Universidad Complutense de Madrid

Trabajo de Fin de Máster
Máster en Ingeniería Matemática

El Problema de Rutas Ecológico con Múltiples Tecnologías y Recargas Parciales

Autor: *Miguel Ambrona Castellanos.*

Tutores: *Ángel Felipe Ortega.*
Gregorio Tirado Domínguez.

Madrid, 2014.

Abstract

A variant of the well-known vehicle routing problem (VRP) has been formulated and studied, the *Green Vehicle Routing Problem with Multiple Technologies and Partial Recarges*. In the *GVRP-MTPR*, the fleet consists of electric vehicles. Therefore, they have a limited range and they need to recharge their batteries in refueling stations. This fact is an additional difficulty, because there are nodes in the network (the refueling stations) that can be visited more than once or never.

Two integer linear programming models have been developed for exact resolution. However, in practice these models are useful in "small" instances of the problem, but not in "large" ones, because their resolution requires excessive computing time. Alternatively, heuristic algorithms have been developed to find good solutions, though maybe not optimal, within short running times. Also three metaheuristic techniques (Simulated Annealing, Tabu Search and Variable Neighbourhood Search) have been implemented and tested in several test instances. Computational results show that metaheuristics perform very well and are a good alternative for solving this combinatorial optimization problem.

Keywords:
vehicle routing, electric vehicles, metaheuristics, simulated annealing, tabu search, variable neighbourhood search

Índice general

Capítulo 1

Introducción

El exponencial desarrollo tecnológico en los últimos años y la globalización de internet, hace que cada vez sea más gente la que se anime a realizar compras a domicilio. Este nuevo mercado en desarrollo crea la necesidad de establecer un modelo de distribución de productos. Así, las organizaciones que inviertan en este campo y sean capaces de crear redes de transporte sostenibles serán las que prosperen en el ámbito económico actual.

1.1 Motivación del tema tratado

Las operaciones logísticas de hoy en día están produciendo efectos muy negativos en la sociedad y el medio ambiente. Las actividades de transporte son responsables de gran parte de las emisiones de gases de efecto invernadero y por ello, las regulaciones europeas son cada vez más exigentes y penalizan gravemente cada gramo de CO_2 por Km emitido a la atmósfera por encima del máximo establecido. El límite marcado de emisiones medias de dióxido de carbono para 2016 es de 130 g/Km, mientras que en 2020 se reducirá a 95 g/Km. De hecho, el libro blanco de la Comisión Europea pretende conseguir ciudades prácticamente libres de emisiones en 2030.

Esta normativa está provocando que cada vez sean más los fabricantes que apuesten por la hibridación de los motores y de hecho el objetivo para 2020 en emisiones es prácticamente una realidad puesto que los diésel más avanzados sólo emiten 81 g/Km (de CO_2). No obstante, la alternativa más ecológica es la del vehículo eléctrico, sin embargo aún queda mucho para que estos vehículos cobren presencia en la sociedad puesto que presentan una serie de dificultades:

- ◇ Tienen baja autonomía, actualmente el rango varía entre unos 130 y 200 kilómetros.
- ◇ Por el momento, los puntos de recarga en la vía pública son aún escasos.
- ◇ El tiempo de recarga es lento (comparado con el repostaje de gasolina o diésel).

Los inconvenientes anteriores serán solventados poco a poco con el desarrollo de la tecnología. Hay fabricantes como *Tesla* que están trabajando en modelos con mayor autonomía, a base de contar con múltiples y potentes baterías. Además, la red española de estaciones de recarga va en aumento gracias al *Programa de Impulso al Vehículo Eléctrico* del Ministerio de Industria. Incluso la investigación en nuevas tecnologías de recarga promete grandes mejoras en este campo. Otra forma de contribuir al uso del vehículo eléctrico además de la evolución tecnológica es el desarrollo de modelos matemáticos para optimizar sus prestaciones y hacerlo rentable.

1.2 Relación con los conocimientos adquiridos durante el Máster

Este trabajo es la continuación del Trabajo de Fin de Grado citado en [2] en el que se planteaba un problema de optimización de rutas de vehículos eléctricos. El problema ha sido ampliado a uno más complejo y se han desarrollado diferentes técnicas para resolverlo tanto de forma exacta como de forma aproximada.

En el capítulo 2 se define el problema y se presentan dos modelos de programación matemática para el cálculo de la solución óptima. Para ello ha sido de especial ayuda la asignatura de *Modelos Determinísticos en Logística* en la que se estudió cómo establecer modelos matemáticos para resolver, mediante modelos de programación lineal entera, problemas complejos. En la práctica, los modelos de programación lineal sólo se pueden resolver en problemas de pequeña dimensión. En los capítulos 3 y 4 se estudian diferentes técnicas aproximativas de resolución, para las cuales ha sido útil cursar la asignatura de *Técnicas Avanzadas de Optimización* del Máster. También han sido útiles asignaturas como *Estadística Aplicada y Minería de Datos*, por ejemplo para el algoritmo de agrupamiento (*cluster*) o la asignatura de *Técnicas de Simulación* para la generación de casos aleatorios.

Los algoritmos descritos en este trabajo han sido implementados en Matlab y recogidos en una interfaz gráfica. En el capítulo 5 se realiza una comparativa de los diferentes algoritmos desarrollados a partir de la resolución de diferentes casos.

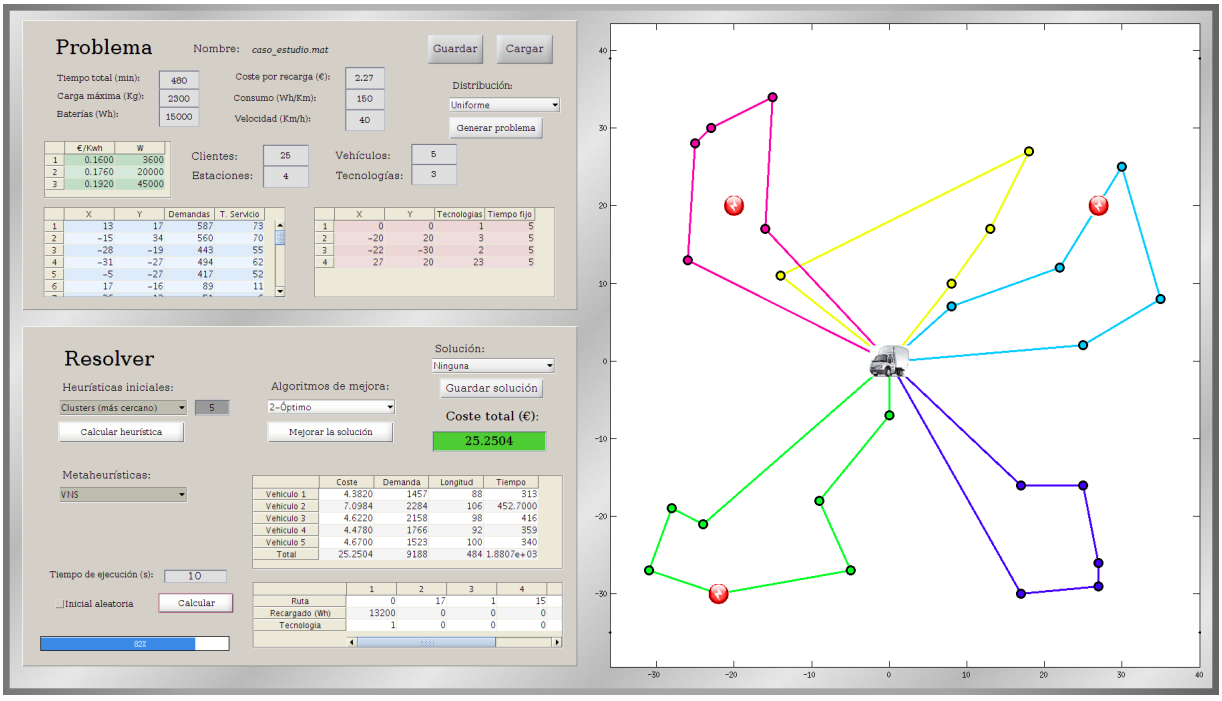

Figura 1.1: Captura de la interfaz gráfica desarrollada en Matlab.

Capítulo 2

El Problema

En este trabajo se estudiará cómo abordar el *Problema de Rutas Ecológico con Múltiples Tecnologías y Recargas Parciales (GVRP-MTPR)*. Se trata de una ampliación del problema planteado por Sevgi Erdogan y Elise Miller-Hooks en [7], conocido como *Green Vehicle Routing Problem (GVRP)*. El *GVRP*, traducido como *Problema de Rutas Ecológico*, es un problema de rutas de vehículos con la particularidad de que los vehículos son eléctricos, utilizan baterías recargables. Esto da lugar a un nuevo tipo de decisión a tomar: cuándo y dónde recargar.

El problema anterior se puede extender incluyendo algunas consideraciones que lo hacen más realista. Una posible extensión del *GVRP* es el *GVRP-MTPR*, propuesto por Ángel Felipe, M^a Teresa Ortuño, Gregorio Tirado y Giovanni Righini en [8]. En este nuevo problema, se contempla la posibilidad de recargar parcialmente las baterías, lo que permite ahorrar tiempo y dinero. Además, cada operación de recarga se puede realizar con diferentes tecnologías, que implican diferentes tiempos de recarga y diferentes costes (generalmente las tecnologías más rápidas son a la vez más caras). De esta forma, no se debe decidir exclusivamente cuándo y dónde deberá recargar cada vehículo, además hay que precisar cuánto recargar y con qué tecnología en cada caso.

2.1 Definición del problema

Se dispone de n vehículos eléctricos, inicialmente ubicados en un depósito D. Mediante éstos, se debe dar servicio a una serie de clientes $\mathcal{C}$ suministrándoles una determinada mercancía. Las demandas son conocidas, se miden en unidades de mercancía y se denotan por $\delta_i \ \forall \ i \in \mathcal{C}$. Además, la visita a un cliente supone un determinado tiempo de servicio $s_i \ \forall i \in \mathcal{C}$.

Los vehículos de reparto tienen una serie de características comunes:

- ◇ Cada vehículo puede transportar un máximo de Q unidades de mercancía.
- ◇ La capacidad máxima de sus baterías es de E unidades de energía.
- ◇ Consumen g unidades de energía por unidad de distancia.

Existe un conjunto $\mathcal{R}$ de estaciones de recarga para las baterías de los vehículos. Dichas estaciones cuentan con una oferta de diferentes tecnologías de recarga, $\mathcal{T}$. Cada

tecnología proporciona una tasa de recarga de ρ_t unidades de energía por unidad de tiempo y tiene un coste de c_t unidades monetarias por unidad de energía $\forall t \in \mathcal{T}$. Cada estación $k \in \mathcal{R}$ ofrece un conjunto $\mathcal{T}_k \subseteq \mathcal{T}$ de las mencionadas tecnologías de recarga y se considera un tiempo fijo de recarga adicional, en cada estación, de f_k unidades de tiempo.

Además, se supone conocida la distancia entre cada par de nodos, d_{ij}, así como el tiempo necesario para viajar entre ellos, t_{ij}.

