Un modelo de optimización binivel en la gestión de residuos

Silvia Álvarez Tena

Trabajo de fin de grado de Matemáticas
Universidad de Zaragoza

Directora del trabajo: María del Carmen Galé Pola
Codirector del trabajo: Aitor Hernández González

6 de julio de 2024

Resumen

La gestión de residuos urbanos es un campo crucial para la sostenibilidad medioambiental, que persigue minimizar la generación de desechos y maximizar su reutilización y reciclaje. Las primeras decisiones al diseñar un sistema de gestión de residuos se refieren a la planificación estratégica de ubicación de instalaciones, como vertederos y centros de reciclaje, y determinación de sus capacidades, que faciliten la gestión eficaz de los residuos en su recogida y tratamiento, y se reduzca el impacto medioambiental. Este trabajo fin de grado (TFG) se centra en la aplicación de modelos de optimización binivel cuando en la gestión de los residuos intervienen varios agentes en la toma de decisiones, situados en una estructura jerárquica de decisión.

En el primer capítulo, se presentan las características de los sistemas de residuos sólidos urbanos que van a considerarse en el TFG, y su relevancia actual para alcanzar las metas establecidas por los Objetivos de Desarrollo Sostenible. En relación con los modelos matemáticos utilizados, se presentan los modelos de optimización binivel así como las dificultades que surgen en la definición de las soluciones factibles del problema y la existencia de cierto tipo de restricciones. El capítulo finaliza describiendo una aproximación habitual en la resolución de los modelos binivel y las principales contribuciones de este TFG.

En el segundo capítulo, se introducen tres modelos de optimización binivel que abordan el problema de ubicación de instalaciones para la gestión de residuos. En el nivel superior de decisión, una autoridad decide qué instalaciones se abren y con qué capacidad, para minimizar el coste total de apertura. En el primer modelo, en el nivel inferior, una empresa determina, una vez conoce las instalaciones abiertas y su capacidad, cómo asigna los clientes a cada una de las instalaciones, atendiendo al beneficio que le reporta. En el segundo modelo, la autoridad determina la apertura de las instalaciones y asigna una capacidad en cada instalación para cada cliente. Cada cliente tiene un rol activo y éstos deciden a qué única instalación se van a dirigir para tratar los residuos, atendiendo a su beneficio. Este modelo se extiende flexibilizando que cada cliente pueda llevar los residuos a más de una instalación. El tercer modelo es una extensión del segundo al considerar varios tipos de residuos. En el TFG, se considera una formulación alternativa del primer modelo, que se denota por **BFLP-E**, y se demuestra su reformulación como un problema de un solo nivel que puede ser resuelto por un software de optimización.

Finalmente, en el tercer capítulo, se realiza una experiencia computacional con el modelo **BFLP-E**. En primer lugar, un ejemplo contribuye a ilustrar las características especiales de los modelos de optimización binivel. En particular, permite visualizar cómo las decisiones tomadas por el nivel superior afectan al nivel inferior, y viceversa. Esto muestra la necesidad de abordar los problemas de decisión con una estructura jerárquica a partir de los modelos binivel. En la segunda parte del capítulo, se han generado aleatoriamente, dos ejemplos de mayor tamaño y varios escenarios para analizar la influencia, en las soluciones del modelo, de dos parámetros del problema: los precios del servicio y la proporción mínima de demanda a satisfacer a los clientes. Esta experiencia computacional se ha programado en Python utilizando el software de optimización Gurobi.

Abstract

Urban waste management is crucial for environmental sustainability, aiming to minimize waste generation and maximize reuse and recycling. The first steps in designing a waste management system include strategically planning the location of facilities like landfills and recycling centers, and determining their capacities to ensure efficient waste collection and treatment, and to reduce environmental impact. This Final Degree Project focuses on applying bilevel optimization models in waste management, where multiple agents are involved in decision-making within a hierarchical structure. These models help solve complex planning and management issues by considering the interactions and potential conflicts between different decision levels, aiming to find optimal, efficient, and sustainable solutions.

Chapter 1 presents the characteristics of the urban solid waste systems to be considered in this project, along with their current relevance in achieving the goals set by the Sustainable Development Goals. Regarding the mathematical models used, bilevel optimization models are introduced, as well as the difficulties that arise from defining feasible solutions to the problem and the existence of certain types of constraints. The chapter concludes by describing a common approach to solving bilevel models and the main contributions of this final degree project.

Chapter 2 introduces three bilevel optimization models addressing the facility location problem for waste management. In the upper decision level, an authority decides which facilities to open and their capacities to minimize the total opening cost. In the first model, at the lower level, a company determines, once the open facilities and their capacities are known, how to assign clients to each facility based on the generated profit. In the second model, the authority determines the facility openings and assigns a capacity to each facility for each client. Each client has an active role and decides which single facility to go to for waste treatment based on their benefit. This model is extended by allowing each client to distribute their waste to more than one facility. The third model is an extension of the second one, considering multiple types of waste. In this project, an alternative formulation of the first model, denoted as **BFLP-E**, is considered, and its reformulation as a single-level problem that can be solved by optimization software is demonstrated.

Finally, in the third chapter, a computational experiment is conducted with the **BFLP-E** model. First, an example helps illustrate the special characteristics of bilevel optimization models. In particular, it allows visualizing how the upper level decisions affect the lower level, and vice versa. This demonstrates the need of addressing decision-making problems with a hierarchical structure using bilevel models. In the second part of the chapter, two larger examples and several scenarios are randomly generated to analyze the influence of two problem parameters on the model solutions: service prices and the minimum proportion of demand to be satisfied for clients. This computational experiment is programmed in Python using the Gurobi optimization software.

Índice general

Capítulo 1

Introducción

1.1. Sistemas de gestión de residuos

Hoy en día, el consumo y la gran cantidad de residuos que produce la industria está generando una grave preocupación mundial. La generación mundial de residuos sólidos urbanos (MSW, del inglés *municipal solid waste*) es catastrófica y se prevé que será de más de 2200 millones de toneladas/año para 2025. La eliminación directa en vertederos (sin ningún tratamiento previo) de los MSW provoca varios problemas ambientales, como emisiones de gases de efecto invernadero y de compuestos orgánicos volátiles peligrosos, así como olores y contaminación de aguas subterráneas debido a filtraciones. Además, el 11 % del metano mundial se genera debido a la mala gestión de los MSW y se considera la tercera fuente antropogénica más grande de gases de efecto invernadero. Por todo ello, gestionar los desechos de manera sostenible, ha de ser el principal objetivo de la sociedad a nivel mundial [10].

La gestión de residuos abarca varios procesos: recolección, transporte, procesado, reciclado, desechado y monitorizado [7]. El objetivo que se persigue es minimizar el impacto de los residuos en la tierra utilizada y en la población a nivel mundial. Actualmente, los Objetivos de Desarrollo Sostenible (ODS) constituyen un llamamiento universal a la acción para, entre otros fines, proteger el planeta [11]. La *Agenda2030* establece las acciones y metas a alcanzar hasta 2030. En lo que concierne a la gestión de residuos cabe destacar los dos siguientes:

- **Objetivo 11:** ***Ciudades y comunidades sostenibles***.
  Recientemente, la expansión urbana está superando el crecimiento de la población en la mayoría de las ciudades, con efectos perjudiciales sobre la sostenibilidad. Es por ello que este objetivo busca ciudades más sostenibles. En concreto, su meta 11.6 hace referencia a la gestión de los desechos municipales, así como a una reducción del impacto medioambiental que éstos conllevan [11].

- **Objetivo 12:** ***Producción y consumo responsables***.
  El consumo poblacional ha aumentado significativamente en los últimos años. Por ello, este objetivo trata de aumentar la eficiencia de recursos, al mismo tiempo que promover una vida sostenible. En particular, en sus metas 12.4 y 12.5, apela a la necesidad de una reducción en la generación de los residuos así como a una adecuada gestión de los mismos [11].

En definitiva, la gestión de los MSW es un problema mundial que se debe manejar de manera sostenible. Este Trabajo Fin de Grado (TFG) se va a centrar en la toma de decisiones sobre la ubicación de instalaciones para el tratamiento de los residuos urbanos tras su recogida y segregación. La Figura 1.1 muestra el esquema de un posible sistema de gestión de residuos urbanos [7]. Los ciudadanos depositan sus residuos en centros de recogida, y a continuación, una autoridad local transporta los residuos a instalaciones de clasificación. Una vez clasificados, los restos son enviados a un vertedero o a un incinerador, todo ello con el principal objetivo de minimizar el impacto medioambiental. Esta gestión se ha de realizar de forma que se minimicen los costes de transporte y de apertura de las instalaciones necesarias, mientras se satisfacen las demandas de la población, en relación al tratamiento de los MSW, y se atiende a criterios medioambientales.

Existen distintas maneras de abordar el problema de gestión de los MSW y una de ellas consiste en modelar el sistema real mediante un problema de optimización matemática. Caramia y Pizzari [7] consideran la gestión de los MSW como un proceso de toma de decisiones jerárquico, en el que el líder o nivel superior de decisión, toma sus decisiones, y el seguidor o nivel inferior, reacciona a las mismas, tomando las decisiones que considera según sus propios objetivos. En la Figura 1.1 se observa que las decisiones sobre la apertura de instalaciones de clasificación y tratamiento son tomadas por el decisor denominado *leader* (rectángulos blancos) y las decisiones sobre el transporte, por otro decisor denominado *follower* (rectángulos grises). El modelo que va a permitir representar adecuadamente este proceso de toma de decisiones, es un modelo de optimización binivel. En el siguiente apartado, se introducen los conceptos fundamentales sobre estos modelos de optimización matemática.

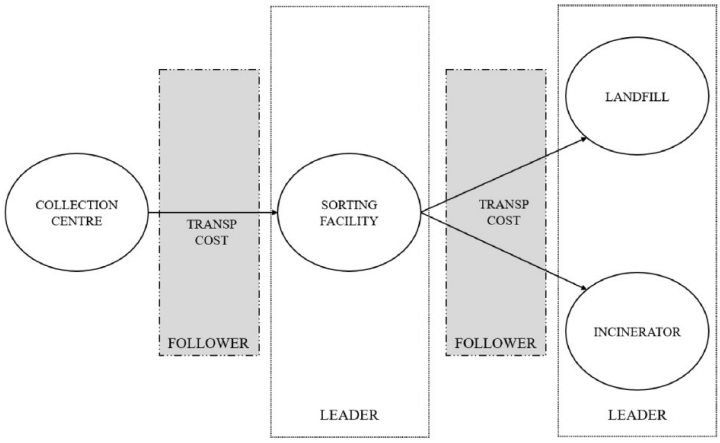


Figura 1.1: Red de gestión de residuos y sus correspondientes áreas de decisión. Fuente: [7]

1.2. Optimización binivel

La optimización binivel (BP, del inglés *Bilevel Programming*) se propone como una generalización de la optimización matemática estándar para modelar adecuadamente la toma de decisiones en un sistema jerárquico con dos niveles de decisión. Estos sistemas se caracterizan por la existencia de una jerarquía organizativa entre ellos, a menudo con objetivos diferentes y conflictivos, y controlando cada uno, únicamente algunas variables de decisión. Es por ello, que el comportamiento de cada decisor se ve afectado por las decisiones tomadas en el otro nivel de decisión [9]. El decisor, situado en el nivel superior y conocido como líder, determina el valor de las variables de decisión que éste controla con el propósito de optimizar su función objetivo. Además, debe tener en cuenta, la reacción del decisor del nivel inferior, llamado seguidor. Este decisor, conocida la decisión del líder, resuelve su propio problema de optimización, cuya función objetivo y restricciones, pueden estar parcialmente determinadas por las decisiones del líder. Por lo tanto, un problema de BP es un modelo de optimización en el que la región de factibilidad está implícitamente determinada por otro problema de optimización. Se puede escribir, en su forma general, de la siguiente manera:

$$
\begin{aligned}
&\text{“}\min_{\mathbf{x}}\text{”} \quad f_1(\mathbf{x},\mathbf{y}) \\
&\text{sujeto a:} \quad g(\mathbf{x},\mathbf{y}) \leq 0 \\
&\qquad\qquad \mathbf{x} \geq \mathbf{0}_{n_1} \\
&\qquad\qquad \text{donde, para cada } \mathbf{x}, \mathbf{y} \text{ es solución de:} \\
&\qquad\qquad \min_{\mathbf{y}} \quad f_2(\mathbf{x},\mathbf{y}) \\
&\qquad\qquad \text{sujeto a:} \quad h(\mathbf{x},\mathbf{y}) \leq 0 \\
&\qquad\qquad\qquad\quad \mathbf{y} \geq \mathbf{0}_{n_2}
\end{aligned}
\tag{1.1}
$$

donde $\mathbf{x} \in \mathbb{R}^{n_1}$ son las variables de decisión controladas por el líder; $\mathbf{y} \in \mathbb{R}^{n_2}$ las variables de decisión controladas por el seguidor; f_1 y f_2 son las funciones objetivo del líder y del seguidor, respectivamente; $g(\mathbf{x},\mathbf{y}) : \mathbb{R}^{n_1+n_2} \to \mathbb{R}^{m_1}$ y $h(\mathbf{x},\mathbf{y}) : \mathbb{R}^{n_1+n_2} \to \mathbb{R}^{m_2}$ determinan el conjunto de restricciones del problema. En la siguiente subsección se detalla la razón de "$\min_{\mathbf{x}}$".

Este TFG se va a centrar en problemas de optimización binivel lineales [3] (LBP, del inglés Linear Bilevel Programming), en los que en el problema (1.1) todas las funciones y restricciones son lineales. A continuación se muestra una formulación general para un LBP:

$$\begin{aligned}
&\text{"}\min_{\mathbf{x}}\text{"} && \mathbf{c_1 x} + \mathbf{d_1 y} && \text{(1.2a)}\\
&\text{sujeto a:} && A_1\mathbf{x} + B_1\mathbf{y} \leq \mathbf{b_1} && \text{(1.2b)}\\
& && \mathbf{x} \geq \mathbf{0}_{n_1} && \text{(1.2c)}
\end{aligned}$$

donde, dado $\mathbf{x}$, $\mathbf{y}$ es solución del problema:

$$\begin{aligned}
&\min_{\mathbf{y}} && \mathbf{d_2 y} && \text{(1.2d)}\\
&\text{sujeto a:} && A_2\mathbf{x} + B_2\mathbf{y} \leq \mathbf{b_2} && \text{(1.2e)}\\
& && \mathbf{y} \geq \mathbf{0}_{n_2} && \text{(1.2f)}
\end{aligned}$$

donde para $i = 1,2$, $\mathbf{c_i}$ y $\mathbf{d_i}$ son vectores fila de dimensión n_1 y n_2, respectivamente; $\mathbf{b_i}$ es un vector de dimensión m_i, A_i es una matriz m_i x n_1 y B_i es una matriz m_i x n_2. Las variables de decisión del líder y del seguidor, $\mathbf{x}$ e $\mathbf{y}$, respectivamente, se asumen no negativas.

Ejemplo 1.2.1. Sea el siguiente problema binivel lineal en el que el nivel superior controla la variable z y las variables x, y son controladas por el nivel inferior:

$$\begin{aligned}
&\text{"}\min_{z}\text{"} && x - z\\
&\text{sujeto a:} && z \leq 3\\
& && z \geq 0\\
& && \text{donde, dado } z, (x,y) \text{ resuelve :}\\
& && \max_{x,y} \quad x + y\\
& && \text{sujeto a:} \quad x + y \leq 5\\
& && \qquad\quad x \geq 0, \quad y \geq 0
\end{aligned}$$

En la Figura 1.2 se puede ver representado el poliedro determinado por todas las restricciones.

1.2.1. Definición de solución factible

La función objetivo del líder (1.2a) consiste en minimizar, así como la función objetivo del seguidor, (1.2d). Sea R el poliedro definido por las restricciones del problema del nivel superior (1.2b)-(1.2c) y S el poliedro definido por las restricciones del problema del nivel inferior (1.2e)-(1.2f). El conjunto de restricciones comunes definen un poliedro $T = R \cap S$.

Sea $R_1 \subset \mathbb{R}^{n_1}$, la proyección del poliedro R en $\mathbb{R}^{n_1}$, entonces dado $x \in R_1$, el problema del nivel inferior viene dado por (1.2d)-(1.2f). Sea $S(x) = \{y \in \mathbb{R}^{n_2} : (x,y) \in S\}$ el conjunto de soluciones factibles del problema del nivel inferior fijado x, y $M(x)$ el conjunto de soluciones óptimas. En la formulación (1.2) se indica "$\min_{x}$" porque cuando en el problema del nivel inferior tiene solución óptima múltiple, esto es, el conjunto $M(x)$ contiene más de un punto, no está claro qué solución $y \in M(x)$ elegirá el decisor del nivel inferior.

La *región factible* del problema de optimización binivel (1.2), denominada *región inducida*, se define:

$$IR = \{(x,y) : (x,y) \in T, y \in M(x)\}$$

En la literatura, se han utilizado principalmente dos aproximaciones para dar respuesta a la existencia de óptimo múltiple en el nivel inferior para una decisión dada del nivel superior. La más habitual o *aproximación optimista* consiste en asumir que el nivel superior tiene influencia sobre el nivel inferior para que éste elija siempre la solución óptima que más le conviene al nivel superior. Así, entre las múltiples soluciones óptimas del nivel inferior, éste siempre se queda con la que da un mejor valor a la función objetivo del nivel superior. En este caso, la región inducida se define de la siguiente manera:

$$IR = \{(x,y) : (x,y) \in T, y \in arg \min_{y \in M(x)} f_1(x,y)\}$$

y en lugar de, "$\min_{x}$", se puede escribir $\min_{x,y}$.

Otra aproximación utilizada en la literatura, asume que el nivel inferior siempre escoge la solución óptima que menos le conviene al nivel superior, es decir, aquella que proporciona un peor valor a su función objetivo. En este caso, se denomina *aproximación pesimista* y la región inducida se define:

$$IR = \{(x,y) : (x,y) \in T, y \in arg \max_{y \in M(x)} f_1(x,y)\}$$

En este TFG se va a utilizar la aproximación optimista en el tratamiento de los modelos binivel formulados.

Ejemplo 1.2.2. Se considera el LBP del Ejemplo 1.2.1. El objetivo del nivel superior consiste en minimizar $f_1 = x - z$, mientras que el objetivo del nivel inferior consiste en maximizar $f_2 = x + y$.

Notar que si el nivel superior toma la decisión $z = 0$, la región de factibilidad del nivel inferior queda definida por $S(0)$, que corresponde a la región azul de la Figura 1.2. El segmento entre $(5,0,0)$ y $(0,5,0)$ es el conjunto de soluciones óptimas del problema inferior fijado $z = 0$, por lo que existe óptimo múltiple, es decir, el conjunto $M(0)$ tiene más de un elemento. Notemos que, dado un valor $z_0 \leq 3$, $M(z_0)$ coincide con el segmento que une los puntos $(5,0,z_0)$ y $(0,5,z_0)$.

