Trabajo Fin de Máster

Modelos de optimización aplicados a la localización de cargadores de vehículos eléctricos

Alfonso Carballo Puebla

Máster en Técnicas Estadísticas

Curso 2024-2025

Propuesta de Trabajo Fin de Máster

Título en galego: Modelos de optimización aplicados á localización de cargadores de vehículos eléctricos

Título en español: Modelos de optimización aplicados a la localización de cargadores de vehículos eléctricos

English title: Optimization models applied to the location of charging stations for electric vehicles

Modalidad: Modalidad A

Autor/a: Alfonso Carballo Puebla, Universidad de Santiago de Compostela

Director/a: María Luisa Carpente Rodríguez, Universidad de Coruña;

Breve resumen del trabajo:

En la actualidad, se está promoviendo el uso de vehículos eléctricos. Contar con este tipo de vehículos es importante por razones de sostenibilidad ambiental, eficiencia energética, ahorros en los costes logísticos, etc. Por todo esto, los modelos de optimización aplicados a vehículos eléctricos son importantes y tratan diversos aspectos de la gestión de dichos vehículos. Por ejemplo, puede ser interesante ubicar de manera óptima las estaciones de carga, gestionar una flota con diferentes particularidades, optimizar el uso de la energía y aumentar la autonomía, etc. En este trabajo, nos centraremos en los modelos de optimización aplicados a la localización óptima de estaciones de carga, y se realizará una revisión de los principales trabajos que aparecen en la literatura, ilustrando las diferentes técnicas descritas en ellos con ejemplos.

Recomendaciones:

Shen, Zuo-Jun Max, et al. "Optimization models for electric vehicle service operations: A literature review." *Transportation Research Part B: Methodological* 128 (2019): 462-477.

Otras observaciones:

Índice general

Capítulo 1

Resumen

Resumen en español

En este trabajo realizamos una revisión de los modelos de optimización aplicados a la localización de estaciones de carga para vehículos eléctricos, un aspecto clave en la expansión de esta infraestructura para fomentar el uso del vehículo eléctrico como medida de reducción de emisiones y transición hacia una movilidad más sostenible.

Analizamos los principales enfoques existentes, clasificados en dos grandes categorías según la forma de modelar la demanda: modelos basados en nodos, habitualmente empleados en entornos urbanos, y modelos basados en flujos de viajes origen-destino, más orientados a áreas geográficas amplias. Se presentan las formulaciones básicas y las principales formas de incorporar otros aspectos relevantes al problema, como las restricciones de capacidad o las particularidades del sistema de transporte. Además, se revisan los métodos de resolución más habituales, generalmente asociados a la programación lineal entera mixta.

Finalmente, se complementa la revisión con un caso práctico aplicado a Galicia, que ilustra la aplicación de estos modelos en un contexto cercano a la realidad y permite comparar su funcionamiento y resultados.

English abstract

This work presents a review of optimization models applied to the location of charging stations for electric vehicles, a key aspect in the expansion of this infrastructure to promote the use of electric vehicles as a measure to reduce emissions and support the transition towards more sustainable mobility.

We analyze the main existing approaches, classified into two broad categories according to how demand is modeled: node-based models, typically applied in urban environments, and flow-based models, where demand is represented by origin-destination trips and is more suitable for larger geographic areas. The basic formulations are presented, along with the main ways of incorporating other relevant aspects of the problem, such as capacity constraints or the specific characteristics of the transport system. In addition, we review the most common solution methods, usually based on mixed-integer linear programming.

Finally, the review is complemented with a practical case study applied to Galicia, which illustrates the application of these models in a context close to reality and allows for a comparison of their performance and results.

Capítulo 2

Introducción

2.1. Estructura del trabajo

En este trabajo hacemos un repaso sobre modelos de optimización aplicados a la localización de cargadores de vehículos eléctricos, en este capítulo introductorio presentamos el contexto y la motivación para estos modelos. A continuación, en el capítulo 3 clasificaremos y veremos ejemplos de las principales formulaciones de estos modelos además de ver otros aspectos que se pueden incluir en la formulación, para complementarlo en el capítulo 4 revisaremos los métodos de solución asociados. Para ver todo esto en la práctica, en el capítulo 5 trabajamos con un ejemplo propio sobre Galicia, donde aplicamos los contenidos anteriores. Finalmente, en el capítulo 6 acabaremos con un comentario sobre la investigación futura y la conclusión.

2.2. Contexto y motivación

El transporte fue responsable de aproximadamente el 24 % de las emisiones globales de CO2 en 2022 [1], las cuales agravan significativamente el cambio climático, que es uno de los grandes problemas que enfrenta la humanidad en la actualidad. Para reducir estas emisiones, una opción es sustituir los vehículos de combustión por vehículos eléctricos, acompañados, por supuesto, de energías renovables que no generen emisiones de CO2. En este segundo aspecto, las energías renovables conformaron el 30 % de la generación global de electricidad, liderando Europa y, en concreto, España con un 47 % y 52 % en 2023 respectivamente[2]. En el primer aspecto, más importante para este trabajo, desde la década de 2010, el coche eléctrico ha comenzado a formar parte del parque automovilístico, representando en 2024 el 4.5 % a nivel mundial y el 1.8 % en España [3]. Sin embargo, estas cifras aún no son suficientes para marcar una diferencia significativa en la reducción de gases de efecto invernadero. Si consideramos las ventas, podemos ser algo más optimistas, ya que los coches eléctricos representaron el 22 % de las ventas globales, con la mayor parte del crecimiento ocurriendo en los últimos cuatro años 2.1 y esperándose un crecimiento similar en los años siguientes.

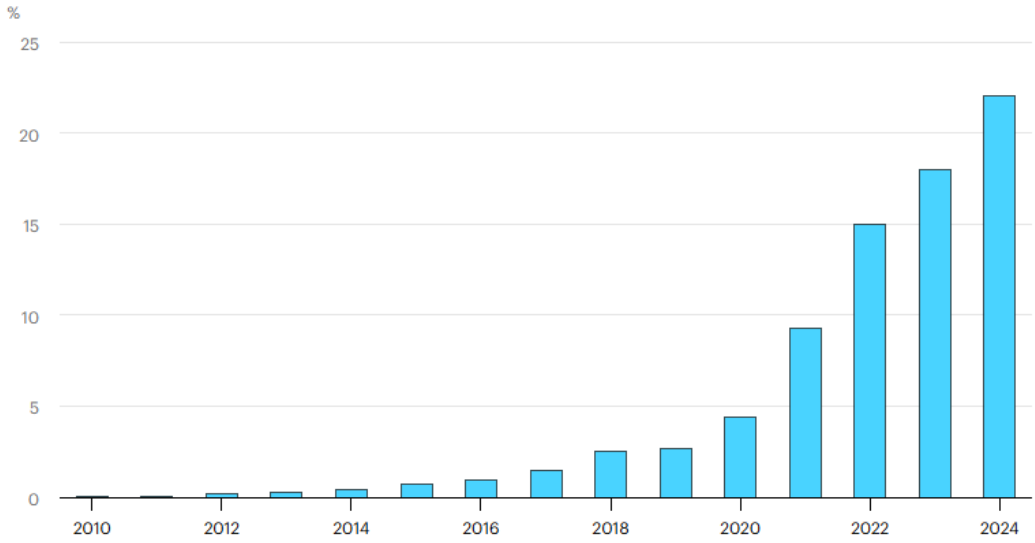

Figura 2.1: Cuota mundial de ventas de turismos eléctricos (%).
Fuente: IEA 2025[4]; "EV sales share, cars, World, 2010–2024", https://www.iea.org/data-and-statistics/data-tools/global-ev-data-explorer, Licencia: CCBY4.0

Algunos de los impedimentos para que estas cifras no sean más altas son los inconvenientes para el usuario del coche eléctrico en comparación con el coche de combustión, estos son:

- Alto precio inicial: Siendo en Europa y Estados Unidos entre un 10 % y un 50 % más caros que su contraparte no eléctrica [3], con una tendencia claramente descendente. En China, en cambio, los vehículos eléctricos son ya más baratos que los vehículos de combustión.
- Baja autonomía: La autonomía estimada de un coche eléctrico nuevo varía entre los 250 y 500 km. Aunque cada vez haya modelos con mayor autonomía estos suelen ir acompañados de un precio mayor.
- Infraestructura de carga insuficiente: no existen suficientes cargadores en comparación, por ejemplo, con el número gasolineras y es necesario adaptar tus viajes a los cargadores existentes. Además no hay incentivos para poner cargadores por los poca cantidad de usuarios de vehículos eléctricos, por lo cuál se crea un dilema de "el huevo y la gallina".

Para aliviar estos problemas, el primero es una cuestión de tecnología y tiempo para que esta se desarrolle, además se suele paliar con ayudas públicas; el segundo problema también es una cuestión de tecnología además de ser un problema que es "multiplicativo" con el tercer problema, por lo que aplacando el tercer problema también suavizamos este. En cuanto al tercer problema, el estado puede invertir en infraestructura pública o dar incentivos para crear infraestructura privada. Aquí entra el problema de la localización de esta infraestructura para cubrir la demanda de la forma más eficiente posible. Así, este trabajo se enfoca en este problema, usando la optimización para determinar las mejores localizaciones para cargadores de forma que la demanda quede lo mejor cubierta atendiendo a diferentes parámetros.

Tecnología de carga

Para plantear el problema de la localización de cargadores necesitamos primero conocer las características técnicas de los posibles cargadores que instalaremos. Actualmente existen tres tipos de cargadores según la velocidad de carga:

- Carga lenta: No un cargador literalmente, se refiere a la carga usando el enchufe Schuko, es decir el enchufe doméstico, por lo tanto no requiere ninguna instalación especial. Proporciona electricidad en corriente alterna, la cual es transformada por el vehículo a corriente contínua. La velocidad de carga depende del cable pero alcanza como mucho 2.3 kW, lo cual permite cargar del 10 % al 80 % de la batería de un coche estándar en 12-20 horas. Este tipo de carga no se suele tener en cuenta para los problemas de localización de cargadores.
- Carga semirrápida: De 4 kW a 22 kW, ya necesitan una instalación propia que puede costar entre 1000 y 3000 euros. Sigues proporcionando electricidad en corriente alterna. Se encuentran en lugares públicos o de trabajo. Se suele utilizar en estaciones de servicio, centros comerciales, estacionamientos y otros lugares de alta rotación de vehículos eléctricos. Permiten cargar del 10 % al 80 % de la batería de un coche estándar en 2-6 horas.
- Carga rápida: De 22 kW a 350 kW, los más rápidos y caros, su coste oscila entre los 15000 y los 70000 euros. Proporciona electricidad directamente en corriente contínua sin ser transformada por el vehículo. Son los cargadores que se encuentran en las estaciones de carga rápida o ultrarrápida (>150kW) y permiten cargar del 10 % al 80 % de la batería de un coche estándar en 20-60 minutos. Este tipo de carga es ideal para viajes largos y lugares de alta demanda.

Según la tecnología y velocidad de carga, unos modelos son más adecuados que otros; es importante tener claras estas diferencias para saber cuál es la tecnología que nos interesa y qué modelos están más orientados a ella.

Situación global de la infraestructura de carga pública.

El parque mundial de cargadores públicos superó los **5 millones** a finales de 2024, el doble que en 2022. Solo durante 2024 se añadieron **1,3 millones** de nuevos puntos, un aumento interanual superior al 30 % [5]. Para poder evaluar la cobertura, podemos comparar con el número de vehículos eléctricos, siendo la proporción global de 11 vehículos eléctricos por cargador y en Europa de 13 vehículos eléctricos por cargador.

Como se indicó anteriormente, las diferencias entre cargadores son importantes; saber cómo se distribuyen este número de cargadores, ya que es muy diferente una red donde los cargadores son en su mayoría lentos, lo cual dificulta la movilidad para grandes distancias, o una red con presencia de cargadores rápidos. En los últimos años, la categoría de cargadores rápidos y ultrarrápidos fue la de mayor crecimiento, al ser su tecnología asociada más reciente. La distribución global de los cargadores según su velocidad es:

- **3 millones** de cargadores semirrápidos ($\leq$ 22 kW).
- **2 millones** de cargadores rápidos ($\geq$ 22 kW).
- Dentro de los rápidos, los **ultrarrápidos** (>150 kW) representan ya ∼10 % (200.000 unidades) y son la categoría de mayor crecimiento (+50 % en 2024).

Una forma alternativa de evaluar la cobertura para la red de cargadores rápidos es ver la cobertura de autovías o vías rápidas según la distancia, así en Europa tenemos que el número de cargadores rápidos cada 50 km es de 1.6, comparado con el número de gasolineras que es 3.8, podemos ver que ya es una cobertura importante pero aún es necesario más crecimiento para alcanzar una cobertura similar a las gasolineras, además si tenemos en cuenta que el tiempo de carga es mayor quizás sea necesaria una cobertura mayor aún para evitar congestiones en las estaciones de carga.

Otro aspecto a remarcar es la complejidad del proceso de recarga: en la práctica, cada operador suele exigir su propia *app* o tarjeta RFID, de modo que un conductor termina gestionando varias

cuentas y métodos de pago para cubrir la red disponible. Este mosaico de sistemas no solo complica los desplazamientos transfronterizos, sino que ya se identifica como un freno directo a la adopción del VE [6].

Para subsanar esta fragmentación, el Reglamento (UE) 2023/1804 (AFIR) establece que todo punto de recarga público de potencia igual o superior a 50 kW instalado a partir del 13 de abril de 2024 debe aceptar el pago *ad hoc* mediante tarjeta o dispositivo contactless y mostrar el precio en €/kWh antes de iniciar la sesión [7]. Estas medidas anticipan un escenario de experiencia de usuario unificada, considerado clave para la generalización del vehículo eléctrico en Europa.

Una vez tenemos clara la situación actual del vehículo eléctrico y su infraestructura de carga, podemos introducir los modelos usados para modelar el problema de la localización de esta.

Capítulo 3

Formulación

En este capítulo repasaremos cómo se suele modelar y formular el problema de localización de estaciones de carga. El *charging station location problem* (CSLP) se puede clasificar dentro de la categoría del problema de localización de instalaciones (*facility location problem*, FLP), siendo una de sus principales características propias la restricción de autonomía de los vehículos eléctricos (*electric vehicles*, EVs). Así se distinguen dos formulaciones principales para abordar el problema:

- la formulación más clásica basada en nodos de una red a los que se asigna demanda.
- una formulación más reciente y frecuente en los trabajos contemporáneos, basada en flujo en la que la demanda se considera como los pares de origen-destino que conforman los viajes de los usuarios.

3.1. Modelos basados en nodos

Una de las formas más comunes de modelar el problema de localización de estaciones de carga es mediante formulaciones basadas en nodos. En este enfoque, la demanda se representa a través de nodos fijos, cada uno con una necesidad estimada de recarga, y el objetivo es determinar en qué ubicaciones instalar estaciones para optimizar ciertos criterios como la cobertura o la distancia.

Este tipo de modelos resulta especialmente adecuado en contextos urbanos o de pequeña escala, donde los usuarios recargan durante estancias prolongadas asociadas a otras actividades (como estar en casa, trabajar o hacer compras). A partir de esta lógica se han adaptado al contexto de la movilidad eléctrica diversas formulaciones clásicas del problema de localización de instalaciones, como el modelo de cobertura, de máxima cobertura, p-mediana o p-centro.

A continuación se presentan estos modelos, junto con sus principales características y limitaciones.

3.1.1. Set-covering

El modelo de *set-covering* fue introducido por Toregas et al. [8] con el objetivo de localizar servicios de emergencia de forma que todos los puntos de demanda quedasen cubiertos con el menor número de instalaciones posibles. En el contexto de infraestructura de recarga para vehículos eléctricos, este enfoque se adapta considerando que un nodo de demanda está cubierto si tiene una estación de carga a una distancia menor o igual a una distancia de cobertura D_c. Por ejemplo, podría tomarse como referencia la mitad de la autonomía de los vehículos si queremos una cobertura mínima o podría tomarse como referencia una distancia que se considere cómoda recorrer en caso de que necesites recargar el coche cotidianamente.

Una formulación típica del modelo es:

$$\text{Minimizar:} \quad Z = \sum_{j \in J} x_j \tag{3.1}$$

$$\text{Sujeto a:} \quad \sum_{j \in N_i} x_j \geq 1, \quad \forall i \in I \tag{3.2}$$

$$x_j \in \{0, 1\}, \quad \forall j \in J \tag{3.3}$$

Donde:

- I: Conjunto de nodos de demanda.
- J: Conjunto de ubicaciones candidatas para estaciones de carga.
- $N_i = \{j \in J : d_{ij} \leq D_c\}$: Conjunto de sitios candidatos que pueden cubrir al nodo de demanda i, siendo D_c la distancia máxima de cobertura.
- d_{ij}: Distancia entre el nodo de demanda i y el sitio candidato j.
- x_j: Variable binaria, igual a 1 si se instala una estación de carga en j, 0 en caso contrario.

El objetivo (3.1) minimiza el número total de estaciones de carga instaladas. La restricción (3.2) asegura que cada nodo de demanda esté cubierto por al menos una estación. Finalmente, (3.3) define la naturaleza binaria de las decisiones de localización.