Se define una jornada, como un periodo de T_{max} unidades de tiempo, en el que se debe satisfacer la demanda de todos los clientes en condiciones de factibilidad, es decir, sin que ningún vehículo se quede sin energía y sin que ningún vehículo transporte más mercancía de la permitida. Al principio de cada jornada los vehículos parten con sus baterías cargadas por completo (o lo suficiente para realizar su ruta), por lo tanto cuando llegan al depósito se reajusta su batería preparando el vehículo para la siguiente jornada. Esta recarga realizada en el depósito tiene lugar con una tecnología muy barata (c^* unidades monetarias por unidad de energía) pero a la vez lenta (solo disponible durante la noche).

El objetivo del problema es encontrar la política óptima que de minimiza el coste total de una jornada. Éste está formado por el coste de recarga nocturna en el depósito, por el coste de recarga en cada una de las estaciones y por un coste fijo, c_f, asociado a cada recarga (que pretende contabilizar el hecho de que una batería tiene un número limitado de ciclos hasta el final de su vida útil).

Por último, se incluyen algunas hipótesis adicionales:

- ◇ Cada cliente será visitado una única vez. Por lo tanto, $\delta_i \leq Q \ \forall i \in \mathcal{C}$.
- ◇ En cada estación se puede recargar *parcialmente* la batería de un vehículo.
- ◇ No se permiten viajes desde una estación de recarga a otra sin pasar por un cliente.

2.2 Modelo matemático del problema

Este problema presenta una dificultad que no aparece en otros problemas de rutas. En la forma general de los problemas de rutas, cada nodo debe ser visitado una sola vez. Sin embargo, en esta ocasión no es así. Es cierto que el conjunto de clientes debe cumplir esta condición, por el contrario, el conjunto de estaciones de recarga no. Cada estación puede ser visitada por varios vehículos e incluso por el mismo vehículo varias veces en una misma ruta. Esto aumenta la dificultad de construir un modelo de programación matemática para resolver el problema.

Usualmente en los problemas de rutas la solución queda caracterizada por unas variables:

$$x_{ij} = \begin{cases} 1 & \text{si algún vehículo viaja desde el cliente } i \text{ hasta el } j \\ 0 & \text{en caso contrario.} \end{cases}$$

Además de otras variables que hacen posible el cumplimiento de todas las restricciones del problema. En el problema estudiado en este trabajo, estas variables binarias x_{ij} no son suficientes para determinar de forma unívoca una solución. Esto se debe a que hay nodos (las estaciones de recarga) que pueden ser visitados por más de un vehículo. En el siguiente ejemplo se muestra este hecho:

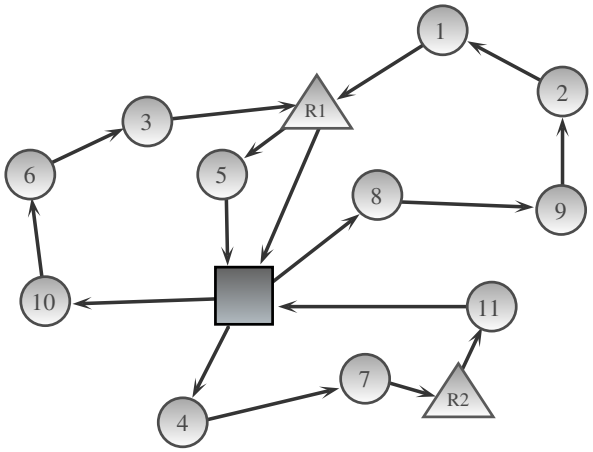

Figura 2.1: Solución representada con las variables x_{ij}.

El cuadrado representa el depósito de vehículos, mediante círculos son representados los clientes del problema y los triángulos son las estaciones de recarga. Es claro que en la representación anterior, uno de los vehículos ha realizado la siguiente ruta:

$$D \to 4 \to 7 \to R2 \to 11 \to D$$

Además, se puede ver que en la solución se utilizan dos vehículos más (hay un total de tres arcos que salen desde el depósito). Sin embargo, no es posible determinar cuál es la ruta que debe realizar cada uno de ellos. Por ejemplo, podrían ser las siguientes rutas:

$$D \to 10 \to 6 \to 3 \to R1 \to 5 \to D \qquad \text{y} \qquad D \to 8 \to 9 \to 2 \to 1 \to R1 \to D$$

Pero también podrían ser éstas:

$$D \to 10 \to 6 \to 3 \to R1 \to D \qquad \text{y} \qquad D \to 8 \to 9 \to 2 \to 1 \to R1 \to 5 \to D$$

Para solucionar este problema se podría *multiplicar* los nodos que representan estaciones de recarga y hacerlos de "un solo uso", pero no es fácil determinar a priori cuántas veces se utilizará cada estación y además, se incrementa mucho el número de nodos y, por tanto, el de variables binarias. Para poder determinar de forma unívoca una solución sin multiplicar las estaciones de recarga se utilizará una formulación de variables ideada en el Trabajo de Fin de Grado [2] que precede a este documento. Consiste en utilizar las variables x_{ij}, sólo entre clientes y el depósito, utilizando otras variables binarias para gestionar los viajes a estaciones de recarga.

Sea $\mathcal{C}$ el conjunto de clientes y sea D el índice correspondiente al depósito. Se establece que $\mathcal{V} = \{\mathcal{C} \cup D\}$. Además, sea $\mathcal{R}$ el conjunto de estaciones de recarga, de forma que el conjunto total de nodos del problema es $\mathcal{V} \cup \mathcal{R}$. Se utilizarán las variables:

$$x_{ij} = \begin{cases} 1 & \text{si algún vehículo viaja desde el cliente } i \text{ hasta el } j \\ 0 & \text{en caso contrario.} \end{cases} \qquad \forall i, j \in \mathcal{C} \cup D,\ i \neq j$$

Para poder gestionar las recargas, se introducen las siguientes variables auxiliares:

$$y_{ik} = \begin{cases} 1 & \text{si tras salir del cliente } i \text{ se reposta en la estación } k \\ 0 & \text{en caso contrario.} \end{cases} \qquad \forall i \in \mathcal{C}\ \ \forall k \in \mathcal{R}$$

Si la variable x_{ij} toma el valor 1 significa que el cliente j está en la misma ruta que i y será el cliente visitado inmediatamente después de i. Ahora, si $y_{ik} = 1$ para algún $k \in \mathcal{R}$, el vehículo no irá directamente desde i hasta j, sino que hará una parada en la estación k. En la figura 2.2 se representa el ejemplo anterior con estas nuevas variables:

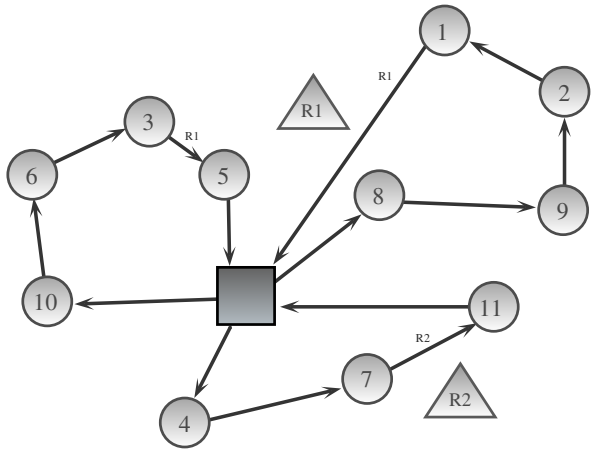

Figura 2.2: Solución representada con las nuevas variables.

Ahora, las solución está perfectamente determinada y corresponde a la primera interpretación de la figura 2.1.

2.2.1 Modelo I (multiples depósitos ficticios)

En el *GVRP* se establece que los vehículos deben recargar al máximo cada vez que visitan una estación de recarga (y que deben salir del depósito con las baterías al máximo). Ésta es una hipótesis que desaparece en el *GVRP-MTPR*, hay que decidir cuánto recargar en cada estación y con cuánto salir desde el depósito.

Para gestionar con cuánta carga saldrá del depósito cada vehículo, parece inevitable almacenar esta información en una variable por vehículo. Por este motivo, surge la idea de considerar tantos depósitos ficticios como vehículos haya. De forma que cada vehículo empezará y terminará su ruta en uno de los depósitos (el suyo propio), aunque realmente todos los depósitos ficticios representen un solo nodo. A continuación presentamos el

modelo de programación matemática desarrollado basado en esta idea:

Denotaremos por $\mathcal{V} = \{\mathcal{C} \cup \mathcal{R} \cup \mathcal{D}\}$ el conjunto de nodos del problema, donde $\mathcal{C}$ representa el conjunto de clientes, $\mathcal{R}$ el conjunto de estaciones de recarga y $\mathcal{D}$ el conjunto de depósitos (habrá un depósito ficticio por cada vehículo disponible).

Sean las variables de decisión:

- $x_{ij} = \begin{cases} 1 & \text{si algún vehículo viaja desde el cliente } i \text{ hasta el } j \\ 0 & \text{en caso contrario.} \end{cases} \quad \forall i, j \in \mathcal{C} \cup \mathcal{D},\ i \neq j$
- $y_{ik} = \begin{cases} 1 & \text{si tras salir del cliente } i \text{ se reposta en la estación } k \\ 0 & \text{en caso contrario.} \end{cases} \quad \forall i \in \mathcal{C} \cup \mathcal{D}\ \ \forall k \in \mathcal{R}$

- $r_{ik}^t \equiv$ cantidad de energía recargada en la estación k con la tecnología t, tras salir del cliente i, $\forall i \in \mathcal{C} \cup \mathcal{D}\ \forall k \in \mathcal{R}\ \forall t \in \mathcal{T}_k$.

- $q_i \equiv$ carga del vehículo al salir del cliente i, $\forall i \in \mathcal{C} \cup \mathcal{D}$. Por definición, $q_i = Q\ \forall i \in \mathcal{D}$.
- $\gamma_i \equiv$ cantidad de energía disponible al salir del nodo i (cliente o depósito), $\forall i \in \mathcal{C} \cup \mathcal{D}$.
- $\gamma_v^D \equiv$ cantidad de energía disponible al llegar al depósito el vehículo v, $\forall v \in \mathcal{D}$.
- $\gamma_i^R \equiv$ cantidad de energía disponible al llegar a la estación que sigue a i, $\forall i \in \mathcal{C} \cup \mathcal{D}$.

Si no existe $k \in \mathcal{R}$ tal que $y_{ik} = 1$, la variable no tiene ningún significado, por simplicidad se establece que, en ese caso, $\gamma_i^R = \gamma_i$.

- $\tau_i \equiv$ instante de salida desde el cliente i, $\forall i \in \mathcal{C} \cup \mathcal{D}$. Si $i \in \mathcal{D}$ el significado de la variable es distinto, representa el instante de llegada al depósito.