En el caso de la aproximación optimista, el seguidor escoge la opción más conveniente para el líder que desea minimizar $x - z$. Luego, la solución del problema inferior viene dada por el punto $(0,5,z_0)$. En la Figura 1.2, el punto en naranja es el punto factible binivel calculado con $z = 0$. Por tanto, la región inducida coincide con el segmento que une los puntos $(0,5,0)$ y $(0,5,3)$. Atendiendo a la función objetivo del líder, la solución del problema binivel viene dada por el punto $(0,5,3)$, marcado en rosa en la Figura 1.2.

Sin embargo, aplicando una aproximación pesimista, el seguidor escoge la opción que menos le conviene al líder, el punto $(5,0,z_0)$, esto es, el punto $(5,0,0)$, marcado en azul en la Figura 1.2 cuando $z = 0$. En este caso, la región inducida es el segmento entre los puntos $(5,0,0)$ y $(5,0,3)$, y la solución al problema binivel es el punto $(5,0,3)$, marcado en verde en la Figura 1.2.

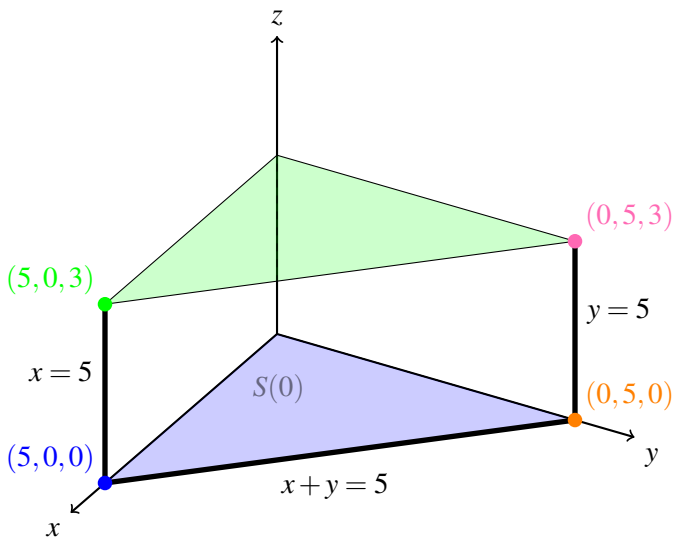


Figura 1.2: Ejemplo ilustrativo de un problema LBP con óptimos múltiples en el nivel inferior.

1.2.2. Restricciones en el nivel superior que incluyen variables del nivel inferior

Cuando las restricciones del nivel superior (1.2b) contienen variables del nivel inferior, se denominan restricciones *coupling* y pueden provocar que el problema binivel sea no factible. Dado $\mathbf{x} \in R_1$, si existe un vector $\mathbf{y} \in \mathbb{R}^{n_2}$ tal que $(x,y) \in IR$, entonces x se dice *admisible*. En el caso de que no existan puntos admisibles, el problema binivel es no factible [2].

Ejemplo 1.2.3. Se considera el Ejemplo 1.2.1, añadiendo la siguiente restricción *coupling*:

$$2x + y - 10z \leq -10$$

El problema del nivel inferior no se ha modificado, por lo que fijado $z = z_0$, el conjunto $M(z_0)$ incluye todos los puntos del segmento $(5,0,z_0)$ y $(0,5,z_0)$. En la Figura 1.3 se ha incluido el plano que determina la restricción *coupling*. Si se considera el conjunto de soluciones óptimas del problema inferior para $z_0 = 0$, el segmento marcado en negro entre los puntos $(5,0,0)$ y $(0,5,0)$, no verifican dicha restricción. Por tanto, el punto $z_0 = 0$ no es un punto admisible.

En el caso de la aproximación optimista, el seguidor escoge la opción más conveniente para el líder que desea minimizar $x - z$. Por tanto, cuando $x = 0$ y $y = 5$ la región factible es el segmento entre los puntos $(0,5,1.5)$ y $(0,5,3)$, y para $z < 1.5$ no son puntos admisibles.

Sin embargo, en la aproximación pesimista, el seguidor escoge la opción que menos le conviene al líder, el punto $(5,0,z_0)$. En este caso, la región factible es el segmento entre los puntos $(5,0,2)$ y $(5,0,3)$, y para $z < 2$ no son puntos admisibles.

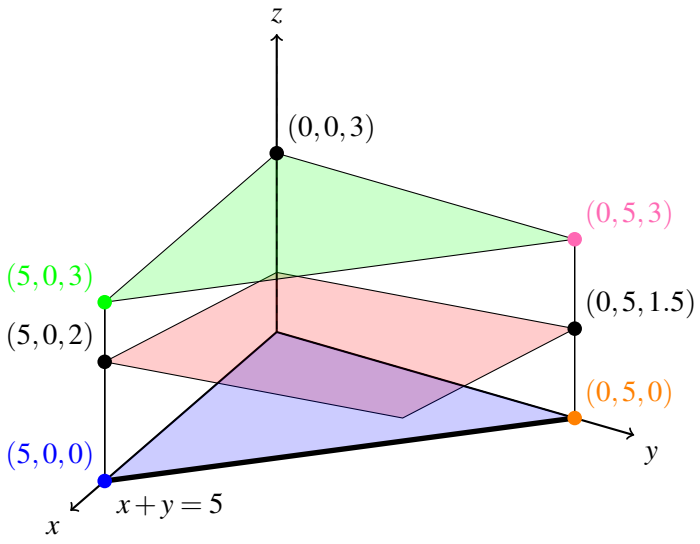


Figura 1.3: Ejemplo ilustrativo de un problema LBP con restricciones *coupling*.

1.3. Reformulación del LBP en un problema de un nivel

Un problema de especial interés relacionado con el problema (1.2) es el denominado *problema relajado*, [3] que no considera el problema de optimización del nivel inferior:

$$\begin{array}{ll} \min\limits_{\mathbf{x},\mathbf{y}} & \mathbf{c_1 x} + \mathbf{d_1 y} \\ \text{sujeto a:} & A_1\mathbf{x} + B_1\mathbf{y} \leq \mathbf{b_1} \\ & A_2\mathbf{x} + B_2\mathbf{y} \leq \mathbf{b_2} \\ & \mathbf{x} \geq \mathbf{0_{n_1}}, \quad \mathbf{y} \geq \mathbf{0_{n_2}} \end{array} \tag{1.3}$$

Si la solución al problema relajado es factible binivel, el problema se considera trivial. En caso contrario, el valor óptimo proporcionado por la solución óptima del problema relajado, proporciona una cota inferior del valor óptimo del problema binivel (1.2).

Para abordar el problema binivel (1.2) se han propuesto en la literatura diferentes aproximaciones. Una de ellas consiste en caracterizar el conjunto de soluciones óptimas del problema del nivel inferior. Al tratarse de un problema lineal, se va a utilizar la teoría de la dualidad [1]. En el Anexo A se incluye la relación entre los problemas primal y dual, junto con un ejemplo.

El problema del nivel inferior en (1.2) se va a sustituir por las condiciones de Karush–Kuhn–Tucker (KKT) que caracterizan a las soluciones óptimas de dicho problema. De esta forma, el problema LBP se podrá resolver como un problema de un solo nivel.

Una vez que el nivel superior fija el valor de las variables que controla, $\mathbf{x} \in R_1$, el problema lineal del nivel inferior queda formulado como sigue:

$$\begin{array}{ll} \min\limits_{\mathbf{y}} & \mathbf{d_2 y} \\ \text{sujeto a:} & B_2\mathbf{y} \leq \mathbf{b_2} - A_2\mathbf{x} \\ & \mathbf{y} \geq \mathbf{0_{n_2}} \end{array} \tag{1.4}$$

El problema dual de (1.4) es:

$$\begin{array}{ll} \max\limits_{\mathbf{u}} & \mathbf{u}(A_2\mathbf{x} - \mathbf{b_2}) \\ \text{sujeto a:} & \mathbf{u}B_2 \geq -\mathbf{d_2} \\ & \mathbf{u} \geq \mathbf{0_{m_2}} \end{array} \tag{1.5}$$

donde $\mathbf{u}$ es el vector fila m_2 de variables duales, cada una asociada con una de las restricciones del problema primal. Aplicando la teoría de la dualidad, dado un valor $\mathbf{x} \in R_1$ fijado por el nivel superior, $\mathbf{y} \in \mathbb{R}^{n_2}$ es solución óptima del problema (1.4) si y solo si existe $\mathbf{u} \in \mathbb{R}^{m_2}$, de variables duales, tal que se verifican las condiciones de KKT, que se muestran a continuación:

$$\begin{array}{l} B_2\mathbf{y} \leq \mathbf{b_2} - A_2\mathbf{x} \\ \mathbf{y} \geq \mathbf{0_{n_2}} \\ \mathbf{u}B_2 \geq -\mathbf{d_2} \\ \mathbf{u} \geq \mathbf{0_{m_2}} \\ (\mathbf{d_2} + \mathbf{u}B_2)\mathbf{y} = 0 \\ \mathbf{u}(\mathbf{b_2} - A_2\mathbf{x} - B_2\mathbf{y}) = 0 \end{array} \tag{1.6}$$

Notar que, las dos últimas restricciones son la suma de productos de dos términos no negativos, lo que permite reformular el problema (1.2), asumiendo la aproximación optimista, como un problema de un solo nivel, en el que se despliegan las dos últimas restricciones en (1.6) en dos conjuntos de n_2 y m_2 restricciones, respectivamente:

$$\begin{array}{ll} \min\limits_{x,y} & \mathbf{c_1 x} + \mathbf{d_1 y} \\ \text{sujeto a:} & A_1\mathbf{x} + B_1\mathbf{y} \leq \mathbf{b_1} \\ & A_2\mathbf{x} + B_2\mathbf{y} \leq \mathbf{b_2} \\ & \mathbf{u}B_2 \geq -\mathbf{d_2} \\ & (\mathbf{d_2} + \mathbf{u}B_2)_i y_i = 0, \qquad i = 1,\ldots,n_2 \\ & u_j(\mathbf{b_2} - A_2\mathbf{x} - B_2\mathbf{y})_j = 0, \quad j = 1,\ldots,m_2 \\ & \mathbf{x} \geq \mathbf{0_{n_1}}, \mathbf{y} \geq \mathbf{0_{n_2}}, \mathbf{u} \geq \mathbf{0_{m_2}} \end{array} \tag{1.7}$$

donde dado un vector $\mathbf{a}$, se ha denotado por a_k a la componente k-ésima del vector.

El problema (1.7) es no lineal por el conjunto de restricciones $(\mathbf{d_2} + \mathbf{u}B_2)_i y_i = 0$, $i = 1,\ldots,n_2$ y $u_j(\mathbf{b_2} - A_2\mathbf{x} - B_2\mathbf{y})_j = 0$, $j = 1,\ldots,m_2$. Estas restricciones garantizan que al menos uno de los dos términos tiene que ser cero. Para linealizar cada una de estas restricciones, se introduce una variable binaria y se sustituye dicha restricción por dos restricciones adicionales. Sean las variables binarias, α_i y

β_j, $i = 1,\ldots,n_2,\ j = 1,\ldots,m_2$:

$$\alpha_i = \begin{cases} 1 & \text{si } y_i = 0 \\ 0 & \text{en otro caso} \end{cases} \qquad \beta_j = \begin{cases} 1 & \text{si } u_j = 0 \\ 0 & \text{en otro caso} \end{cases}$$

En el caso del primer conjunto de restricciones, $i = 1,\ldots,n_2$:

$$(\mathbf{d_2} + \mathbf{u}B_2)_i y_i = 0 \Longleftrightarrow \begin{array}{l} (\mathbf{d_2} + \mathbf{u}B_2)_i \leq M_1 \alpha_i \\ y_i \leq M_2(1-\alpha_i) \end{array}$$

donde M_1, M_2 son constantes suficientemente grandes. Cuando $\alpha_i = 0$, la primera restricción garantiza que $(\mathbf{d_2} + \mathbf{u}B_2)_i \leq 0$ luego $(\mathbf{d_2} + \mathbf{u}B_2)_i = 0$, por ser éste un término no negativo, y en la segunda restricción, el valor de M_2 ha de ser una cota válida para y_i. Cuando $\alpha_i = 1$, la segunda restricción garantiza que $y_i = 0$, ya que y_i es una variable no negativa. En la primera restricción, $(\mathbf{d_2} + \mathbf{u}B_2)_i \leq M_1$, se verifica por ser M_1 una cota válida para $(\mathbf{d_2} + \mathbf{u}B_2)_i$.

Del mismo modo, el segundo conjunto de restricciones, $j = 1,\ldots,m_2$:

$$u_j(\mathbf{b_2} - A_2\mathbf{x} - B_2\mathbf{y})_j = 0 \Longleftrightarrow \begin{array}{l} (\mathbf{b_2} - A_2\mathbf{x} - B_2\mathbf{y})_j \leq M_3(1-\beta_j) \\ u_j \leq M_4 \beta_j \end{array}$$

donde M_3 y M_4 son cotas válidas para $(\mathbf{b_2} - A_2\mathbf{x} - B_2\mathbf{y})_j$ y u_j, respectivamente. En este caso, si $\beta_j = 0$ se garantiza que $u_j = 0$ y si $\beta_j = 1$, que la holgura $(\mathbf{b_2} - A_2\mathbf{x} - B_2\mathbf{y})_j = 0$.

1.4. Contribución del TFG

En este TFG se aborda el problema de ubicación de instalaciones para la gestión de residuos urbanos mediante la optimización binivel. Se ha revisado la literatura sobre este tipo de modelos y estudiado con mayor detalle la formulación matemática de tres modelos binivel que atienden a diferentes características de los sistemas de gestión de residuos. En los tres modelos, se presenta un líder o autoridad municipal y uno o varios seguidores. Es posible que los seguidores sean directamente los clientes que utilizan las instalaciones, o puede haber un único seguidor, por ejemplo, una empresa que hace de intermediario y asigna a los clientes qué instalación han de utilizar. Además, también es posible considerar la gestión de varios tipos de residuos (papel, plástico, vidrio, etc.).

En este TFG, se ha propuesto una formulación alternativa para los dos modelos binivel en los que se trata con un solo tipo de residuos. En uno de ellos hay un único decisor, y en el segundo de los modelos, hay múltiples seguidores, ya que son los clientes quienes utilizan directamente las instalaciones.

En la resolución de los tres modelos, los autores de los trabajos proponen un algoritmo heurístico, es decir, utilizan un método que solo proporciona una solución aproximada sin garantía de ser óptima. En este TFG, se ha reformulado el modelo de un único seguidor, a partir de la formulación alternativa propuesta, como un problema de un solo nivel, aplicando los resultados de la teoría de la dualidad. El problema, una vez reformulado, precisa de la linealización de un conjunto de restricciones, que requieren de la obtención de cotas válidas. Tras demostrar la validez de las cotas, el problema lineal entero mixto se resuelve con un software de optimización.

Finalmente, la experiencia computacional se ha realizado mediante el uso del lenguaje de programación Python y al software de optimización matemática Gurobi. En una primera parte, se han proporcionado varios ejemplos ilustrativos. En la segunda parte de la experimentación se han generado diversos escenarios para evaluar el efecto en el tiempo computacional y las soluciones propuestas. En particular, se ha estudiado el comportamiento del problema variando dos parámetros relevantes, la demanda mínima a satisfacer de los clientes y el precio de los residuos.

Capítulo 2

Modelos de optimización binivel en gestión de residuos

En este capítulo se realiza una revisión de la literatura sobre algunos problemas en gestión de residuos que se han abordado a través de modelos de optimización binivel. En particular, se introducen tres modelos de localización de instalaciones en sistemas de gestión de residuos con diferentes características. En dos de los modelos se va a proponer una formulación alternativa y extender a un caso más general. Además, el primer modelo, se va a reformular como un modelo de optimización de un solo nivel para así resolverlo con un software de optimización.

2.1. Descripción del problema

En esta sección se aborda el problema de ubicación de instalaciones para la gestión de residuos a partir del modelo introducido en [6], BFLP, (del inglés Bilevel Facility Location Problem). Dado un conjunto de ubicaciones potenciales, este problema consiste en determinar qué instalaciones han de abrirse y con qué capacidad para atender la demanda de un conjunto de clientes mientras se minimizan los costes de gestión del sistema.

Sea $J = \{1,\dots,m\}$ el conjunto de instalaciones potenciales. El coste fijo de abrir una instalación $j \in J$, es f_j. Además, se denota por g_j, el coste por unidad de capacidad en cada instalación $j \in J$. Cuando una sola empresa decide qué instalaciones se abren, es posible asumir que $g_j = g$, $j \in J$, es decir, que el coste por unidad de capacidad sea el mismo para todas las instalaciones $j \in J$. Por otro lado, sea $I = \{1,\dots,n\}$, el conjunto de clientes. Para cada cliente $i \in I$, se define d_i como la demanda de servicio del cliente, y p_i el precio que paga el cliente i por unidad de demanda atendida. Además, sea c_{ij} el coste por el envío de una unidad de demanda desde el cliente i hasta la instalación $j \in J$, la ganancia unitaria por la gestión de la demanda del cliente i, atendida por la instalación $j \in J$, se obtiene como $\pi_{ij} = p_i - c_{ij}$.

Caramia y Costa [4] utilizan el BFLP para modelar un sistema de gestión de residuos con una estructura jerárquica. El nivel superior decide qué instalaciones se abren, esto es, dónde se abren y cuál es la capacidad de las mismas, con el objetivo de minimizar el coste total que conlleva abrir las instalaciones necesarias. Luego, las variables de decisión controladas por el líder son las variables binarias que deciden sobre la apertura de las instalaciones y las variables continuas no negativas, que representan la capacidad que se le asigna a cada instalación:

$$x_j = \begin{cases} 1 & \text{si la instalación j se abre } j \in J \\ 0 & \text{en otro caso} \end{cases}, \qquad z_j \geq 0, \qquad j \in J$$

El nivel inferior va a decidir sobre qué instalaciones va a utilizar, de las que haya abierto el nivel superior. Por ello, las variables de decisión controladas por el nivel inferior, son las variables continuas no negativas que representan el porcentaje de demanda enviada por el cliente i a la instalación j:

$$y_{ij} \geq 0, \quad i \in I,\ j \in J$$

De ahora en adelante, para simplificar la notación, al conjunto de variables $\{x_j\}_{j\in J}$, $\{y_{ij}\}_{i\in I,\, j\in J}$ y $\{z_j\}_{j\in J}$, se les denotará como $\mathbf{x}$, $\mathbf{y}$ y $\mathbf{z}$, respectivamente.