Este modelo, si bien es sencillo e intuitivo, presenta algunas limitaciones. No considera la magnitud de la demanda en cada nodo, no impone un límite máximo al número de estaciones (lo que puede derivar en soluciones costosas), y puede tener muchas soluciones *equivalentes* en términos de cobertura pero distintas desde el punto de vista operativo o de eficiencia.

3.1.2. Max-covering

Partiendo del modelo anterior, si consideramos que no es posible cubrir toda la demanda con el número de puntos de carga que podemos instalar, podemos tomar como función a optimizar la demanda cubierta y añadir el límite de estaciones como una restricción. Así, el modelo de máxima cobertura (*maximum covering location problem*), introducido por Church y ReVelle [9], maximiza la demanda cubierta por p estaciones de carga. Además, tiene en cuenta la demanda en cada nodo y permite que existan nodos no cubiertos.

Una formulación básica del modelo es la siguiente:

$$\text{Maximizar:} \quad Z = \sum_{i \in I} w_i z_i \tag{3.4}$$

$$\text{Sujeto a:} \quad \sum_{j \in J} x_j = p \tag{3.5}$$

$$z_i - \sum_{j \in N_i} x_j \leq 0, \quad \forall i \in I \tag{3.6}$$

$$x_j \in \{0,1\}, \quad \forall j \in J \tag{3.7}$$

$$z_i \in \{0,1\}, \quad \forall i \in I \tag{3.8}$$

Donde:

- I: Conjunto de nodos de demanda.
- J: Conjunto de ubicaciones candidatas para estaciones de carga.
- $N_i = \{j \in J : d_{ij} \leq D_c\}$: Conjunto de sitios candidatos que pueden cubrir al nodo i.
- d_{ij}: Distancia entre el nodo i y el sitio j.
- w_i: Demanda asociada al nodo $i \in I$.
- x_j: Variable binaria, igual a 1 si se instala una estación de carga en j.
- z_i: Variable binaria, igual a 1 si el nodo de demanda i está cubierto.
- p: Número total de estaciones de carga que se pueden instalar.

El objetivo (3.4) maximiza la suma ponderada de la demanda cubierta. La restricción (3.5) impone un límite al número total de estaciones instaladas. La restricción (3.6) asegura que un nodo de demanda esté cubierto ($z_i = 1$) solo si al menos una estación en su conjunto de cobertura N_i está activa.

Una debilidad respecto al modelo anterior es que se debe fijar de antemano el número de estaciones a instalar. Además, si es posible cubrir todos los puntos de demanda, el modelo puede tener múltiples soluciones equivalentes en cuanto a cobertura total. Por otro lado, tanto este modelo como el de *set-covering* únicamente consideran si un nodo está cubierto o no, pero no miden cómo de "bien.ésta cubierto. Este aspecto se explora en los modelos siguientes.

3.1.3. Problema p-mediana

Una forma alternativa de modelar la localización de estaciones de carga consiste en minimizar la distancia entre los usuarios y las estaciones, en lugar de asegurar únicamente su cobertura. El problema *p-mediana*, propuesto inicialmente por Hakimi [10, 11], busca determinar la ubicación de p estaciones de carga de forma que la suma ponderada de las distancias desde los nodos de demanda a su estación más cercana sea mínima.

Una formulación básica del modelo es:

$$\text{Minimizar:} \quad Z = \sum_{i \in I} \sum_{j \in J} w_i d_{ij} y_{ij} \tag{3.9}$$

$$\text{Sujeto a:} \quad \sum_{j \in J} y_{ij} = 1, \quad \forall i \in I \tag{3.10}$$

$$y_{ij} \leq x_j, \quad \forall i \in I, \forall j \in J \tag{3.11}$$

$$\sum_{j \in J} x_j = p \tag{3.12}$$

$$y_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{3.13}$$

$$x_j \in \{0,1\}, \quad \forall j \in J \tag{3.14}$$

Donde:

- I: Conjunto de nodos de demanda.
- J: Conjunto de ubicaciones candidatas para estaciones de carga.
- d_{ij}: Distancia entre el nodo de demanda i y el candidato j.
- w_i: Demanda o peso asociado al nodo i.
- y_{ij}: Variable binaria, igual a 1 si el nodo i se asigna al sitio j.
- x_j: Variable binaria, igual a 1 si se instala una estación en j.
- p: Número total de estaciones de carga a instalar.

La función objetivo (3.9) minimiza la distancia total, ponderada por demanda, entre los nodos de demanda y sus estaciones asignadas. La restricción (3.10) asegura que cada nodo de demanda se asigne a exactamente una estación. La restricción (3.11) garantiza que un nodo solo puede asignarse a una estación si esta ha sido instalada. La restricción (3.12) fija el número total de estaciones de carga a instalar en p. Finalmente, (3.13) y (3.14) definen las variables como binarias.

Este modelo permite optimizar la eficiencia general de la red al reducir las distancias promedio entre los usuarios y sus estaciones asignadas. No obstante, presenta algunas limitaciones, como la necesidad de fijar previamente el número de estaciones a instalar, así como el riesgo de que ciertos nodos queden asignados a estaciones muy alejadas si la red no está equilibrada, quedando ”aislados” en la práctica.

3.1.4. Problema p-centro

Una alternativa al modelo *p-mediana*, especialmente útil cuando se desea garantizar un servicio equitativo, es el modelo *p-centro*. Este problema, también introducido por Hakimi [10, 11], busca minimizar la peor situación posible: la máxima distancia entre un nodo de demanda y su estación de carga más cercana.

El objetivo del modelo es localizar p estaciones de carga en la red de forma que se minimice dicha distancia máxima, evitando así que haya nodos ”aislados” con tiempos de acceso excesivos.

La formulación típica del problema *p-centro* es:

$$\text{Minimizar:} \quad Z \tag{3.15}$$

$$\text{Sujeto a:} \quad y_{ij} \leq x_j, \quad \forall i \in I, \forall j \in J \tag{3.16}$$

$$\sum_{j \in J} y_{ij} = 1, \quad \forall i \in I \tag{3.17}$$

$$\sum_{j \in J} x_j = p \tag{3.18}$$

$$\sum_{j \in J} d_{ij} y_{ij} \leq Z, \quad \forall i \in I \tag{3.19}$$

$$y_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{3.20}$$

$$x_j \in \{0,1\}, \quad \forall j \in J \tag{3.21}$$

Donde:

- Z: Distancia máxima a minimizar.
- I: Conjunto de nodos de demanda.
- J: Conjunto de ubicaciones candidatas para estaciones de carga.
- d_{ij}: Distancia entre el nodo de demanda i y el candidato j.
- w_i: Demanda o peso asociado al nodo i.
- y_{ij}: Variable binaria, igual a 1 si el nodo i se asigna al sitio j.
- x_j: Variable binaria, igual a 1 si se instala una estación en j.
- p: Número total de estaciones de carga a instalar.

La función objetivo (3.15) minimiza la mayor distancia entre cualquier nodo de demanda y su estación asignada. Las otras restricciones son análogas al problema *p-mediana*.

Este modelo puede interpretarse como una versión inversa del *set-covering*, en la que se determina la mínima distancia de cobertura necesaria para garantizar la cobertura completa de la red. A diferencia de los modelos de cobertura clásicos, aquí se consideran explícitamente las distancias individuales, aunque únicamente en función del peor caso.

Conclusión sobre los modelos basados en nodos

Todos estos modelos basados en nodos comparten una característica fundamental: modelan la demanda de forma estática y agregada, considerando únicamente la ubicación de los usuarios y su asignación a estaciones de carga cercanas. Esta aproximación resulta útil para representar el problema en términos simples y resolverlo con técnicas clásicas de optimización combinatoria. De hecho, estos modelos son especialmente aplicables en contextos urbanos o de pequeña escala, como la planificación de estaciones dentro de una ciudad o una región con pocos puntos de demanda claramente identificados.

Sin embargo, presentan importantes limitaciones cuando se quiere representar de forma más realista el uso de vehículos eléctricos. En primer lugar, no capturan el comportamiento dinámico asociado a los desplazamientos reales, ya que no modelan trayectos ni flujos entre nodos. Además, suelen requerir que el número de estaciones a instalar esté fijado de antemano, y no permiten incorporar restricciones derivadas de la autonomía de los vehículos, la posibilidad de recarga durante un viaje o los desvíos de ruta. Por último, estos modelos no distinguen entre distintas configuraciones de red que puedan ofrecer un mismo nivel de cobertura o distancia total, lo que puede dar lugar a soluciones poco robustas o subóptimas en la práctica.

Por estas razones, su aplicabilidad se reduce cuando el objetivo es planificar infraestructuras de recarga para viajes interurbanos o a gran escala. En tales casos, resulta más apropiado adoptar una visión basada en los flujos de tráfico, como la que se introduce en la siguiente sección.

3.2. Modelos basados en flujo

Una perspectiva más reciente es abordar el problema modelando la demanda como un conjunto de viajes origen-destino, donde cargadores rápidos son colocados de forma que la distancia entre cargadores no excede la autonomía, permitiendo así a los conductores completar los viajes sin quedarse sin batería. Algunas de las características principales son la estructura mono-nivel o bi-nivel, restricciones de capacidad de las estaciones, posibilidad de desviarse del trayecto más corto, elección de la función objetivo y cómo se cubre la demanda.
A continuación se presentan los modelos básicos basados en flujo los cuales hacen una serie de suposiciones para simplificar el problema real, las cuales son: (1) el tráfico entre un par Origen-Destino fluye por un solo camino (el trayecto más rápido normalmente); (2) el volumen de tráfico entre pares OD (origen-destino) es conocido; (3) los conductores tienen conocimiento sobre la localización de todos los cargadores y recargan lo necesario para completar sus viajes; (4) solo se puede colocar estaciones de carga en los nodos de la red; (5) los vehículos tienen una autonomía similar; (6) el consumo es directamente proporcional a la distancia recorrida; y (7) las estaciones de carga no tienen una capacidad fija. El contraste de estas suposiciones con la realidad es abordado al extender los modelos con más variables de decisiones o modificar algunas de las variables como veremos más adelante.

3.2.1. Captura de flujo

El modelo de captura de flujo (*Flow capturing location model*) (FCLM) fue introducido por Hodgson [12] y es el primero en plantear el problema de localización de estaciones de carga como un problema de flujo, basándose en el conocido modelo de "maximal covering facility location model". En el FCLM, un viaje q se considera cubierto si por lo menos una estación de carga se coloca a lo largo de q. Así, la formulación sería:

$$\text{Maximizar } Z = \sum_{q \in Q} f_q y_q \tag{3.22}$$

$$\text{sujeto a} \quad \sum_{k \in N_q} x_k \geq y_q \quad \forall q \in Q \tag{3.23}$$

$$\sum_{k \in N} x_k = p \tag{3.24}$$

$$x_k, y_q \in \{0,1\} \quad \forall q \in Q, k \in N \tag{3.25}$$

Donde:

- Q: Conjunto de viajes, pares OD.
- N: Conjunto de nodos k, donde se pueden instalar estaciones de carga.
- p: Número de estaciones de carga a instalar.
- f_q: Flujo en el viaje q, demanda.
- x_k: Variable binaria, igual a 1 si se instala una estación en k.
- y_q: Variable binaria, igual a 1 si el viaje q está cubierto por una estación.

Tenemos la función objetivo (3.22) que maximiza el flujo total que puede ser cubierto por las estaciones abiertas. La restricción (3.23) nos asegura que para que un viaje esté cubierto necesitamos por lo menos una estación en uno de sus nodos, mientras que la restricción (3.24) fija el número de estaciones construidas en p y la restricción 3.25 caracteriza las variables binarias. La principal limitación de esta formulación es que no tiene en cuenta la autonomía de los vehículos eléctricos.

3.2.2. Recarga de flujo

El primer modelo basado en flujo en tener en cuenta la autonomía de los vehículos fue el modelo de localización de recarga de flujo (*Flow refueling location model*) (FRLM) presentado por Kuby y Lim [13]. En este modelo se supone una autonomía fija R para todos los vehículos y para cubrir los viajes se usan combinaciones de estaciones de carga generadas en un paso previo. Entonces, una combinación h puede cubrir un viaje q cuando se puede recorrer q usando las estaciones de h sin quedarse sin batería. Un viaje es cubierto si una de las combinaciones capaces de cubrirlo es usada, es decir, si se colocan estaciones de carga en todos los nodos de h. Además de las variables binarias anteriores se usan también variables binarias v_h para indicar si se usa la combinación h.

$$\text{Maximizar } Z = \sum_{q \in Q} f_q y_q \tag{3.26}$$

$$\text{sujeto a} \quad \sum_{h \in H} b_{qh} v_h \geq y_q \quad \forall q \in Q \tag{3.27}$$

$$a_{hk} x_k \geq v_h \quad \forall h \in H, k \in N | a_{hk} = 1 \tag{3.28}$$

$$\sum_{kinN} x_k = p \tag{3.29}$$

$$x_k, y_q \in \{0, 1\} \quad \forall q \in Q, k \in N \tag{3.30}$$

Donde:

- Q: Conjunto de viajes, pares OD.
- N: Conjunto de nodos k, donde se pueden instalar estaciones de carga.
- p: Número de estaciones de carga a instalar.
- f_q: Flujo en el viaje q, demanda.
- x_k: Variable binaria, igual a 1 si se instala una estación en k.
- y_q: Variable binaria, igual a 1 si el viaje q está cubierto.
- a_{hk}: Variable binaria que caracteriza la combinación h, igual a 1 si el nodo k pertenece a h.
- b_{qh}: Variable binaria que caracteriza la combinación h, igual a 1 si la combinación h cubre el viaje q.

Así la función objetivo 3.26 representa el flujo cubierto como en el FCLM, la restricción 3.27 se traduce en que para cubrir un viaje ($y_q = 1$= por lo menos una de las combinaciones que lo cubre ($b_{qh} = 1$) es usada ($v_h = 1$), la restricción 3.28 conecta las variables binarias de las combinaciones con las de los nodos que las conforman y 3.29 fija el número total de estaciones construidas.
El principal punto negativo del modelo de localización de recarga de flujo es la ineficiencia a la hora de generar las combinaciones de nodos para casos reales, ya que el tiempo para generarlos crece exponencialmente con el número de arcos.

3.2.3. Cobertura de arcos

Para evitar la generación de combinaciones del modelo de recarga de flujo y mejorar la eficiencia computacional se propuso [14] una nueva formulación basada en la cobertura de arcos, según la cual un viaje q se considera cubierto si cada arco en q está cubierto. La idea se centra en incluir un conjunto de nodos K_{ij}^q que puede cubrir el arco (i,j) del viaje q, que sería como los puntos donde al recargar se puede recorrer el arco (i,j) sin quedarse sin batería.

$$\text{Maximizar} \sum_{q \in Q} f_q y_q \tag{3.31}$$

$$\text{sujeto a} \sum_{k \in K_{ij}^q} x_k \geq y_q \quad \forall q \in Q, (i,j) \in A_q \tag{3.32}$$

$$\sum_{kinN} x_k = p \tag{3.33}$$

$$x_k, y_q \in \{0,1\} \quad \forall q \in Q, k \in N \tag{3.34}$$

Donde:

- Q: Conjunto de viajes, pares OD.
- N: Conjunto de nodos k, donde se pueden instalar estaciones de carga.
- p: Número de estaciones de carga a instalar.
- f_q: Flujo en el viaje q, demanda.
- x_k: Variable binaria, igual a 1 si se instala una estación en k.
- y_q: Variable binaria, igual a 1 si el viaje q está cubierto.
- A_q: Conjunto de arcos (i,j) que conforman el viaje q.
- K_{ij}^q: Conjunto de nodos k que pueden cubrir el arco $(i,j) \in A_q$ del viaje q.

La función objetivo (3.31) representa el flujo cubierto, la restricción (3.33) enlaza las estaciones con los viajes y (3.34) fija el número de estaciones, manteniéndose como en el FCLM y se incluye la restricción (3.32) que nos asegura que para cubrir un viaje q, cada uno de sus arcos ($(i,j) \in A_q$) tiene al menos una estación abierta en uno de los nodos que permite cubrir ese arco.

