. DASKIN Y D. SCHILLING, *Discrete Network Location Models*, en: Z. DREZNER Y H.W. HAMACHER, *Facility Location: Applications and Theory*, Springer-Verlag, 2002.

[4] H. EISELT Y G. LAPORTE, Location of a New Facility on a Linear Market in the Presence of Weights, *University of Montreal: Centre de Recherche sur les Transports* **583** (1987).

[5] R.Z. FARAHANI Y M. HEKMATFAR, *Facility Location. Concepts, Models, Algorithms and Case Studies*, Contributions to Management Science, Physica Heidelberg, 2009.

[6] S.L. HAKIMI, Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph, *Operations Research* **12** (2) (1964), 450-459.

[7] S.L. HAKIMI, On Locating New Facilities in a Competitive Environment, *European Journal of Operational Research* **12** (1) (1983), 29-35.

[8] H. HOTELLING, Stability in Competition, *The Economic Journal* **39** (153) (1929), 41-57.

[9] A. KARAKITSIOU, *Modeling Discrete Competitive Facility Location*, SpringerBriefs in Optimization, Springer, 2015.

[10] G. LAPORTE, S. NICKEL Y F. SALDANHA DA GAMA, *Location Science*, Springer, 2015.

[11] F. PLASTRIA, Static Competitive Facility Location: An Overview of Optimisation Approaches, *European Journal of Operational Research* **129** (3) (2001), 461-470.

[12] F. PLASTRIA Y L. VANHAVERBEKE, Maximal Covering Location Problem with Price Decision for Revenue Maximization in a Competitive Environment, *OR Spectrum* **31** (2009), 555-571.

[13] C. REVELLE, The Maximum Capture or Sphere of Influence Location Problem: Hotelling Revisited on a Network, *Journal of Regional Science* **26** (2) (1986), 343-358.

[14] C. REVELLE Y D. SERRA, The Maximum Capture Problem Including Relocation, *INFOR: Information Systems and Operational Research* **29** (2) (1991), 130-138.

[15] D. SERRA, V. MARIANOV Y C. REVELLE, The Maximum Capture Hierarchical Location Problem, *European Journal of Operational Research* **62** (3) (1992), 363-371.

[16] D. SERRA Y C. REVELLE, *Competitive Location in Discrete Space*, en: Z. DREZNER, *Facility Location: A Survey of Applications and Methods*, New York: Springer, 1995

[17] D. SERRA, S. RATICK Y C. REVELLE, The Maximum Capture Problem with Uncertainty, *Environment and Planning B: Planning and Design* **23** (1) (1996), 49-59.

[18] D. SERRA Y C. REVELLE, Competitive Location and Pricing on Networks, *Geographical Analysis* **31** (1) (1999), 109-129.

[19] M.B. TEITZ, Locational Strategies for Competitive Systems, *Journal of Regional Science* **8** (2) (1968), 135-148.

[20] A. WEBER, *Über den Standort des Industrien*, Mohr Siebeck, Tübingen, 1909.