La función objetivo del problema es la siguiente:

$$\text{mín}\, c^* \sum_{v \in \mathcal{D}} \gamma_v + \sum_{i \in \mathcal{C}} \sum_{k \in \mathcal{R}} \sum_{t \in \mathcal{T}_k} c_t r_{ik}^t + c_f \left(\sum_{i \in \mathcal{C}} \sum_{k \in \mathcal{R}} y_{ik} + \sum_{i \in \mathcal{D}} \sum_{j \in \mathcal{C} \cup \mathcal{D}} x_{ij} \right) \tag{2.1}$$

La restricciones son:

$$\sum_{i \in \mathcal{C} \cup \mathcal{D}} x_{ij} = 1 \qquad \forall j \in \mathcal{C} \tag{2.2}$$

$$\sum_{i \in \mathcal{C} \cup \mathcal{D}} (x_{ij} - x_{ji}) = 0 \qquad \forall j \in \mathcal{C} \cup \mathcal{D} \tag{2.3}$$

$$\sum_{j \in \mathcal{C} \cup \mathcal{D}} x_{vj} \leq 1 \qquad \forall v \in \mathcal{D} \tag{2.4}$$

$$\sum_{k \in \mathcal{R}} y_{ik} \leq 1 \qquad \forall i \in \mathcal{C} \tag{2.5}$$

$$q_j \leq q_i - \delta_j + L_1(1 - x_{ij}) \qquad \forall i \in \mathcal{C} \cup \mathcal{D}\ \ \forall j \in \mathcal{C} \tag{2.6}$$

$$\tau_j \geq \tau_i + t_{ij} + s_i - L_2(1 - x_{ij} + \textstyle\sum_{k\in\mathcal{R}} y_{ik}) \qquad \forall i \in \mathcal{C} \;\; \forall j \in \mathcal{C}\cup\mathcal{D} \quad (2.7)$$

$$\tau_j \geq t_{vj} - L_3(1 - x_{vj}) \qquad \forall v \in \mathcal{D} \;\; \forall j \in \mathcal{C} \quad (2.8)$$

$$\tau_j \geq \tau_i + t_{ik} + t_{kj} + s_i + \sum_{t\in\mathcal{T}_k} \frac{1}{\rho_t} r_{ik}^t + f_k - L_4(2 - x_{ij} - y_{ik}) \qquad \forall i \in \mathcal{C} \;\; \forall j \in \mathcal{C}\cup\mathcal{D} \;\; \forall k \in \mathcal{R} \quad (2.9)$$

$$\gamma_j \leq \gamma_i - g d_{ij} + L_5(1 - x_{ij} + \textstyle\sum_{k\in\mathcal{R}} y_{ik}) \qquad \forall i \in \mathcal{C}\cup\mathcal{D} \;\; \forall j \in \mathcal{C} \quad (2.10)$$

$$\gamma_j \leq \gamma_i - g(d_{ik} - d_{kj}) + \sum_{t\in\mathcal{T}_k} r_{ik}^t + L_6(2 - x_{ij} - y_{ik}) \qquad \forall i \in \mathcal{C}\cup\mathcal{D} \;\; \forall j \in \mathcal{C} \;\; \forall k \in \mathcal{R} \quad (2.11)$$

$$\gamma_j^D \leq \gamma_i - g d_{ij} + L_5(1 - x_{ij} + \textstyle\sum_{k\in\mathcal{R}} y_{ik}) \qquad \forall i \in \mathcal{C}\cup\mathcal{D} \;\; \forall j \in \mathcal{D} \quad (2.12)$$

$$\gamma_j^D \leq \gamma_i - g(d_{ik} - d_{kj}) + \sum_{t\in\mathcal{T}_k} r_{ik}^t + L_6(2 - x_{ij} - y_{ik}) \qquad \forall i \in \mathcal{C}\cup\mathcal{D} \;\; \forall j \in \mathcal{D} \;\; \forall k \in \mathcal{R} \quad (2.13)$$

$$\gamma_i^R = \gamma_i - g \sum_{k\in\mathcal{R}} d_{ik} y_{ik} \qquad \forall i \in \mathcal{C}\cup\mathcal{D} \quad (2.14)$$

$$\sum_{t\in\mathcal{T}_k} r_{ik}^t \leq E y_{ik} \qquad \forall i \in \mathcal{C} \;\; \forall k \in \mathcal{R} \quad (2.15)$$

$$\gamma_i^R + \sum_{t\in\mathcal{T}_k} r_{ik}^t \leq E \qquad \forall i \in \mathcal{C} \;\; \forall k \in \mathcal{R} \quad (2.16)$$

Por último:

$$\begin{aligned}
&x_{ij} \in \{0,1\} && \forall i,j \in \mathcal{C}\cup\mathcal{D}\\
&y_{ik} \in \{0,1\} && \forall i \in \mathcal{C} \;\; \forall k \in \mathcal{R}\\
&0 \leq r_{ik}^t \leq E && \forall i \in \mathcal{C} \; \forall k \in \mathcal{R} \; \forall t \in \mathcal{T}_k\\
&0 \leq q_i \leq Q && \forall i \in \mathcal{C}\cup\mathcal{D}\\
&0 \leq \tau_i \leq T_{max} && \forall i \in \mathcal{C}\cup\mathcal{D}\\
&0 \leq \gamma_i \leq E && \forall i \in \mathcal{C}\cup\mathcal{D}\\
&0 \leq \gamma_v^D \leq E && \forall v \in \mathcal{D}\\
&0 \leq \gamma_i^R \leq E && \forall i \in \mathcal{C}
\end{aligned}$$

Con (2.2) se consigue que los clientes sean visitados una y sólo una vez mientras que la igualdad (2.3) recoge las condiciones de conservación de flujo. Con (2.4) permite restringir el número de vehículos, cada vehículo llegará a uno de los depósitos ficticios, en concreto no podrán utilizarse más de $\#\mathcal{D} = n$ vehículos. La restricción (2.5) sirve para indicar que sólo se puede recargar en una estación tras salir de un cliente. Por otro lado, (2.6) restringe la carga de mercancía total de los vehículos, para que no excedan su capacidad. La constante L_1 debe ser tal que desaparezca la desigualdad si $x_{ij} = 0$. Podría tomarse,

$$L_1 = Q + \max_j \{\delta_j\}$$

Las restricciones (2.7)-(2.9) permiten darle sentido a las variables de tiempo. Con (2.7) gestiona las variables de tiempo entre clientes, mientras que (2.8) lo hace en las salidas del depósito y (2.9) en los viajes que incluyen recargas. Se podría elegir como constantes:

$$L_2 = T_{max} + \text{máx}_{i,j\in\mathcal{V}}\{t_{ij}\} + \text{máx}_j\{s_j\}$$

$$L_3 = T_{max} + 2\,\text{máx}_{i,j\in\mathcal{V}}\{t_{ij}\} + \text{máx}_j\{s_j\} + \frac{E}{\text{mín}_{t\in\mathcal{T}}\{\rho_t\}} + \text{máx}_{k\in\mathcal{R}}\{f_k\}$$

$$L_4 = \text{máx}_{j\in\mathcal{V}}\{t_{Dj}\} + \text{máx}_j\{s_j\}$$

Además, con (2.10)-(2.14) se controla las variables de energía. Las restricciones (2.10) y (2.11) se encargan de los trayectos que llegan a un cliente, mientras que (2.12) y (2.13) lo hacen para las llegadas al depósito. La restricción (2.14) relaciona las variables de energía con las variables binarias como se indica en la definición de variables. Se debería elegir,

$$L_5 = E + g\,\text{máx}_{i,j\in\mathcal{V}}\{d_{ij}\}$$

$$L_6 = E + 2g\,\text{máx}_{i,j\in\mathcal{V}}\{d_{ij}\}$$

Por último, (2.15) relaciona las variables binarias y las cantidades de recarga, mientras que (2.16) sirve para que no se recargue más energía de la que cabe en una batería.

El modelo anterior tiene un inconveniente: se crean nodos ficticios al multiplicar el depósito, lo que aumenta el número de variables binarias. Se puede intentar diseñar un modelo que no duplique el depósito sino que lo contabilice una única vez, de forma que el número de variables binarias se reduzca. Con esta idea se ha desarrollado el modelo de programación matemática que sigue a continuación.

2.2.2 Modelo II (un solo depósito)

El mayor problema de considerar solo un depósito es poder gestionar la energía inicial de cada vehículo. Para solucionarlo, se introducen las variables γ_j^D, que permiten almacenar la energía disponible al salir del depósito hacia el cliente j. Se presenta el modelo detallado:

Sea $\mathcal{C}$ el conjunto de clientes y sea D el índice correspondiente al depósito. Se establece que $\mathcal{V} = \{\mathcal{C} \cup D\}$. Además, sea $\mathcal{R}$ el conjunto de estaciones de recarga, de forma que el conjunto total de nodos del problema es $\mathcal{V} \cup \mathcal{R}$. Por último, dada una estación $k \in \mathcal{R}$, $\mathcal{T}_k$ representa el conjunto de tecnologías disponibles en dicha estación.

Sean las variables de decisión:

- $x_{ij} = \begin{cases} 1 & \text{si algún vehículo viaja desde el cliente } i \text{ hasta el } j \\ 0 & \text{en caso contrario.} \end{cases} \quad \forall i,j \in \mathcal{V},\ i \neq j$
- $y_{ik} = \begin{cases} 1 & \text{si tras salir del cliente } i \text{ se reposta en la estación } k \\ 0 & \text{en caso contrario.} \end{cases} \quad \forall i \in \mathcal{C}\ \ \forall k \in \mathcal{R}$
- $r_{ik}^t \equiv$ cantidad de energía recargada en la estación k con la tecnología t, tras salir del cliente i, $\forall i \in \mathcal{C}\ \forall k \in \mathcal{R}\ \forall t \in \mathcal{T}_k$.

- $q_i \equiv$ carga (de mercancía) del vehículo que visita al cliente i, al salir de dicho cliente, $\forall i \in \mathcal{V}$. Por definición, $q_D = Q$.
- $\tau_i \equiv$ instante de salida desde el cliente i, $\forall i \in \mathcal{C}$.
- $\gamma_i \equiv$ cantidad de energía disponible al salir del cliente i, $\forall i \in \mathcal{C}$.
- $\gamma_j^D \equiv$ cantidad de energía disponible al salir del depósito hacia el cliente j, $\forall j \in \mathcal{C}$. Si $x_{Dj} = 0$, debe ser $\gamma_j^D = 0$.
- $\gamma_i^R \equiv$ cantidad de energía disponible al llegar a la estación que sigue a i, $\forall i \in \mathcal{C}$. Si no existe $k \in \mathcal{R}$ tal que $y_{ik} = 1$, la variable no tiene ningún significado, por simplicidad se establece que, en ese caso, $\gamma_i^R = \gamma_i$.