3.2.4. Path-segment

Un segmento (*path-segment*) $[i,j]$ de un viaje q es una parte del camino o trayecto usado para llegar del origen del viaje O_q al destino D_q, así está conformado por un conjunto de arcos consecutivos en q usados para viajar del nodo i al nodo j, siendo su longitud la suma de las longitudes de estos arcos. Esta idea se basa en descomponer cada viaje en secuencias de segmentos que al recorrerlos consecutivamente se completa el viaje q, para esto la longitud de estos segmentos no debe superar la autonomía R (o $R/2$ si uno de sus extremos es O_q o D_q) y se construye una estación en el origen de cada segmento (excepto en O_q).
Uno de los primeros trabajos en introducir esta idea y formularla es MirHassani and Ebrazi [15], en forma de una red expandida ($\hat{N}_q, \hat{A}_q$ para cada viaje q. En esta red se crean nodos artificiales fuente

s y sumidero t. La formulación completa es la siguiente:

$$\text{Maximizar } Z = \sum_{q \in Q} f_q (1 - y^q_{st}) \tag{3.35}$$

$$\text{sujeto a} \quad \sum_{j|(i,j)\in \hat{A}_q} y^q_{ij} - \sum_{j|)j,i)\in \hat{A}_q} y^q_{ji} = \begin{cases} 1, & \text{if } i = s \\ -1, & \text{if } i = t \\ 0, & \text{if } i \neq s,t \end{cases} \qquad \forall q \in Q, i \in \hat{N}_q \tag{3.36}$$

$$\sum_{j|(j,i)\in \hat{A}_q} y^q_{ji} \leq x_i \qquad \forall i \in \mathcal{N}, q \in Q_i \tag{3.37}$$

$$\sum_{k \in N} x_k = p \tag{3.38}$$

$$y^q_{ij} \geq 0 \qquad \forall q \in Q, (i,j) \in \hat{A}_q \tag{3.39}$$

$$x_k \in \{0,1\} \qquad \forall k \in \mathcal{N} \tag{3.40}$$

Donde:

- Q: Conjunto de viajes, pares OD.
- N: Conjunto de nodos k, donde se pueden instalar estaciones de carga.
- N_q: Conjunto de nodos k en el viaje q.
- $\hat{N}_q$: Conjunto de nodos k en el viaje q con los nodos artificiales fuente s y sumidero t para el viaje q.
- p: Número de estaciones de carga a instalar.
- f_q: Flujo en el viaje q, demanda.
- x_k: Variable binaria, igual a 1 si se instala una estación en k.
- $\hat{A}_q$: Conjunto de arcos en la red artificial, obtenido al conectar cualquier par de nodos en $\hat{N}_q$ para el cual la distancia del segmento que los une no supera la autonomía R (o $R/2$ si uno de sus extremos es O_q o D_q).
- y^q_{st}: Variable que indica si el viaje q está cubierto, en tal caso vale 0, si no esta cubierto vale 1.
- y^q_{ij}: Variable que indica si el segmento $[i,j]$ del viaje q es utilizado para cubrir el viaje q, en tal caso vale 1.

La función objetivo 3.35 vuelve a representar el flujo cubierto, se ve que en este caso la variable y^q_{st} funciona al revés que en los otros modelos (hay flujo cuando $y^q_{st} = 0$). Las restricciones 3.36 mantienen el balance de masa, la restricción 3.37 nos asegura que se construyen estaciones en los nodos de los segmentos usados, 3.38 fija el número total de estacione, mientras que 3.39 y 3.40 caracterizan las variables correspondientes. Las variables y^q_{st} y y^q_{ij} aunque no se fijen como binarias, lo son en la práctica debido principalmente a la restricción 3.36. Si $y^q_{st} = 1$, el resto de las y^q_{ij} del viaje q valen 0, en caso contrario existe un camino $[s, i_1, ..., i_n, t]$ para el viaje q cuyas variables asociadas $y^q_{s,i_1}, y^q_{i_1,i_2}, ..., y^q_{i_n,t}$ valen 1.

3.3. Otras extensiones de modelos

Los modelos vistos anteriormente se basan principalmente en dos tipos de variables de decisión fundamentales. Por un lado, las variables de localización, que determinan en qué ubicaciones candidatas se instalan estaciones de carga; y por otro, las variables de asignación, que indican cómo se asigna la demanda a las estaciones instaladas. En los modelos basados en nodos, estas variables suelen reflejar qué nodo de demanda utiliza qué estación de carga. En los modelos basados en flujo, en cambio, representan qué trayectos origen-destino están cubiertos por la infraestructura instalada, es decir, si un flujo determinado puede completarse sin agotar la batería gracias a las estaciones presentes en la ruta.

Para representar de forma más realista el problema, muchos trabajos extienden esta formulación básica con variables adicionales. A continuación se describen algunas de las más relevantes, tanto para modelos basados en nodos como basados en flujo.

Capacidad

Una de las variables más importantes para reflejar el funcionamiento real de una red de recarga es la capacidad de las estaciones. Esta variable determina cuántos puntos de carga se instalan en cada ubicación seleccionada, lo cual permite tener en cuenta la posibilidad de saturación, los tiempos de espera y el nivel de servicio ofrecido.

La capacidad está estrechamente asociada con la demanda que se espera cubrir, ya sea en términos medios o en situaciones de máxima exigencia. En algunos enfoques, se modela la capacidad a partir de la demanda pico estimada; en otros, se consideran horizontes de planificación donde la demanda evoluciona en el tiempo. Esto permite incorporar decisiones dinámicas sobre ampliación de capacidad o ubicación escalonada.

Al introducir esta variable, también se invita a considerar elementos como las restricciones presupuestarias, las limitaciones técnicas de los cargadores, el impacto sobre la red eléctrica, como podemos ver en [16], o incluso la incorporación de modelos de colas. Varios trabajos muestran que tener en cuenta esta variable mejora notablemente la utilidad práctica de las soluciones obtenidas.

Desvío respecto a la ruta óptima

En los modelos basados en flujo, muchos enfoques asumen que los vehículos siguen siempre el camino más corto entre origen y destino. Sin embargo, esta restricción puede limitar significativamente la cobertura alcanzable o aumentar el número necesario de estaciones de recarga. Permitir que los vehículos se desvíen de su ruta óptima ofrece la posibilidad de cubrir una mayor parte de la demanda o hacerlo de forma más eficiente.

Modelar esta posibilidad de desvío presenta importantes desafíos computacionales, ya que calcular todas las rutas alternativas posibles es costoso. Por ello, en la práctica se suele proporcionar como entrada un conjunto limitado de rutas alternativas prefijadas, o bien se recurre a heurísticas que seleccionan subconjuntos relevantes. En algunos casos, esta decisión se integra con variables de enrutamiento, lo que permite determinar simultáneamente la ubicación de estaciones y los trayectos a seguir por los vehículos.

Algunos modelos también incorporan penalizaciones por desviación en términos de tiempo o distancia, lo que permite modular cuánto se desvía un vehículo respecto a su trayecto original. Esta flexibilidad resulta clave para lograr soluciones eficientes en redes con múltiples alternativas viables.

Una formulación destacada que introduce esta posibilidad es el modelo de desvío de flujo (*Deviation Flow Refueling Location Model*) (DFRLM), desarrollado por Kim y Kuby [17] como extensión del modelo original de recarga de flujo de Kuby y Lim [13]. Otro ejemplo relacionado es el *Multi-Path Refueling Location Model* (MPRLM) [18], que, si bien no modela desvíos respecto a una única ruta base, permite seleccionar entre múltiples rutas alternativas predefinidas, lo que introduce un grado adicional de flexibilidad en la cobertura sin asumir una única trayectoria óptima.

Además, Hosseini et al. [19] proponen una versión capacitada del modelo de desvío, conocida como *Capacitated Deviation Flow Refueling Location Model* (CDFRLM), que combina la posibilidad de desviarse de la ruta más corta con restricciones operativas sobre la capacidad de servicio de las estaciones. Este modelo se basa en la formulación de cobertura de arcos desarrollada por Capar et al. [14], y permite representar de forma más realista la saturación potencial en estaciones con alta demanda.

Enrutamiento de vehículos

Algunos modelos amplían la formulación básica incorporando variables de enrutamiento, es decir, determinando no solo dónde ubicar las estaciones de carga, sino también qué trayectos siguen los vehículos para completar sus viajes. Este tipo de formulaciones suele aparecer en contextos como el reparto de mercancías o los servicios de taxi eléctrico, donde las rutas no están prefijadas y pueden optimizarse junto con la infraestructura.

Integrar decisiones de enrutamiento permite modelar de forma conjunta la planificación de la red de carga y los patrones de movilidad, aunque también incrementa la complejidad computacional. Uno de los primeros trabajos de este tipo de modelos es el de Yang y Sun [20] donde se combinan decisiones de enrutamiento con estaciones de intercambio de baterías.

Este tipo de enfoque es especialmente útil en entornos urbanos o logísticos, donde el comportamiento de los vehículos puede controlarse o anticiparse de forma más precisa.

Tecnología de recarga

Normalmente, los trabajos sobre localización de estaciones de recarga se centran en una única tecnología concreta, como la recarga lenta, la rápida o el intercambio de baterías. Sin embargo, otra posibilidad es considerar la tecnología como una variable de decisión, permitiendo elegir entre diferentes tecnologías de carga con sus respectivos costes y niveles de servicio.

Incorporar estas decisiones tecnológicas permite adaptar mejor la red de estaciones a las necesidades de los usuarios y a las características de los viajes. Por ejemplo, las estaciones rápidas pueden ser más adecuadas para trayectos largos o zonas de alto tráfico, mientras que las lentas pueden ser suficientes en entornos residenciales o laborales.

Aunque esta variable no se incluye de forma tan habitual como las anteriores, algunos trabajos la han abordado explícitamente. Un ejemplo es el de Wang y Lin [21], que estudian la localización conjunta de estaciones de carga y de intercambio de baterías, considerando el coste total y la cobertura de la red como criterios de optimización.

3.3.1. Modelos con incertidumbre y formulaciones estocásticas

La localización de infraestructuras de carga implica decisiones estratégicas a lo largo de un horizonte amplio; durante ese periodo, parámetros como la demanda, los costes o los patrones de movilidad pueden variar de forma significativa. Dado que resulta difícil predecir con precisión dichas variaciones, se recurre a modelos estocásticos que incorporan la incertidumbre explícitamente. Una alternativa es plantear modelos multi-período, que permiten reubicar o ampliar la red conforme evolucionen los parámetros. Ambas líneas de investigación son relativamente recientes en la literatura sobre localización de cargadores para vehículos eléctricos.

En los modelos estocásticos la incertidumbre puede afectar a varias variables, siendo la demanda la más habitual, pues prever la velocidad de adopción del vehículo eléctrico y los desplazamientos futuros es complejo. También se modelan como aleatorias la autonomía efectiva (SOC y consumo por kilómetro), los costes de instalación o energía, los tiempos de viaje y tiempos de servicio en horas pico.

Una vez caracterizada la incertidumbre, el analista debe decidir cómo gestionarla:

1. *Criterio de valor esperado*: se optimiza la esperanza del coste o del flujo cubierto.

2. *Restricciones de probabilidad*: se impone que eventos críticos, como quedarnos sin batería, ocurran con probabilidad inferior a un umbral.

3. *Optimización robusta*: se diseña la red para el peor caso dentro de un conjunto plausible de realizaciones.

Cada enfoque conlleva un compromiso entre fidelidad al fenómeno, complejidad computacional y grado de conservadurismo de la solución.

Capítulo 4

Métodos de solución

Una vez tenemos formulado el problema, podemos proceder a su resolución; todos los modelos planteados en su forma básica son MILP, problemas de programación lineal entera mixta, y la principal diferencia entre modelos es su escala combinatoria y complejidad, teniendo normalmente los modelos basados en flujo mayores dimensiones que los modelos basados en nodos.

El procedimiento principal para resolver un problema de programación lineal entera mixta consiste en relajar las restricciones de integralidad, las cuales en nuestro caso suelen ser restricciones que caracterizan las variables binarias que modelizan decidir si instalamos una estación en un nodo, si un nodo de demanda está cubierto o asignado a una estación concreta, si un viaje está cubierto o asignado a cierta estación, etc. Una vez tenemos nuestro problema relajado a su versión lineal, al resolverlo obtenemos una cota para nuestro problema; la solución del problema MILP será peor que la del problema relajado ya que el problema relajado tiene menos restricciones, entonces podemos seguir un algoritmo de Branch–and–Bound para conseguir la solución del problema original. Branch–and–Bound consiste en elegir una de las variables x_i cuya restricción de integralidad habíamos relajado y tenga un valor no entero en la solución obtenida y crear dos nuevos problemas con una restricción extra $x_i \leq 0$ y $x_i \geq 1$ ($x_i \leq \lfloor x_i \rfloor$ y $x_i \geq \lceil x_i \rceil$ si la variable x_i no fuese binaria), cada uno de estos nuevos problemas se resuelve y sobre cada nuevo problema se vuelve a elegir otra variable para aplicar el mismo procedimiento a no ser que el nodo/problema a explorar cumpla alguna condición que nos indique que explorar ese nodo no nos va a proporcionar más información, esto se da en tres posibles casos:

- La solución cumple todas las restricciones del problema original, en cuyo caso es una cota de la peor solución posible del problema original si no teníamos una solución mejor como cota y además añadir más restricciones no conseguiría mejorar la solución de este nodo.

- El problema de optimización a resolver en ese nodo no es factible, por lo cual, añadir más restricciones al problema no lo hará factible y no obtendremos ninguna solución factible explorando este nodo.

- La solución obtenida es peor que la cota de la peor solución del problema original, como en el primer caso, añadir más restricciones no conseguiría mejorar la solución de este nodo.

Repetimos este proceso con nuevas variables hasta no tener más nodos por explorar y entonces nos quedamos con la mejor solución conocida. Dos aspectos importantes del algoritmo de Branch–and–Bound son cómo se seleccionan el nodo a explorar y la variable no entera sobre la que creamos las nuevas ramas, el procedimiento habitual es elegir el nodo con mejor solución y la variable con valor más próximo a un entero, aunque existen otros criterios para tomar estas decisiones. Este método de relajar el problema original y aplicar Branch–and–Bound es el que suelen seguir normalmente los solvers comerciales, ayudándose de algoritmos para reducir la complejidad del problema inicial.

El método anterior es el principal método dentro de los métodos exactos, estos son métodos que obtienen la solución óptima con certeza de que la solución obtenida es la óptima, aparte del método de Branch–and–Bound existen más métodos exactos para resolver estos modelos, tanto variantes del Branch–and–Bound como técnicas de descomposición, como por ejemplo Benders, apropiadas para problemas con variables binarias como los nuestros. Las técnicas de descomposición y Benders dividen el problema en dos niveles, problema maestro el cual contiene las variables difíciles (binarias o enteras) y subproblemas de programación lineal sencillos que añaden cortes al problema maestro acotando la región factible.

Los métodos exactos descritos son adecuados para problemas no muy complejos como los producidos por modelos basados en nodos o instancias pequeñas de modelos basados en flujo, pero si la complejidad del problema es alta, ya sea por la modelización o por el número de variables probablemente, el tiempo requerido por estos métodos exactos exceda cantidades razonables y tengamos que recurrir a otros métodos, otros motivos para recurrir a otros métodos es usar función objetivo o restricciones no lineales, tener un problema bi-nivel, etc. En estas situaciones es necesario recurrir a otro métodos para obtener una solución aproximada en tiempo razonable, entre estos métodos distinguimos dos categorías:

Heurísticas:

Los algoritmos heurísticos consisten en métodos para encontrar una solución lo suficientemente buena en un tiempo de computación breve, existen multitud de heurísticas que se pueden además modificar para adaptar al problema a tratar aunque las principales usadas en los modelos de localización suelen ser algoritmos greedy, algoritmos genéticos o algoritmos de relajación lagrangiana.

- **Heurísticas basadas en relajación *lagrangiana*.** Muy empleadas en modelos nodales [22]. La idea es *dualizar* las restricciones complicantes, trasladándolas al objetivo mediante multiplicadores de Lagrange y obteniendo así subproblemas lineales más manejables. La solución resultante suele ser infactible para el modelo original, pero proporciona una cota inferior; una sencilla fase de reparación produce una solución factible y, por tanto, una cota superior. Estas dos cotas pueden (i) sustituir a la relajación lineal dentro de un esquema Branch–and–Bound, reforzando las podas y acelerando la convergencia, o bien (ii) emplearse como procedimiento heurístico autónomo que ofrece con rapidez soluciones de buena calidad aunque sin garantía de optimalidad.

- **Heurísticas *greedy*.** Procedimientos voraces que construyen (o depuran) la solución paso a paso, seleccionando en cada iteración la alternativa con la mejor contribución local según un criterio de beneficio o razón beneficio/coste. En su versión *constructiva* se parte del conjunto vacío y se añaden emplazamientos hasta cumplir el número o la cobertura deseada; en la *destructiva* se parte de la solución completa y se eliminan nodos mientras la calidad permanezca dentro de un umbral. Estas heurísticas son muy rápidas ($\mathcal{O}(n^2)$ en los casos simples) y, aunque no garantizan cercanía al óptimo o proporcionan información sobre esta, producen buenas soluciones iniciales que pueden refinarse con búsqueda local o emplearse como arranque en algoritmos más sofisticados.

- **Algoritmos genéticos (GA).** Metaheurística evolutiva que mantiene una *población* de soluciones candidatas codificadas como cromosomas binarios. Cada generación se produce mediante operadores de selección (elige los individuos más aptos), cruce (combina pares de cromosomas) y mutación (altera aleatoriamente algunos genes), equilibrando exploración y explotación del espacio de búsqueda. Con mecanismos como elitismo y control adaptativo de tasas de cruce/mutación, los GA consiguen aproximaciones de alta calidad para problemas combinatorios de gran escala en tiempos de cómputo moderados. Junto con las heurísticas *greedy* son las heurísticas más comunes al tratar con modelos basados en flujo [23].

Aparte de estas heurísticas existen otras como recocido simulado, búsqueda adaptativa en vecindad amplia (ALNS), heurísticas basadas en relajación lagrangiana, búsqueda tabú, optimización por reacción química, algoritmos evolutivos híbridos, métodos inspirados en el comportamiento de las ballenas

y optimización por enjambre de partículas. No existe una heurística mejor que otras y depende mucho del problema a tratar. Una estrategia común es combinar heurísticas para abordar problemas complejos.