Con la notación introducida, se van a presentar tres modelos diferentes que proporcionan sistemas de gestión alternativos.

2.1.1. Modelo de ubicación de instalaciones con una empresa en el nivel inferior

En este primer caso, todos los clientes son gestionados por una empresa [6] que toma la decisión correspondiente a todos ellos. La empresa, en el nivel inferior, reacciona a cada decisión del nivel superior, es decir, según las instalaciones que decida abrir y su capacidad, la empresa determina cómo va a realizar el envío de los residuos de cada cliente $i \in I$ a cada instalación abierta $j \in J$.

El nivel inferior, la empresa, una vez informado de las instalaciones abiertas y su capacidad, trata de satisfacer la demanda de los clientes, atendiendo a su objetivo de maximizar beneficios y determinando qué instalación va a satisfacer la demanda de cada cliente. Dado que la empresa no está obligada a satisfacer el total de la demanda de los clientes y el nivel superior no puede controlar ni aplicar ninguna sanción por ello, se incluye una restricción en el nivel superior que garantiza que a cada cliente $i \in I$ se le ha de satisfacer al menos una proporción mínima de demanda λ_i conocida.

La formulación matemática de este primer modelo, denotado por **BFLP-e**, es la siguiente:

$$\min_{\mathbf{x},\mathbf{z},\mathbf{y}} \quad \sum_{j\in J} f_j x_j + \sum_{j\in J} g z_j \tag{2.1a}$$

$$\text{sujeto a:} \quad \sum_{j\in J} y_{ij} \geq \lambda_i, \qquad i \in I \tag{2.1b}$$

$$x_j \in \{0,1\}, \qquad z_j \geq 0, \qquad j \in J \tag{2.1c}$$

dado $(\mathbf{x}, \mathbf{z})$, y resuelve:

$$\max_{\mathbf{y}} \quad \sum_{i\in I}\sum_{j\in J} d_i \pi_{ij} y_{ij} \tag{2.1d}$$

$$\text{sujeto a:} \quad \sum_{j\in J} y_{ij} \leq 1, \qquad i \in I \tag{2.1e}$$

$$\sum_{i\in I} d_i y_{ij} \leq z_j, \qquad j \in J \tag{2.1f}$$

$$y_{ij} \leq x_j, \qquad i \in I, \quad j \in J \tag{2.1g}$$

$$y_{ij} \geq 0, \qquad i \in I, \quad j \in J \tag{2.1h}$$

La función objetivo del nivel superior (2.1a) consiste en minimizar el coste fijo de apertura de las instalaciones y el coste total correspondiente a la capacidad de las instalaciones abiertas.

El conjunto de restricciones (2.1b) garantiza que para cada cliente $i \in I$ se satisfaga, entre todas las instalaciones, una proporción mínima λ_i de la demanda. Las restricciones (2.1c) indican el carácter binario de las variables $\mathbf{x}$ y que las variables $\mathbf{z}$ sean continuas no negativas.

El problema de la empresa queda formulado por (2.1d)-(2.1h). La función objetivo (2.1d) consiste en maximizar el beneficio neto obtenido por la atención de la demanda de todos los clientes por todas las instalaciones.

El conjunto de restricciones (2.1e), asegura que la demanda de cada uno de los clientes, $i \in I$ no se exceda. Las restricciones (2.1f) garantizan que para cada instalación no se supere la capacidad disponible, y las restricciones (2.1g) no permiten a la empresa asignar un cliente i a una instalación j en el caso de que esta última no este abierta, ya que si $x_j = 0$, la instalación j está cerrada, y por la correspondiente restricción (2.1g) todas las variables $y_{ij} = 0, \;\; i \in I, \;\; j \in J$. Finalmente, las restricciones (2.1h) son de signo para las variables del nivel inferior. Aunque las variables $\mathbf{y}$ son continuas no negativas, expresan

una proporción, por ello tomarán siempre un valor menor o igual a 1. Esto queda garantizado a partir de las restricciones (2.1e).

Cabe destacar que, en este modelo, las restricciones (2.1b) son *coupling*. Esto significa que es posible que ciertas decisiones del nivel superior sobre la apertura y la capacidad de las instalaciones, al resolver el problema la empresa, el valor de las variables **y**, no verifiquen las restricciones (2.1b). En ese caso, no se considerará una decisión admisible del nivel superior.

2.1.2. Modelo de ubicación de instalaciones con clientes en el nivel inferior

En este caso, los clientes toman sus propias decisiones, es decir, cada uno toma sus decisiones sin ser gestionados por una empresa [4]. Se denota por **BFLP-u** al modelo binivel correspondiente. En la Figura 2.1 se muestra la estructura binivel de este sistema de gestión de residuos.

La compañía municipal encargada de la recogida y eliminación de residuos se sitúa en el nivel superior de la toma de decisiones y su objetivo consiste en minimizar los costes de localización y de capacidad asignada a las instalaciones abiertas. En el nivel inferior, a diferencia del modelo **BFLP-e**, se sitúan cada una de las familias y trabajadores residentes, que han de utilizar adecuadamente los recursos, para lo cual, son incentivados con algún tipo de utilidad. Por ejemplo, una posible forma de incentivar es que las familias puedan reducir las tasas a pagar si realizan una buena gestión de los residuos.

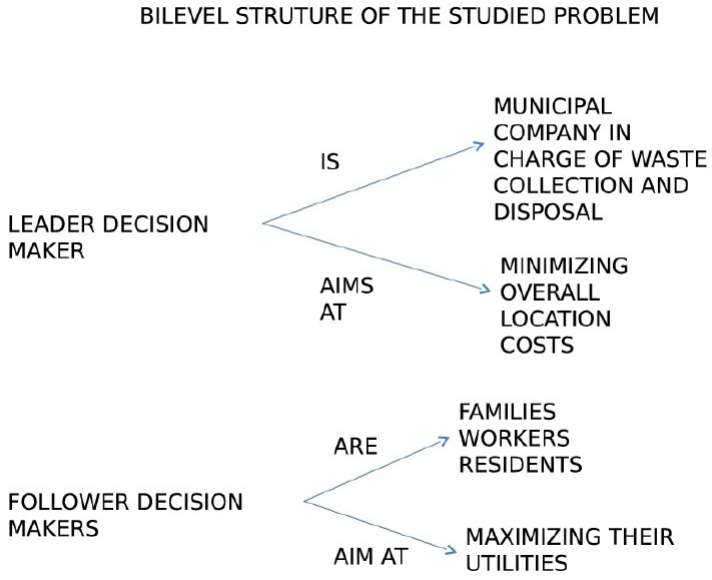


Figura 2.1: Esquema conceptual del modelo **BFLP-u**. Fuente: [4]

En este caso, al considerar los clientes de forma individual, el nivel superior va asignar una capacidad en cada instalación $j \in J$ a cada cliente $i \in I$, que se representa por las nuevas variables continuas no negativas:

$$z_{ij} \geq 0, \quad i \in I, \ , j \in J$$

Además, se va a asumir que cada cliente i solo puede ser atendido por una instalación. Para ello, se introducen las siguientes variables binarias:

$$x'_{ij} = \begin{cases} 1 & \text{si la instalación } j \text{ atiende al cliente } i \\ 0 & \text{en otro caso} \end{cases} \qquad i \in i, \ j \in J$$

De ahora en adelante, como en el problema anterior (2.1), por simplificar la notación, el conjunto $\{z_{ij}\}_{i\in I,\ j\in J}$ se denotará como **z**.

La formulación matemática del modelo **BFLP-u** es la siguiente:

$$\min_{\mathbf{x},\mathbf{z},\mathbf{y}} \quad \sum_{j\in J} f_j x_j + \sum_{i\in I}\sum_{j\in J} g z_{ij} \tag{2.2a}$$

$$\text{sujeto a:} \quad \sum_{j\in J} y_{ij} \geq \lambda_i, \qquad i \in I \tag{2.2b}$$

$$z_{ij} \leq d_i x_j, \qquad i \in I, \quad j \in J \tag{2.2c}$$

$$x_j \in \{0,1\}, \quad z_{ij} \geq 0, \qquad i \in I, \quad j \in J \tag{2.2d}$$

dado $(\mathbf{x}, \mathbf{z})$, cada cliente $i \in I$, resuelve:

$$\max_{\mathbf{y}} \quad \sum_{j\in J} d_i \pi_{ij} y_{ij} \tag{2.2e}$$

$$\text{sujeto a:} \quad \sum_{j\in J} x'_{ij} \leq 1 \tag{2.2f}$$

$$d_i y_{ij} \leq z_{ij}, \qquad j \in J \tag{2.2g}$$

$$y_{ij} \leq x'_{ij}, \qquad j \in J \tag{2.2h}$$

$$x'_{ij} \in \{0,1\}, \quad y_{ij} \geq 0, \qquad j \in J \tag{2.2i}$$

La función objetivo del nivel superior (2.2a) y el conjunto de restricciones (2.2b) coinciden con las correspondientes al modelo (2.1), ya que el nivel superior continúa minimizando los costes de apertura y capacidad, y desea garantizar una demanda mínima atendida a cada cliente. La restricción (2.2c) garantiza que la capacidad de cada instalación $j \in J$ asignada a cada cliente i es mayor que 0 solo si la instalación j ha sido abierta. Las restricciones (2.2d) indican el carácter binario de las variables $\mathbf{x}$ y que las variables $\mathbf{z}$ sean continuas no negativas.

El problema para cada uno de los clientes queda formulado por (2.2e)-(2.2i). La función objetivo (2.2e) consiste en maximizar la utilidad del cliente i.

El conjunto de restricciones (2.2f) asegura que cada cliente i solo puede ser atendido por una instalación. Las restricciones (2.2g) imponen que el cliente i no supere la capacidad que se le ha asignado en cada instalación $j \in J$. Dado el cliente, si $x'_{ij} = 0$ entonces el cliente i no puede ser atendido por la instalación j. Esto se garantiza por la restricción (2.2h) correspondiente. Finalmente, las restricciones (2.2i) indican el carácter binario de las variables x'_{ij} y que las variables y sean continuas no negativas. Al igual que en el problema (2.1), las variables y son continuas no negativas cuyo valor está acotado por 1, debido a las restricciones (2.2h), expresando la proporción de demanda atendida de un cliente por una instalación.

En este modelo también se incluyen las restricciones *coupling* (2.2b). Por tanto, de nuevo, es posible que ciertas decisiones del nivel superior sobre la apertura y la capacidad de las instalaciones, provoquen una reacción en los clientes, tal que el valor propuesto de las variables $\mathbf{y}$, no verifique las restricciones (2.2b). En ese caso, no se considerará como decisión admisible del nivel superior de decisión.

El modelo anterior (2.2) se puede extender, de forma que, cada cliente pueda ser atendido por más de una instalación como en el modelo previo (2.1). En la literatura, se conocen por modelos *single source* y *multiple source*, respectivamente. La formulación alternativa matemática del nuevo modelo es la siguiente:

$$\min_{\mathbf{x},\mathbf{z},\mathbf{y}} \quad \sum_{j\in J} f_j x_j + \sum_{i\in I}\sum_{j\in J} g z_{ij} \tag{2.3a}$$

$$\text{sujeto a:} \quad \sum_{j\in J} y_{ij} \geq \lambda_i, \qquad i \in I \tag{2.3b}$$

$$z_{ij} \leq d_i x_j, \qquad i \in I, \quad j \in J \tag{2.3c}$$

$$x_j \in \{0,1\}, \quad z_{ij} \geq 0, \qquad i \in I, \quad j \in J \tag{2.3d}$$

dado $(\mathbf{x}, \mathbf{z})$, cada cliente $i \in I$, resuelve:

$$\max_{\mathbf{y}} \quad \sum_{j\in J} d_i \pi_{ij} y_{ij} \tag{2.3e}$$

$$\text{sujeto a:} \quad \sum_{j\in J} y_{ij} \leq 1, \tag{2.3f}$$

$$d_i y_{ij} \leq z_{ij}, \qquad j \in J \tag{2.3g}$$

$$y_{ij} \geq 0, \qquad j \in J \tag{2.3h}$$

Notar que en el modelo (2.3) se han suprimido las variables $\mathbf{x}'$. La función objetivo del líder (2.3a) y el conjunto de restricciones (2.3b), (2.3c) y (2.3d) coinciden con las correspondientes al modelo (2.2), ya que el nivel superior continúa minimizando los costes de apertura y capacidad, y desea garantizar una demanda mínima atendida a cada cliente.

El problema de los clientes queda formulado por (2.3e)-(2.3h). La función objetivo (2.3e) y la restricción (2.3g) se mantienen como en el problema (2.2). Las restricciones (2.2f) y (2.2h) se han suprimido y se han sustituido por las restricciones (2.3f), que garantizan que la demanda atendida del cliente i por las instalaciones j, no se exceda. Finalmente, las restricciones (2.3h) son de signo para las variables del nivel inferior.

2.1.3. Modelo de ubicación de instalaciones de gestión de residuos diferenciados y clientes en el nivel inferior

Este tercer modelo, denotado por **BFLP-r**, extiende el modelo anterior **BFLP-u**, al considerar clientes que toman decisiones en el nivel inferior y añadir la recolección de residuos diferenciados para el reciclaje. Sea K el conjunto de tipos de residuos. En este caso, en el nivel superior, una autoridad municipal, trata de incentivar a los clientes a utilizar centros de recolección de residuos de varios tipos [5].

Al considerar residuos diferenciados, cada instalación $j \in J$ tiene un conjunto asociado K_j tal que, en dicha instalación se puede reciclar el residuo k si $k \in K_j$ y la capacidad viene dada por el parámetro Q_j^k. En este modelo, cada cliente $i \in I$ tiene una demanda de gestión d_i^k de residuo $k \in K$. Se asume que el cliente i no conoce la capacidad de las instalaciones $j \in J$ para cada residuo $k \in K_j$.

Para cada residuo $k \in K$ se define el conjunto $J^k \subseteq J$ de instalaciones en las que se puede reciclar el residuo k.

En este modelo se introducen nuevas variables continuas no negativas controladas por el nivel superior para representar la capacidad que el líder reserva para el cliente i, en cada instalación $j \in J$ para el residuo $k \in K$:

$$z_{ij}^k \geq 0, \qquad i \in I,\ j \in J,\ k \in K_j$$

En el nivel inferior, se define una nueva variable continua no negativa que representa el porcentaje de demanda d_i^k del cliente i satisfecha en la instalación j del residuo k:

$$y_{ij}^k \geq 0, \qquad i \in I,\ j \in J,\ k \in K_j$$

Como en el modelo (2.2) se impone que cada cliente $i \in I$ puede ser asignado para atender su demanda de residuo de tipo $k \in K$ a lo sumo por una instalación $j \in J^k$, por lo que se introducen las siguientes

variables binarias:

$$x'^{k}_{ij} = \begin{cases} 1 & \text{si la instalación } j \text{ atiende al cliente } i \text{ para el residuo } k, \quad i \in i,\ k \in K,\ j \in J^k \\ 0 & \text{en otro caso} \end{cases}$$

Por último, λ^k representa la proporción mínima de demanda del tipo de residuo $k \in K$ que ha de ser atendida en todos los clientes; g^k_j es el coste de instalar una unidad de capacidad del residuo $k \in K$ en la instalación $j \in J^k$; y U_{ij} es la utilidad para cada cliente $i \in I$ por el hecho de utilizar la instalación $j \in J$. Notar que, en este modelo, los costes de instalar capacidad en una instalación, a diferencia de los modelos anteriores, no son iguales.

La formulación matemática del modelo **BFLP-r** es la siguiente:

$$\min_{\mathbf{x},\mathbf{z},\mathbf{y}} \quad \sum_{j\in J} f_j x_j + \sum_{j\in J}\sum_{k\in K_j} g^k_j \sum_{i\in I} z^k_{ij} \tag{2.4a}$$

$$\text{sujeto a:} \quad \sum_{j\in J^k} y^k_{ij} \geq \lambda^k, \qquad i \in I, \quad k \in K \tag{2.4b}$$

$$z^k_{ij} \leq d^k_i x_j, \qquad i \in I, \quad j \in J, \quad k \in K_j \tag{2.4c}$$

$$x_j \in \{0,1\}, \quad z^k_{ij} \geq 0, \qquad i \in I, \quad j \in J, \quad k \in K_j \tag{2.4d}$$

dado $(\mathbf{x}, \mathbf{z})$, cada cliente $i \in I$, resuelve:

$$\max_{\mathbf{y}} \quad \sum_{j\in J}\sum_{k\in K_j} U_{ij} d^k_i y^k_{ij} \tag{2.4e}$$

$$\text{sujeto a:} \quad \sum_{j\in J^k} x'^{k}_{ij} \leq 1, \qquad k \in K \tag{2.4f}$$

$$d_{ik} y^k_{ij} \leq z^k_{ij}, \qquad j \in J, \quad k \in K_j \tag{2.4g}$$

$$y^k_{ij} \leq x'^{k}_{ij}, \qquad j \in J, \quad k \in K_j \tag{2.4h}$$

$$x'^{k}_{ij} \leq 1 - x'^{k'}_{ij'}, \qquad j, j' \in J,\ j' \neq j,\ k, k' \in K_{j'},\ k \neq k' \tag{2.4i}$$

$$y^k_{ij} \geq 0, \qquad j \in J, \quad k \in K_j \tag{2.4j}$$

$$x'_{ikj} \in \{0,1\}, \qquad j \in J, \quad k \in K_j \tag{2.4k}$$

La función objetivo del nivel superior (2.4a) consiste en minimizar el coste fijo de apertura de las instalaciones y el coste total correspondiente a la capacidad de las instalaciones abiertas de todos los tipos de residuos $k \in K$.

El conjunto de restricciones (2.4b) garantiza, para cada cliente $i \in I$, que el porcentaje reciclado de la demanda de residuos de tipo $k \in K$ es al menos la proporción mínima λ^k. El conjunto de restricciones (2.4c) establece que la capacidad z^k_{ij} asignada a un cliente i de tipo de residuo k será 0 si la instalación j está cerrada y a lo más, la demanda de ese tipo de residuo del cliente, si está abierta. Las restricciones (2.4d) indican el carácter binario de las variables x_j y que las variables z^k_{ij} sean continuas no negativas.

El problema que resuelve cada cliente $i \in I$ es (2.4e)-(2.4k). La función objetivo (2.4e) consiste en maximizar la utilidad derivada del uso de las instalaciones $j \in J$ para gestionar los residuos $k \in J^k$.