La función objetivo del problema es la siguiente:

$$\text{mín}\, c^* \sum_{j \in \mathcal{C}} \gamma_j^D + \sum_{i \in \mathcal{C}} \sum_{k \in \mathcal{R}} \sum_{t \in \mathcal{T}_k} c_t r_{ik}^t + c_f \left(\sum_{i \in \mathcal{C}} \sum_{k \in \mathcal{R}} y_{ik} + \sum_{j \in \mathcal{C}} x_{Dj} \right) \tag{2.17}$$

La restricciones son:

$$\sum_{i \in \mathcal{V}} x_{ij} = 1 \qquad \forall j \in \mathcal{C} \tag{2.18}$$

$$\sum_{i \in \mathcal{V}} (x_{ij} - x_{ji}) = 0 \qquad \forall j \in \mathcal{V} \tag{2.19}$$

$$\sum_{i \in \mathcal{C}} x_{iD} \leq n \tag{2.20}$$

$$\sum_{k \in \mathcal{R}} y_{ik} \leq 1 \qquad \forall i \in \mathcal{C} \tag{2.21}$$

$$q_j \leq q_i - \delta_j + L_1(1 - x_{ij}) \qquad \forall i \in \mathcal{V} \;\; \forall j \in \mathcal{C} \tag{2.22}$$

$$\tau_j \geq \tau_i + t_{ij} + s_j - L_2(1 - x_{ij} + \textstyle\sum_{k \in \mathcal{R}} y_{ik}) \qquad \forall i \in \mathcal{C} \;\; \forall j \in \mathcal{C} \tag{2.23}$$

$$\tau_j \geq \tau_i + t_{ik} + t_{kj} + s_j + \textstyle\sum_{t \in \mathcal{T}_k} (\frac{1}{\rho_t} r_{ik}^t) + f_k - L_3(2 - x_{ij} - y_{ik}) \qquad \forall i \in \mathcal{C} \;\; \forall j \in \mathcal{C} \;\; \forall k \in \mathcal{R} \tag{2.24}$$

$$\tau_j \geq t_{Dj} + s_j - L_4(1 - x_{Dj}) \qquad \forall j \in \mathcal{C} \tag{2.25}$$

$$T_{max} \geq \tau_i + t_{iD} - L_2(1 - x_{iD} + \textstyle\sum_{k \in \mathcal{R}} y_{ik}) \qquad \forall i \in \mathcal{C} \tag{2.26}$$

$$T_{max} \geq \tau_i + t_{ik} + t_{kD} + \textstyle\sum_{t \in \mathcal{T}_k} (\rho_t^{-1} r_{ik}^t) + f_k - L_3(2 - x_{iD} - y_{ik}) \qquad \forall i \in \mathcal{C} \;\; \forall k \in \mathcal{R} \tag{2.27}$$

$$\gamma_j \leq \gamma_i - gd_{ij} + L_5(1 - x_{ij} + \sum_{k\in\mathcal{R}} y_{ik}) \qquad \forall i \in \mathcal{C} \;\; \forall j \in \mathcal{C} \qquad (2.28)$$

$$\gamma_j \leq \gamma_i - g(d_{ik} + d_{kj}) + \sum_{t\in\mathcal{T}_k} r_{ik}^t + L_6(2 - x_{ij} - y_{ik}) \qquad \forall i \in \mathcal{C} \;\; \forall j \in \mathcal{C} \;\; \forall k \in \mathcal{R} \qquad (2.29)$$

$$\gamma_j \leq \gamma_j^D - gd_{Dj} + L_5(1 - x_{Dj}) \qquad \forall j \in \mathcal{C} \qquad (2.30)$$

$$0 \leq \gamma_i - gd_{iD} + L_5(1 - x_{iD} + \sum_{k\in\mathcal{R}} y_{ik}) \qquad \forall i \in \mathcal{C} \qquad (2.31)$$

$$0 \leq \gamma_i - g(d_{ik} + d_{kD}) + \sum_{t\in\mathcal{T}_k} r_{ik}^t + L_6(2 - x_{iD} - y_{ik}) \qquad \forall i \in \mathcal{C} \;\; \forall k \in \mathcal{R} \qquad (2.32)$$

$$\gamma_j^D \leq Ex_{Dj} \qquad \forall j \in \mathcal{C} \qquad (2.33)$$

$$\gamma_i^R = \gamma_i - g\sum_{k\in\mathcal{R}} d_{ik}y_{ik} \qquad \forall i \in \mathcal{C} \qquad (2.34)$$

$$\sum_{t\in\mathcal{T}_k} r_{ik}^t \leq Ey_{ik} \qquad \forall i \in \mathcal{C} \;\; \forall k \in \mathcal{R} \qquad (2.35)$$

$$\gamma_i^R + \sum_{t\in\mathcal{T}_k} r_{ik}^t \leq E \qquad \forall i \in \mathcal{C} \;\; \forall k \in \mathcal{R} \qquad (2.36)$$

Las variables están definidas en las mismas condiciones que en el modelo anterior y además se pueden elegir exactamente las mismas cotas L_i que en el modelo previo. La nueva variable γ_j^D está definida entre 0 y E $\forall j \in \mathcal{C}$.

Con (2.18) se asegura que todos los clientes son visitados una única vez, mientras que con (2.19) se establecen las condiciones de conservación de flujo, nótese que al depósito puede llegar más de un vehículo. La desigualdad (2.20) garantiza que no se utilizan más vehículos de los permitidos. Y (2.21) da lugar a que un vehículo, tras salir de un cliente, no puede visitar más de una estación. Además, la restricción (2.22) indica que ningún vehículo puede transportar más mercancía de la permitida.

Las restricciones (2.23)-(2.27) gestionan las variables de tiempo. Con (2.23) y (2.24) se gestionan las variables de tiempo entre clientes. La desigualdad (2.25) está relacionada con los tiempos tras salir del depósito mientras que (2.26) y (2.27) garantizan que ningún vehículo realiza una ruta más duradera de lo permitido.

Análogamente, las restricciones (2.28)-(2.32) gestionan las variables de energía. Las restricciones (2.28) y (2.29) se encargan de los trayectos entre clientes, mientras que (2.30) está pensada para las salidas desde el depósito. En este caso, (2.31) y (2.32) sirven para prohibir que los vehículos se queden sin energía al llegar al depósito.

La restricción (2.33) establece las cantidades iniciales de energía de cada ruta y (2.34) permite calcular la energía con la que se llega a las estaciones de recarga. Por último, (2.35) relaciona las variables binarias y las cantidades de recarga, mientras que (2.36) sirve para que no se recargue más energía de la que cabe en una batería.

2.3 Complejidad del problema

Se trata de un problema de rutas de vehículos. Dar una solución es equivalente a determinar cuáles serán los clientes que visitará cada vehículo y en qué orden (además de algunos detalles

adicionales). Esto significa que el problema es una generalización del TSP (Travelling Salesman Problem), por lo que el *GVRP-MTPR* es $\mathcal{NP}$*-completo*.

En la siguiente sección se hace una comparación de los dos modelos de programación matemática propuestos para resolver de forma exacta el problema. Dada la dificultad de éste, sólo se ha conseguido llegar a la solución óptima (y asegurar que es óptima) en problemas con un número muy reducido de clientes (véase el capítulo 5). Para resolver problemas grandes se utilizarán métodos aproximados, que no buscan la solución óptima, pero sí soluciones razonablemente buenas (véanse los capítulos 3 y 4).

2.4 Comparación de los modelos propuestos

En esta sección se pretende comparar los dos modelos desarrollados. El segundo de ellos suponía una mejora con respecto al primero puesto que no multiplicaba el depósito creando nodos ficticios. A continuación se analiza cómo influye esto en el número de variables binarias del modelo.

Sea c el número total de clientes, r el número total de estaciones de recarga y n el número total de vehículos.

Modelo I:

Las variables x_{ij} existen para $i, j \in \mathcal{C} \cup \mathcal{D}$, $i \neq j$, por lo tanto, el número total de variables x_{ij} es de $(c+n)(c+n-1)$. Por otro lado, las variables y_{ik} están definidas con $i \in \mathcal{C} \cup \mathcal{D}$, $k \in \mathcal{R}$, lo que hace que haya un total de $(c+n)r$ variables y_{ik}. De este modo, el número total de variables binarias de este modelo es

$$(c+n)(c+n-1) + (c+n)r = c^2 + n^2 + 2cn + cr + nr - c - n$$

Modelo II:

En este nuevo modelo, las variables x_{ij} están definidas $\forall i, j \in \mathcal{V}$, $i \neq j$, de forma que hay un total de $(c+1)c$ variables. Por otro lado, las variables y_{ik} están defindas para $i \in \mathcal{C}$, $k \in \mathcal{R}$, por lo que hay un total de cr. El número total de variables binarias del modelo es:

$$(c+1)c + cr = c^2 + cr + c$$

Si se quiere resolver un problema con $c = 12$ clientes, $r = 4$ estaciones y $n = 6$ vehículos, el *Modelo I* contaría con $(12+6)(12+6-1)+(12+6)4 = 378$ variables binarias, por el contrario, el número de variables binarias con el *Modelo II* sería de $(12+1)12+12\cdot 4 = 204$, una cantidad significativamente menor.

A pesar de ser modelos bastante similares, cabe esperar que sea más eficiente el *Modelo II* al tener muchas menos variables binarias. Nótese que en el segundo modelo el número de variables binarias no depende del número de vehículos, n.

Capítulo 3

Algoritmos heurísticos

Como se ha visto anteriormente, el *GVRP-MTPR* es considerado un problema $\mathcal{NP}$*-completo*. Los problemas $\mathcal{NP}$*-completos* son muy difíciles de resolver, hasta hoy no se conoce ningún algoritmo que proporcione la solución óptima en tiempo polinomial y no cabe esperar que tal algoritmo exista a menos que $\mathcal{P} = \mathcal{NP}$.

A la hora de abordar este tipo de problemas, no es viable utilizar algoritmos exactos que proporcionen la solución óptima, debido a que el tiempo necesario para obtenerla sería desorbitado. Excepcionalmente, en problemas de pequeño tamaño (en el caso del *GVRP-MTPR*, con pocos clientes) se podría utilizar métodos exactos, "*ramificación y acotación*", "*ramificación y corte*", mejorados con técnicas de preproceso. Cuando el problema sea más grande, es necesario utilizar técnicas de aproximación para encontrar soluciones razonablemente buenas en un tiempo factible. Estos métodos de aproximación se conocen como "*heurísticas*".

"Una **heurística** *es un procedimiento simple, a menudo basado en el sentido común, que se supone que ofrecerá una buena solución (aunque no necesariamente la óptima) a problemas difíciles, de un modo fácil y rápido".* (Zanakis y Evans, 1981 [31]).

Las heurísticas clásicas de podrían dividir en dos grupos:

- ◇ Heurísticas **constructivas**: Parten de cero y construyen gradualmente una solución. Suelen ser algoritmos muy rápidos, pero en general ofrecen soluciones de baja calidad.
- ◇ Heurísticas **de mejora**: Comienzan con una solución factible e intentan mejorarla. Introducen pequeñas modificaciones en la solución que la van mejorando, hasta que se atascan en un *óptimo local* o se verifica algún criterio de parada.

Las heurísticas clásicas suelen ser algoritmos muy rápidos que proporcionan buenas soluciones, sin embargo sólo buscan en una región limitada del espacio de soluciones y esto hace que con gran probabilidad acaben en un óptimo local. Existen otras técnicas, conocidas como *metaheurísticas* que suelen ser más efectivas pues exploran de forma más eficiente el espacio de soluciones. En el capítulo 4 se profundizará en las técnicas metaheurísticas, describiendo con más detalle en qué consisten y analizando sus ventajas e inconvenientes.

En este capítulo se proponen algunas heurísticas constructivas y de mejora para el *GVRP-MTPR*. Además, se incluye una sección en la que se estudia cómo saber lo acertadas que han sido las soluciones proporcionadas por las heurísticas (lo lejos que están de la solución óptima) mediante la búsqueda de cotas en la función objetivo.