Métodos aproximados:

Cuando los modelos de localización de cargadores incorporan componentes no lineales, ya sea en restricciones o en función objetivo, por incorporar incertidumbre, estructura bi-nivel u otras características, el modelo puede volverse no convexo o intratable con otros métodos. Entonces se puede recurrir a reformular el modelo o aproximarlo a un modelo que suprima estas complejidades para obtener una solución aproximada. Dependiendo del origen y forma de la complicación el método para la reformulación varía, siendo desarrollado en función de la estructura del término no lineal y la información disponible.

Capítulo 5

Caso práctico: Galicia

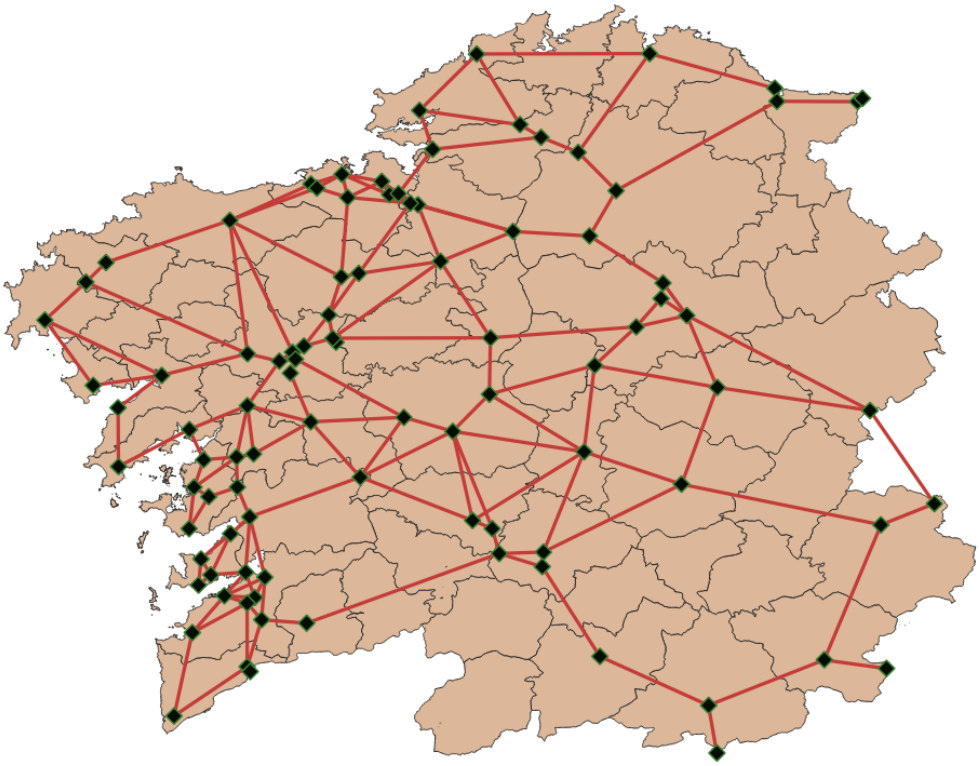

Figura 5.1: Red de arcos entre nodos de Galicia

5.1. Propósito y alcance

En este capítulo, de carácter demostrativo, vamos a ver en la práctica los modelos presentados en un ejemplo con datos inspirados de un caso real, basado en Galicia. Este capítulo tiene un carácter eminentemente *demostrativo*: ilustrar, con datos de Galicia, cómo se instancian los modelos de locali-

zación presentados en los capítulos 3 y 4. El objetivo no es un plan operativo, sino un *proof-of-concept* que permita

1. **Validar empíricamente** las formulaciones seleccionadas sobre una red realista;
2. **Comparar su comportamiento** en términos de cobertura y accesibilidad en un contexto de demanda dispersa como el gallego;

Acotación del estudio.

- La red viaria utilizada se construye, en el capítulo siguiente, a partir de los *nodos de población relevante* (municipios $> 7\,000$ hab.) y/o situación geográfica relevante para la red y los ejes que les dan servicio (autovías completas, la mayor parte de carreteras nacionales y ciertos tramos secundarios indispensables).
- Se fija un **horizonte único** (situación 2025) y se mantienen supuestos estáticos de autonomía y demanda; la dinámica temporal y la incertidumbre quedan fuera de este ejercicio preliminar.

Bajo este alcance reducido mostraremos, paso a paso, cómo se definen las variables y restricciones de los modelos y cómo se interpretan los resultados obtenidos.

5.2. Datos y construcción de la red

5.2.1. Nodos

- **Población.** Se partió del listado de municipios gallegos en `Wikipedia` [24] (consulta: enero 2025). Se retuvieron aquellos con población censada $> 7\,000$ hab. Para evitar una sobre–segmentación urbana se fusionaron núcleos contiguos (p. ej. *Ferrol+Narón*), resultando en $|OD| = 58$ *nodos-población*.
- **Conexiones externas.** Se añadieron $|OD_C| = 6$ nodos que representan los accesos viarios clave con Portugal (A55, A75) y con España (A6, A8, A52, N120).
- **Aeropuertos.** Tres nodos adicionales ($|OD_A| = 3$) para SCQ, LCG y VGO.
- **Intersecciones estratégicas.** Para dotar de granularidad a la red se crearon $|J| = 30$ nodos de cruce (enlaces de autovía, cambios de corredor), guiándose por cartografía.

Así, el conjunto total es $|N| = |OD| + |OD_C| + |OD_A| + |J| = 97$ nodos.

Para los pesos de los nodos de conexiones externas y aeropuertos se usó una cifra provisional de población que al calcular los flujos se ajustó con el tráfico de tramos seleccionados con los datos del ministerio de transportes [25]. La lista completa de nodos con sus poblaciones se puede ver en el Anexo

5.2.2. Digitalización de la malla y cálculo de atributos

1. **Vectorización.** Mediante *QGIS 3.34* se digitalizaron los arcos que conectan nodos adyacentes a través de: *i)* toda la red de autovías (AP–9, A–6, A–8, A–52, CG-n); *ii)* la mayor parte de carreteras nacionales y, *iii)* tramos secundarios imprescindibles para asegurar conectividad. Se obtuvieron $|A| = 167$ arcos. Como se ve en la figura 5.1.
2. **Distancias y tiempos.** Para cada arco (i, j) se interrogó la *Google Directions API* registrando: distancia d_{ij} y tiempo t_{ij}.
3. **Matrices de coste mínimo.** Con un algoritmo de Dijkstra (`Python`, ponderado por t_{ij}) se generaron las matrices $\mathbf{T} = (T_{ij})$ y $\mathbf{D} = (D_{ij})$ de tiempo (min) y distancia (km) mínimos entre pares de nodos.

5.2.3. Estimación del flujo OD mediante un modelo gravitacional

Para obtener o estimar los flujos de tráfico tenemos varias opciones: Disponemos de datos diarios de movilidad entre municipios por el Ministerio de Transportes, con el paquete `spanishoddata` (versión 0.2.0) [27] de los cuales podemos seleccionar días 2024 para obtener una representación de la movilidad que se aproxime a la realidad. La otra principal opción es usar un modelo gravitacional, los cuales se usan ampliamente en estudios de movilidad y planificación urbana [26], para el cual podemos tomar parámetros usados en otros ejemplos de aplicación de nuestros modelos o calibrar el modelo usando los datos de `spanishoddata`. Si nos decantamos por usar los datos reales o un modelo basado en estos veremos que la gran parte de la movilidad y flujo está conformada por viajes relativamente cortos (20 Km) que no son los que nos interesan para comprobar nuestros modelos de flujo más orientados a trayectos más largos. Así si ajustáramos un modelo gravitacional de forma logarítmica a la movilidad de una selección de días de 2024 obtenemos el siguiente modelo:

El modelo adoptado asume simetría en los flujos ($F_{ij} = F_{ji}$) e introduce una única elasticidad poblacional λ para ambos extremos del viaje. La forma funcional utilizada es la habitual en la literatura, combinando la masa poblacional de origen y destino, la distancia entre ambos y el número de peajes en la ruta mínima:

$$F_{ij} = k \cdot (P_i P_j)^{0,93} \cdot D_{ij}^{-2,54} \cdot (1 + Peaje_{ij})^{-0,33}, \qquad i \neq j, \tag{5.1}$$

donde P_i y P_j representan las poblaciones de origen y destino, D_{ij} es la distancia en kilómetros por red viaria y $Peaje_{ij}$ el número de peajes en la ruta. El factor k se ajusta mediante $k = e^{\beta_0}$, con $\beta_0 = 10{,}74$, si bien este valor no afecta a los resultados relativos de los modelos de localización que se utilizarán más adelante. No obstante, podría ajustarse a posteriori para reflejar diferentes magnitudes de la movilidad total, por ejemplo, para diferenciar entre movilidad general y movilidad eléctrica.
Con este modelo obtenemos vemos que a pesar que los viajes están repartidos en distancia entre 0 y 300 km, la mayor parte del flujo se concentra en viajes muy cortos 5.2, lo cual, a pesar de aproximarse a la realidad no es práctico para nuestra intención de aplicar modelos basados en flujo como el FCLM o el FRLM.

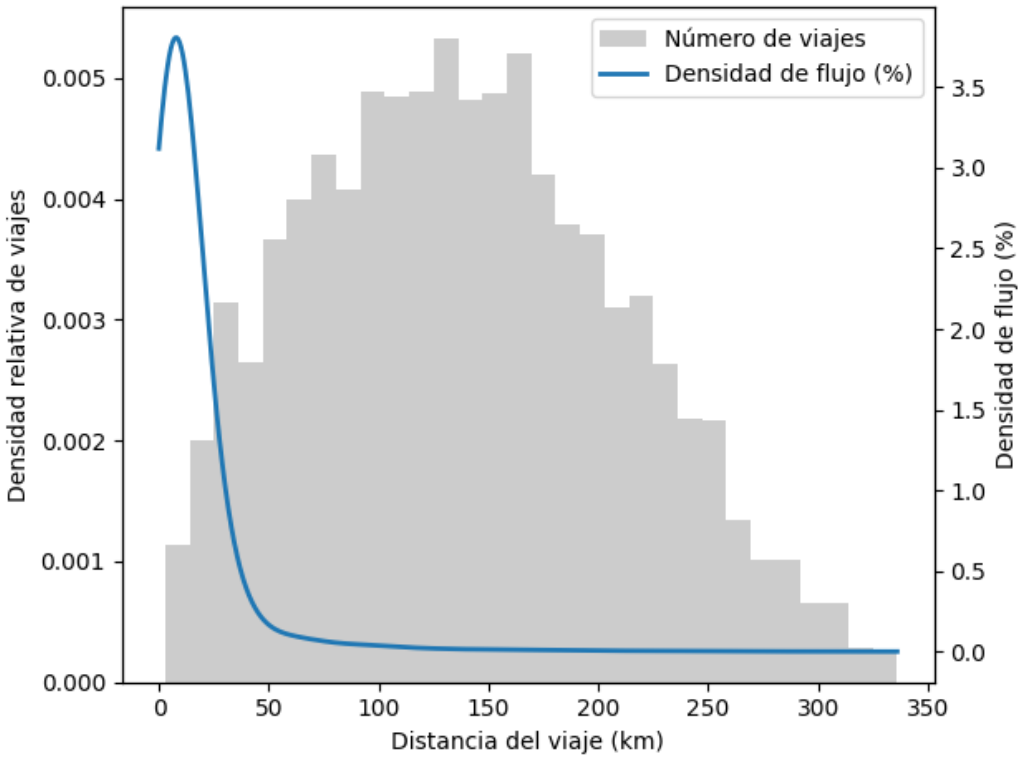

Figura 5.2: Distribuciones de viajes y flujo

Para solventar esto podemos o bien ajustar los parámetros para que la distancia no sea tan perjudicial para la estimación o cortar los viajes de menos de cierta distancia que consideremos no sean el objetivo de nuestra red de cargadores. Al final optamos por la primera opción ya que no tenemos una cifra fija para determinar cuáles son los viajes objetivo y la distribución sigue penalizando excesivamente la distancia aunque cortemos los viajes más cortos.

Así sustituimos el modelos gravitacional reduciendo las penalizaciones por distancia y peajes y obtenemos una distribución para la cual tienen más interés los modelos de localización de estaciones de carga a aplicar 5.3.

$$F_{ij} = k \cdot (P_i P_j)^{0,93} \cdot D_{ij}^{-0,5} \cdot (1 + Peaje_{ij})^{-0,1}, \qquad i \neq j, \tag{5.2}$$

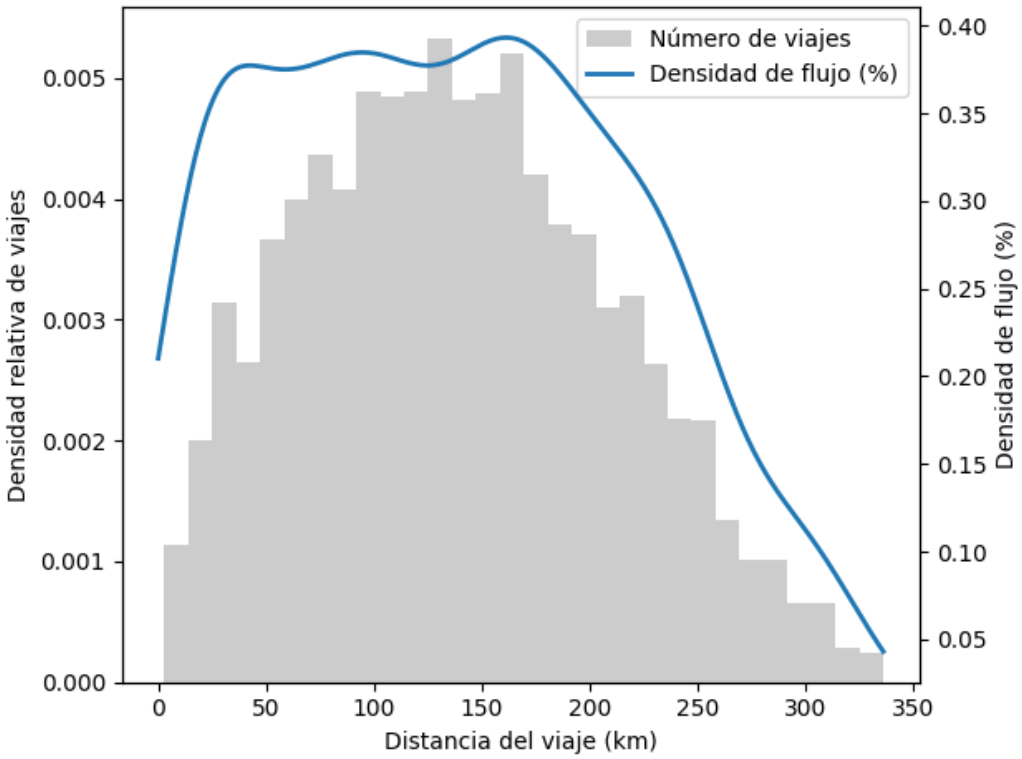

Figura 5.3: Distribuciones de viajes y flujo con el flujo modificado

Por otro lado, es necesario realizar ciertos ajustes para los nodos especiales, como aeropuertos y conexiones exteriores. En estos casos, se asumió que la distancia recorrida dentro de Galicia representa solo una parte del trayecto total, lo que podría alterar la relación entre distancia y flujo respecto a los viajes estrictamente internos. Ante la falta de datos más precisos, se introdujeron factores de corrección razonables a partir de suposiciones simplificadas: se consideró que, de media, los trayectos hacia conexiones exteriores recorren aproximadamente un 80 % de su distancia dentro de Galicia, y los que se dirigen a aeropuertos un 60 %. A partir del cálculo inicial de F_{ij}, se ajustaron las poblaciones *sustitutivas* de estos nodos para obtener resultados compatibles con la intensidad de tráfico media diaria en ciertas carreteras. [25].

La matriz final **F** se utilizará como demanda exógena en los modelos de localización desarrollados en el capítulo 5.3.

5.3. Aplicación de modelos

Como describimos anteriormente los modelos, es más interesante aplicar a este ejemplo modelos basados en flujo, usando el flujo estimado en el apartado anterior. Aún así, podemos aplicar modelos basados en nodos usando la población de cada nodo, con el fin de ver en práctica estos modelos y comparar los resultados con los modelos basados en flujo. Un aspecto importante a tener en cuenta al trabajar con los modelos basados en nodos, es que nuestros nodos de conexiones externas OD_C no representan demanda estática; por lo tanto, no tiene sentido tenerlos en cuenta para estos modelos.

5.3.1. Modelos basados en nodos

Set-covering

Si aplicamos el modelo de *set-covering* 3.1.1 podemos ver cuántos puntos de recarga necesitamos según la distancia de cobertura escogida. Como la extensión de Galicia no es muy grande para distancias de cobertura que se acerquen a la autonomía actual de los coches eléctricos, esperamos necesitar pocos nodos de demanda con instalación. Si, en cambio, seleccionamos una distancia de cobertura menor para asegurar una cobertura más completa que ayude más a dar servicio diario a la población, esperamos necesitar más nodos con instalaciones. Así, si probamos con las distancias de cobertura $D_c = \{15,\ 50,\ 100,\ 150\}$ (km) obtenemos que necesitamos, respectivamente, 40, 10, 4 y 2 puntos de recarga.