El conjunto de restricciones (2.4f) garantiza para cada tipo de residuo $k \in K$ que el cliente solo puede ser asignado a lo más a una instalación $j \in J_k$. Para cada instalación $j \in J$ y tipo de residuo gestionado en ella $k \in K_j$, la correspondiente restricción (2.4g) asegura que a lo más se utiliza la capacidad z^k_{ij} asignada. En el caso de que una instalación $j \in J$ no haya sido asignada al cliente para atender la demanda de un tipo de residuo, esto es, $x'^{k}_{ij} = 0$, dicha instalación no atenderá ninguna demanda del cliente por la correspondiente restricción (2.4h).

Dada una instalación $j \in J$, el conjunto de restricciones (2.4i) garantiza que si el cliente es atendido por otra instalación j' diferente para la demanda de un residuo k' que también es tratado en la instalación j, entonces el cliente no puede ser atendido en dicha instalación $j \in J$ para ningún otro residuo $k \in K_j$.

Finalmente, las restricciones (2.4j) y (2.4k) indican el carácter binario de la variable $x_{ij}^{\prime k}$ y que las variables y_{ij}^{k} sean continuas no negativas.

2.2. Reformulación del modelo binivel BFLP-e

De los tres modelos propuestos se va a considerar el modelo **BFLP-e** (2.1) de ubicación de instalaciones donde en el nivel inferior hay una empresa. Antes de reformularlo como un problema de un solo nivel utilizando la aproximación dada en la sección 1.3, se plantea una formulación alternativa.

En el modelo binivel (2.1), la demanda satisfecha a cada cliente $i \in I$ desde una instalación $j \in J$ se expresa como un porcentaje. En el tratamiento del problema resulta más intuitivo que la variable y_{ij} represente directamente la cantidad de demanda satisfecha. Por ello, se propone una nueva formulación (denotada **BFLP-E**), del modelo binivel (2.1):

$$\min_{\mathbf{x},\mathbf{z},\mathbf{y}} \quad \sum_{j\in J} f_j x_j + \sum_{j\in J} g z_j \tag{2.5a}$$

$$\text{sujeto a:} \quad \sum_{j\in J} y_{ij} \geq \lambda_i d_i, \qquad i \in I \tag{2.5b}$$

$$x_j \in \{0,1\}, \qquad z_j \geq 0, \qquad j \in J \tag{2.5c}$$

dado $(\mathbf{x}, \mathbf{z})$, $\mathbf{y}$ resuelve:

$$\max_{\mathbf{y}} \quad \sum_{i\in I}\sum_{j\in J} \pi_{ij} y_{ij} \tag{2.5d}$$

$$\text{sujeto a:} \quad \sum_{j\in J} y_{ij} \leq d_i, \qquad i \in I \tag{2.5e}$$

$$\sum_{i\in I} y_{ij} \leq z_j \qquad j \in J \tag{2.5f}$$

$$y_{ij} \leq d_i x_j, \qquad i \in I, \quad j \in J \tag{2.5g}$$

$$y_{ij} \geq 0, \qquad i \in I, \quad j \in J \tag{2.5h}$$

donde el conjunto de restricciones (2.5b) garantiza, para cada cliente, que se satisfaga la cantidad mínima de demanda, en sustitución del conjunto de restricciones (2.1b). Las restricciones (2.5c) se mantienen iguales que (2.1c).

En el problema del nivel inferior, (2.5d)-(2.5h), la función objetivo (2.5d), maximiza el beneficio neto de la asignación de clientes a las instalaciones, en sustitución de la función objetivo (2.1d).
Las restricciones (2.5e) garantizan que para cada cliente $i \in I$, la cantidad recibida de todas las instalaciones, no supera su demanda, lo que corresponde a la restricción (2.1e). El conjunto de restricciones (2.5f) sustituye al conjunto (2.1f), y asegura que la suma de la cantidad de la demanda de todos los clientes, satisfecha por la instalación j, es menor o igual que la capacidad de dicha instalación. La restricción (2.5g), como (2.1g), asegura que la cantidad de demanda del cliente i, satisfecha por la instalación j, es menor o igual que la demanda del cliente i, en caso de que la instalación j resulte abierta. En caso contrario, obliga a que $y_{ij} = 0$, es decir, si la instalación j no es abierta por el nivel superior, $x_j = 0$, y no se puede atender demanda de ningún cliente i. Finalmente, la restricción (2.5h), igual que (2.1h), es de signo para las variables controladas por el nivel inferior.

2.2.1. Caracterización de las soluciones óptimas del problema del nivel inferior

Con el objetivo de caracterizar las soluciones óptimas del problema del nivel inferior, dado un valor de las variables controladas por el nivel superior, $\mathbf{x}$, se considera el conjunto:

$$J(\mathbf{x}) = \{j \in J : x_j = 1\},$$

que identifica el conjunto de instalaciones abiertas.

El problema del nivel inferior se puede reescribir utilizando el conjunto $J(\mathbf{x})$ como:

$$\max_{y} \quad \sum_{i\in I}\sum_{j\in J(\mathbf{x})} \pi_{ij} y_{ij} \tag{2.6a}$$

$$\text{sujeto a:} \quad \sum_{j\in J(\mathbf{x})} y_{ij} \le d_i, \qquad i \in I \tag{2.6b}$$

$$\sum_{i\in I} y_{ij} \le z_j, \qquad j \in J(\mathbf{x}) \tag{2.6c}$$

$$y_{ij} \ge 0, \qquad i \in I, \quad j \in J(\mathbf{x}) \tag{2.6d}$$

donde se mantienen todas restricciones del problema (2.5), a excepción de las restricciones (2.5g). Estas restricciones no son necesarias porque al introducir el conjunto $J(\mathbf{x})$, en el problema (2.6) solo se definen las variables y_{ij} con $i \in I$, $j \in J(\mathbf{x})$.

A continuación, se formula el problema dual del problema (2.6). Sean w_i, $i \in I$, las variables duales asociadas a cada una de las restricciones en (2.6b) y u_j, $j \in J(\mathbf{x})$, las variables duales correspondientes al conjunto de restricciones (2.6c). El problema dual del problema (2.6) se formula:

$$\begin{aligned}
\min_{\mathbf{w},\mathbf{u}} \quad & \sum_{i\in I} w_i d_i + \sum_{j\in J(\mathbf{x})} z_j u_j & & \\
\text{sujeto a:} \quad & w_i + u_j \ge \pi_{ij}, & i \in I, \quad & j \in J(\mathbf{x}) \\
& w_i \ge 0, & & i \in I \\
& u_j \ge 0, & & j \in J(\mathbf{x})
\end{aligned} \tag{2.7}$$

Aplicando la teoría de la dualidad como se ha explicado en la sección 1.3, las condiciones de KKT que caracterizan las soluciones óptimas del problema del nivel inferior son:

$$\begin{aligned}
& \sum_{j\in J(\mathbf{x})} y_{ij} \le d_i, & & i \in I \\
& \sum_{i\in I} y_{ij} \le z_j, & & j \in J(\mathbf{x}) \\
& w_i + u_j \ge \pi_{ij}, & i \in I, \quad & j \in J(\mathbf{x}) \\
& (w_i + u_j - \pi_{ij}) y_{ij} = 0, & i \in I, \quad & j \in J(\mathbf{x}) \\
& \left(d_i - \sum_{j\in J(\mathbf{x})} y_{ij}\right) w_i = 0, & & i \in I \\
& \left(z_j - \sum_{i\in I} y_{ij}\right) u_j = 0 \qquad , & & j \in J(\mathbf{x}) \\
& w_i \ge 0, & & i \in I \\
& u_j \ge 0, & & j \in J(\mathbf{x}) \\
& y_{ij} \ge 0, & i \in I, \quad & j \in J(\mathbf{x})
\end{aligned} \tag{2.8}$$

El conjunto de restricciones (2.8) caracteriza la solución del problema del nivel inferior dado el valor $\mathbf{x}, \mathbf{z}$ de variables del nivel superior. Para reformular el problema binivel (2.5) como un problema de un solo nivel, se sustituye el problema del nivel inferior (2.5d)-(2.5h) por el conjunto de restricciones (2.8). Si bien hay que definir el valor de todas las variables y_{ij}, $i \in I$, $j \in J$ y las correspondientes variables u_j, $j \in J$, garantizando que tomarán el valor 0 si $j \notin J(\mathbf{x})$ y que las correspondientes restricciones en el sistema (2.8) no restrinjan el valor de ninguna variable. Por lo tanto, el problema de optimización binivel (2.5), queda reformulado a un problema de optimización de un solo nivel:

$$\min_{\mathbf{x},\mathbf{y},\mathbf{z}} \quad \sum_{j\in J} f_j x_j + \sum_{j\in J} g z_j \tag{2.9a}$$

$$\text{sujeto a:} \quad \sum_{j\in J} y_{ij} \geq \lambda_i d_i, \qquad i \in I \tag{2.9b}$$

$$\sum_{j\in J} y_{ij} \leq d_i, \qquad i \in I \tag{2.9c}$$

$$\sum_{i\in I} y_{ij} \leq z_j, \qquad j \in J \tag{2.9d}$$

$$y_{ij} \leq d_i x_j, \qquad i \in I, \quad j \in J \tag{2.9e}$$

$$w_i + u_j \geq \pi_{ij} x_j, \qquad i \in I, \quad j \in J \tag{2.9f}$$

$$u_j \leq M x_j, \qquad j \in J \tag{2.9g}$$

$$(w_i + u_j - \pi_{ij}) y_{ij} = 0, \qquad i \in I, \quad j \in J \tag{2.9h}$$

$$w_i \left(d_i - \sum_{j\in J} y_{ij} \right) = 0, \qquad i \in I \tag{2.9i}$$

$$u_j \left(z_j - \sum_{i\in I} y_{ij} \right) = 0, \qquad j \in J \tag{2.9j}$$

$$x_j \in \{0,1\}, \quad z_j \geq 0, \qquad j \in J \tag{2.9k}$$

$$y_{ij} \geq 0, \qquad i \in I, \quad j \in J \tag{2.9l}$$

$$w_i \geq 0, \qquad i \in I \tag{2.9m}$$

$$u_j \geq 0, \qquad j \in J \tag{2.9n}$$

donde M es una constante suficientemente grande. La función objetivo (2.9a) coincide con la función objetivo del nivel superior del problema (2.5). El conjunto de restricciones (2.9b) y (2.9k) son restricciones del nivel superior del problema (2.5). Las restricciones (2.9e) y (2.9g) garantizan que las correspondientes variables $y_{ij} = u_j = 0$ para $i \in I$ y $j \notin J(\mathbf{x})$. El resto de restricciones son las restricciones del conjunto (2.8). En el caso de las restricciones (2.9f) se ha multiplicado el término de la derecha por x_j para que sea una restricción válida para cualquier $j \in J$. De forma que si $j \in J(\mathbf{x})$ entonces $x_j = 1$ y la restricción coincide con la dada en (2.8).

El problema (2.9) es no lineal por los conjuntos de restricciones (2.9h), (2.9i) y (2.9j). Para linealizarlo se introducen tantas variables binarias como restricciones no lineales incluye el modelo. Sean las variables γ_{ij}, α_i y β_j, $i \in I$, $j \in J$ asociadas a las restricciones (2.9h), (2.9i) y (2.9j), respectivamente:

$$\gamma_{ij} = \begin{cases} 1 & \text{si } y_{ij} = 0 \\ 0 & \text{en otro caso} \end{cases} \qquad \alpha_i = \begin{cases} 1 & \text{si } d_i - \sum_{j\in J} y_{ij} = 0 \\ 0 & \text{en otro caso} \end{cases} \qquad \beta_j = \begin{cases} 1 & \text{si } z_j - \sum_{i\in I} y_{ij} = 0 \\ 0 & \text{en otro caso} \end{cases}$$

y se añaden las correspondientes restricciones al problema, siendo finalmente el problema (2.9) formulado como un problema lineal:

$$\min_{\mathbf{x},\mathbf{y},\mathbf{z}} \quad \sum_{j\in J} f_j x_j + \sum_{j\in J} g z_j \tag{2.10a}$$

sujeto a:

$$\sum_{j\in J} y_{ij} \geq \lambda_i d_i, \qquad i \in I \tag{2.10b}$$

$$\sum_{j\in J} y_{ij} \leq d_i, \qquad i \in I \tag{2.10c}$$

$$\sum_{i\in I} y_{ij} \leq z_j, \qquad j \in J \tag{2.10d}$$

$$y_{ij} \leq d_i x_j, \qquad i \in I, \quad j \in J \tag{2.10e}$$

$$u_j \leq M_j^5 x_j, \qquad j \in J \tag{2.10f}$$

$$w_i + u_j \geq \pi_{ij} x_j, \qquad i \in I, \quad j \in J \tag{2.10g}$$

$$y_{ij} \leq M_i^1 (1-\gamma_{ij}), \qquad i \in I, \quad j \in J \tag{2.10h}$$

$$d_i - \sum_{j\in J} y_{ij} \leq M_i^2 (1-\alpha_i), \qquad i \in I \tag{2.10i}$$

$$z_j - \sum_{i\in I} y_{ij} \leq M^3 (1-\beta_j), \qquad j \in J \tag{2.10j}$$

$$w_i \leq M_i^4 \alpha_i, \qquad i \in I \tag{2.10k}$$

$$u_j \leq M_j^5 \beta_j, \qquad j \in J \tag{2.10l}$$

$$w_i + u_j - \pi_{ij} \leq M_{ij}^6 \gamma_{ij}, \qquad i \in I, \quad j \in J \tag{2.10m}$$

$$x_j \in \{0,1\}, \quad z_j \geq 0, \quad y_{ij} \geq 0, \qquad i \in I, \quad j \in J \tag{2.10n}$$

$$w_i \geq 0, \quad u_j \geq 0, \qquad i \in I, \quad j \in J \tag{2.10ñ}$$

$$\alpha_i, \beta_j, \gamma_{ij} \in \{0,1\}, \qquad i \in I, \quad j \in J \tag{2.10o}$$

Los conjuntos de restricciones (2.10h)-(2.10m) se han introducido junto con las variables binarias γ_{ij}, α_i y β_j, $i \in I$, $j \in J$, para linealizar los términos no lineales. En particular, las restricciones (2.10h) y (2.10m) corresponden a las restricciones (2.9h). Las restricciones (2.10i) y (2.10k) sustituyen a las restricciones (2.9i). Las restricciones (2.10j) y (2.10l) garantizan que se cumplan las restricciones (2.9j). Los valores de las constantes M se han de calcular para cada una de las restricciones para garantizar que sean válidas.

2.2.2. Cálculo de las gran M

En esta sección, con el objetivo de resolver el problema (2.10), se van a buscar cotas adecuadas para los valores de $M_i^1, M_i^2, M_j^3, M_i^4, M_j^5, M^6$ que sean suficientemente grandes para garantizar que no se eliminan soluciones factibles binivel y suficientemente pequeñas para que la solución obtenida sea factible binivel [8].

Lema 2.2.1. Una cota válida en las restricciones (2.10h) es $M_i^1 = d_i,\ i \in I$.

Demostración. Dado $i \in I$, $j \in J$, se obtiene:

$$y_{ij} \leq \sum_{j\in J} y_{ij} \leq d_i$$

donde la segunda desigualdad es válida por la restricción (2.9c). Por tanto, $M_1^i = d_i, i \in I$ es una cota válida. □

Lema 2.2.2. Una cota válida en las restricciones (2.10i) es $M_i^2 = (1-\lambda_i)d_i,\ i \in I$.

Demostración. Dado $i \in I$,

$$d_i - \sum_{j\in J} y_{ij} \leq d_i - \lambda_i d_i = (1-\lambda_i)d_i$$

el primer término representa la demanda no satisfecha del cliente i que puede ser acotado superiormente aplicando la restricción (2.10b). Por tanto, $M_2^i = (1-\lambda_i)d_i, i \in I$ es una cota válida para la demanda no satisfecha. □

Lema 2.2.3. Una cota válida en las restricciones (2.10j) es $M^3 = \sum_{i\in I} d_i,\ j \in J$.

Demostración. En el caso de que $\beta_j = 0$, entonces M^3 ha de ser una cota válida:

$$z_j - \sum_{i\in I} y_{ij} \leq M^3$$

y el valor máximo que puede alcanzar $z_j - \sum_{i\in I} y_{ij}$, se tiene cuando el total de la capacidad de la instalación j está disponible, es decir, cuando $\sum_{i\in I} y_{ij} = 0$. Y como z_j se puede acotar por el total de la demanda de los clientes,

$$z_j \leq \sum_{i\in I} d_i$$

se obtiene que, $M^3 = \sum_{i\in I} d_i, j \in J$, es una cota válida.

□

Lema 2.2.4. Una cota válida en las restricciones (2.10k) es $M_i^4 = \text{máx}\{0, \underset{j\in J}{\text{máx}}\, \pi_{ij}\},\ i \in I$.

Demostración. Dado $i \in I,\ j \in J$, cuando $x_j = 1$, la restricción (2.10g) establece que

$$w_i + u_j \geq \pi_{ij}$$

Por las restricciones de signo (2.10n), $w_i \geq 0$ y $u_j \geq 0$. Entonces,

$$w_i \leq w_i + u_j$$

Así, el máximo valor que toma w_i se da cuando $u_j = 0$, y en ese caso, por la restricción (2.10g)

$$w_i \geq \pi_{ij}$$

Y dado que en el problema (2.7) se está minimizando y los coeficientes de w_i en la función objetivo son mayores o iguales que 0, es decir, $d_i \geq 0$, se tendrá que:

$$w_i \leq \underset{j\in J}{\text{máx}}\, \pi_{ij}$$

Sin embargo, cabe la posibilidad de que $\pi_{ij} \leq 0$ para todo $j \in J$, en ese caso, $w_i = 0$. En general, se obtiene que:

$$w_i \leq \text{máx}\{0, \underset{j\in J}{\text{máx}}\, \pi_{ij}\}$$

Por tanto, $M_i^4 = \text{máx}\{0, \underset{j\in J}{\text{máx}}\, \pi_{ij}\}$ es una cota válida para $i \in I$. □

Lema 2.2.5. Una cota válida en las restricciones (2.10l) es $M_j^5 = \text{máx}\{0, \underset{i\in I}{\text{máx}}\, \pi_{ij}\},\ j \in j$.

Demostración. Con un razonamiento similar, al proporcionado en el Lema 2.2.4, se demuestra que $M_5^i = \text{máx}\{0, \underset{i\in I}{\text{máx}}\, \pi_{ij}\}$, es una cota válida para $j \in j$.

□

Lema 2.2.6. Una cota válida en las restricciones (2.10m) es $M_{ij}^6 = \max\{0, \max_{i\in I, j\in J} \pi_{ij}\} - \pi_{ij}$.