3.1 Caso de estudio

Con el objetivo de ilustrar cómo funcionan los algoritmos desarrollados que se irán proponiendo a lo largo del trabajo para resolver el *GVRP-MTPR*, se aplicarán en un problema de ejemplo que ha sido generado aleatoriamente con la interfaz desarrollada.
Las tecnologías de recarga disponibles en el problema son las siguientes:

Tecnologías	Coste (€/ Kwh)	Potencia de recarga (W)
1	0.160	3600
2	0.176	20000
3	0.192	45000

Por otro lado, el problema consta de 25 clientes y 3 estaciones de recarga (además del depósito). El depósito se encuentra en el punto de coordenadas $(0, 0)$ y es la única estación que permite la utilización de la tecnología 1. Los datos detallados de cada nodo son:

Cliente	Posición en X	Posición en Y	Demanda (Kg)	Tiempo de servicio (min)
1	13	17	587	73
2	-15	34	560	70
3	-28	-19	443	55
4	-31	-27	494	62
5	-5	-27	417	52
6	14	-16	89	11
7	-26	13	51	6
8	25	2	267	33
9	35	8	537	67
10	0	-7	553	69
11	25	-16	344	43
12	-24	-21	76	10
13	27	-29	549	69
14	-25	28	161	20
15	18	27	123	15
16	-16	17	370	46
17	8	10	314	39
18	-23	30	381	48
19	8	7	468	59
20	-14	11	433	54
21	-9	-18	301	38
22	22	12	323	40
23	30	25	563	70
24	27	-26	341	43
25	17	-30	443	55

Estación	Posición en X	Posición en Y	Tecnologías	Tiempo fijo (min)
1	-20	20	$\{3\}$	5
2	-22	-30	$\{2\}$	5
3	27	20	$\{2, 3\}$	5

Se cuenta con una flota de 5 vehículos, con una capacidad de carga de 2300 Kg de mercancía y cuyas baterías pueden almacenar 15000 Wh, llevan una velocidad media de 40Km/h y consumen 150 Wh/Km. Por último, el coste fijo por cada recarga es de 2,27 € y los vehículos deben realizar su recorrido en menos de 8 horas.

Para construir la matriz de distancias se ha calculado el techo de la norma euclídea entre cada par de puntos. La matriz de tiempos entre nodos se construye a partir de la de distancias y la velocidad media de los vehículos.

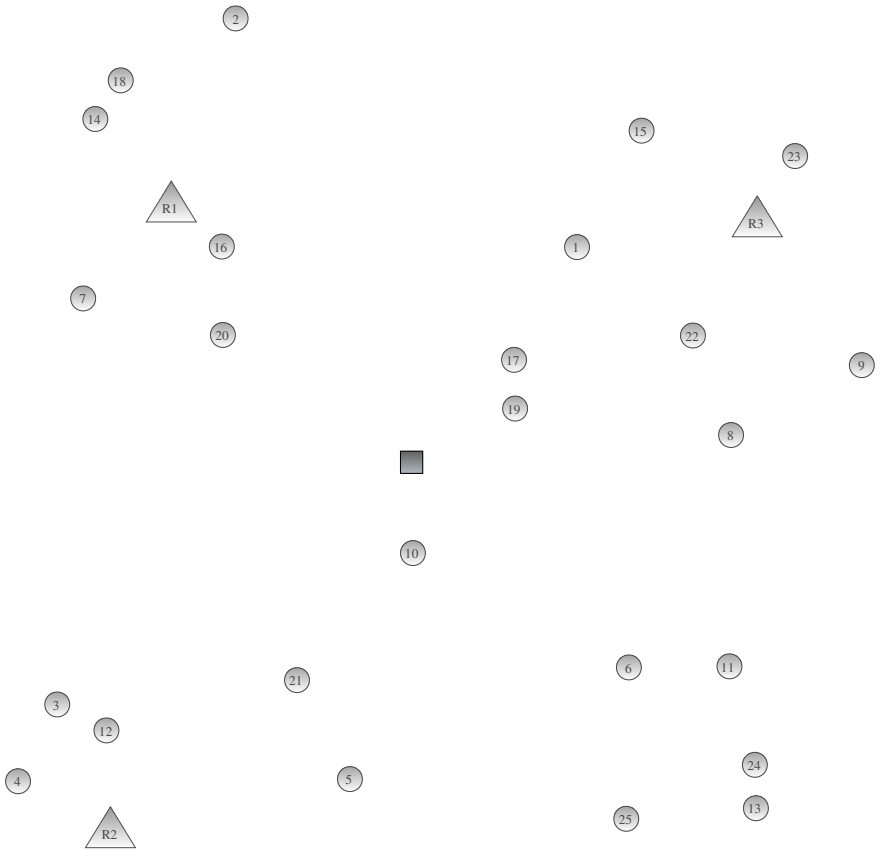

Figura 3.1: Disposición de los nodos en el problema de ejemplo.

3.2 Heurísticas constructivas

En esta sección se proponen tres heurísticas constructivas para el *GVRP-MTPR*. Las dos primeras: "*búsqueda del vecino más cercano*" y "*algoritmo de agrupamiento*" son modificaciones de las heurísticas propuestas en el Trabajo de Fin de Grado [2] que se adaptan al nuevo problema. La tercera heurística, más elaborada, es propia de este trabajo y se basa en utilizar el modelo de programación matemática con un número reducido de variables binarias.

Los parámetros del problema son:

- ◇ Número máximo de vehículos permitido, n.
- ◇ Matriz de distancias, (d_{ij}) $\forall i, j \in \mathcal{V} \cup \mathcal{R}$ (u. distancia).
- ◇ Matriz de tiempos, (t_{ij}) $\forall i, j \in \mathcal{V} \cup \mathcal{R}$ (u. tiempo).
- ◇ Demandas de los clientes, δ_i $\forall i \in \mathcal{C}$ (u. mercancía).
- ◇ Tiempos de servicio de los clientes, s_i $\forall i \in \mathcal{C}$ (u. tiempo).
- ◇ Capacidad de carga de los vehículos, Q (u. mercancía).
- ◇ Capacidad de energía de las baterías, E (u. energía).
- ◇ Consumo de energía por unidad de distancia, g (u.energía / u. distancia).
- ◇ Tasa de recarga con la tecnología t, ρ_t $\forall t \in \mathcal{T}$ (u. energía / u. tiempo).
- ◇ Tiempo fijo de recarga en la estación k, f_k $\forall k \in \mathcal{R}$ (u. tiempo).
- ◇ Conjunto de tecnologías disponibles en la estación k, $\mathcal{T}_k$ $\forall k \in \mathcal{R}$.
- ◇ Tiempo máximo permitido, T_{max} (u. tiempo).
- ◇ Coste fijo por cada recarga c_f (u. monetarias).
- ◇ Coste por unidad de energía de la recarga con tecnología t, c_t $\forall t \in \mathcal{T}$ (u. monetarias).

3.2.1 Búsqueda del vecino más cercano

Ésta es una heurística constructiva que se puede incluir en la clase de algoritmos *ávidos* o *voraces* (greedy). Se trata de algoritmos que toman decisiones óptimas en cada paso local con la esperanza de llegar o acercarse al óptimo global del problema. Esto no ofrece ninguna garantía de éxito, pero en general son algoritmos sencillos de programar y muy rápidos.

El algoritmo de la "búsqueda del vecino más cercano" construye una solución desde cero, ruta por ruta. Comienza con un vehículo, que visitará al cliente más cercano al depósito. Después, continúa visitando a los clientes que aún no han sido visitados, siempre dirigiéndose al más cercano de su posición en cada momento. Cuando, por motivos de factibilidad (ya sean de tiempo, de carga o de energía) el vehículo no puede continuar su trayecto, éste vuelve al depósito cerrando su ruta. El proceso se repite con otro vehículo hasta que todos los clientes han sido atendidos.

A continuación se incluye el pseudocódigo de este algoritmo que ha sido desarrollado:

Entrada: Datos del problema.
Salida: Una solución solucion $S = \{\mu_1, \mu_2, \dots \mu_n\}$ (conjunto de rutas).

$S \leftarrow \{\}$, $k \leftarrow 1$, $sin\_visitar = \{1, 2, \dots, \#\mathcal{C}\}$.
$\mu_k \leftarrow \{dep\}$, $tiempo \leftarrow 0$, $energia \leftarrow E$, $carga \leftarrow 0$, $pos \leftarrow dep$.

mientras $sin\_visitar \neq \emptyset$ (hay clientes sin visitar) **hacer**
  $dest \leftarrow$ lista ordenada de $sin\_visitar$ por proximidad a pos.
  $movido \leftarrow$ **falso**
  para $i = 1$ **hasta** $\#dest$ **hacer**
    si $(carga + \delta(dest(i)) > Q$ **or**
    $tiempo + t(pos, dest(i)) + s(dest(i)) > T_{max} - t(dest(i), dep)$) **entonces**
      avanzar en el bucle (No se puede ir a $dest(i)$).
    fin si
    si $(energia - g \cdot d(pos, dest(i)) \geq g \cdot d(dest(i), dep))$ **entonces**
      (Es factible ir a $dest(i)$).
      $carga \leftarrow carga + c(dest(i))$.
      $tiempo \leftarrow tiempo + t(pos, dest(i)) + s(dest(i))$.
      $energia \leftarrow energia - g \cdot d(pos, dest(i))$.
      $pos \leftarrow dest(i)$.
      Añadir $dest(i)$ a μ_k y Borrar $dest(i)$ de $sin\_visitar$.
      $movido \leftarrow$ **verdadero**
      terminar bucle
    si no
      (Si es posible se irá a la estación de recarga más cercana)
      $estacion \leftarrow \text{argmin}_{r \in \mathcal{R}}\{d(pos, r)\}$
      $t\_carga \leftarrow (E - (energia + g \cdot d(pos, estacion)))/\rho_{t^*}$.
      si $(energia - g \cdot d(pos, estacion) \geq 0$ **and**
      $tiempo + t(pos, estacion) + t\_carga + f(estacion) \leq T_{max} -$
      $t(estacion, dep))$ **entonces**
        (Es factible ir a $estacion$).
        $tiempo \leftarrow tiempo + t(pos, estacion) + f(estacion)$.
        $energia \leftarrow E$. (Se recarga a tope)
        $pos \leftarrow estacion$.
        Añadir $estacion$ a μ_k.
        $movido \leftarrow$ **verdadero**
        terminar bucle
      fin si
    fin si
  fin para
  si (**no** movido) (Hay que volver al depósito) **entonces**
    Añadir μ_k a S.
    $k \leftarrow k + 1$.
    $\mu_k \leftarrow \{dep\}$, $tiempo \leftarrow 0$, $energia \leftarrow E$, $carga \leftarrow 0$, $pos \leftarrow dep$.
  fin si
fin mientras

devolver S.

Algoritmo 1: Algoritmo de la búsqueda del vecino más cercano.

Es necesario hacer algunas observaciones acerca del algoritmo. En primer lugar, no garantiza soluciones factibles ya que puede necesitar más de n vehículos para visitar a todos los clientes. Además, el algoritmo trata de minimizar la distancia y con ello el coste total de una jornada. Sin embargo, puede haber rutas muy cortas en distancia, pero que no sean las más baratas. Otro aspecto a resaltar es que el algoritmo recarga siempre con la tecnología más rápida en cada caso (para ganar en factibilidad), para referirse a la tasa de recarga más rápida disponible, se ha utilizado la notación ρ_{t^*}. Cabe destacar que en cada paso por una estación, se recarga al máximo las baterías. Esta idea debe ir acompañada de un algoritmo que, una vez decididas las rutas, sepa ajustar las cantidades a recargar de forma óptima. Se propone el algoritmo de la sección 3.3.1 (heurísticas de mejora) pensado para ajustar las cantidades de recarga.