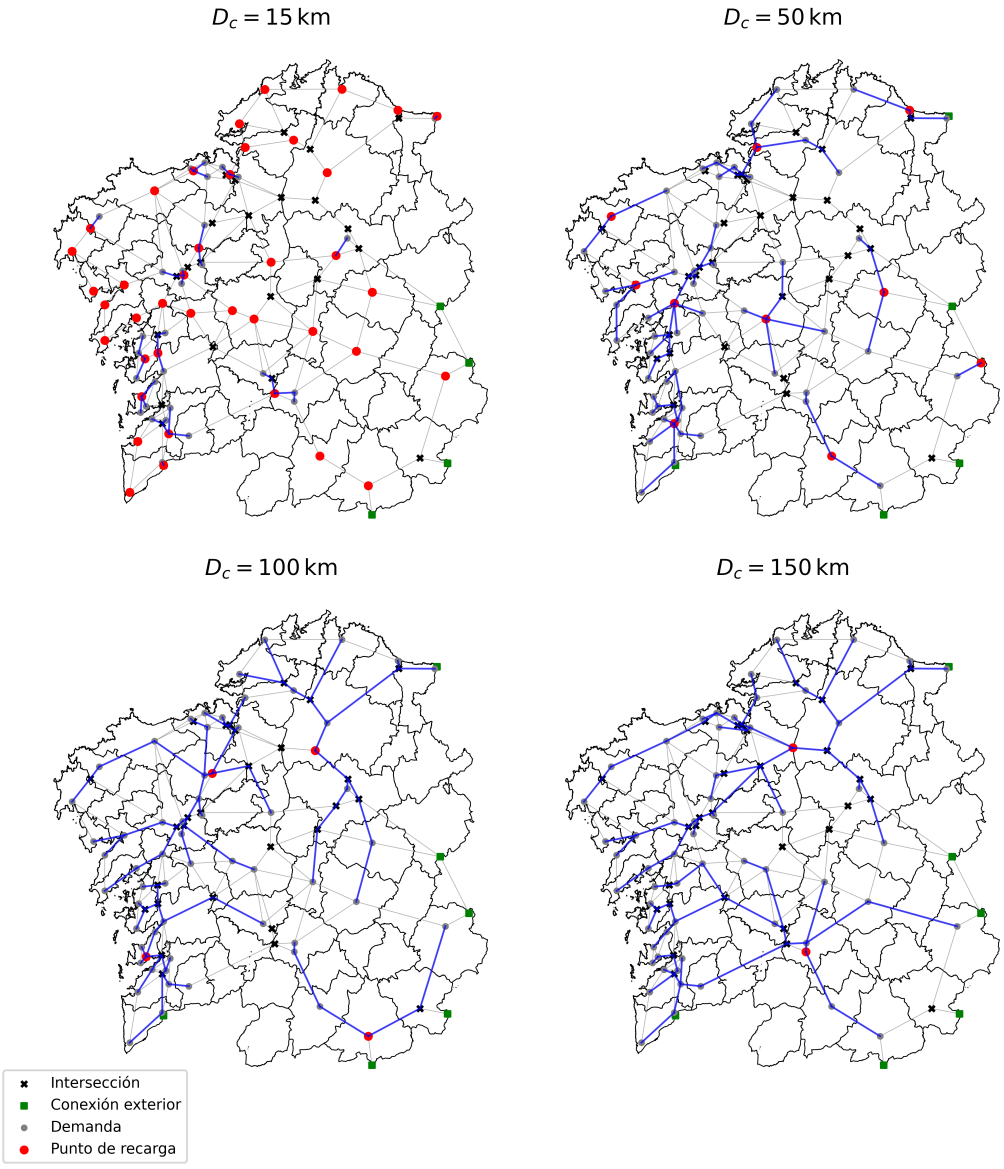

Figura 5.4: Solución set-covering

El código usado para este modelo se puede ver en .2.1.

Max-covering

Con el modelo de max-covering 3.1.2 realizamos un poco el proceso inverso al anterior, fijamos los puntos de recarga y cubrimos el máximo de demanda posible, teniendo en cuenta la demanda (población) de cada nodo. Obtenemos, por ejemplo, que para una distancia de cobertura de 50 km y con 5 puntos de recarga cubrimos el 94.8 % de la demanda.
Otra forma de aplicar este modelo pensando más en eficiencia de cada cargador es ir aumentando progresivamente el número de puntos de recarga hasta que la mejora de demanda no compense la instalación de un nuevo punto de recarga. Por ejemplo, si realizamos esto para una distancia de cobertura de 15 km, obtenemos que con 8 puntos de recarga cubrimos el 81.4 % de la demanda y si instalásemos 9 puntos de recarga *solo* cubriríamos un 2.5 % más de la demanda, hasta un 83.9 %.
Estamos tomando distancias de cobertura relativamente pequeñas e interesantes para el ejemplo de Galicia, ya que para distancias que se acerquen a la autonomía real de los vehículos eléctricos actuales no es muy interesante la aplicación de este modelo a este ejemplo.

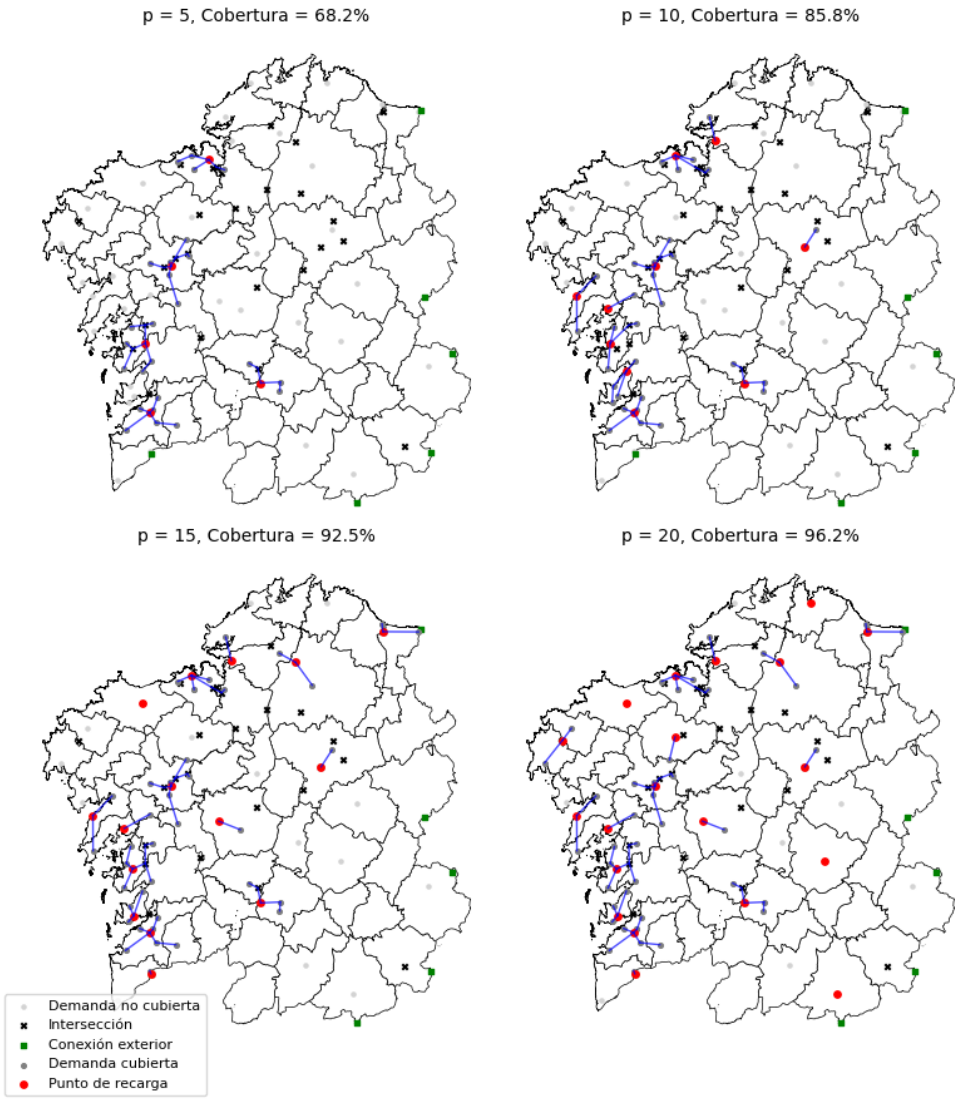

Figura 5.5: Soluciones max-covering con $d_c = 20\ km$

p-mediana

Con los modelos de p-mediana y p-centro no tenemos que fijar una distancia de cobertura, lo cual tiene más sentido en nuestro caso, ya que, aunque nos interesa reducir la distancia, no tenemos una cifra que podamos usar como referencia. La preferencia entre uno y otro se da en si nos importa más reducir la distancia media o el peor caso.

Con la p-mediana 3.1.3 reducimos la distancia media y esperamos obtener más estaciones de carga localizadas en poblaciones grandes como Vigo, A Coruña o Santiago. Como en el caso anterior, podemos orientar el modelo hacia la eficiencia para ver cómo se reduce la distancia media según aumentamos el número de cargadores.

La función para resolver p-mediana se puede ver en.2.3

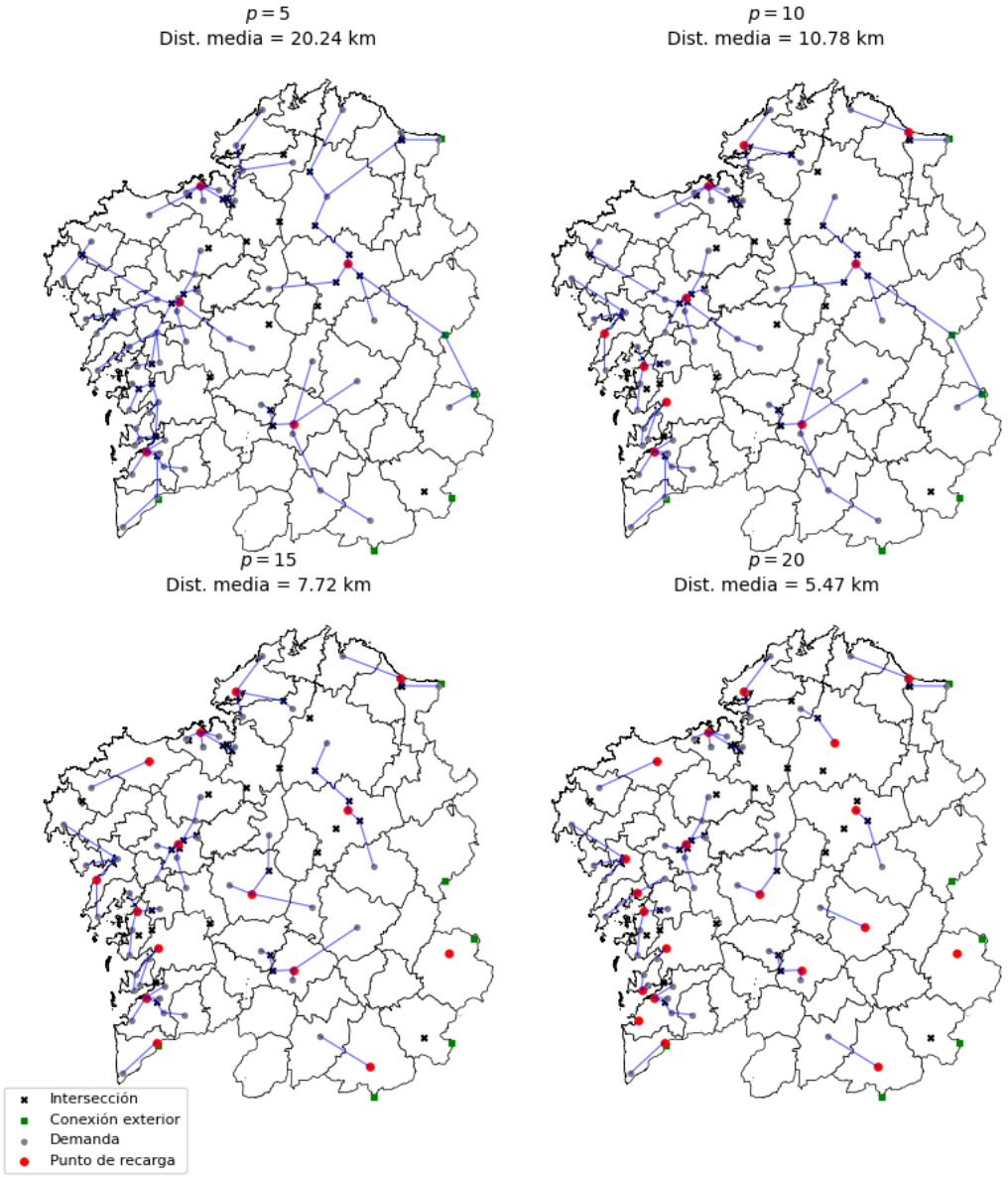

Figura 5.6: Soluciones p-mediana

p-centro

Con p-centro 3.1.4, el modelo prioriza no aislar nodos de demanda, así que esperamos más puntos de recarga distribuidos y menos concentrados; por lo tanto, más puntos de recarga en las provincias de Lugo y Ourense.

Adaptando el código .2.4 obtenemos las soluciones de p-centro.

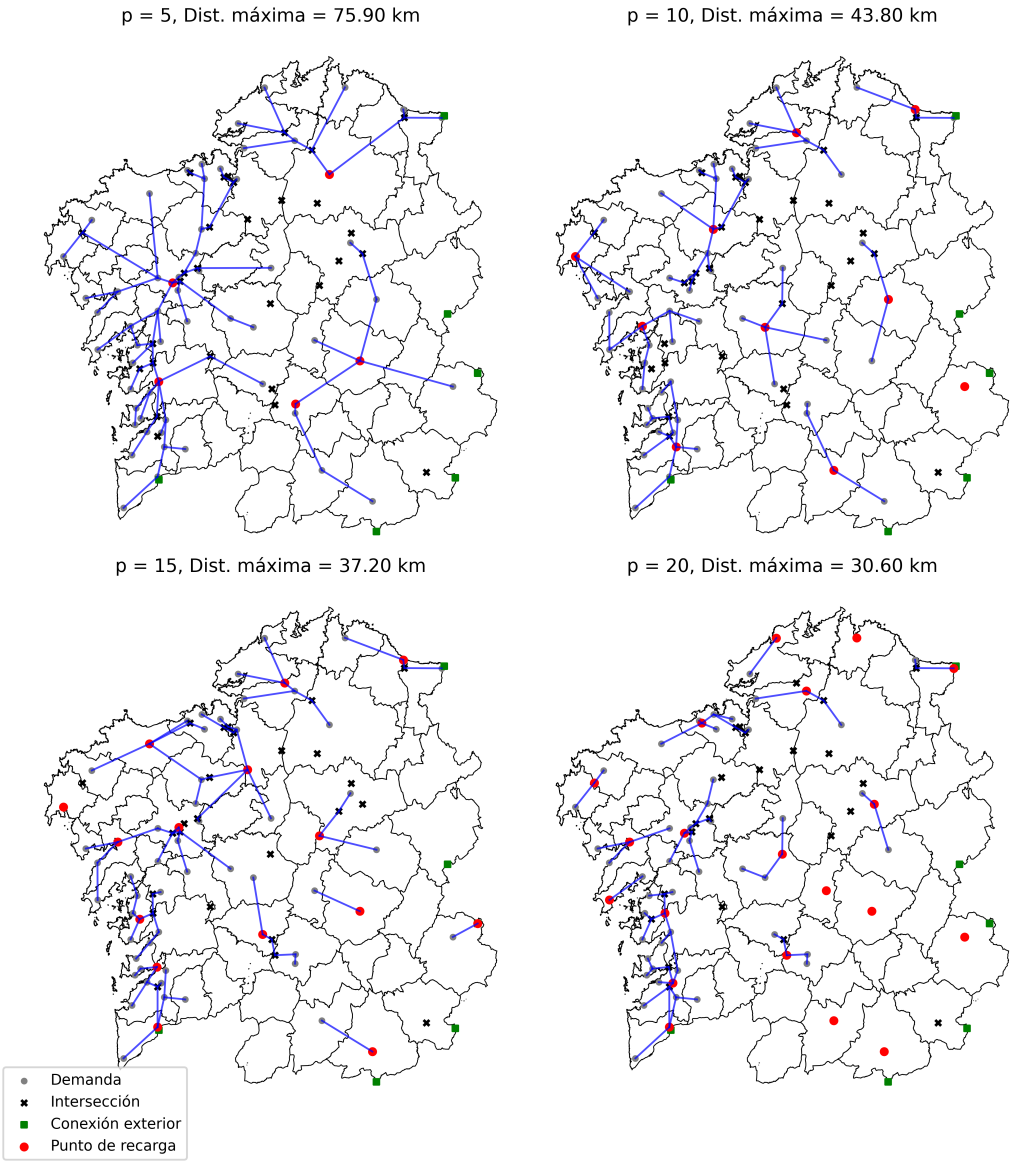

Figura 5.7: Soluciones p-centro

5.3.2. Comparación modelos basados en nodos

Podemos ahora comparar estos modelos con las diferentes métricas que optimizan para ver como se comportan en las otras métricas y cuanto mejores son unas u otras.