Demostración. Dado $i \in I$, $j \in J$, cuando $x_j = 1$, la restricción (2.10g) establece que

$$w_i + u_j \geq \pi_{ij}$$

En el problema (2.7), la función objetivo es de mínimo y por las restricciones de signo (2.10n), $w_i \geq 0$ y $u_j \geq 0$, entonces:

$$w_i + u_j \leq \max\{0, \max_{i\in I, j\in J} \pi_{ij}\}$$

Por tanto, se deduce que $M_{ij}^6 = \max\{0, \max_{i\in I, j\in J} \pi_{ij}\} - \pi_{ij}$ es una cota válida para la restricción (2.10m). □

Capítulo 3

Experiencia computacional con el modelo BFLP-E

En este capítulo se consideran varios escenarios para los que se resolverá el modelo **BFLP-E** (2.5). En primer lugar, se muestra un ejemplo pequeño para ilustrar las características del modelo. A continuación, se describe la experiencia computacional llevada a cabo y se analizan los resultados obtenidos para conocer la influencia del número de clientes, la proporción de la demanda mínima y de los precios.

La experiencia computacional ha sido llevada a cabo en un ordenador con procesador Intel Core i3-6006U de 200 GHz, 4.00 GB de RAM y Windows 10 64-bits como sistema operativo. Se ha utilizado el lenguaje de programación Python para la definición de los parámetros de entrada en los modelos, la ejecución del modelo de optimización y la escritura de los resultados. El software de optimización utilizado ha sido Gurobi. La versión de Pyhton utilizada ha sido la versión 3.10.12, y en el caso de Gurobi, se ha hecho uso de la licencia académica (ya que se trata de un solver de optimización de pago), utilizando la versión 11.0.2. El código programado se encuentra en el Anexo C.

3.1. Ejemplo ilustrativo

Para ilustrar las características del modelo **BFLP-E** se introduce un ejemplo con 4 clientes ($|I| = 4$) y 3 instalaciones potenciales ($|J| = 3$), donde $|A|$ indica el cardinal del conjunto A.

Los valores de los parámetros del modelo se incluyen en la Tabla 3.1. En la segunda columna se indica la demanda de cada cliente. En la tercera columna se indica el porcentaje mínimo a atender al cliente y, entre paréntesis, el valor de esa demanda mínima. En la cuarta columna se muestran dos precios, ya que se resuelve el problema en dos escenarios. En el primer escenario, el precio es constante para todos los usuarios, y en el segundo, el precio para cada cliente es igual al máximo correspondiente de los costes de envío a las tres instalaciones, que se incluyen en las columnas quinta a séptima. El coste por abrir cada una de las instalaciones y por unidad de capacidad se incluye en las dos últimas filas de la tabla.

	Demanda d_i, $i \in I$	Porcentaje mínimo a atender, λ_i, $i \in I$	Precio p_i, $i \in I$	Costes, c_{ij}, $i \in I$, $j \in J$		
				Inst.1	Inst.2	Inst.3
Cliente 1	10	0.7 (7)	50, 90	90	60	70
Cliente 2	30	0.9 (27)	50, 30	30	20	5
Cliente 3	50	0.8 (40)	50, 60	60	50	40
Cliente 4	5	0.6 (3)	50, 20	20	20	10
Coste fijo de apertura f_j, $j \in J$				100	120	90
Coste por unidad de capacidad, g_j, $j \in J$				1	1	1

Tabla 3.1: Valor de los parámetros en el ejemplo ilustrativo

De ahora en adelante, en todos los casos en los que se proporciona una solución, se identifican solo las variables no nulas.

Solución del problema relajado

El nivel superior no atiende al beneficio ni los costes de transporte de los clientes a las instalaciones, por ello, sea el precio, constante o diferente para cada cliente, el problema relajado es el mismo en ambos escenarios. En la Figura 3.1 se ha representado el sistema mediante un grafo, con los valores de los parámetros en negro y gris, y la solución del problema relajado en azul. La solución óptima consiste en abrir la instalación 3 ($x_3 = 1$) con una capacidad de 77 unidades ($z_3 = 77$), que le permite satisfacer la demanda mínima exigida de los 4 clientes. El coste de esta solución para el nivel superior es $90 + 77 = 167$.

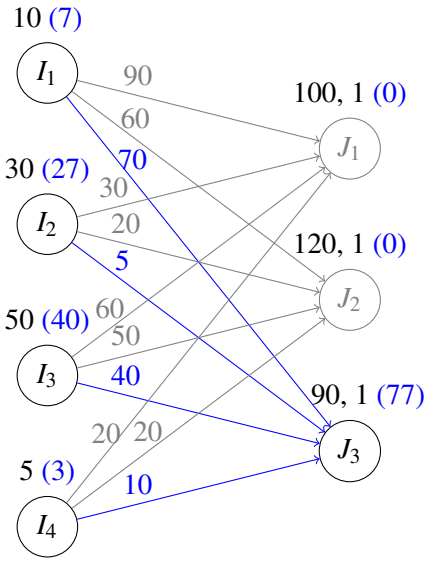


Figura 3.1: Solución del problema relajado en el ejemplo ilustrativo

Notar que, el beneficio para el nivel inferior con la asignación dada por la solución del relajado es, en el primer escenario con precios constantes para todos los clientes:

$$((50-70)\times 7)+((50-5)\times 27)+((50-40)\times 40)+((50-10)\times 3)=1595$$

y en el segundo escenario:

$$((90-70)\times 7)+((30-5)\times 27)+((60-40)\times 40)+((20-10)\times 3)=1645$$

Solución del problema del nivel inferior con $x_3 = 1, z_3 = 77$

Para ilustrar lo que significa considerar la reacción del nivel inferior en la toma de decisiones por parte del nivel superior, se resuelve el problema del nivel inferior, en ambos escenarios, fijadas las variables controladas por el nivel superior en el valor dado por el relajado: $x_3 = 1, z_3 = 77$.

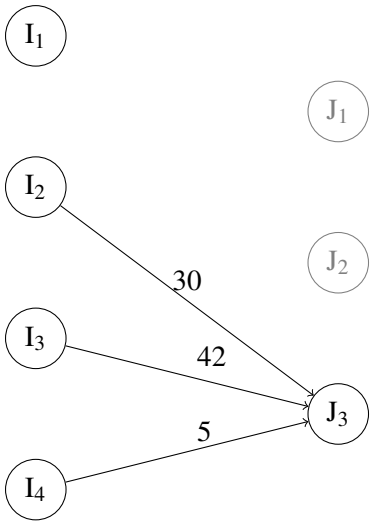
Cuando el nivel superior establece la apertura de la tercera ubicación con una capacidad de 77 unidades, el segundo nivel decide satisfacer primero la demanda de aquellos clientes que le proporcionan un mayor beneficio.

En la Figura 3.2a se representa la solución óptima en el caso de que los precios sean iguales para todos los clientes. En este caso, se atiende la demanda total de los clientes 2 y 4, y 42 unidades del cliente 3, lo que le reporta el siguiente beneficio:

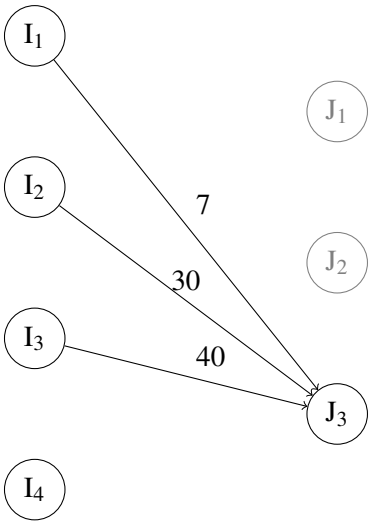
$$((50-5)\times 30)+((50-40)\times 42)+((50-10)\times 5)=1970$$

que es superior al beneficio dado por la solución del relajado. Esta solución no verifica las restricciones *coupling*, ya que no se satisface la demanda mínima al cliente 1. Notar que en la resolución del problema relajado, el nivel superior de decisión no considera la reacción del nivel inferior, y dado que al cliente

1 no se le satisface el mínimo establecido (restricción *coupling*), la decisión del nivel superior ($x_3 = 1$, $z_3 = 77$), no es un punto admisible, al no proporcionar una solución factible binivel.



(a) Solución del problema del nivel inferior con $p_i = 50$, $i \in I$



(b) Solución del problema del nivel inferior con precios $p_i = \max_{j \in J}(c_{ij})$, $i \in I$

Figura 3.2: Soluciones del problema del nivel inferior para $x_3 = 1$, $z_3 = 77$ en el ejemplo ilustrativo

La solución óptima del problema del nivel inferior, cuando para cada cliente el precio es igual al máximo de los costes de envío correspondientes, se muestra en la Figura 3.2b. En este caso, el valor óptimo es:

$$((90-70)\times 7)+((30-5)\times 30)+((60-40)\times 40) = 1690$$

superior al dado por la solución del relajado. En este caso, tampoco proporciona una solución factible binivel ya que no se atiende el mínimo de la demanda del cliente 4.

Solución del problema binivel

En ambos escenarios, no hay ninguna decisión del nivel superior admisible y por tanto, el problema binivel es no factible.

3.2. Descripción de la experiencia computacional

En la experiencia computacional se consideran $|J| = 10$ instalaciones potenciales y dos valores del número de clientes, $|I| \in \{20, 100\}$, con el propósito de ver su influencia en el tiempo computacional requerido para resolver el modelo. Las coordenadas de la localización de los clientes y de las instalaciones se han generado según una distribución uniforme en un plano 100×100. En el Anexo B, las Figuras B.1 y B.2 muestran la distribución de instalaciones y clientes en el plano para $|I| = 20$ y $|I| = 100$, respectivamente. El coste de envío c_{ij} de un cliente $i \in I$ a cada instalación $j \in J$ se ha fijado como la distancia euclídea obtenida a partir de las coordenadas de localización correspondientes. El coste fijo de instalación en cada ubicación $j \in J$ se genera a partir de una distribución uniforme $\mathscr{U}(100, 150)$ y el coste por unidad de capacidad se considera constante e igual a 1. La demanda de los clientes se ha generado a partir de una distribución uniforme $\mathscr{U}(20, 100)$. Los valores de todos estos parámetros quedan recogidos en el Anexo B y el código utilizado para generarlos en el Anexo C.

El modelo **BFLP-E** se resuelve en varios escenarios para evaluar la influencia del valor de dos parámetros del modelo: los precios y la proporción mínima de demanda a satisfacer. La descripción de cada escenario $[A, B]$ indica el valor del precio A y el valor de la proporción mínima B, con:

$$A \in \{F_{min}, F_{max}, F_{mean}, V_{min}, V_{max}, V_{mean}\} \qquad B \in \{0.1, 0.5, 0.9, r\}$$

donde F indica el mismo precio para todos los clientes (precio fijo) y V indica un precio diferente para cada cliente (precio variable). El valor F_{min} es el menor precio fijo entero para el que el correspondiente problema binivel es factible, F_{max} es máximo de los costes ($\max_{i\in I, j\in J} c_{ij}$) y F_{mean}, el valor medio de los dos anteriores ($(F_{min}+F_{max})/2$). En el caso de los precios variables, V_{min} y V_{max}, se define como precio para cada cliente el mínimo y el máximo, respectivamente, de sus costes de envío. Con V_{mean}, el precio para cada cliente es el precio medio entre el mínimo y el máximo. El valor r (random) indica que la proporción mínima de demanda se ha generado según una distribución uniforme $\mathscr{U}(0.2, 0.8)$.

3.3. Resultados de la experiencia computacional

Problema relajado

En primer lugar, se resuelve el problema relajado, con $|I| \in \{20, 100\}$ y se analiza la influencia de la proporción mínima B, ya que los valores de A no le afectan.

En la Tabla 3.2 se resume la información de los 8 problemas resueltos. La primera columna indica el escenario. Para cada valor de $|I|$, la primera columna proporciona el tiempo de ejecución en segundos del modelo. La segunda y tercera columnas, el valor de las variables del nivel superior y la función objetivo en el óptimo.

Solución	$\|I\| = 20$			$\|I\| = 100$		
relajado	**T(s)**	**(x, z)**	**VFO$_1$**	**T(s)**	**(x, z)**	**VFO$_1$**
$[A, 0.1]$	0.04	$(4, 121.3)$	223.3	0.05	$(4, 577.9)$	679.9
$[A, 0.5]$	0.03	$(4, 606.5)$	708.5	0.05	$(4, 2889.5)$	2991.5
$[A, 0.9]$	0.03	$(4, 1091.7)$	1193.7	0.05	$(4, 5201.1)$	5303.1
$[A, r]$	0.04	$(4, 543.7)$	645.7	0.06	$(4, 2941.3)$	3043.3

Tabla 3.2: Solución del problema relajado modificando valores de $|I|$ y B

Los tiempos de resolución están por debajo de los 0.06 segundos, siendo ligeramente más costoso resolver el problema con 100 clientes. Al nivel superior, para minimizar los costes de apertura le interesa abrir una única planta y asignarle toda la capacidad necesaria para atender el mínimo de demanda, ya que se está asumiendo que el coste por capacidad es el mismo en todas las instalaciones. En todos los escenarios abre la instalación 4, de menor coste de apertura. Conforme aumenta la proporción de demanda mínima a satisfacer a los clientes, aumenta la capacidad necesaria y con ella, el valor de VFO_1. En todos los escenarios, el problema binivel va a ser no trivial, ya que ninguna de las soluciones del problema relajado es factible binivel.

Influencia de la proporción de demanda mínima

Fijado el valor $A = V_{max}$, se resuelve el modelo **BFLP-E** en los escenarios generados por el valor de B. La Tabla 3.3 tiene la misma estructura que la Tabla 3.2. En todas las soluciones óptimas del modelo binivel con $|I|$ fijado, el valor de la función objetivo del segundo nivel coincide y se ha incluido su valor en la última fila.

En relación con los tiempos computacionales, la resolución del modelo binivel requiere más tiempo que el problema relajado y estos tiempos se incrementan al pasar de 20 a 100 clientes, como cabía esperar. En todos los escenarios, las soluciones del modelo binivel consideran la apertura de la instalación 3 con una capacidad suficiente para atender las demandas mínimas, que se incrementan según el valor de B. Notemos que, el valor óptimo de z en la solución del relajado de la Tabla 3.2 proporciona la suma total de las demandas mínimas. Por tanto, en la solución del modelo binivel, se satisface siempre más demanda que ese valor mínimo para algunos clientes. Por ejemplo, en el escenario $[V_{max}, 0.1]$ con $|I| = 20$, la demanda mínima total es 121.3 y en la solución óptima binivel se satisface 745.9, de modo que a 12 de los 20 clientes se les satisface el 100 % de la demanda.

Solución	$\|I\| = 20$				$\|I\| = 100$			
BFLP-E	**T(s)**	**(x,z)**	**VFO$_1$**	**100%**	**T(s)**	**(x,z)**	**VFO$_1$**	**100%**
$[V_{max}, 0.1]$	10.42	$(3, 745.9)$	871.9	12	53.33	$(3, 3555.1)$	3681.1	61
$[V_{max}, 0.5]$	2.17	$(3, 953.5)$	1079.5	12	136.46	$(3, 4543.5)$	4669.5	61
$[V_{max}, 0.9]$	1.41	$(3, 1161.1)$	1287.1	12	55.10	$(3, 5531.9)$	5657.9	61
$[V_{max}, r]$	5.88	$(3, 882.8)$	1008.8	12	144.78	$(3, 4470.7)$	4596.7	61
		VFO$_2$	24323			**VFO$_2$**	121886	

Tabla 3.3: Solución del modelo **BFLP-E** modificando $|I|$ y B

Influencia de los precios

Fijado el valor $B = r$, se resuelve el problema (2.5) en los escenarios generados por el valor de A. A partir de la solución del relajado, se conoce que la suma de las demandas mínimas de los clientes es 543.7. La Tabla 3.4 tiene la misma estructura que las anteriores.

Solución	$\|I\| = 20$					$\|I\| = 100$				
BFLP-E	**T(s)**	**(x,z)**	**VFO$_1$**	**VFO$_2$**	**100%**	**T(s)**	**(x,z)**	**VFO$_1$**	**VFO$_2$**	**100%**
$[F_{min}, r]$	0.18	$(1, 302.5)$ $(2, 448.3)$ $(6, 338.1)$	1492.9	51797.8	16	0.06	Infactible			
$[F_{max}, r]$	12.92	$(10, 1145)$	1264	80154	19	626.75	$(10, 5737.5)$	5856.5	418600.5	99
$[F_{mean}, r]$	6.77	$(10, 1145)$	1264	54391.5	19	54.87	$(10, 5737.5)$	5856.5	289506.8	99
$[V_{min}, r]$	0.02	Infactible				0.03	Infactible			
$[V_{max}, r]$	5.88	$(3, 882.8)$	1008.8	24323	12	144.78	$(3, 4470.7)$	4596.7	121886	61
$[V_{mean}, r]$	0.02	Infactible				0.03	Infactible			

Tabla 3.4: Solución del problema **BFLP-E** modificando $|I|$ y A

Notar que, el tiempo para detectar la infactibilidad es inferior a 0.18 segundos en todos los escenarios. En la resolución del modelo binivel aumenta con el número de clientes y los escenarios con precios iguales valores máximos de costes de envío tienen tiempos mayores de resolución.

De todos los escenarios, $[F_{min}, r]$ con $|I| = 20$ es el único en el que se abre más de una instalación, satisfaciendo el 100% de la demanda a 16 clientes de los 20. La capacidad total con la que se abren las tres plantas es 1088.9, casi el doble de la demanda mínima 543.7. En esta solución, la instalación 1 atiende a los clientes $\{2, 3, 5, 6, 9, 15\}$, la instalación 2 a los clientes $\{7, 8, 12, 14, 16 - 18, 20\}$, y la instalación 6 a los clientes $\{1, 4, 10, 11, 13, 19\}$.

En los escenarios $[F_{max}, r]$ y $[F_{mean}, r]$, con $|I| = 20$, se abre la instalación 10 con capacidad 1145 y se atiende el 100% de la demanda de 19 clientes. El correspondiente valor de $\mathbf{f_2}$ cambia porque en el escenario $[F_{max}, r]$ los precios son más elevados. El mismo comportamiento se observa con $|I| = 100$. La instalación 10 con capacidad 5737.5 atiende el 100% de la demanda de 99 clientes, siendo la suma total de la demanda mínima 2941.3.

Los escenarios $[V_{min}, r]$ y $[V_{mean}, r]$ con $|I| \in \{20, 100\}$ son no factibles, esto es, no hay ninguna apertura que pueda proporcionar el nivel superior para la que las decisiones del nivel inferior verifiquen la restricción de demanda mínima a todos los clientes. Se observa, que los precios fijos obtienen en más ocasiones soluciones factibles.