Cuando el algoritmo greedy devuelve soluciones factibles, éstas no suelen ser demasiado buenas (comparadas con las soluciones que proporcionan otros métodos desarrollados). Sin embargo, las soluciones obtenidas con el algoritmo de la "búsqueda del vecino más cercano" pueden ser sometidas a algoritmos de mejora (véase la sección 3.3) para obtener mejores resultados.

A continuación se presenta la solución al caso de estudio obtenida con el algoritmo de la "búsqueda del vecino más cercano":

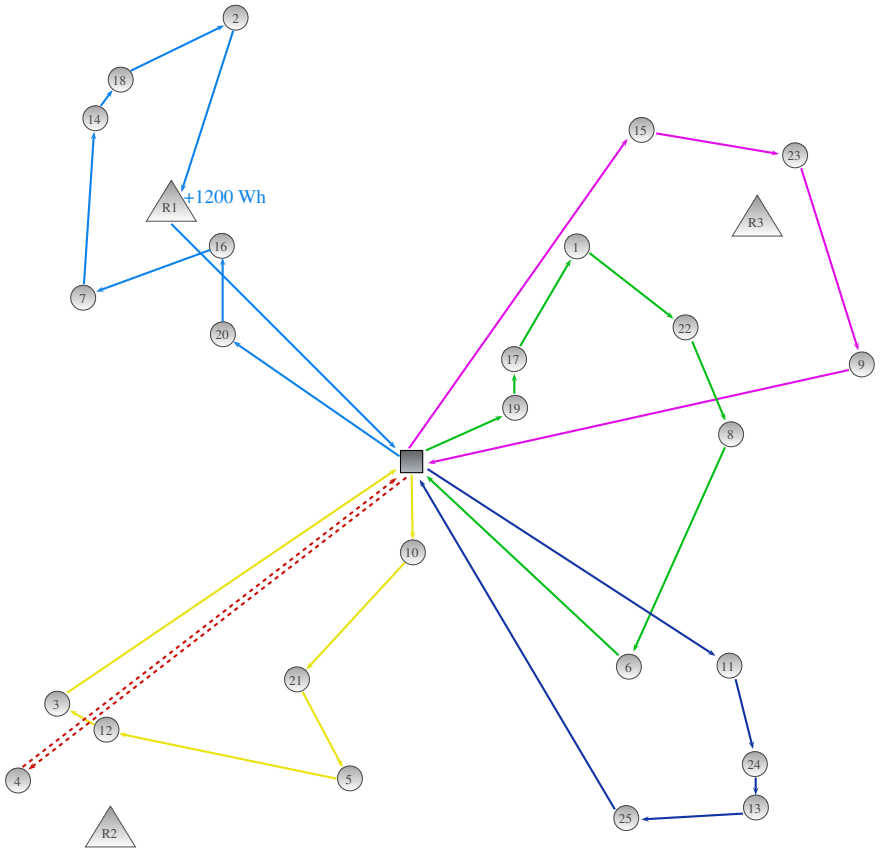

Figura 3.2: Solución de la "búsqueda del vecino más cercano". Coste: 29.42 €.

Como se puede apreciar en la figura 3.2, con 5 vehículos no ha sido suficiente y el algoritmo ha necesitado un vehículo adicional para visitar al cliente número 4.

3.2.2 Algoritmo de agrupamiento

Este algoritmo, desarrollado en [2], ha sido modificado y adaptado al nuevo problema. Su idea es muy básica: en un primer paso se decide cuáles serán los clientes visitados por cada vehículo y a continuación, en qué orden. La decisión de cómo agrupar los clientes es un punto clave de esta heurística y del que dependerá la calidad de las soluciones obtenidas.

Paso I: Agrupamiento (Clustering)

Se pretende agrupar a los clientes en n grupos o *clusters* para que cada grupo sea visitado por uno de los n vehículos. Esta agrupación se realizará con un algoritmo jerárquico de estrategia aglomerativa. Inicialmente cada cliente pertenece a su propio cluster (al principio hay tantos grupos como clientes). En cada iteración del algoritmo dos de los grupos se fusionan formando uno solo, de forma que el número de clusters se reduce en una unidad. El proceso finaliza cuando el número de clusters es el deseado.

En cada iteración, los grupos que son unificados son los más cercanos según la distancia definida entre clusters. Al comienzo del algoritmo, como cada grupo tiene un solo cliente, las distancias entre los grupos son las distancias entre clientes. Sin embargo, cada vez que se fusionan dos grupos hay que definir la distancia del nuevo grupo a todos los demás. Esto se puede hacer siguiendo varios criterios (cada uno dará lugar a diferentes agrupapciones y por lo tanto a soluciones diferentes). Supóngase que se acaban de unificar los grupos A y B formando el grupo AB. La nueva distancia entre el grupo AB y un grupo anterior C se puede definir de las siguientes formas:

* Distancia mínima: $d(AB, C) = \min(d(A, C), d(B, C))$.
* Distancia máxima: $d(AB, C) = \max(d(A, C), d(B, C))$.
* Distancia media: $d(AB, C) = \frac{|A|}{|A|+|B|} d(A, C) + \frac{|B|}{|A|+|B|} d(B, C)$.

Merece la pena resaltar que la suma de demandas de los clientes de un mismo grupo no debe superar Q (la capacidad de carga de mercancía de un vehículo). Esto puede hacer que el algoritmo jerárquico se "atasque" antes de haber llegado al número de clusters deseado porque no se pueda fusionar ningún par de grupos (al violar la condición de carga). Más adelante se verá cómo resolver este inconveniente.

A continuación se presenta el pseudocódigo de este algoritmo de agrupamiento:

Entrada: Datos del problema.
Salida: Un conjunto de n grupos de clientes: $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$.

Definir $\mathcal{G} \leftarrow \{g_1, g_2, \ldots, g_{\#\mathcal{C}}\}$, donde $g_i = \{i\}$ $\forall i \in \mathcal{C}$. (Un grupo por cada cliente)
Definir la matriz de distancias entre grupos:
$D(g_i, g_j) \leftarrow d_{ij}$ si $i \neq j$ $\forall i, j \in \mathcal{C}$.
$D(g_i, g_i) \leftarrow \infty$ $\forall i \in \mathcal{C}$.
mientras $\#\mathcal{G} > n$ **hacer**
    Sean i^* y j^* tales que $D(g_{i^*}, g_{j^*}) = \text{mín}_{i,j}\{D(g_i, g_j)\}$.
    si $\sum_{i \in g_{i^*}} c_i + \sum_{j \in g_{j^*}} c_j \leq Q$ (Se pueden unir los grupos) **entonces**
        $g_{i^*} \leftarrow g_{i^*} \cup g_{j^*}$
        Eliminar g_{j^*} de $\mathcal{G}$.
        para $k \in \{1, \ldots, \#\mathcal{G}\}$ $(k \neq i^*, j^*)$ **hacer**
            $D(g_k, g_{i^*}) \leftarrow \text{mín}\,(D(g_k, g_{i^*}), D(g_k, g_{j^*}))$.
            $D(g_{i^*}, g_k) \leftarrow D(g_k, g_{i^*})$. (Se actualiza la matriz de distancias)
            Borrar la fila y la columna j^*-ésimas de D.
        fin para
    si no
        si $D(g_{i^*}, g_{j^*}) == \infty$ **entonces**
            Salir del bucle **mientras**. (No se pueden unir más grupos)
        fin si
        $D(g_{i^*}, g_{j^*}) \leftarrow \infty$. (Ya no se podrá unir esos grupos)
        $D(g_{j^*}, g_{i^*}) \leftarrow \infty$.
    fin si
fin mientras

Al salir del bucle **mientras**, es posible que el cardinal de $\mathcal{G}$ sea mayor que n. Esto sucede si es imposible fusionar cualquier par de grupos (pues se sobrepasaría la capacidad de carga de los vehículos). El siguiente pseudocódigo se encarga de este problema:

mientras $\#\mathcal{G} > n$ **hacer**
    Sea k^* tal que g_{k^*} es el cluster de menor cardinal.
    para $i \in g_{k^*}$ **hacer**
        Sea $d_k = \text{mín}_{j \in g_k}\{d_{ij}\}$ $\forall k \in \{1, \ldots, \#\mathcal{G}\}$ con $k \neq k^*$.
        Sea $\alpha(k)$ una ordenación ascendente de los índices $\{1, \ldots, \#\mathcal{G}\} \setminus k^*$ según d_k.
        para $k \in \{1, \ldots, \#\mathcal{G}\} \setminus k^*$ **hacer**
            si $c_i + \sum_{j \in g_{\alpha(k)}} c_j \leq Q$ **entonces**
                Añadir i a $g_{\alpha(k)}$ y borrarlo de g_{k^*}.
            fin si
        fin para
    fin para
    Eliminar g_{k^*} de $\mathcal{G}$.
fin mientras
devolver $\mathcal{G}$.

Algoritmo 2: Paso I del algoritmo de agrupamiento (distancia mínima).

Se localiza el cluster más pequeño y se reparten sus individuos entre los demás cluster según un criterio de proximidad. Esto se hace hasta que el número de clusters es igual a n.

Geométricamente, el criterio de distancia mínima tiende a crear conjuntos *alargados*, mientras que el de distancia máxima suele crear conjuntos más *redondeados*.

Paso II: TSP

Una vez decidido cuáles serán los clientes que visitará cada vehículo, se debe determinar en qué orden. Éste es justo el problema del viajante (TSP), que se resolverá de forma aproximada.

El TSP es sin duda el problema más estudiado en Optimización Combinatoria, y existen diferentes algoritmos que encuentran soluciones aproximadas. En este caso se ha optado por partir de una solución aleatoria y aplicarle un método 2-óptimo. Si lo anterior se realiza desde diferentes soluciones iniciales se pueden conseguir buenos resultados. El pseudocódigo para resolver el TSP de cada vehículo es el siguiente:

Entrada: Una agrupación de los clientes $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$, un número natural L.
Salida: Una solución "razonablemente buena" (no necesariamente factible).

Inicializar $S \leftarrow \{\}$.
para $k \in \{1, \ldots, n\}$ **hacer**
    $mejor\_valor \leftarrow \infty$.
    para $i \in 1, \ldots L$ (se elegirá la mejor solución de las L calculaldas) **hacer**
        $\mu \leftarrow permutacion\_aleatoria(g_k)$.
        $\mu \leftarrow dos\_opt(\mu)$. (Se aplica un algoritmo 2-óptimo)
        si $f(\mu) < mejor\_valor$ **entonces**
            $mejor\_valor \leftarrow f(\mu)$.
            $\mu_k \leftarrow \mu$.
        fin si
    fin para
    Añadir μ_k a S.
fin para
devolver S.

Algoritmo 3: Paso II del algoritmo de agrupamiento.

En este punto, la única restricción que se verifica con seguridad es la de demanda. No se ha atendido a las restricciones de tiempo ni de energía (aún no se ha visitado ninguna estación de recarga y puede que sea necesario).

Paso III: Conseguir factibilidad de energía:

En este paso se pretende respetar las restricciones de energía, es decir, que ningún vehículo se quede sin ella. Esto se puede hacer con el procedimiento detallado en el siguiente pseudocódigo:

Entrada: Una solución $S = \{\mu_1, \ldots, \mu_n\}$ (posiblemente no factible).
Salida: Otra solución en la que la restricción de energía queda satisfecha.

mientras haya rutas en S que se queden sin energía **hacer**
  para $k \in \{1, \ldots, n\}$ **hacer**
    si El vehículo de μ_k se queda sin energía en algún momento **entonces**
      Sea p la última posición de μ_k a la que se ha podido llegar sin agotar la batería.
      Añadir a μ_k la estación de recarga r^* en la posición $p^* \in \{1, \ldots, p\}$ de forma que la distancia total recorrida aumente en lo mínimo posible (y recargar al máximo).
    fin si
  fin para
  Actualizar $S \leftarrow \{\mu_1, \ldots, \mu_n\}$.
fin mientras
devolver S.