Modelo	**Parámetros**	**Nº Inst.**	**Cobertura (%)**		**Dist. media**	**Dist. máx.**
			15 km	30 km		
Set-covering	$d_c = 15$	40	100.0	100.0	6.00	14.20
Set-covering	$d_c = 30$	21	66.9	100	11.97	29.00
Max-covering	$p = 15, d_c = 15$	15	86.3	89.8	15.60	120.00
Max-covering	$p = 15, d_c = 30$	15	62.0	97.5	14.60	48.40
Max-covering	$p = 30, d_c = 15$	30	96.1	98.2	7.85	43.80
Max-covering	$p = 30, d_c = 30$	23*	62.2	100	13.24	29.10
p-mediana	$p = 15$	15	77.9	93.8	7.72	56.30
p-mediana	$p = 30$	30	92.2	98.4	2.89	35.30
p-centro	$p = 15$	15	27.5	74.3	21.20	37.20
p-centro	$p = 30$	30	86.3	100.0	8.00	18.90

Cuadro 5.1: Resumen comparativo de modelos con evaluación de cobertura a dos radios 15 km y 30 km

Así, en la tabla observamos qué modelos se comportan mejor según el criterio que optimizan. El modelo de *set-covering* no es directamente comparable con el resto, ya que en su caso el número de puntos de recarga es la variable a minimizar, mientras que en los demás modelos es un parámetro fijado.

En el caso de *max-covering*, vemos que efectivamente maximiza la cobertura para el número de instalaciones y distancia de cobertura dadas. Sin embargo, si evaluamos el resultado con otro radio, *p-mediana* o *p-centro* logran una mejor cobertura. Además, observamos que con una distancia de cobertura de 30 km, *max-covering* requiere 23 puntos de carga para alcanzar el 100 % de cobertura, mientras que *set-covering* logra ese mismo nivel con solo 21 instalaciones, lo que indica que *max-covering* no obtiene una solución óptima en ese aspecto.

Los modelos más relevantes para nuestro problema —en el que no nos interesa fijar una distancia de cobertura concreta— son *p-mediana* y *p-centro*, según si priorizamos reducir la distancia media o la distancia máxima, respectivamente.

Si se desea un compromiso entre ambos criterios, es posible aplicar un modelo de *p-mediana acotado*, en el que se impone una restricción sobre la distancia máxima permitida y se minimiza la distancia media bajo dicha condición.

5.3.3. Modelos basados en flujo

FCLM

EL modelo de captura de flujo 3.2.1 trata básicamente de identificar las localizaciones por las que pasan más flujo sin tener en cuenta la distancia de los viajes. Así, si resolvemos para 1 estación,

básicamente estamos identificando el nodo por el que pasa más flujo, que en este caso es el asociado a Pontevedra, que cubre un 25 % del flujo total por sí solo.
Si resolvemos para 5 estaciones ya cubrimos un 75 % del flujo y estas se reparten en torno a las ciudades principales (A Coruña, Santiago, Ourense y Pontevedra) con una estación en la conexión de la A-6 en Lugo. Al aumentar el número de estaciones podemos ver que cada vez aumenta menos la proporción de flujo cubierto según nos aproximamos al total y las instalaciones se concentran más en el eje atlántico, como es de esperar por la mayor población.
El código usado puede verse en .2.5.

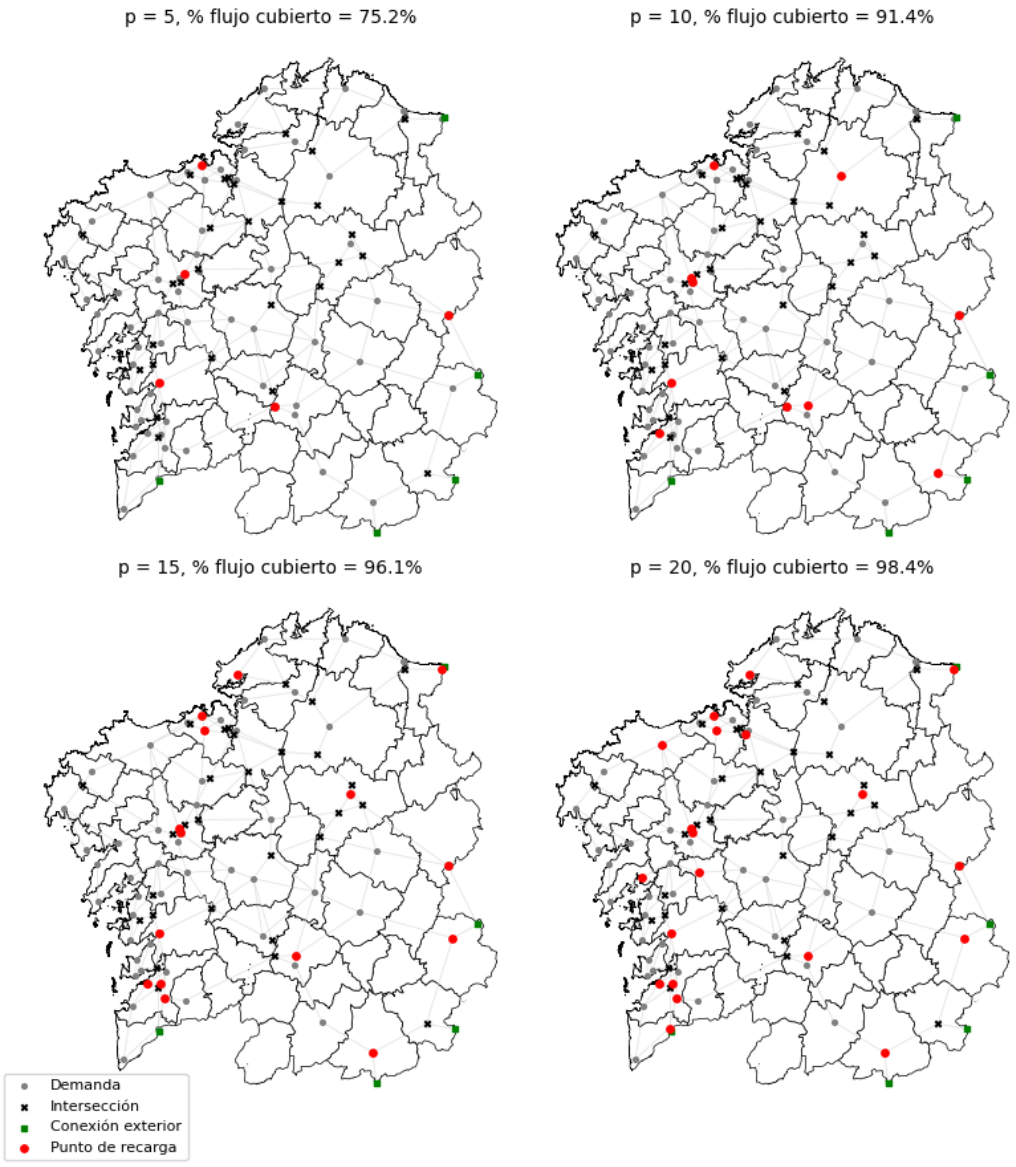

Figura 5.8: Soluciones FCLM

FRLM

Para aplicar el modelo de recarga de flujo 3.2.2, donde ya tenemos en cuenta las distancias y la autonomía, necesitamos primero generar las combinaciones de nodos que cubren nuestros viajes.. Antes de esto necesitamos fijar la autonomía con la que vamos a trabajar,podemos tomar $R = 150\ km$ que es una generosa cota inferior para la autonomía de la mayor parte de coches eléctricos actuales. Con esta autonomía procedemos a generar las combinaciones, como son bastantes viajes las combinaciones de estaciones que permiten cubrir cada viaje son bastantes y lleva bastante tiempo generarlas, en total para 2209 viajes se generaron 6180 combinaciones en un tiempo de 284 segundos.
Ahora con las combinaciones generadas continuamos con la resolución del modelo; si utilizamos el solver de Gurobi como anteriormente con el código .2.6, veremos que, dada la dimensión del modelo, el tiempo requerido aumenta considerablemente. Si probamos para un número pequeño de p, veremos que aumenta exponencialmente, necesitando media hora para el problema con 10 estaciones 5.2.

p	Estaciones seleccionadas	Viajes cubiertos	% Flujo cubierto	Tiempo (s)
2	{6, 53}	369	18.24	11.66
4	{1, 6, 109, 116}	701	30.44	121.70
6	{6, 18, 31, 109, 112, 117}	905	44.2	308.46
8	{2, 6, 18, 31, 104, 110, 112, 117}	1006	52.8	1241.12
10	{2, 6, 18, 31, 103, 104, 110, 112, 117, 129}	1236	61	1823.26

Cuadro 5.2: Resultados del FRLM con solver Gurobi.

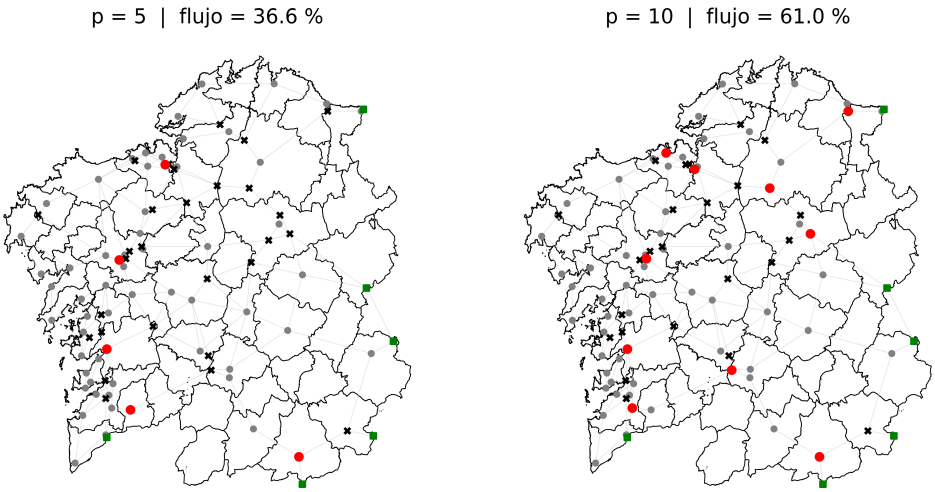

Figura 5.9: Soluciones FRLM con solver

Para evitar estos altos tiempos de computación podemos recurrir a una heurística, por ejemplo en

[28] recurren a un algoritmo genético combinado con una heurística de búsqueda profunda para un problema de FRLM, aunque algo más complejo que el nuestro.
Siguiendo su ejemplo, podemos aplicar entonces un algoritmo genético para resolver nuestro caso. El algoritmo genético usado consta de una población formada por soluciones generadas aleatoriamente, aunque con pesos para dar prioridad a nodos que funcionan bien por sí solos. Después, esta población se modifica con cruces y mutaciones, dando prioridad a las soluciones que cubren mayor parte del flujo; este proceso se repite en un número de generaciones, en nuestro caso 200, y nos quedamos con la mejor solución obtenida, el código completo puede verse en .2.7.
Si comparamos las soluciones obtenidas para valores de p donde ya teníamos la solución óptima con el solver, podemos ver que el algoritmo genético se aproxima bastante, aunque no suele encontrar la solución óptima. Por ejemplo, para el caso $P = 10$ tenemos que la solución encontrada por el algoritmo genético es $[6, 18, 44, 59, 103, 109, 112, 117, 128, 129]$, que difiere de la del solver 5.2, pero el porcentaje del flujo cubierto es 60,62 %, muy próximo al 61 % de la solución óptima; y consumiendo mucho menos tiempo (187.3s).

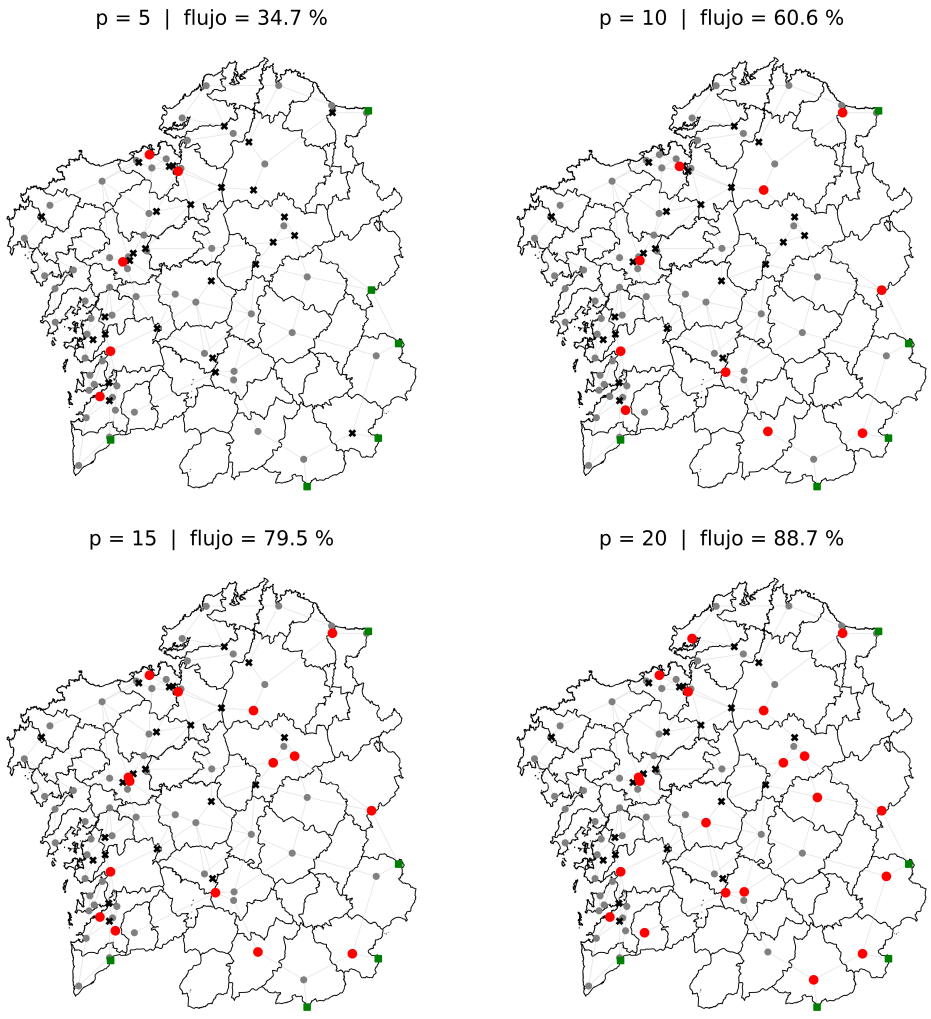

Figura 5.10: Soluciones FRLM con algorítmo genético

Si comparamos la localización de las estaciones con el modelo de captura de flujo, podemos ver que la principal diferencia es que las estaciones de carga, exceptuando para el caso de $P = 5$, no se concentran tanto en el eje atlántico y se reparten más por Galicia.

Aunque sería posible estudiar la aplicación de más modelos al ejemplo, como *Arc-covering* o *Path-segment*, o añadir más variables de decisión como capacidad, desvío, etc..., la complejidad añadida implicaría el desarrollo de heurísticas adicionales y tareas de calibración que exceden el alcance de este trabajo. Los diferentes modelos implementados ya permiten comparar diversas estrategias y ver cómo se comportan en nuestro ejemplo concreto.

Sería posible comparar las soluciones obtenidas con la situación actual de la distribución de cargadores en Galicia, pero actualmente el número de cargadores (cerca de 400 cargadores rápidos, aunque muchos no 100 % operativos [29]) supera con creces a los números con los que trabajamos en los ejemplos, tendríamos que incluir variables como capacidad y restricciones de servicio de los cargadores para poder hacer una comparación honesta con la capacidad de carga real.

Capítulo 6

Investigaciones futuras y conclusiones

Investigaciones futuras

Uno de los retos más señalados por la literatura es la **estimación fiable de los parámetros de entrada**. La escasez de datos reales sobre recarga y consumo energético —debida a la limitada penetración actual del VE— impide generar pronósticos que contemplen simultáneamente *cuánto* se recarga, *cuándo* y *dónde*. Los registros de taxis o flotas de vehículos de combustión, aunque accesibles, no reflejan el patrón de viaje ni el comportamiento de recarga propios del VE; además, la hipótesis habitual de que el consumo es proporcional a la distancia ignora efectos de orografía, tráfico, clima o estilo de conducción. Se necesitarán campañas de medición específicas —trials a pequeña escala y, a medida que aumente el parque, bases de datos de cargadores inteligentes— para estimar distribuciones de demanda y de consumo más realistas y alimentar así los modelos estocásticos de forma rigurosa.