En este modelo, el primer nivel de decisión solo está obligado a satisfacer el mínimo de la demanda $\lambda_i d_i$ a cada uno de los clientes $i \in I$. Sin embargo, el seguidor decide cómo atender la demanda de los clientes en su beneficio. Por eso, cuando dispone de una capacidad atiende primero toda la demanda de aquellos clientes con los que obtiene mayor beneficio, dejando, por tanto, a otros clientes con el 0% de demanda atendida. Esto obliga al líder a aumentar la capacidad de las instalaciones para que a todos los clientes les llegue al menos la demanda mínima.

Bibliografía

[1] Herminia I. Calvete. Apuntes de la asignatura investigación operativa, unizar, 2023.

[2] Herminia I. Calvete and Carmen Galé. On linear bilevel problems with multiple objectives at the lower level. *Omega*, 39:33–40, 2011.

[3] Herminia I. Calvete and Carmen Galé. Algorithms for linear bilevel optimization. In *Bilevel Optimization*. Springer, 2020.

[4] Massimiliano Caramia and Mattia Dalla Costa. An application of bilevel optimisation to the waste collection centres location problem, 2020.

[5] Massimiliano Caramia and Stefano Giordani. Location of differentiated waste collection centers with user cooperation: a bilevel optimization approach. *Optimization Letters*, 14:85–99, 2 2020.

[6] Massimiliano Caramia and Renato Mari. A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints. *Optimization Letters*, 10:997–1019, 6 2016.

[7] Massimiliano Caramia and Emanuele Pizzari. Novel bilevel formulations for waste management. *Discrete Applied Mathematics*, 2022.

[8] Thomas Kleinert, Martine Labbé, Fränk Plein, and Martin Schmidt. There's no free lunch: On the hardness of choosing a correct big-m in bilevel optimization.

[9] Martine Labbé and Alessia Violin. Bilevel programming and price setting problems. *Annals of Operations Research*, 240:141–169, 5 2016.

[10] Yash Pujara, Pankaj Pathak, Archana Sharma, and Janki Govani. Review on indian municipal solid waste management practices for reduction of environmental impacts to achieve sustainable development goals, 10 2019.

[11] Naciones Unidas. La agenda 2030 y los objetivos de desarrollo sostenible, 2015-2023.

Anexos

Anexo A

Relación entre el problema primal y dual

Se llama *problema dual*, al problema asociado al problema primal que tiene: mismo número de restricciones que variables tiene el problema primal, y mismo número de variables que restricciones tiene el problema primal. Los coeficientes de la función objetivo del problema primal, son los términos independientes de las restricciones del problema dual, y los coeficientes de la función objetivo del problema dual, son los términos independientes de las restricciones del problema primal. Además, si el problema primal es de máximo, el problema dual es de mínimo, y viceversa. Las relaciones que se establecen entre el signo de las restricciones y las variables del problema primal y dual vienen dadas en la Tabla A.1. Si el problema es de máximo, se lee de izquierda a derecha, y si el problema es de mínimo, al contrario [1].

Máximo	Mínimo
restricciones $\leq$	variables ≥ 0
variables ≥ 0	restricciones $\geq$
restricciones $=$	variables no restringidas
variables no restringidas	restricciones $=$
restricciones ≥ 0	variables≤ 0
variables ≤ 0	restricciones ≤ 0

Tabla A.1: Relación entre las restricciones y las variables el problema primal y dual

A continuación, se ilustra un ejemplo:
Sea el problema primal:

$$\begin{array}{ll} \max\limits_{x_1,x_2,x_3} & x_1+2x_2+x_3 \\ \text{sujeto a:} & x_1+x_2-x_3 \leq 2 \\ & x_1-x_2+x_3 = 1 \\ & 2x_1+x_2+x_3 \geq 2 \\ & x_1 \geq 0, \quad x_2 \leq 0, \quad x_3 \quad \text{no restringida} \end{array} \tag{A.1}$$

entonces, su correspondiente problema dual es:

$$\begin{array}{ll} \min\limits_{w_1,w_2,w_3} & 2w_1+w_2+2w_3 \\ \text{sujeto a:} & w_1+w_2+2w_3 \geq 1 \\ & w_1-w_2+w_3 \leq 2 \\ & -w_1+w_2+w_3 = 1 \\ & w_1 \geq 0, \quad w_2 \quad \text{no restringida}, \quad w_3 \leq 0 \end{array} \tag{A.2}$$

Anexo B

Valores en la experiencia computacional

- Posiciones de las instalaciones $|J| = 10$:

  $(74,62),(64,12),(11,93),(67,78),(56,96),(15,70),(80,19),(95,90),(88,72),(47,81)$

- Posiciones de los clientes $|I| = 20$:

  $(18,47),(78,96),(75,52),(15,64),(97,74),(70,88),(98,12),(24,4),(92,57),(13,52),$
  $(4,38),(58,14),(19,72),(26,15),(58,53),(34,26),(95,22),(29,32),(47,94),(84,35)$

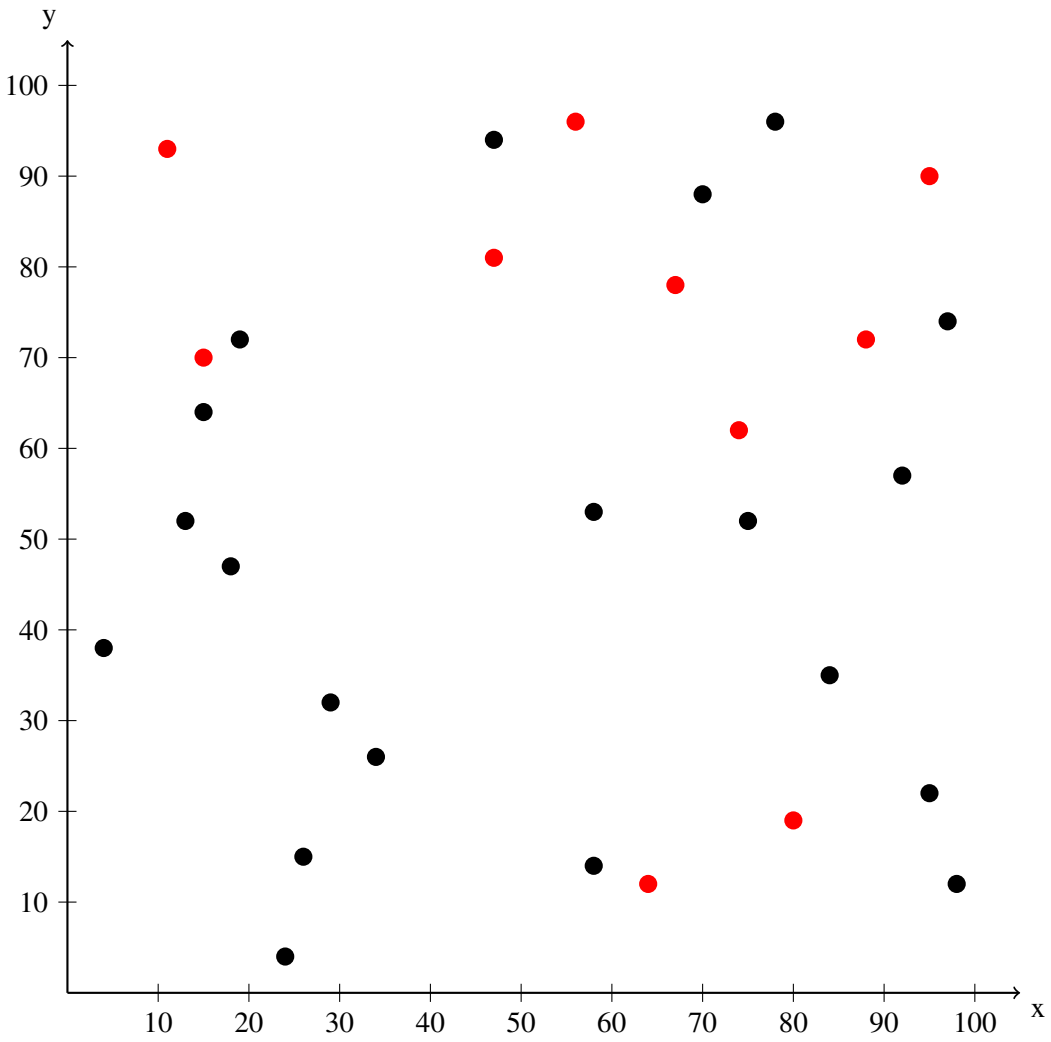


Figura B.1: Plano con las posiciones de 20 clientes (negro) y 10 instalaciones (rojo).

Con todo ello, considerando la distancia euclídea, se calcula **C** la matriz de distancias (y costes) c_{ij}, desde cada cliente i, a cada instalación j.

$$\mathbf{C} = \begin{pmatrix}
58 & 58 & 47 & 58 & 63 & 24 & 69 & 89 & 75 & 45 \\
35 & 86 & 68 & 22 & 22 & 69 & 78 & 19 & 26 & 35 \\
11 & 42 & 77 & 28 & 48 & 63 & 34 & 43 & 24 & 41 \\
60 & 72 & 30 & 54 & 53 & 6 & 80 & 85 & 74 & 37 \\
26 & 71 & 89 & 31 & 47 & 83 & 58 & 17 & 10 & 51 \\
27 & 77 & 60 & 11 & 17 & 58 & 70 & 26 & 25 & 25 \\
56 & 34 & 119 & 73 & 94 & 102 & 20 & 79 & 61 & 86 \\
77 & 41 & 90 & 86 & 98 & 67 & 58 & 112 & 94 & 81 \\
19 & 53 & 89 & 33 & 54 & 79 & 40 & 34 & 16 & 51 \\
62 & 65 & 42 & 60 & 62 & 19 & 75 & 91 & 78 & 45 \\
74 & 66 & 56 & 75 & 78 & 34 & 79 & 105 & 91 & 61 \\
51 & 7 & 92 & 65 & 83 & 71 & 23 & 85 & 66 & 68 \\
56 & 75 & 23 & 49 & 45 & 5 & 81 & 79 & 69 & 30 \\
68 & 39 & 80 & 76 & 87 & 57 & 55 & 102 & 85 & 70 \\
19 & 42 & 62 & 27 & 44 & 47 & 41 & 53 & 36 & 31 \\
54 & 34 & 71 & 62 & 74 & 48 & 47 & 89 & 71 & 57 \\
46 & 33 & 110 & 63 & 84 & 94 & 16 & 68 & 51 & 77 \\
55 & 41 & 64 & 60 & 70 & 41 & 53 & 88 & 72 & 53 \\
42 & 84 & 37 & 26 & 10 & 40 & 82 & 49 & 47 & 13 \\
29 & 31 & 94 & 47 & 68 & 78 & 17 & 57 & 38 & 60
\end{pmatrix}$$

Además, se considera una demanda de clientes también generada aleatoriamente, con una distribución uniforme $\mathscr{U}(20,100)$:

$$\mathbf{d_i} = [96\ 51\ 39\ 66\ 66\ 37\ 85\ 51\ 47\ 43\ 40\ 22\ 35\ 25\ 88\ 94\ 97\ 73\ 83\ 75]$$

Y se obtiene la proporción mínima de demanda a satisfacer λ_i, con otra distribución uniforme $\mathscr{U}(0.2,0.8)$, es:

$$\boldsymbol{\lambda_i} = [0.3\ 0.5\ 0.6\ 0.7\ 0.8\ 0.7\ 0.2\ 0.7\ 0.6\ 0.7\ 0.2\ 0.3\ 0.5\ 0.7\ 0.3\ 0.5\ 0.2\ 0.2\ 0.7\ 0.2]$$

Así mismo, se proporcionan aleatoriamente, valores entre $[100,150]$ de f_j, es decir, de los costes fijos por abrir cada una de las instalaciones j:

$$\mathbf{f_j} = [124,148,126,102,116,132,131,125,150,119]$$

Se repite el proceso para $|I| = 100$:

- Posiciones de los clientes $|I| = 100$:

$$(17,72),(97,8),(32,15),(63,97),(57,60),(83,48),(26,12),(62,3),(49,55),(77,97),$$
$$(98,0),(89,57),(34,92),(29,75),(13,40),(3,2),(3,83),(69,1),(48,87),(27,54),$$
$$(92,3),(67,28),(97,56),(63,70),(29,44),(29,86),(28,97),(58,37),(2,53),(71,82),$$
$$(12,23),(80,92),(37,15),(95,42),(92,91),(64,54),(64,85),(24,38),(36,75),(63,64),$$
$$(50,75),(4,61),(31,95),(51,53),(85,22),(46,70),(89,99),(86,94),(47,11),(56,84)$$

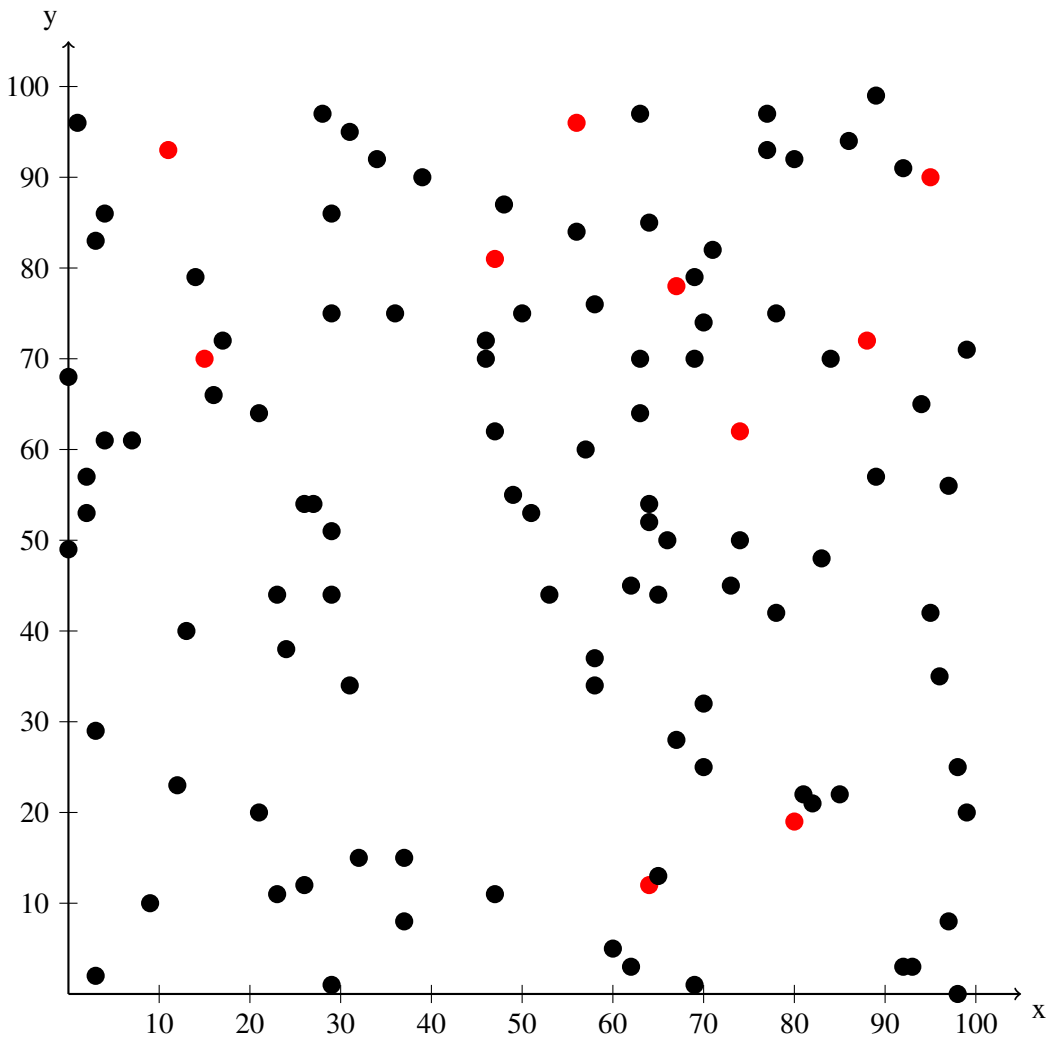


Figura B.2: Plano con las posiciones de 100 clientes (negro) y 10 instalaciones (rojo).