Algoritmo 4: Paso III del algoritmo de agrupamiento.

En el Trabajo de Fin de Grado [2], la heurística finalizaba en este punto. En el problema abordado ahora, se considera la posibilidad de recargar las baterías parcialmente. Por eso sería conveniente aplicarle la heurística de mejora 3.3.1 (descrita en la siguiente sección) a la solución obtenida hasta el momento con el fin de ajustar las cantidades de energía recargada de forma óptima.

Del mismo modo que en la heurística de la "búsqueda del vecino más cercano", es posible que la solución devuelta por este algoritmo no sea factible, en este caso debido a la restricción de tiempo. Sin embargo, al haber intentado minimizar la distancia de cada ruta y bajo la hipótesis de que el tiempo entre rutas es directamente proporcional a la distancia, se espera que se haya respetado la cota T_{max}.

Se ha comprobado experimentalmente que, en los problemas generados aleatoriamente con el software desarrollado, esta heurística devuelve soluciones factibles en la mayoría de los casos. En algunas ocasiones la función objetivo es comparable a la obtenida mediante técnicas metaheurísticas (véase el capítulo 5).

La figura 3.3 recoge la solución obtenida mediante al algoritmo de agrupamiento con distancia mínima:

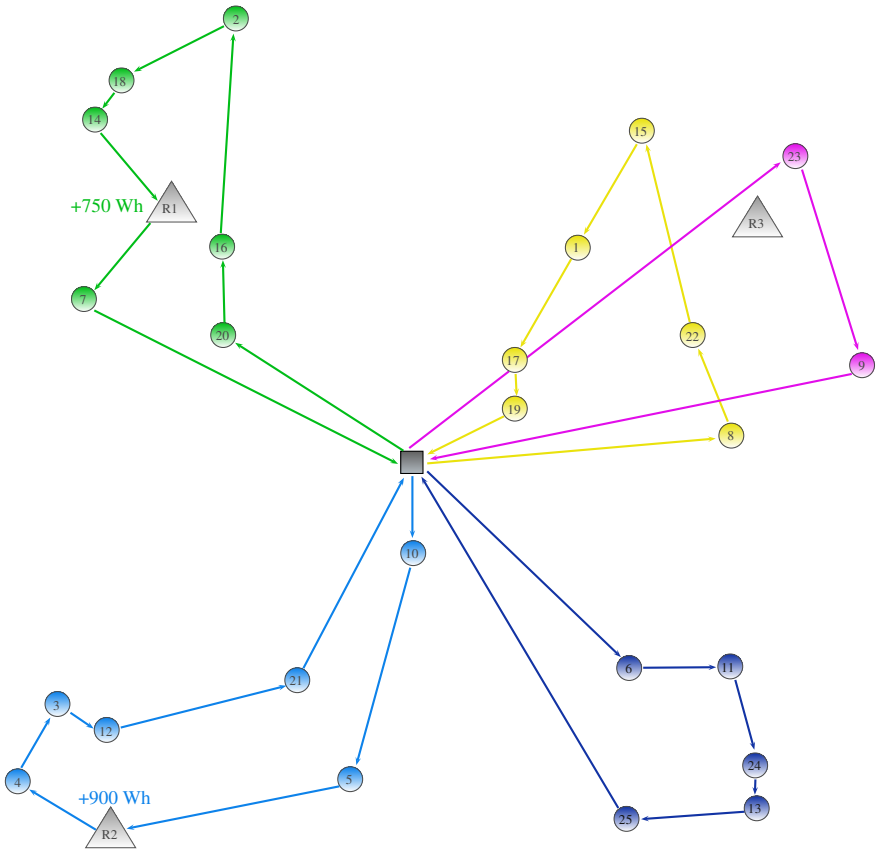

Figura 3.3: Solución del "algoritmo de agrupamiento". Coste: 27.57 €.

3.2.3 Heurística basada en programación matemática

El mayor problema a la hora de resolver el *GVRP-MTPR* con programación matemática (por ejemplo con los modelos propuestos en las secciones 2.2.1 y 2.2.2) es la elevada cantidad de variables binarias que aparecen en los modelos. Esto hace que en problemas "relativamente grandes", tras varias horas de ejecución, ni siquiera se llegue a una solución entera factible. Además, la memoria se convierte en un problema, pues el archivo que almacena la información del árbol de nodos a examinar puede llegar a ocupar varios GB. Este hecho también lo comentan Schneider, Stenger y Goeke en [26], que no pueden resolver en 2 horas de CPU con CPLEX, problemas de 15 clientes e incluso algunos casos más pequeños de 10 clientes.

Estos inconvenientes hacen que se descarte la programación matemática para resolver problemas de esta envergadura (por lo menos de forma exacta). Sin embargo, si se consigue tener un modelo consistente con menos variables binarias, a cambio de perder la garantía de convergencia a la solución óptima, se puede llegar a buenas soluciones del problema planteado. En concreto, en esta sección y como heurística adicional para abordar el *GVRP-MTPR*, se propone una reducción de variables binarias en el modelo de 2.2.2.

Unas de las variables binarias consideradas en el modelo eran las llamadas x_{ij}, definidas

como sigue:

$$x_{ij} = \begin{cases} 1 & \text{si algún vehículo viaja desde el cliente } i \text{ hasta el } j \\ 0 & \text{en caso contrario.} \end{cases} \qquad \forall i, j \in \mathcal{V},\ i \neq j$$

Donde $\mathcal{V}$ era el conjunto de clientes y el depósito.

Se propone no considerar todas las variables x_{ij}, sino solo las "mejores" en el sentido de que corresponden a arcos más cortos. Deben considerarse todas las conexiones con el depósito, es decir, x_{iD} y x_{Di} existirán $\forall i \in \mathcal{C}$. Pero, dado un cliente $i \in \mathcal{C}$, sólo existirán las variables x_{ij} si $j \in \mathcal{C}$ es uno de los K clientes más cercanos a i.

La figura 3.4 presenta la solución obtenida mediante esta algoritmo con $K = 9$:

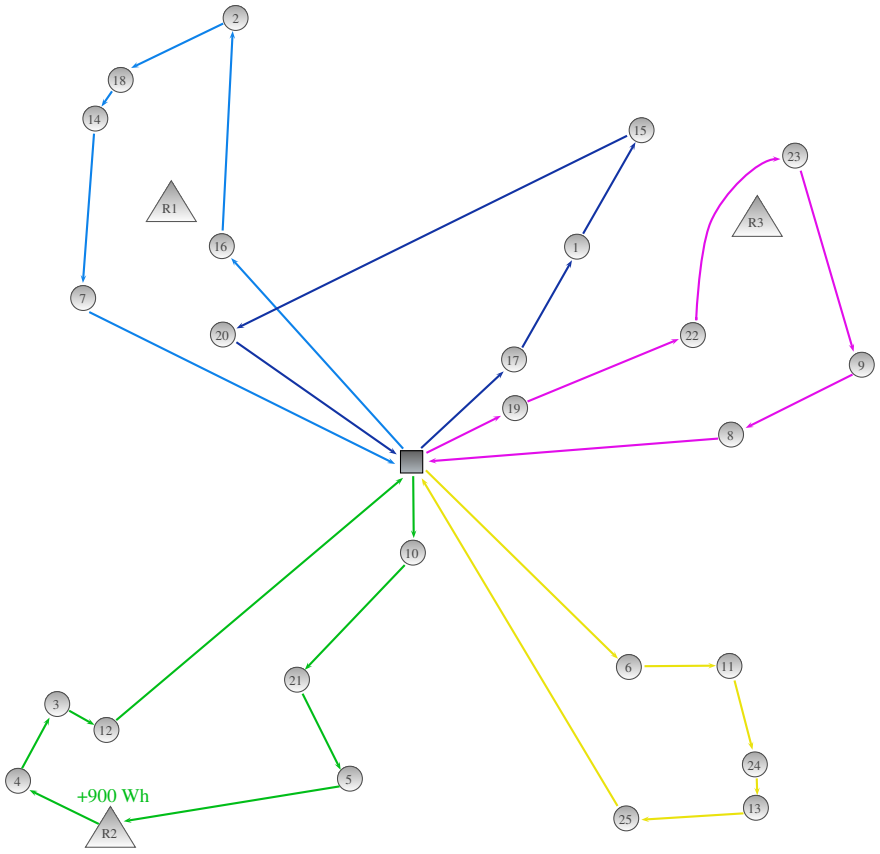

Figura 3.4: Solución de la "heurística de programación matemática". Coste: 25.25 €.

3.3 Heurísticas de mejora

Estas heurísticas tratan de mejorar soluciones ya construidas. En esta sección se proponen dos tipos de heurísticas de mejora. La primera trata de explotar la idea de que se puede decidir

cuánta energía recargar en cada estación (no es necesario recargar a tope). A continuación se proponen otras heurísticas de mejora que se basan en abordar el TSP asociado a cada una de las rutas.

3.3.1 Algoritmo de ajuste de recargas

Una vez que se ha decidido cómo serán las rutas y en qué momento se realizarán las recargas, es necesario ajustar adecuadamente las cantidades a recargar en cada estación. Esto se puede hacer resolviendo el modelo de programación matemática propuesto (nótese que las variables binarias estarían fijadas por lo que se trata de un problema de programación lineal continua, rápidamente resoluble).

A continuación se propone un algoritmo alternativo que resuelve de forma óptima este apartado (en caso de devolver una solución factible).

Entrada: Datos del problema y una ruta de una solución μ.
Salida: Un conjunto $\mathcal{E}$ de números (tantos como estaciones de recarga en μ más uno).

si (número de estaciones en μ) $= 0$ **entonces**
    $\mathcal{E} = \{g \cdot \text{longitud}(\mu)\}$
    devolver
si no
    $\mathcal{E} = \{E\}$
fin si

para $k \in$ estaciones(μ) **hacer**
    Actualizar *energia*, la energía disponible al llegar a la estación k.
    Sea k' la siguiente estación más barata a k en μ (o el depósito).
    si $g \cdot \text{longitud}(\mu_k^{k'}) \leq E$ **entonces**
        $\mathcal{E} \leftarrow \mathcal{E} \cup (g \cdot \text{longitud}(\mu_k^{k'}) - energia)$ (lo justo hasta la siguiente más barata).
    si no
        $\mathcal{E} \leftarrow \mathcal{E} \cup (E - energia)$ (lo máximo).
    fin si
fin para

Algoritmo 5: Algoritmo de ajuste de recargas

El algoritmo anterior se aplicará en cada ruta de una solución dada. A continuación se eliminarán de las rutas las estaciones en las que se ha decidido recargar 0 unidades de energía, lo cual puede suceder.

Merece la pena hacer algunas consideraciones. En primer lugar, siempre se recargará con la energía más barata disponible en cada estación. Esto hace que no se tenga en cuenta la

restricción de tiempo de las rutas, la cual puede ser violada. Sin embargo, si no se sobrepasa el tiempo en las rutas, el ajuste de recargas realizado resulta óptimo.