El segundo frente de investigación deriva del **crecimiento previsto de la demanda y de los picos de recarga**. Con la prohibición progresiva del vehículo de combustión (objetivo 100 % cero emisiones entre 2030 y 2040 en varios países), la congestión simultánea de red y estación cobrará protagonismo. Los trabajos futuros deberán integrar, en un único marco, la decisión de ubicación y capacidad de las estaciones con soluciones para casar oferta y demanda: inversiones de refuerzo en la red, ubicación y dimensionamiento de sistemas de almacenamiento, y estrategias de precios variables que modelen la elasticidad de la demanda. La combinación de decisiones estratégicas (long-term) con decisiones operativas (programación y tarificación horaria) exigirá modelos estocásticos de gran tamaño y nuevos métodos de resolución. Dado que la mayoría de trabajos actuales emplean solvers comerciales o heurísticas, se abre la necesidad de desarrollar *métodos exactos especializados* y descomposiciones que proporcionen garantías de optimalidad en tiempos de cómputo aceptables.

Más allá del avance científico, la transición hacia una movilidad 100 % eléctrica necesita de un impulso decidido al uso del vehículo eléctrico: facilitar su adquisición con incentivos económicos estables y simplificados; divulgar de forma rigurosa sus beneficios ambientales y de coste total de propiedad; reforzar la infraestructura de recarga en los "puntos ciegos" detectados por los estudios de accesibilidad; y coordinar normativa, fiscalidad y planificación energética para que la expansión de la red no dependa únicamente de la iniciativa privada.

Conclusiones

En este trabajo hemos dado un contexto a la situación del coche eléctrico y su infraestructura de carga para introducir el problema de la localización de estaciones de carga. Después vimos las formulaciones básicas de modelos de localización de cargadores para vehículos eléctricos, diferenciando entre modelos basados en nodos y modelos basados en flujo, según cómo se modela la demanda; viendo las principales características de cada tipo. En el aspecto de la formulación de este problema también repasamos qué otras variables de decisión pueden incluir versiones más completas de esos modelos, aparte de qué otras particularidades se pueden modelar. Para complementar la formulación de estos modelos, revisamos los principales métodos de solución, tanto exactos como aproximados, siendo estos principalmente el uso de solvers comerciales con soluciones exactas de la mano del algoritmo de *branch-and-bound* y heurísticas y meta-heurísticas para casos con mayor complejidad computacional. Esto todo lo aplicamos a un caso práctico basado en la región de Galicia para comprobar cómo se comportan en un caso real y compararlos entre sí, aunque para poder comparar los resultados con la situación actual de la infraestructura de carga en Galicia requeriríamos modelos más desarrollados. Finalmente, señalamos los campos donde se esperan avances próximos en la investigación futura.

Bibliografía

[1] IEA, International Energy Agency, Greenhouse Gas Emissions

[2] IEA, International Energy Agency, Renewable Energy Progress Tracker

[3] IEA, International Energy Agency, Global EV Data Explorer

[4] IEA, International Energy Agency, Global EV Data Explorer, *EV sales share, cars, World, 2010–2024*

[5] IEA, International Energy Agency, Global EV Outlook 2025, Electric Vehicle Charging

[6] European Court of Auditors. *Special Report 05/2021: Infrastructure for charging electric vehicles.* Luxembourg, 2021.

[7] European Union. Regulation (EU) 2023/1804 of the European Parliament and of the Council of 13 September 2023 on the deployment of alternative fuels infrastructure (AFIR).

[8] C. Toregas, C. ReVelle, R. Swain, and L. Bergman, The Location of Emergency Service Facilities, *Operations Research* XIX, No,5 (October, 1971)

[9] Church, R. L., & ReVelle, C. (1974). The maximal covering location problem. *Papers in Regional Science* 32(1), 101–118.

[10] Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3), 450–459.

[11] Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13(3), 462–475.

[12] M.J. Hodgson, A flow capturing location-allocation model, *Geogr. Anal.* 22 (1990) 270–279.

[13] M. Kuby, S. Lim, The flow-refueling location problem for alternative-fuel vehicles, *Soc.-Econ. Plann. Sci.* 39 (2005) 125–145.

[14] I. Capar, M. Kuby, V.J. Leon, Y.J. Tsai, An arc cover–path-cover formulation and strategic analysis of alternative-fuel station locations, *European J. Oper. Res.* 227 (2013) 142–151.

[15] S.A. MirHassani, R. Ebrazi, A flexible reformulation of the refueling station location problem, *Transp. Sci. 47* (2013) 617–628.

[16] H. Sadeghi-Barzani, A. Rajabi-Ghahnavieh, H. Kazemi-Karegar, Optimal fast charging station placing and sizing, *Applied Energy* 125 (2014) 289–299.

[17] J.-G. Kim, M. Kuby, The deviation-flow refueling location model for optimizing a network of refueling stations, *International Journal of Hydrogen Energy*, 37(6) (2012) 5406–5420.

[18] Huang, Y., Li, S. Qian, Z.S. Optimal Deployment of Alternative Fueling Stations on Transportation Networks Considering Deviation Paths. *Netw Spat Econ* 15, 183–204 (2015).

[19] M. Hosseini, S.A. MirHassani, F. Hooshmand, Deviation-flow refueling location problem with capacitated facilities: Model and algorithm, *Transportation Research Part D: Transport and Environment*, 54, 269-281 (2017).

[20] J. Yang, H. Sun, Battery swap station location-routing problem with capacitated electric vehicles, *Computers & Operations Research* 55 (2015) 217–232.

[21] Y.-W. Wang, C.-C. Lin, Locating multiple types of charging stations for battery-powered electric vehicle transport, *Transportation Research Part E: Logistics and Transportation Review*, 58 (2013) 76–87.

[22] Mark S. Daskin, Network and Discrete Location: Models, Algorithms, and Applications (1995)

[23] Mouna Kchaou-Boujelben, Charging station location problem: A comprehensive review on models and solution approaches, *Transportation Research Part C: Emerging Technologies*, 132, (2021)

[24] Consulta Enero 2025, Población de 2022

[25] Mapa de trafico 2022 Ministerio de Transportes

[26] A Primer on Gravity Models in Urban Analysis

[27] spanishoddata: Get Spanish Origin-Destination Data

[28] Y. Wang, J. Shi, R. Wang, Z. Liu, L. Wang, Siting and sizing of fast charging stations in highway network with budget constraint, *Appl. Energy* 228 (2018b) 1255-1271

[29] Electromaps, Julio 2025

Anexo

.1. Listado completo de nodos

A continuación se presenta el listado completo de nodos que conforma la red de Galicia en nuestro ejemplo, cada nodo tiene el tipo de nodo: OD (origen-destino), OD_C (origen-destino con conexión exterior), OD_A (origen-destino de aeropuerto) o J (intersección). Los nodos de origen-destino tienen su población o población equivalente en miles, además para nodos que junten varios municipios, los municipios en la unión se pueden ver en *Municipio extra*. Finalmente se proporcionan las coordenadas de cada nodo.

Nodos de Galicia

id	Nombre	Tipo	Población	Municipio extra	lon	lat
1	Vigo	OD	292.0	-	-8.719	42.229
2	A Coruña	OD	244.0	-	-8.408	43.343
3	Ourense	OD	104.0	-	-7.876	42.345
4	Santiago	OD	102.0	-	-8.539	42.873
5	Lugo	OD	97.0	-	-7.565	43.015
6	Pontevedra	OD	100.0	Poio	-8.652	42.438
7	Ferrol	OD	103.0	Narón	-8.203	43.512
8	Vilagarcia	OD	38.0	-	-8.774	42.589
9	Oleiros	OD	54.0	Sada	-8.303	43.324
10	Arteixo	OD	33.0	-	-8.49	43.317
11	Carballo	OD	31.0	-	-8.705	43.221
12	Redondela	OD	29.0	-	-8.612	42.278
13	Ribeira	OD	27.0	-	-8.999	42.571
14	Cangas	OD	27.0	-	-8.787	42.258
15	Culleredo	OD	55.0	Cambre	-8.393	43.282
16	Marín	OD	24.0	-	-8.703	42.393
17	Ponteareas	OD	23.0	-	-8.502	42.157
18	Porriño	OD	20.0	-	-8.621	42.166
19	AEstrada	OD	20.0	-	-8.491	42.689
20	Lalín	OD	20.0	-	-8.116	42.664
21	Moaña	OD	19.0	-	-8.755	42.285
22	Val Minor	OD	45.0	Nigrán G+B	-8.804	42.132
23	Monforte	OD	18.0	-	-7.511	42.524
24	Sanxenxo	OD	18.0	-	-8.812	42.407
25	Tui	OD	31.0	Tomiño	-8.659	42.042
26	Viveiro	OD	15.0	-	-7.596	43.662
27	Noia	OD	14.0	-	-8.885	42.812
28	Carballiño	OD	14.0	-	-8.063	42.428
29	Villalba	OD	14.0	-	-7.684	43.3
30	Cambados	OD	14.0	-	-8.799	42.516
31	Verín	OD	14.0	-	-7.44	41.94
32	Barco	OD	13.0	-	-6.984	42.417
33	Sarria	OD	13.0	-	-7.417	42.78
34	Betanzos	OD	13.0	-	-8.21	43.28
35	Ordes	OD	13.0	-	-8.41	43.072

Nodos de Galicia (continuación)

id	Nombre	Tipo	Población	Nota	lon	lat
36	Bueu	OD	12.0	-	-8.781	42.327
37	Barbadás	OD	11.0	-	-7.879	42.306
38	Boiro	OD	30.0	Rianxo	-8.812	42.669
39	Foz	OD	10.0	-	-7.265	43.57
40	A Garda	OD	10.0	-	-8.852	41.912
41	As Pontes	OD	10.0	-	-7.882	43.441
42	Ribadeo	OD	10.0	-	-7.046	43.535
43	Caldas	OD	10.0	-	-8.642	42.604
44	Xinzo	OD	10.0	-	-7.727	42.069
45	Porto Son	OD	9.0	-	-9.0	42.726
46	Silleda	OD	9.0	-	-8.245	42.701
47	Muros	OD	8.0	-	-9.066	42.785
48	Padrón	OD	8.0	-	-8.658	42.732
49	Chantada	OD	8.0	-	-7.769	42.61
50	Brión	OD	8.0	-	-8.658	42.869
51	Pontedeume	OD	8.0	-	-8.169	43.409
52	Cee	OD	8.0	-	-9.193	42.958
53	Sigüeiro	OD	8.0	Oroso	-8.443	42.972
54	Melide	OD	7.0	-	-8.016	42.911
55	Vimianzo	OD	7.0	-	-9.032	43.11
56	Cedeira	OD	7.0	-	-8.052	43.661
57	Ames	OD	51.0	Teo	-8.545	42.818
58	Mos	OD	15.0	-	-8.639	42.225
59	A6	OD_C	120.0	-	-7.013	42.719
60	N-120	OD_C	54.0	-	-6.843	42.473
61	A-52	OD_C	190.0	-	-6.97	42.037
62	A-75	OD_C	80.0	-	-7.418	41.814
63	A-55	OD_C	35.0	-	-8.651	42.029
64	A-8	OD_C	100.0	-	-7.033	43.544
65	SCQ	OD_A	18.0	-	-8.424	42.899
66	VGO	OD_A	5.0	-	-8.639	42.225
67	LCQ	OD_A	6.0	-	-8.393	43.282
101	AG-64 43	J	-	-	-7.784	43.401
102	AG-64 26	J	-	-	-7.938	43.474
103	A-6 A-8	J	-	-	-7.754	43.18
104	Nadela	J	-	-	-7.497	42.97
105	O Ceao	J	-	-	-7.56	43.056
106	A-54 12	J	-	-	-7.631	42.94
107	A-6 540	J	-	-	-7.956	43.192
108	Curtis	J	-	-	-8.148	43.113
109	A-6 AP-9 1	J	-	-	-8.283	43.289
110	A-6 AP-9 2	J	-	-	-8.228	43.267
111	Rande	J	-	-	-8.663	42.292
112	A-52 AG-53	J	-	-	-7.993	42.341
113	AG-53AG-54	J	-	-	-8.011	42.407
114	AP-9 A-55	J	-	-	-8.659	42.21
115	A-6 AG-55	J	-	-	-8.475	43.307
116	AP-9 AG-56	J	-	-	-8.574	42.849
117	AP-9 AP-53	J	-	-	-8.531	42.855
118	AP-9 AG-41	J	-	-	-8.685	42.516
119	AP-9 110	J	-	-	-8.687	42.596
120	N-541	J	-	-	-8.36	42.543
121	AG-41VG4.2	J	-	-	-8.761	42.491
122	Berdoias	J	-	-	-9.085	43.058
123	Golada	J	-	-	-8.019	42.762
124	N-540N-640	J	-	-	-7.741	42.838
125	SC-20 N	J	-	-	-8.509	42.889
126	A-54 SCQ	J	-	-	-8.433	42.91
127	AP-9 Ordes	J	-	-	-8.364	43.081

Nodos de Galicia (continuación)

id	Nombre	Tipo	Población	Nota	lon	lat
128	La Gudiña	J	-	-	-7.134	42.06
129	A-8 524	J	-	-	-7.259	43.536
130	N-6 AP-9	J	-	-	-8.259	43.292

.2. Código solución de modelos

En esta sección se comparte el código usado para modelar y solucionar nuestro ejemplo con los diferentes modelos:

.2.1. Set-covering:

```
import pandas as pd
from pulp import LpProblem, LpMinimize, LpVariable, lpSum, LpBinary, PULP_CBC_CMD
import geopandas as gpd
import matplotlib.pyplot as plt
import networkx as nx
import os
# === PARÁMETROS ===
RADIOS = [15, 50, 100, 150]  # radios de cobertura en km

# === CARGA DE DATOS ===
matriz_dist = pd.read_csv("matriz_distancias_extendida.csv", index_col=0)
matriz_dist.index = matriz_dist.index.astype(str)
matriz_dist.columns = matriz_dist.columns.astype(str)

# Cargar Nodos_Galicia con tipos
nodos_df = pd.read_csv("Nodos_Galicia.csv")
nodos_df["id"] = nodos_df["id"].astype(str)

# Filtrar nodos de demanda válidos: OD y OD_A
nodos_demanda = sorted(
    nodos_df.loc[nodos_df["Tipo"].isin(["OD", "OD_A"]), "id"].tolist()
)

nodos_candidatos = list(matriz_dist.columns)

resultados = {}  # clave: radio, valor: lista de instalaciones

for radio in RADIOS:

    cobertura = {
        i: [j for j in nodos_candidatos if matriz_dist.loc[i, j] <= radio]
        for i in nodos_demanda
    }

    model = LpProblem(f"Set_Covering_{radio}", LpMinimize)
    y = {j: LpVariable(f"y_{j}", cat=LpBinary) for j in nodos_candidatos}

    # Función objetivo
    model += lpSum(y[j] for j in nodos_candidatos)

    # Restricciones
    for i in nodos_demanda:
        if cobertura[i]:
            model += lpSum(y[j] for j in cobertura[i]) >= 1
        else:
            print(f" Nodo de demanda {i} no cubierto para radio {radio} km")

    # Resolver
    solver = PULP_CBC_CMD(msg=False)
    model.solve(solver)

    instalaciones = [j for j in nodos_candidatos if y[j].value() == 1]
    resultados[radio] = instalaciones
    print(f" {len(instalaciones)} instalaciones necesarias para radio = {radio} km")

# === TABLA FINAL ===
```

```
tabla_resultados = pd.DataFrame({
    "Radio_km": RADIOS,
    "N_Instalaciones": [len(resultados[r]) for r in RADIOS]
})
print("\n Resultados:\n", tabla_resultados)

tabla_resultados.to_csv("set_covering.csv", index=False)
```

.2.2. Max-covering:

```
def resolver_max_covering(p):
    m = gp.Model(f"MaxCover_p{p}")
    x = m.addVars(nodos_candidatos, vtype=GRB.BINARY, name="x")
    z = m.addVars(pesos.keys(), vtype=GRB.BINARY, name="z")

    m.setObjective(gp.quicksum(pesos[i] * z[i] for i in pesos), GRB.MAXIMIZE)
    m.addConstr(gp.quicksum(x[j] for j in nodos_candidatos) == p)
    for i in pesos:
        if cobertura[i]:
            m.addConstr(z[i] <= gp.quicksum(x[j] for j in cobertura[i]))
        else:
            m.addConstr(z[i] == 0)

    m.setParam("OutputFlag", 0)
    m.optimize()

    instalaciones = [j for j in nodos_candidatos if x[j].X > 0.5]
    nodos_cubiertos = [i for i in pesos if z[i].X > 0.5]
    demanda_cubierta = sum(pesos[i] for i in nodos_cubiertos)
    porcentaje_cubierta = 100 * demanda_cubierta / demanda_total
    return instalaciones, nodos_cubiertos, porcentaje_cubierta
```