Y con ello, considerando de nuevo la distancia euclídea, se calcula **C** la matriz de distancias (y costes) c_{ij}, desde cada cliente i a cada instalación j:

$$\mathbf{C} = \begin{pmatrix}
58 & 77 & 22 & 51 & 46 & 3 & 83 & 81 & 71 & 32 \\
59 & 34 & 121 & 77 & 98 & 103 & 21 & 83 & 65 & 89 \\
64 & 33 & 81 & 73 & 85 & 58 & 49 & 98 & 80 & 68 \\
37 & 86 & 53 & 20 & 8 & 56 & 80 & 33 & 36 & 23 \\
18 & 49 & 57 & 21 & 37 & 44 & 48 & 49 & 34 & 24 \\
17 & 41 & 85 & 34 & 56 & 72 & 30 & 44 & 25 & 49 \\
70 & 38 & 83 & 78 & 90 & 60 & 55 & 105 & 87 & 73 \\
61 & 10 & 104 & 76 & 94 & 82 & 25 & 94 & 74 & 80 \\
26 & 46 & 54 & 30 & 42 & 38 & 48 & 58 & 43 & 27 \\
36 & 86 & 67 & 22 & 22 & 68 & 79 & 20 & 28 & 34 \\
67 & 37 & 128 & 84 & 105 & 109 & 27 & 91 & 73 & 96 \\
16 & 52 & 86 & 31 & 52 & 76 & 40 & 34 & 16 & 49 \\
50 & 86 & 24 & 36 & 23 & 30 & 87 & 62 & 58 & 18 \\
47 & 73 & 26 & 39 & 35 & 15 & 76 & 68 & 60 & 19 \\
65 & 59 & 54 & 67 & 71 & 31 & 71 & 97 & 82 & 54 \\
93 & 62 & 92 & 100 & 108 & 70 & 79 & 128 & 111 & 91 \\
75 & 94 & 13 & 65 & 55 & 18 & 101 & 93 & 86 & 45 \\
62 & 13 & 109 & 78 & 96 & 88 & 22 & 93 & 74 & 83 \\
37 & 77 & 38 & 22 & 13 & 38 & 76 & 48 & 43 & 7 \\
48 & 56 & 43 & 47 & 52 & 20 & 64 & 77 & 64 & 34 \\
62 & 30 & 122 & 80 & 100 & 103 & 20 & 88 & 70 & 91 \\
35 & 17 & 86 & 50 & 69 & 67 & 16 & 69 & 49 & 57 \\
24 & 55 & 94 & 38 & 58 & 84 & 41 & 35 & 19 & 56 \\
14 & 59 & 57 & 9 & 27 & 48 & 54 & 38 & 26 & 20 \\
49 & 48 & 53 & 51 & 59 & 30 & 57 & 81 & 66 & 42 \\
51 & 82 & 20 & 39 & 29 & 22 & 85 & 67 & 61 & 19 \\
58 & 93 & 18 & 44 & 29 & 30 & 94 & 68 & 65 & 25 \\
30 & 26 & 74 & 42 & 60 & 55 & 29 & 65 & 47 & 46 \\
73 & 75 & 41 & 70 & 70 & 22 & 86 & 101 & 89 & 53 \\
21 & 71 & 61 & 6 & 21 & 58 & 64 & 26 & 20 & 25 \\
74 & 54 & 71 & 78 & 86 & 48 & 69 & 107 & 91 & 68 \\
31 & 82 & 70 & 20 & 25 & 69 & 73 & 16 & 22 & 35 \\
60 & 28 & 83 & 70 & 84 & 60 & 44 & 95 & 77 & 67 \\
29 & 44 & 99 & 46 & 67 & 85 & 28 & 48 & 31 & 62 \\
35 & 84 & 82 & 29 & 37 & 80 & 73 & 4 & 20 & 47 \\
13 & 42 & 66 & 25 & 43 & 52 & 39 & 48 & 30 & 32 \\
26 & 73 & 54 & 8 & 14 & 52 & 68 & 32 & 28 & 18 \\
56 & 48 & 57 & 59 & 67 & 34 & 60 & 89 & 73 & 49 \\
41 & 69 & 31 & 32 & 29 & 22 & 72 & 61 & 53 & 13 \\
12 & 53 & 60 & 15 & 33 & 49 & 49 & 42 & 27 & 24 \\
28 & 65 & 43 & 18 & 22 & 36 & 64 & 48 & 39 & 7 \\
71 & 78 & 33 & 66 & 63 & 15 & 87 & 96 & 85 & 48 \\
55 & 90 & 21 & 40 & 26 & 30 & 91 & 65 & 62 & 22 \\
25 & 44 & 57 & 30 & 44 & 40 & 45 & 58 & 42 & 29 \\
42 & 24 & 103 & 59 & 80 & 85 & 6 & 69 & 51 & 71 \\
30 & 61 & 42 & 23 & 28 & 31 & 62 & 53 & 43 & 12 \\
40 & 91 & 79 & 31 & 34 & 80 & 81 & 11 & 28 & 46 \\
35 & 85 & 76 & 25 & 31 & 75 & 76 & 10 & 23 & 42 \\
58 & 18 & 90 & 70 & 86 & 68 & 34 & 93 & 74 & 70 \\
29 & 73 & 46 & 13 & 12 & 44 & 70 & 40 & 35 & 10 \\
50 & 2 & 97 & 66 & 84 & 76 & 17 & 83 & 64 & 71 \\
49 & 36 & 115 & 67 & 88 & 98 & 20 & 71 & 54 & 81 \\
15 & 39 & 70 & 29 & 48 & 55 & 35 & 50 & 32 & 37 \\
27 & 53 & 48 & 26 & 36 & 33 & 55 & 56 & 43 & 19 \\
62 & 31 & 122 & 80 & 101 & 103 & 21 & 88 & 70 & 91 \\
59 & 9 & 101 & 74 & 92 & 80 & 25 & 92 & 73 & 78 \\
45 & 82 & 29 & 31 & 19 & 32 & 82 & 56 & 53 & 13 \\
14 & 65 & 70 & 12 & 31 & 64 & 57 & 23 & 11 & 32 \\
12 & 40 & 77 & 29 & 50 & 63 & 32 & 46 & 27 & 42 \\
42 & 21 & 102 & 59 & 80 & 84 & 3 & 71 & 52 & 70 \\
54 & 68 & 31 & 49 & 48 & 9 & 75 & 79 & 68 & 32 \\
76 & 37 & 94 & 86 & 99 & 71 & 55 & 111 & 93 & 82 \\
45 & 37 & 111 & 62 & 83 & 95 & 19 & 66 & 49 & 76 \\
10 & 59 & 63 & 9 & 30 & 54 & 53 & 33 & 20 & 25 \\
47 & 53 & 46 & 47 & 53 & 24 & 61 & 77 & 63 & 35 \\
21 & 33 & 73 & 35 & 53 & 57 & 30 & 55 & 37 & 42 \\
18 & 35 & 79 & 34 & 54 & 64 & 27 & 51 & 31 & 45 \\
33 & 23 & 76 & 45 & 63 & 57 & 27 & 68 & 49 & 49 \\
13 & 62 & 77 & 19 & 39 & 69 & 52 & 23 & 5 & 39 \\
32 & 83 & 66 & 19 & 22 & 67 & 75 & 19 & 24 & 33 \\
76 & 74 & 46 & 74 & 74 & 26 & 86 & 104 & 91 & 57 \\
21 & 61 & 88 & 30 & 50 & 80 & 49 & 26 & 10 & 50 \\
59 & 73 & 28 & 53 & 50 & 5 & 80 & 83 & 73 & 35 \\
27 & 69 & 91 & 33 & 50 & 85 & 56 & 20 & 12 & 53 \\
49 & 57 & 42 & 48 & 52 & 20 & 65 & 78 & 65 & 35 \\
68 & 76 & 33 & 63 & 61 & 13 & 85 & 93 & 82 & 45 \\
30 & 63 & 41 & 22 & 26 & 32 & 63 & 53 & 42 & 10 \\
38 & 15 & 91 & 54 & 73 & 72 & 12 & 70 & 51 & 61 \\
15 & 40 & 68 & 27 & 45 & 53 & 37 & 50 & 32 & 34 \\
21 & 34 & 71 & 34 & 52 & 54 & 32 & 56 & 38 & 39 \\
28 & 34 & 65 & 37 & 53 & 47 & 37 & 63 & 45 & 38 \\
75 & 86 & 28 & 68 & 63 & 16 & 94 & 98 & 89 & 49 \\
18 & 68 & 60 & 3 & 22 & 55 & 61 & 29 & 21 & 23 \\
21 & 34 & 85 & 38 & 59 & 69 & 24 & 51 & 32 & 50 \\
22 & 65 & 50 & 10 & 21 & 44 & 62 & 40 & 31 & 13 \\
79 & 64 & 65 & 81 & 86 & 43 & 78 & 111 & 96 & 69 \\
41 & 20 & 100 & 58 & 79 & 82 & 4 & 70 & 51 & 69 \\
13 & 63 & 62 & 5 & 27 & 56 & 56 & 30 & 19 & 25 \\
73 & 42 & 83 & 81 & 92 & 60 & 58 & 107 & 90 & 74 \\
31 & 21 & 85 & 47 & 66 & 67 & 17 & 64 & 44 & 55 \\
74 & 96 & 10 & 64 & 53 & 20 & 102 & 92 & 86 & 44 \\
84 & 56 & 84 & 90 & 99 & 61 & 72 & 118 & 101 & 81 \\
73 & 77 & 38 & 69 & 67 & 19 & 87 & 99 & 88 & 51 \\
81 & 105 & 11 & 69 & 55 & 30 & 111 & 95 & 91 & 49 \\
35 & 40 & 103 & 52 & 73 & 89 & 23 & 56 & 38 & 68 \\
52 & 40 & 63 & 57 & 67 & 40 & 52 & 86 & 69 & 50 \\
63 & 84 & 15 & 54 & 46 & 10 & 90 & 82 & 75 & 34 \\
55 & 53 & 51 & 56 & 62 & 28 & 63 & 86 & 71 & 45 \\
66 & 28 & 89 & 77 & 91 & 66 & 45 & 101 & 82 & 74 \\
68 & 44 & 74 & 75 & 84 & 51 & 60 & 102 & 85 & 67
\end{pmatrix}$$

De nuevo, se considera una demanda de clientes también generada aleatoriamente, con una distribución uniforme $\mathscr{U}(20,100)$:

$\mathbf{d_i} = [64\ 77\ 68\ 64\ 54\ 72\ 55\ 91\ 97\ 51\ 83\ 62\ 65\ 94\ 26\ 27\ 22\ 87\ 82\ 90$
$98\ 84\ 57\ 82\ 29\ 71\ 31\ 96\ 62\ 53\ 41\ 82\ 56\ 65\ 22\ 69\ 69\ 69\ 95\ 75$
$49\ 55\ 76\ 25\ 73\ 74\ 37\ 30\ 45\ 49\ 66\ 55\ 99\ 28\ 37\ 33\ 72\ 40\ 57\ 40$
$33\ 29\ 73\ 31\ 36\ 49\ 86\ 28\ 87\ 28\ 98\ 57\ 98\ 68\ 79\ 23\ 43\ 30\ 44\ 29$
$45\ 53\ 25\ 75\ 65\ 41\ 62\ 28\ 66\ 94\ 45\ 73\ 31\ 77\ 43\ 35\ 67\ 22\ 86\ 20]$

Y se obtiene la proporción mínima de demanda a satisfacer λ_i, con otra distribución uniforme $\mathscr{U}(0.2,0.8)$, es:

$\boldsymbol{\lambda_i} = [0.6\ 0.4\ 0.6\ 0.8\ 0.3\ 0.5\ 0.6\ 0.5\ 0.3\ 0.8\ 0.5\ 0.7\ 0.6\ 0.4\ 0.7\ 0.4\ 0.7\ 0.5\ 0.7\ 0.6$
$0.6\ 0.5\ 0.8\ 0.6\ 0.5\ 0.6\ 0.2\ 0.4\ 0.6\ 0.4\ 0.6\ 0.5\ 0.3\ 0.4\ 0.5\ 0.6\ 0.5\ 0.6\ 0.6\ 0.5$
$0.7\ 0.4\ 0.5\ 0.7\ 0.7\ 0.6\ 0.3\ 0.8\ 0.6\ 0.8\ 0.3\ 0.7\ 0.3\ 0.6\ 0.3\ 0.7\ 0.7\ 0.5\ 0.4\ 0.2$
$0.6\ 0.5\ 0.6\ 0.7\ 0.8\ 0.7\ 0.2\ 0.4\ 0.6\ 0.3\ 0.5\ 0.2\ 0.3\ 0.2\ 0.7\ 0.3\ 0.4\ 0.8\ 0.6\ 0.2$
$0.3\ 0.6\ 0.5\ 0.3\ 0.8\ 0.6\ 0.5\ 0.6\ 0.6\ 0.4\ 0.4\ 0.3\ 0.3\ 0.8\ 0.6\ 0.5\ 0.3\ 0.4\ 0.2\ 0.5]$