.2.3. p-mediana

```
def resolver_p_median(p):
    nodos_candidatos = list(matriz_dist.columns)
    nodos_demanda_df = nodos_df[nodos_df["Tipo"].isin(["OD", "OD_A"])].copy()
    pesos = nodos_demanda_df.set_index("id")["Población"].astype(float).to_dict()

    m = gp.Model("p-median")
    x = m.addVars(nodos_candidatos, vtype=GRB.BINARY, name="x")
    y = m.addVars(pesos.keys(), nodos_candidatos, vtype=GRB.BINARY, name="y")

    m.setObjective(gp.quicksum(pesos[i] * matriz_dist.loc[i, j] * y[i, j]
                               for i in pesos for j in nodos_candidatos), GRB.MINIMIZE)

    for i in pesos:
        m.addConstr(gp.quicksum(y[i, j] for j in nodos_candidatos) == 1)
        for j in nodos_candidatos:
            m.addConstr(y[i, j] <= x[j])

    m.addConstr(gp.quicksum(x[j] for j in nodos_candidatos) == p)
    m.setParam("OutputFlag", 0)
    m.optimize()

    instalaciones = [j for j in nodos_candidatos if x[j].x > 0.5]
    asignaciones = {i: [j for j in nodos_candidatos if y[i, j].x > 0.5][0] for i in pesos}

    total_pesos = sum(pesos.values())
    costo_total = sum(pesos[i] * matriz_dist.loc[i, asignaciones[i]] for i in pesos)
    costo_medio = costo_total / total_pesos

    return instalaciones, asignaciones, costo_medio
```

.2.4. p-centro

```
def resolver_p_center(p):
    nodos_demanda = nodos_df[nodos_df["Tipo"].isin(["OD", "OD_A"])]
    nodos_candidatos = list(matriz_dist.columns)
    demanda_ids = nodos_demanda["id"].tolist()

    m = gp.Model("p-center")
    x = m.addVars(nodos_candidatos, vtype=GRB.BINARY, name="x")
    y = m.addVars(demanda_ids, nodos_candidatos, vtype=GRB.BINARY, name="y")
    w = m.addVar(vtype=GRB.CONTINUOUS, name="w")
```

```
    m.setObjective(w, GRB.MINIMIZE)

    for i in demanda_ids:
        m.addConstr(gp.quicksum(y[i, j] for j in nodos_candidatos) == 1)
        for j in nodos_candidatos:
            m.addConstr(y[i, j] <= x[j])
            m.addConstr(w >= matriz_dist.loc[i, j] * y[i, j])

    m.addConstr(gp.quicksum(x[j] for j in nodos_candidatos) == p)

    m.setParam("OutputFlag", 0)
    m.optimize()

    instalaciones = [j for j in nodos_candidatos if x[j].X > 0.5]
    asignaciones = {i: [j for j in nodos_candidatos if y[i, j].X > 0.5][0] for i in demanda_ids}
    dist_maxima = max(matriz_dist.loc[i, asignaciones[i]] for i in demanda_ids)

    return instalaciones, asignaciones, dist_maxima
```

.2.5. FCLM

```
import pandas as pd
import pickle
from gurobipy import Model, GRB, quicksum

# === CARGA DE DATOS ===
flujo_df = pd.read_csv("matriz_trafico.csv", index_col=0)
with open("caminos_mas_cortos.pkl", "rb") as f:
    caminos = pickle.load(f)

# Asegurar tipos correctos
flujo_df.index = flujo_df.index.astype(int)
flujo_df.columns = flujo_df.columns.astype(int)
caminos = {
    int(i): {int(j): [int(n) for n in path] for j, path in destinos.items()}
    for i, destinos in caminos.items()
}

# === CONJUNTOS DEL MODELO ===
# Nodos válidos: los de dos dígitos
nodos_validos = sorted([n for n in flujo_df.index if n < 100])

# Viajes con flujo positivo
viajes = [(i, j) for i in nodos_validos for j in nodos_validos if i < j and flujo_df.loc[i, j] > 0]
f_q = {(i, j): flujo_df.loc[i, j] for (i, j) in viajes}

# N_q: nodos en el camino (incluidos intermedios)
N_q = {}
nodos_en_caminos = set()
errores = 0
for (i, j) in viajes:
    try:
        camino = caminos[i][j]
        N_q[(i, j)] = set(camino)
        nodos_en_caminos.update(camino)
    except KeyError:
        errores += 1
        print(f" Camino no disponible para viaje {i}-{j}")

print(f" Viajes ignorados por falta de camino: {errores}")

# Unión de nodos válidos e intermedios
nodos = sorted(set(nodos_validos).union(nodos_en_caminos))

# === PARÁMETRO: número de estaciones ===
p = 5  # Puedes cambiarlo según necesites

# === CONSTRUCCIÓN DEL MODELO GUROBI ===
m = Model("FCLM_caminos")

# Variables: x_k = estación en nodo k
x = m.addVars(nodos, vtype=GRB.BINARY, name="x")

# Variables: y_q = si el viaje (i,j) está cubierto
y = m.addVars(viajes, vtype=GRB.BINARY, name="y")

# Objetivo: maximizar flujo cubierto
```

```
m.setObjective(quicksum(f_q[q] * y[q] for q in viajes), GRB.MAXIMIZE)

# Restricciones de cobertura de viajes
for q in viajes:
    if q in N_q:
        m.addConstr(quicksum(x[k] for k in N_q[q]) >= y[q], name=f"cover_{q}")
    else:
        m.addConstr(y[q] == 0, name=f"nocover_{q}")

# Restricción: número exacto de estaciones
m.addConstr(quicksum(x[k] for k in nodos) == p, name="n_estaciones")

# === RESOLVER ===
m.optimize()

# === SALIDA ===
estaciones = [k for k in nodos if x[k].X > 0.5]
viajes_cubiertos = [q for q in viajes if y[q].X > 0.5]

# === CÁLCULO DEL PORCENTAJE DE FLUJO CUBIERTO ===
flujo_total = sum(f_q.values())
porcentaje_flujo_cubierto = (m.objVal / flujo_total) * 100 if flujo_total > 0 else 0

# === SALIDA ===
print(f"\n Estaciones colocadas en: {estaciones}")
print(f" Viajes cubiertos: {len(viajes_cubiertos)} de {len(viajes)}")
print(f" Flujo total cubierto: {m.objVal:.2f}")
print(f" Porcentaje del flujo cubierto: {porcentaje_flujo_cubierto:.2f}%")
```

.2.6. FRLM

```
import time
import pandas as pd
from gurobipy import Model, GRB, quicksum

#  parámetros de usuario
P_ESTACIONES = 10         # número de estaciones a ubicar
PRINT_STEP   = 0.50       # progreso (restricciones) cada 50 %
PRINT_EVERY  = 60         # segundos entre mensajes del callback

#  cronómetro global
t0 = time.perf_counter()

#  leer datos (flujo, A, B)
flujo = pd.read_csv("matriz_trafico.csv", index_col=0).astype(float)
flujo.index, flujo.columns = flujo.index.astype(int), flujo.columns.astype(int)

B = pd.read_csv("b_matrix.csv", index_col=0)
A = pd.read_csv("a_matrix.csv", index_col=0)
A.index   = A.index.astype(int)
A.columns = A.columns.astype(int)
B.columns = B.columns.astype(int)

t_read = time.perf_counter()
print(f"  Lectura datos    : {t_read - t0: .2f} s")

#  conjuntos y parámetros
f_q = (
    flujo.unstack()
         .dropna()
         .loc[lambda s: s > 0]
)
f_q.index = f_q.index.map(lambda t: f"{int(t[0])}-{int(t[1])}")
f_q = f_q.to_dict()

Q = B.index.tolist()        # viajes i-j con flujo >0
H = B.columns.tolist()      # hiper-caminos
N = A.columns.tolist()      # nodos candidatos

t_sets = time.perf_counter()
print(f"  Crear conjuntos  : {t_sets - t_read: .2f} s")

#  modelo Gurobi
m = Model("FRLM")
m.setParam("OutputFlag", 0)        # oculta log por defecto
```

```
# variables
x = m.addVars(N, vtype=GRB.BINARY, name="x")   # estación en nodo k
v = m.addVars(H, vtype=GRB.BINARY, name="v")   # hiper-camino activo
y = m.addVars(Q, vtype=GRB.BINARY, name="y")   # viaje cubierto

# objetivo
m.setObjective(quicksum(f_q[q] * y[q] for q in Q), GRB.MAXIMIZE)

# - (a) cada viaje cubierto requiere 1 hiper-camino activo
step = max(1, int(len(Q) * PRINT_STEP))
for i, q in enumerate(Q, start=1):
    hiper_q = B.columns[B.loc[q] == 1]
    m.addConstr(quicksum(v[h] for h in hiper_q) >= y[q], name=f"cover_{q}")
    if not i % step:
        print(f"   progreso {(i / len(Q)) * 100:4.0f} %")

# - (b) un hiper-camino solo está activo si todos sus nodos tienen estación
A_coo = A.stack().loc[lambda s: s == 1]     # formato ‘‘tripleta’’ (h,k) con 1
step2 = max(1, int(len(A_coo) * PRINT_STEP))
for i, (h, k) in enumerate(A_coo.index, start=1):
    m.addConstr(x[k] >= v[h], name=f"link_{h}_{k}")
    if not i % step2:
        print(f"   progreso {(i / len(A_coo)) * 100:4.0f} %")

# - (c) número fijo de estaciones
m.addConstr(quicksum(x[k] for k in N) == P_ESTACIONES, name="p_stations")

t_build = time.perf_counter()
print(f"  Modelo construido: {t_build - t_sets: .2f} s  "
      f"(#restricciones = {m.numConstrs})")

#  callback de progreso
def progreso(model, where):
    """Imprime la mejor solución y el gap cada PRINT_EVERY segundos."""
    if where != GRB.Callback.MIPSOL:
        return

    runtime = model.cbGet(GRB.Callback.RUNTIME)
    if runtime - progreso.last_print < PRINT_EVERY:
        return                      # aún no toca

    obj_best  = model.cbGet(GRB.Callback.MIPSOL_OBJ)
    obj_bound = model.cbGet(GRB.Callback.MIPSOL_OBJBND)
    gap = (abs(obj_bound - obj_best) / abs(obj_best)) if obj_best else float("inf")

    # solución incumbente
    xsol = model.cbGetSolution(model._x)
    ysol = model.cbGetSolution(model._y)

    estaciones = [k for k, val in xsol.items() if val > 0.5]
    viajes_cov = sum(1 for val in ysol.values() if val > 0.5)

    print(f"[{runtime:7.1f}s]  OBJ = {obj_best:,.0f}   GAP = {gap * 100:5.2f} %")
    print(f"            estaciones      : {sorted(estaciones)}")
    print(f"            viajes cubiertos : {viajes_cov} / {len(Q)}\n")

    progreso.last_print = runtime

progreso.last_print = 0.0      # variable ‘‘estática’’ en la función
m._x = x                       # referencias accesibles dentro del callback
m._y = y

#  optimizar
m.optimize(progreso)

t_opt = time.perf_counter()
print(f"  Optimización     : {t_opt - t_build: .2f} s  (status {m.Status})")

#  resultados finales
estaciones     = [k for k in N if x[k].X > 0.5]
viajes_cubiertos = int(sum(round(y[q].X) for q in Q))
flujo_total    = sum(f_q.values())
pct            = 100 * m.objVal / flujo_total if flujo_total else 0.0

print("\n=== Resultados FRLM ===")
print(f"Nº de estaciones        : {len(estaciones)} de {P_ESTACIONES}")
```

```
print(f"Nodos con estación      : {sorted(estaciones)}")
print(f"Viajes cubiertos        : {viajes_cubiertos} / {len(Q)}")
print(f"Flujo cubierto (objetivo): {m.objVal:,.2f}  ({pct:.2f} % del total)")

print(f"\n  Tiempo total         : {time.perf_counter() - t0: .2f} s")
```

.2.7. FRLM con algorítmo genético:

```
import random, time
import pandas as pd
from collections import defaultdict
import numpy as np

#  parámetros GA
P_ESTACIONES   = 5
POP_SIZE       = 100
GENERATIONS    = 200
CROSS_RATE     = 0.8
MUT_RATE       = 0.5
TOURNAMENT     = 3
RANDOM_SEED    = 42

np.random.seed(RANDOM_SEED)

#  cargar matrices y pesos
A = pd.read_csv("a_matrix.csv", index_col=0).astype(int)
B = pd.read_csv("b_matrix.csv", index_col=0).astype(int)
weights = pd.read_csv("nodos_weights.csv", index_col=0)

# Aseguramos tipos homogéneos
A.index, A.columns = A.index.astype(int), A.columns.astype(int)
B.columns          = B.columns.astype(int)
weights.index      = weights.index.astype(int)

# flujos por viaje
flujo = (
    pd.read_csv("matriz_trafico.csv", index_col=0).astype(float)
      .stack()
      .loc[lambda s: s > 0]
)
flujo.index = flujo.index.map(lambda t: f"{int(t[0])}-{int(t[1])}")
f_q = flujo.to_dict()
Q   = list(B.index)

#  pre-procesado bit-a-bit
# Bitmask del nodo k = 1 << pos[k]
NODES = sorted(A.columns)
pos   = {k: i for i, k in enumerate(NODES)}          # mapa nodo → bit
bit  = {k: 1 << pos[k]   for k in NODES}

# 1. máscaras de cada hiper-camino
mask_h = {h: int((A.loc[h] == 1).dot([1 << i for i in range(len(NODES))]))
          for h in A.index}

# 2. para cada viaje q, lista de máscaras de hiper-caminos que lo cubren
h_by_q = {q: [mask_h[h] for h in B.columns[B.loc[q] == 1]]
          for q in Q}

# 3. flujo total (para porcentaje)
FLUJO_TOTAL = sum(f_q.values())

#  funciones GA
def fitness(chrom: frozenset[int]) -> float:
    """Suma de flujos cubiertos por el conjunto de nodos 'chrom'."""
    mask_S = 0
    for k in chrom:
        mask_S |= bit[k]

    flow = 0.0
    for q, h_masks in h_by_q.items():
        for m in h_masks:
            if m & ~mask_S == 0:
                flow += f_q[q]
                break
    return flow
```

```
def random_chrom(weight_col="weight_best") -> frozenset[int]:
    """Devuelve un cromosoma de p nodos sin repetición, sesgado por los pesos."""
    probs = weights[weight_col].values
    probs = probs / probs.sum()

    seleccion = np.random.choice(
        weights.index.values,
        size=P_ESTACIONES,
        replace=False,
        p=probs
    )
    return frozenset(map(int, seleccion))


def tournament(pop, k=TOURNAMENT):
    return max(random.sample(pop, k), key=lambda ind: ind["fit"])


def crossover(p1: frozenset[int], p2: frozenset[int]) -> frozenset[int]:
    if random.random() > CROSS_RATE:
        return p1

    child = set()
    for k in p1:
        if random.random() < 0.5:
            child.add(k)
    child.update(k for k in p2 if len(child) < P_ESTACIONES)
    # reparar tamaño (si faltan nodos, añade al azar; si sobran, quita al azar)
    while len(child) < P_ESTACIONES:
        child.add(random.choice(NODES))
    while len(child) > P_ESTACIONES:
        child.remove(random.choice(tuple(child)))
    return frozenset(child)


def mutate(chrom: frozenset[int]) -> frozenset[int]:
    if random.random() > MUT_RATE:
        return chrom
    chrom = set(chrom)
    out = random.choice(tuple(chrom))
    chrom.remove(out)
    chrom.add(random.choice([k for k in NODES if k not in chrom]))
    return frozenset(chrom)

#  ciclo evolutivo
# población inicial
population = []
for _ in range(POP_SIZE - 2):
    c = random_chrom()
    population.append({"chrom": c, "fit": fitness(c)})

# Semillas deterministas: mejores nodos individuales
best_single = weights["score_abs"].nlargest(P_ESTACIONES).index.tolist()
seed_union  = frozenset(best_single[:P_ESTACIONES])
population.append({"chrom": seed_union, "fit": fitness(seed_union)})

best_node  = weights["score_abs"].idxmax()
population.append({"chrom": frozenset([best_node] + random.sample(
                    [k for k in NODES if k != best_node], P_ESTACIONES - 1)),
                   "fit": 0.0})   # se evaluará en el loop

# evalúa semillas
population[-1]["fit"] = fitness(population[-1]["chrom"])

# evolución
t_start = time.perf_counter()
for gen in range(1, GENERATIONS + 1):
    new_pop = []

    # elitismo: mejor actual pasa directo
    elite = max(population, key=lambda ind: ind["fit"])
    new_pop.append(elite)

    # cruce + mutación
    while len(new_pop) < POP_SIZE:
        p1 = tournament(population)
```

```
        p2 = tournament(population)
        child = crossover(p1["chrom"], p2["chrom"])
        child = mutate(child)
        new_pop.append({"chrom": child, "fit": fitness(child)})

    population = new_pop

    # log simple cada 10 generaciones
    if not gen % 10 or gen == 1:
        best = elite["fit"]
        print(f"Gen {gen:3d} | best={best:,.0f}  "
              f"({best / FLUJO_TOTAL * 100:5.2f} % del flujo)")

best_solution = max(population, key=lambda ind: ind["fit"])
t_end = time.perf_counter()

print("\n Resultado GA ")
print(f"Mejor flujo   : {best_solution['fit']:,.0f} "
      f"({best_solution['fit'] / FLUJO_TOTAL * 100:.2f} % del total)")
print(f"Nodos estación: {sorted(best_solution['chrom'])}")
print(f"Generaciones  : {GENERATIONS}")
print(f"Tiempo total  : {t_end - t_start: .1f} s")
```