Anexo C

Código programado en Python

```

!pip install gurobipy
import gurobipy as gp
from gurobipy import GRB, Env

import numpy as np
import random
import pandas as pd

env = Env(params = {"WLSACCESSID" : " ",
                    "WLSSECRET" : " ",
                    "LICENSEID" :  })

##Problema relajado en el ejemplo ilustrativo
mrelajado =gp.Model("mip", env = env)
# Conjuntos
I = [i for i in range(0,4)]
J = [j for j in range(0,3)]

# Parametros
d = [10,30,50,5]
lambda_i=[0.7,0.9,0.8,0.6]

f = [100,120,90]
g = 1 #se puede asumir g_j=g

# Variables
x = mrelajado.addVars(J, vtype=GRB.BINARY, name="x")
z = mrelajado.addVars(J, vtype=GRB.CONTINUOUS, name="z")
y = mrelajado.addVars(I, J, vtype=GRB.CONTINUOUS, name="y")

# Funcion objetivo
mrelajado.setObjective(gp.quicksum(f[j] * x[j] + g * z[j] for j in J),GRB.
    MINIMIZE)

# Restricciones
mrelajado.addConstrs((gp.quicksum(y[i, j] for j in J) >= lambda_i[i] *d[i] for i
     in I))
mrelajado.addConstrs((gp.quicksum(y[i, j] for j in J) <= d[i] for i in I))
mrelajado.addConstrs((gp.quicksum(y[i, j] for i in I) <= z[j] for j in J))
mrelajado.addConstrs((y[i, j] <= d[i] * x[j] for i in I for j in J))

# Optimize the model
mrelajado.optimize()

## Problema primal para el nivel inferior (empresa) con 4 clientes y 3
    instalaciones, precio fijo.
```

```
45
46 mprimal = gp.Model("mip", env = env)
47
48 # Conjuntos
49 I = [i for i in range(0,4)]
50 J = [j for j in range(0,3)]
51
52 # Parametros
53 matriz_distancias = np.array([
54     [90, 60, 70],
55     [30, 20, 5],
56     [60, 50, 40],
57     [20, 20, 10]
58 ])
59
60 matriz_costes = np.array([
61     [50, 50, 50, 50]
62 ]).T
63 d=[10,30,50,5]
64 z=[0,0,77]
65
66 pi=matriz_costes-matriz_distancias
67
68 # Variables
69 y = mprimal.addVars(I, J, vtype=GRB.CONTINUOUS, name="y")
70
71 mprimal.update()
72
73 #Funcion Objetivo
74 objective = gp.quicksum(pi[i][j] * y[i, j] for i in I for j in J)
75 mprimal.setObjective(objective, GRB.MAXIMIZE)
76
77 mprimal.update()
78
79 # Restricciones
80 for i in I:
81     mprimal.addConstr(gp.quicksum(y[i, j] for j in J) <= d[i])
82
83 for j in J:
84     mprimal.addConstr(gp.quicksum (y[i, j] for i in I) <= z[j])
85
86 # Optimizar
87 mprimal.optimize()
88
89 ## Problema primal para el nivel inferior (empresa) con 4 clientes y 3
       instalaciones, precio variable.
90
91 mprimal = gp.Model("mip", env = env)
92
93 # Conjuntos
94 I = [i for i in range(0,4)]
95 J = [j for j in range(0,3)]
96
97 # Parametros
98 matriz_distancias = np.array([
99     [90, 60, 70],
100     [30, 20, 5],
101     [60, 50, 40],
102     [20, 20, 10]
103 ])
104
105 matriz_costes = np.array([
106     [90, 30, 60, 20]
```

```
]).T
d=[10,30,50,5]
z=[0,0,77]
#pi = 50- matriz_distancias
pi=matriz_costes-matriz_distancias

# Variables
y = mprimal.addVars(I, J, vtype=GRB.CONTINUOUS, name="y")

mprimal.update()

#Funcion Objetivo
objective = gp.quicksum(pi[i][j] * y[i, j] for i in I for j in J)
mprimal.setObjective(objective, GRB.MAXIMIZE)

mprimal.update()

# Restricciones
for i in I:
    mprimal.addConstr(gp.quicksum(y[i, j] for j in J) <= d[i])

for j in J:
    mprimal.addConstr(gp.quicksum (y[i, j] for i in I) <= z[j])

# Optimizar
mprimal.optimize()

##Generar posiciones de  20 clientes y 10 instalaciones

random.seed(0)

# Plano
ancho_plano = 100
alto_plano = 100

# Numero de clientes y plantas
num_clientes = 20
num_instalaciones = 10

# Posiciones aleatorias para los clientes
posiciones_clientes = [(random.randint(0, ancho_plano-1), random.randint(0,
    alto_plano-1)) for _ in range(num_clientes)]

# Posiciones aleatorias para las instalaciones
posiciones_instalaciones = [(random.randint(0, ancho_plano-1), random.randint(0,
     alto_plano-1)) for _ in range(num_instalaciones)]

print("Posiciones de los clientes:", posiciones_clientes)
print("Posiciones de las instalaciones:", posiciones_instalaciones)

## Matriz distancias c_{ij} de cada cliente, a cada instalacion
import math
matriz_distancias = []

for cliente in posiciones_clientes:
    fila_distancias = []
    for instalacion in posiciones_instalaciones:
        distancia = math.ceil(math.sqrt((cliente[0] - instalacion[0])**2 + (
    cliente[1] - instalacion[1])**2))
        fila_distancias.append(distancia)
    matriz_distancias.append(fila_distancias)
matriz_distancias=np.array(matriz_distancias)
matriz_distancias
```

```
167
168 ##Generar demanda clientes
169
170 np.random.seed(0)
171
172 #Numero de clientes
173 num_clientes = 20
174
175 # Generar demanda con distribucion uniforme en el intervalo [20, 100]
176 demanda_i = np.array(np.round(np.random.uniform(20, 100, num_clientes)))
177 lambda_i = np.array(np.round(np.random.uniform(0.2, 0.8, num_clientes)))
178
179 print("Demanda de los clientes:")
180 print(demanda_i)
181
182 print("Proporcion minima de demanda a satisfacer de los clientes:")
183 print(lambda_i)
184
185 import random
186 random.seed(0)
187 f_values = [random.randint(100, 150) for _ in range(10)]
188 print(f_values)
189
190 ##Problema relajado con  20 clientes y 10 instalaciones
191
192 mrelajado =gp.Model("mip", env = env)
193 # Conjuntos
194 I = [i for i in range(0,20)]
195 J = [j for j in range(0,10)]
196
197 # Parametros
198 d = {i: demanda_i[i] for i in I}
199 #lambda_i={i: 0.1 for i in I}
200 #lambda_i={i: 0.5 for i in I}
201 lambda_i={i: 0.9 for i in I}
202 #lambda_i
203 f = f_values
204 g = 1 #se puede asumir g_j=g
205
206 # Variables
207 x = mrelajado.addVars(J, vtype=GRB.BINARY, name="x")
208 z = mrelajado.addVars(J, vtype=GRB.CONTINUOUS, name="z")
209 y = mrelajado.addVars(I, J, vtype=GRB.CONTINUOUS, name="y")
210
211 # Funcion objetivo
212 mrelajado.setObjective(gp.quicksum(f[j] * x[j] + g * z[j] for j in J),GRB.
        MINIMIZE)
213
214 # Restricciones
215 mrelajado.addConstrs((gp.quicksum(y[i, j] for j in J) >= lambda_i[i] *d[i] for i
         in I))
216 mrelajado.addConstrs((gp.quicksum(y[i, j] for j in J) <= d[i] for i in I))
217 mrelajado.addConstrs((gp.quicksum(y[i, j] for i in I) <= z[j] for j in J))
218 mrelajado.addConstrs((y[i, j] <= d[i] * x[j] for i in I for j in J))
219
220 # Optimize the model
221 mrelajado.optimize()
222
223 ## Problema primal para el nivel inferior (empresa) con los valores de z_j
        obtenidos en el problema relajado
224 mprimal = gp.Model("mip", env = env)
225
226 # Conjuntos
```

```
I = [i for i in range(0,20)]
J = [j for j in range(0,10)]

# Parametros
d=demanda_i
z=[0,0,0,543.7,0,0,0,0,0,0]
pi = 50- matriz_distancias

# Variables
y = mprimal.addVars(I, J, vtype=GRB.CONTINUOUS, name="y")

mprimal.update()

#Funcion Objetivo
objective = gp.quicksum(pi[i][j] * y[i, j] for i in I for j in J)
mprimal.setObjective(objective, GRB.MAXIMIZE)

mprimal.update()

# Restricciones
for i in I:
    mprimal.addConstr(gp.quicksum(y[i, j] for j in J) <= d[i])

for j in J:
    mprimal.addConstr(gp.quicksum (y[i, j] for i in I) <= z[j])

# Optimizar
mprimal.optimize()

## Problema lineal para los escenarios propuestos con 20 clientes y 10
    instalaciones

def modelo_lineal(instalaciones, clientes, distancias, demandas, precios, pi,
                  lambdaa, costes_fijos, costes_variables, M1, M2, M3, M4, M5,
    M6 ):
    '''
    instalaciones: conjunto de las instalaciones de las que se dispone
    clientes: conjunto de clientes que tienen demanda
    distancias: distancia desde el cliente i a  la  instalacion j. Se asume que
    el coste es igual a la distancia
    demandas: demanda del cliente i
    precios: precio que paga el cliente  i por que se  le atienda su demanda
    pi: ganancia unitaria por atender la demanda, es el precio-distancias
    lambdaa: porcentaje  minimo de demanda que se esta  obligado a satisfacer
    costes_fijos: costes que tiene el lider por abrir una instalacion j
    costes_variables: costes que tiene el lider por cada unidad de capacidad  de
     la instalacion j
    M1, M2, M3, M4, M5,  M6: cotas validas para sus correspondientes
    restricciones
    '''
    mlineal = gp.Model("mip", env = env)

    #mlineal.setParam("OutputFlag", 0)

    # Variables
    x = mlineal.addVars(instalaciones, vtype=GRB.BINARY, name="x")
    z = mlineal.addVars(instalaciones, vtype=GRB.CONTINUOUS, name="z")
    y = mlineal.addVars(clientes, instalaciones, vtype=GRB.CONTINUOUS, name="y")

    w = mlineal.addVars(clientes, vtype=GRB.CONTINUOUS, name="w")
    u = mlineal.addVars(instalaciones, vtype=GRB.CONTINUOUS, name="u")

    alpha = mlineal.addVars(clientes, vtype=GRB.BINARY, name="alpha")
```

```
285     beta = mlineal.addVars(instalaciones, vtype=GRB.BINARY, name="beta")
286     gamma = mlineal.addVars(clientes, instalaciones, vtype=GRB.BINARY, name="
        gamma")
287
288     # Funcion Objetivo
289     objective=gp.quicksum(costes_fijos[j] * x[j] for j in instalaciones) + gp.
        quicksum(costes_variables * z[j] for j in instalaciones)
290     mlineal.setObjective(objective, GRB.MINIMIZE)
291
292     # Restricciones
293     mlineal.addConstrs((gp.quicksum(y[i, j] for j in instalaciones) >= lambdaa[i
        ] *demandas[i] for i in clientes))
294     mlineal.addConstrs((gp.quicksum(y[i, j] for j in instalaciones) <= demandas[
        i] for i in clientes))
295     mlineal.addConstrs((gp.quicksum(y[i, j] for i in clientes) <= z[j] for j in
        instalaciones))
296     mlineal.addConstrs((w[i] + u[j] >= pi[i, j] * x[j] for i in clientes for j
        in instalaciones))
297     mlineal.addConstrs((y[i, j] <= demandas[i] * x[j] for i in clientes for j in
         instalaciones))
298     mlineal.addConstrs((y[i, j] <= M1[i] * (1 - gamma[i, j]) for i in clientes
        for j in instalaciones))
299     mlineal.addConstrs((demandas[i] - gp.quicksum(y[i, j] for j in instalaciones
        ) <= M2[i] * (1 - alpha[i]) for i in clientes))
300     mlineal.addConstrs((z[j] - gp.quicksum(y[i, j] for i in clientes) <= M3[j] *
         (1 - beta[j]) for j in instalaciones))
301     mlineal.addConstrs((w[i] <= M4[i] * alpha[i] for i in clientes))
302     mlineal.addConstrs((u[j] <= M5[j] * beta[j] for j in instalaciones))
303     mlineal.addConstrs((u[j] <= M5[j] * x[j] for j in instalaciones))
304     mlineal.addConstrs((w[i] + u[j] - pi[i, j] <= M6 * gamma[i, j] for i in
        clientes for j in instalaciones))
305
306     # Optimizar
307     mlineal.optimize()
308
309     if mlineal.status == GRB.INFEASIBLE:
310         print('El modelo es infactible')
311         return None, None, None, None, None, None, None
312     else:
313
314         #Instalaciones abiertas
315         decision_instalaciones=pd.DataFrame()
316         for j in instalaciones:
317             decision_instalaciones.loc[j,'x']=x[j].X
318             decision_instalaciones.loc[j,'z']=z[j].X
319             # decision_instalaciones.loc[j,'Capacidad_utilizada']=sum(y[i,j].X
        for i in clientes)
320
321         #Valores funciones objetivo
322         #lider
323         vfo_lider=mlineal.ObjVal
324         #seguidor
325         vfo_seguidor=sum(pi[i,j]*y[i, j].X for i in clientes for j in
        instalaciones)
326
327         #demanda satisfecha
328         demanda_satisfecha=pd.DataFrame()
329         for i in clientes:
330             demanda_satisfecha.loc[i,'Demanda_satisfecha']=(sum(y[i,j].X for j
        in instalaciones)/demandas[i])*100
331
332         #numero de clientes con demanda satisfecha al 100%
333         clientes_demanda_satisfecha=len(demanda_satisfecha[demanda_satisfecha.
```

```
    Demanda_satisfecha==100].index)
334
335         #numero de clientes con demanda satisfecha al min
336         clientes_min_demanda_satisfecha=len(demanda_satisfecha[
    demanda_satisfecha.Demanda_satisfecha==lambdaa*100].index)
337
338         #variables y_{ij}
339         variables_y_ij = pd.DataFrame(index=clientes, columns=instalaciones)
340         for i in clientes:
341             for j in instalaciones:
342                 variables_y_ij.loc[i, j] = y[i, j].X
343
344
345         return decision_instalaciones, vfo_lider, vfo_seguidor,
    demanda_satisfecha, clientes_demanda_satisfecha,
    clientes_min_demanda_satisfecha, variables_y_ij
346
347 escenarios=range(0,6)
348
349 resultados=dict()
350
351 for e in escenarios:
352   resultados[e]=dict()
353
354   # Conjuntos
355   I = [i for i in range(0,20)]
356   J = [j for j in range(0,10)]
357
358   # Parametros
359   d=demanda_i
360   p_i = np.array([
361         [74 for _ in I],
362         [119 for _ in I],
363         [96.5 for _ in I],
364         [min(matriz_distancias[i, :]) for i in I],
365         [max(matriz_distancias[i, :]) for i in I],
366         [(max(matriz_distancias[i, :]) + min(matriz_distancias[i, :])) / 2 for i
     in I]
367     ]).T
368
369   pi = p_i[:,e:e+1] - matriz_distancias
370   #lambda_i=np.array([0.1 for i in I])
371   #lambda_i=np.array([0.5 for i in I])
372   #lambda_i=np.array([0.9 for i in I])
373
374   f = f_values
375   g = 1 #se puede asumir g_j=g
376
377   # Valores M_n
378   M1 = {i: d[i] for i in I}
379   M2 = {i: (1 - lambda_i[i]) * d[i] for i in I}
380   M3 = {j: sum(d[i] for i in I) for j in J}
381   M4 = {i: max(0,max([pi[i, j] for j in J])) for i in I}
382   M5 = {j: max(0,max([pi[i, j] for i in I])) for j in J}
383   M6 = max(0, max([pi[i,j] for i in I for j in J] ))
384
385
386   result=modelo_lineal(J,I,matriz_distancias,demanda_i,p_i,pi,lambda_i,f,g,M1,M2
      ,M3,M4,M5,M6)
387   resultados[e]['decision_instalaciones']=result[0]
388   resultados[e]['vfo_lider']=result[1]
389   resultados[e]['vfo_seguidor']=result[2]
390   resultados[e]['demanda_satisfecha']=result[3]
```

```
  resultados[e]['clientes_demanda_satisfecha']=result[4]
  resultados[e]['clientes_min_demanda_satisfecha']=result[5]
  resultados[e]['variables_y_ij']=result[6]

resultados

## Generar 100 clientes
random.seed(1)
num_clientes = 100

# Posiciones aleatorias para los clientes
posiciones_clientes = [(random.randint(0, ancho_plano-1), random.randint(0, alto_plano-1)) for _ in range(num_clientes)]
print("Posiciones de los clientes:", posiciones_clientes)

## Matriz distancias c_{ij} de cada cliente, a cada instalacion
import math
matriz_distancias = []

for cliente in posiciones_clientes:
    fila_distancias = []
    for instalacion in posiciones_instalaciones:
        distancia = math.ceil(math.sqrt((cliente[0] - instalacion[0])**2 + (cliente[1] - instalacion[1])**2))
        fila_distancias.append(distancia)
    matriz_distancias.append(fila_distancias)
matriz_distancias=np.array(matriz_distancias)
matriz_distancias

##Generar demanda 100 clientes

np.random.seed(0)

#Numero de clientes
num_clientes = 100

# Generar demanda con distribucion uniforme en el intervalo [20, 100]
demanda_i = np.round(np.random.uniform(20, 100, num_clientes))
lambda_i = np.round(np.random.uniform(0.2, 0.8, num_clientes),1)

print("Demanda de los clientes:")
print(demanda_i)

print("Proporcion minima de demanda a satisfacer de los clientes:")
print(lambda_i)

##Problema relajado para 100 clientes

mrelajado =gp.Model("mip", env = env)
# Conjuntos
I = [i for i in range(0,100)]
J = [j for j in range(0,10)]

# Parametros
d = {i: demanda_i[i] for i in I}

#lambda_i={i: 0.1 for i in I}
#lambda_i={i: 0.5 for i in I}
#lambda_i={i: 0.9 for i in I}
#lambda_i
f = f_values
g = 1 #se puede asumir g_j=g

```

```
# Variables
x = mrelajado.addVars(J, vtype=GRB.BINARY, name="x")
z = mrelajado.addVars(J, vtype=GRB.CONTINUOUS, name="z")
y = mrelajado.addVars(I, J, vtype=GRB.CONTINUOUS, name="y")

# Funcion objetivo
mrelajado.setObjective(gp.quicksum(f[j] * x[j] + g * z[j] for j in J),GRB.
    MINIMIZE)

# Restricciones
mrelajado.addConstrs((gp.quicksum(y[i, j] for j in J) >= lambda_i[i] *d[i] for i
     in I))
mrelajado.addConstrs((gp.quicksum(y[i, j] for j in J) <= d[i] for i in I))
mrelajado.addConstrs((gp.quicksum(y[i, j] for i in I) <= z[j] for j in J))
mrelajado.addConstrs((y[i, j] <= d[i] * x[j] for i in I for j in J))

# Optimize the model
mrelajado.optimize()

## Problema lineal para los seis escenarios propuestos

def modelo_lineal(instalaciones, clientes, distancias, demandas, precios, pi,
                  lambdaa, costes_fijos, costes_variables, M1, M2, M3, M4, M5,
    M6 ):
    '''
    instalaciones: conjunto de las instalaciones de las que se dispone
    clientes: conjunto de clientes que tienen demanda
    distancias: distancia desde el cliente i a  la  instalacion j. Se asume que
    el coste es igual a la distancia
    demandas: demanda del cliente i
    precios: precio que paga el cliente  i por que se  le atienda su demanda
    pi: ganancia unitaria por atender la demanda, es el precio-distancias
    lambdaa: porcentaje  minimo de demanda que se esta  obligado a satisfacer
    costes_fijos: costes que tiene el lider por abrir una instalacion j
    costes_variables: costes que tiene el lider por cada unidad de capacidad  de
     la instalacion j
    M1, M2, M3, M4, M5,  M6: cotas validas para sus correspondientes
    restricciones
    '''
    mlineal = gp.Model("mip", env = env)

    #mlineal.setParam("OutputFlag", 0)

    # Variables
    x = mlineal.addVars(instalaciones, vtype=GRB.BINARY, name="x")
    z = mlineal.addVars(instalaciones, vtype=GRB.CONTINUOUS, name="z")
    y = mlineal.addVars(clientes, instalaciones, vtype=GRB.CONTINUOUS, name="y")

    w = mlineal.addVars(clientes, vtype=GRB.CONTINUOUS, name="w")
    u = mlineal.addVars(instalaciones, vtype=GRB.CONTINUOUS, name="u")

    alpha = mlineal.addVars(clientes, vtype=GRB.BINARY, name="alpha")
    beta = mlineal.addVars(instalaciones, vtype=GRB.BINARY, name="beta")
    gamma = mlineal.addVars(clientes, instalaciones, vtype=GRB.BINARY, name="
    gamma")

    # Funcion Objetivo
    objective=gp.quicksum(costes_fijos[j] * x[j] for j in instalaciones) + gp.
    quicksum(costes_variables * z[j] for j in instalaciones)
    mlineal.setObjective(objective, GRB.MINIMIZE)

    # Restricciones
    mlineal.addConstrs((gp.quicksum(y[i, j] for j in instalaciones) >= lambdaa[i
```

```
    ] *demandas[i] for i in clientes))
507     mlineal.addConstrs((gp.quicksum(y[i, j] for j in instalaciones) <= demandas[
    i] for i in clientes))
508     mlineal.addConstrs((gp.quicksum(y[i, j] for i in clientes) <= z[j] for j in
    instalaciones))
509     mlineal.addConstrs((w[i] + u[j] >= pi[i, j] * x[j] for i in clientes for j
    in instalaciones))
510     mlineal.addConstrs((y[i, j] <= demandas[i] * x[j] for i in clientes for j in
     instalaciones))
511     mlineal.addConstrs((y[i, j] <= M1[i] * (1 - gamma[i, j]) for i in clientes
    for j in instalaciones))
512     mlineal.addConstrs((demandas[i] - gp.quicksum(y[i, j] for j in instalaciones
    ) <= M2[i] * (1 - alpha[i]) for i in clientes))
513     mlineal.addConstrs((z[j] - gp.quicksum(y[i, j] for i in clientes) <= M3[j] *
     (1 - beta[j]) for j in instalaciones))
514     mlineal.addConstrs((w[i] <= M4[i] * alpha[i] for i in clientes))
515     mlineal.addConstrs((u[j] <= M5[j] * beta[j] for j in instalaciones))
516     mlineal.addConstrs((u[j] <= M5[j] * x[j] for j in instalaciones))
517     mlineal.addConstrs((w[i] + u[j] - pi[i, j] <= M6 * gamma[i, j] for i in
    clientes for j in instalaciones))
518
519     # Optimizar
520     mlineal.optimize()
521
522     if mlineal.status == GRB.INFEASIBLE:
523         print('El modelo es infactible')
524         return None, None, None, None, None, None, None
525     else:
526
527         #Instalaciones abiertas
528         decision_instalaciones=pd.DataFrame()
529         for j in instalaciones:
530             decision_instalaciones.loc[j,'x']=x[j].X
531             decision_instalaciones.loc[j,'z']=z[j].X
532             # decision_instalaciones.loc[j,'Capacidad_utilizada']=sum(y[i,j].X
    for i in clientes)
533
534         #Valores funciones objetivo
535         #lider
536         vfo_lider=mlineal.ObjVal
537         #seguidor
538         vfo_seguidor=sum(pi[i,j]*y[i, j].X for i in clientes for j in
    instalaciones)
539
540         #demandad satisfecha
541         demanda_satisfecha=pd.DataFrame()
542         for i in clientes:
543             demanda_satisfecha.loc[i,'Demanda_satisfecha']=(sum(y[i,j].X for j
    in instalaciones)/demandas[i])*100
544
545         #numero de clientes con demanda satisfecha al 100%
546         clientes_demanda_satisfecha=len(demanda_satisfecha[demanda_satisfecha.
    Demanda_satisfecha==100].index)
547
548         #numero de clientes con demanda satisfecha al min
549         clientes_min_demanda_satisfecha=len(demanda_satisfecha[
    demanda_satisfecha.Demanda_satisfecha==lambdaa*100].index)
550
551         #variables y_{ij}
552         variables_y_ij = pd.DataFrame(index=clientes, columns=instalaciones)
553         for i in clientes:
554             for j in instalaciones:
555                 variables_y_ij.loc[i, j] = y[i, j].X
```

```
556
557
558         return decision_instalaciones, vfo_lider, vfo_seguidor,
        demanda_satisfecha, clientes_demanda_satisfecha,
        clientes_min_demanda_satisfecha, variables_y_ij
559
560 escenarios=range(0,2)
561
562 resultados=dict()
563
564 for e in escenarios:
565   resultados[e]=dict()
566
567   # Conjuntos
568   I = [i for i in range(0,100)]
569   J = [j for j in range(0,10)]
570
571   # Parametros
572   d=demanda_i
573   p_i = np.array([
574         [74 for _ in I],
575         [119 for _ in I],
576         [96.5 for _ in I],
577         [min(matriz_distancias[i, :]) for i in I],
578         [max(matriz_distancias[i, :]) for i in I],
579         [(max(matriz_distancias[i, :]) + min(matriz_distancias[i, :])) / 2 for i
        in I]
580       ]).T
581
582   pi = p_i[:,e:e+1] - matriz_distancias
583   #lambda_i=np.array([0.1 for i in I])
584   #lambda_i=np.array([0.5 for i in I])
585   #lambda_i=np.array([0.5 for i in I])
586
587   f = f_values
588   g = 1 #se puede asumir g_j=g
589
590   # Valores M_n
591   M1 = {i: d[i] for i in I}
592   M2 = {i: (1 - lambda_i[i]) * d[i] for i in I}
593   M3 = {j: sum(d[i] for i in I) for j in J}
594   M4 = {i: max(0,max([pi[i, j] for j in J])) for i in I}
595   M5 = {j: max(0,max([pi[i, j] for i in I])) for j in J}
596   M6 = max(0, max([pi[i,j] for i in I for j in J] ))
597
598
599   result=modelo_lineal(J,I,matriz_distancias,demanda_i,p_i,pi,lambda_i,f,g,M1,M2
        ,M3,M4,M5,M6)
600   resultados[e]['decision_instalaciones']=result[0]
601   resultados[e]['vfo_lider']=result[1]
602   resultados[e]['vfo_seguidor']=result[2]
603   resultados[e]['demanda_satisfecha']=result[3]
604   resultados[e]['clientes_demanda_satisfecha']=result[4]
605   resultados[e]['clientes_min_demanda_satisfecha']=result[5]
606   resultados[e]['variables_y_ij']=result[6]
```