La idea detrás del algoritmo es una estrategia "greedy". Se recarga siempre el mínimo entre lo máximo posible y lo justo hasta llegar a una estación más barata que la actual. Nótese que si existe alguna estación de recarga en el trayecto, lo óptimo es partir con la batería llena (puesto que la tecnología inicial es la más barata).

3.3.2 Algoritmos k-óptimos

Otra forma de mejorar una solución del *GVRP-MTPR* es mejorar cada una de las rutas individualmente, abordando el problema del TSP. Son famosas las heurísticas de búsqueda local utilizadas para resolver este problema, como los algoritmos k-óptimos.

Dada una solución al TSP, es decir, un ciclo de n vértices, se consideran todos los subconjuntos de k aristas. Para cada subconjunto, se eliminan dichas aristas y se comprueba si se pueden recombinar los caminos resultantes (añadiendo k nuevas aristas) volviendo a formar un ciclo con mejor función objetivo. En caso afirmativo, el nuevo ciclo remplaza al primero y se reinicia el proceso.

En la práctica, los algoritmos k-óptimos se suelen utilizar con $k = 2, 3$ pues el número de subconjuntos a examinar crece exponencialmente con k. Además, un inconveniente que presentan es que no es posible conocer a priori el número de mejoras que se realizarán en el algoritmo hasta llegar a un ciclo k-óptimo (un ciclo que no se puede mejorar con intercambios de k aristas).

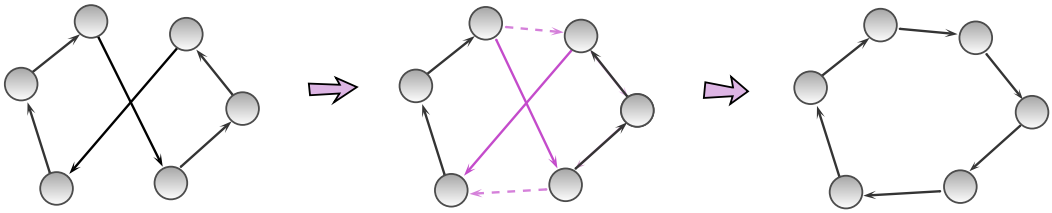

Figura 3.5: Movimiento 2-óptimo".

Algoritmo de Lin-Kernighan

En 1973, Shen Lin y Brian Kernighan [19], desarrollaron una heurística para el TSP que ofrece muy buenos resultados en la práctica. En esencia, el algoritmo es un método k-óptimo, con dos características adicionales:

- ◇ El valor de k es variable.
- ◇ Es posible que no se utilicen inmediatamente los intercambios de mejora localizados.

Hasta 1989 se consideró la mejor herramienta para abordar el TSP, sin embargo a día de hoy se considera mejor el algoritmo *Chained Lin-Kernighan*, una modificación de éste.

3.4 Cotas inferiores del problema

Al haber optado por la utilización de algoritmos aproximados para resolver el *GVRP-MTPR*, sería ideal poder estimar cómo de cerca (o de lejos) de la solución óptima están las soluciones obtenidas. Esta tarea se puede realizar mediante la búsqueda de cotas inferiores, de valores con seguridad menores que la mejor solución existente, sin embargo resulta realmente difícil encontrar buenas cotas.

En general, si se conoce una cota inferior del problema $\underline{f}$ para una solución $\mathbf{x}_0$ cuya función objetivo es $f(\mathbf{x}_0)$, su porcentaje de desviación relativa al óptimo es a lo sumo

$$\left(\frac{f(\mathbf{x}_0)}{\underline{f}} - 1\right) \cdot 100$$

Existen algoritmos que, *a priori*, aseguran una solución con desviación relativa acotada. Un ejemplo es el algoritmo heurístico de Christofides, que proporciona soluciones para el TSP como mucho a un 50 % del óptimo. Se trata de un caso excepcional y de una cota bastante débil. En general, este tipo de cotas se obtienen mediante relajaciones del problema original.

3.4.1 Relajación lineal

Sea el problema de programación lineal entera

$$\begin{array}{ll} \text{mín}_x & c^{\,T} x \\ s.a. & Ax \leq b \\ & x \geq 0 \text{ enteras.} \end{array}$$

Su problema de relajación lineal asociado es

$$\begin{array}{ll} \text{mín}_x & c^{\,T} x \\ s.a. & Ax \leq b \\ & x \geq 0. \end{array}$$

Esta relajación no suele ofrecer muy buenos resultados. En la práctica resulta más interesante la relajación lagrangiana.

3.4.2 Relajación lagrangiana

Sea el problema de programación lineal entera

$$\begin{array}{ll} \text{mín}_x & c^{\,T} x \\ s.a. & Ax \leq b \\ & Dx \leq e \\ & x \geq 0 \text{ enteras.} \end{array}$$

Supongamos que las restricciónes $Ax \leq b$ son las "restricciones difíciles" y sin embargo $Dx \leq e$ son "fáciles", por ejemplo porque la matriz D es totalmente unimodular. La idea que reside bajo la relajación lagrangiana es pasar las restricciones difíciles a la función objetivo.

Se define la relajación lagrangiana (de parámetro $\lambda > 0$) del problema original, como sigue:

$$\begin{array}{rll} L(\lambda) = \text{mín}_x & c^{\,T} x + \lambda^T (Ax - b) \\ s.a. & Dx \leq e \\ & x \geq 0 \text{ enteras.} \end{array}$$

Como se puede apreciar, se pretende penalizar la violación de las restricciones "difíciles", sin embargo éstas ya no aparecen como restricciones del problema.

Se puede comprobar que el valor $L(\lambda)$ es una cota inferior del óptimo del problema entero original para cualquier elección de parámetros $\lambda \geq 0$. Esto da lugar a plantearse el problema dual lagrangiano

$$LD = \max_{\lambda \geq 0} L(\lambda)$$

que es equivalente a buscar la elección de parámetros que permite hacer más grande la cota inferior.

Teorema: (para problemas de minimización)
Sea LP la cota inferior proporcionada la relajación lineal y sea LD la cota inferior del problema dual lagrangiano. Sea F^* la solución óptima al problema de programación lineal entera original. Se verifica que,

$$LP \leq LD \leq F^*$$

Demostración:
La segunda desigualdad es trivial, pues al resolverse la relajación de un problema de minimización se obtiene siempre una solución menor o igual que la solución del problema original. Se probará que $LP \leq LD$:

$$
\begin{aligned}
LP = \min_{x} \left\{ c^{\,T} x \mid Ax \leq b, Dx \leq e,\ x \geq 0 \right\} &= \text{(dualidad)} \\
= \max_{\lambda, y} \left\{ b^{\,T} \lambda + e^{\,T} y \mid A^T \lambda + D^T y \leq c,\ \ y \leq 0,\ \lambda \leq 0 \right\} &= \\
= \max_{\lambda \leq 0} \left\{ b^{\,T} \lambda + \max_{y} \left\{ e^{\,T} y \mid D^T y \leq (c - A^T \lambda),\ \ y \leq 0 \right\} \right\} &= \text{(dualidad)} \\
= \max_{\lambda \leq 0} \left\{ b^{\,T} \lambda + \min_{x} \left\{ (c^{\,T} - \lambda^T A) x \mid Dx \leq e,\ \ x \geq 0 \right\} \right\} &\leq \\
\leq \max_{\lambda \leq 0} \left\{ b^{\,T} \lambda + \min_{x} \left\{ (c^{\,T} - \lambda^T A) x \mid Dx \leq e,\ \ x \geq 0 \ \text{ enteras} \right\} \right\} &= \\
= \max_{\lambda \geq 0} \left\{ \min_{x} \left\{ c^{\,T} x + \lambda^T (Ax - b) \mid Dx \leq e,\ \ x \geq 0 \ \text{ enteras} \right\} \right\} &= LD
\end{aligned}
$$

El teorema permite afirmar que la relajación lagrangiana nunca ofrece cotas peores que la relajación lineal, lo cual anima a utilizar la primera como método de estimación de cotas inferiores.

3.4.3 Búsqueda de cotas sobre el grafo

En esta sección se pretende acotar el problema desde un punto de vista menos abstracto, sin utilizar exclusivamente los modelos de programación matemática. En el *GVRP-MTPR* hay dos tipos de coste en la función objetivo:

- ◇ Costes fijos por recarga.
- ◇ Costes por cantidad de energía recargada.

Se puede intentar encontrar una cota para ambos costes por separado y unirlas en una sola.

Cotas sobre el número de vehículos:

Nótese que cada vehículo utilizado supone por lo menos una recarga (la realizada en el depósito durante la noche). Para estimar el número total de recargas realizadas puede ser útil conocer cuál es el mínimo número de vehículos necesarios.

Dado un conjunto de demandas $\delta_i \ \forall i \in \mathcal{C}$ y vehículos de capacidad Q, hay que determinar cuál es el menor número de vehículos, $\underline{n}$, necesarios para cargar todas las demandas de una sola vez. Éste es el conocido *Bin Packing Problem*, que resulta ser $\mathcal{NP}-$completo. Sin embargo se pueden encontrar cotas inferiores de este problema. La cota más sencilla es la siguiente:

$$\underline{n} \approx \left\lceil \frac{\sum_{i \in \mathcal{C}} \delta_i}{Q} \right\rceil$$

Cota sobre la cantidad total de energía necesitada:

La energía necesitada es directamente proporcional (con constante de proporcionalidad g) a la distancia total recorrida. Esto permite acotar la mínima energía necesitada a partir de la distancia mínima recorrida. Existen diferentes técnicas para acotar inferiormente la distancia recorrida en un problema de rutas, una de ellas podría ser utilizar la cota de Held-Karp para el TSP, aunque en [18] se pueden encontrar estrategias mejores, específicas para problemas de varios vehículos.

Supóngase que en una solución se utilizan exactamente n_0 vehículos y que se ha estimado que la distancia total recorrida (utilizando n_0 vehículos) no puede ser menor que una cota $\underline{D}_{n_0}$. En ese caso, la energía total necesitada será mayor o igual que $g\underline{D}_{n_0}$ y una cota inferior para el número mínimo de recargas realizadas sería:

$$\underline{N}_{n_0} = \text{mín}\left(n_0, \left\lceil \frac{g\underline{D}_{n_0}}{E} \right\rceil\right)$$

donde E denota la capacidad de las baterías de los vehículos. Además, el gasto en energía recargada no podría ser menor que

$$\underline{G}_{n_0} = \begin{cases} c^* \ g\underline{D}_{n_0} & \text{si} \quad g\underline{D}_{n_0} \leq n_0 E \\ c^* \ n_0 E + c^{**}\left(g\underline{D}_{n_0} - n_0 E\right) & \text{si} \quad g\underline{D}_{n_0} > n_0 E \end{cases}$$

donde se recuerda que c^* es el coste por unidad de energía de recargar en el depósito y se establece que c^{**} es el precio de la tecnología alternativa al depósito más barata.

Cota para el GVRP-MTPR:

Las cotas anteriores conducen a la cota para el *GVRP-MTPR*:

$$\min_{n_0 \in [\underline{n},\ n]} \left\{ c_f \underline{N}_{n_0} + \underline{G}_{n_0} \right\}$$

expresada en unidades monetarias.

Capítulo 4

Metaheurísticas

Existen algoritmos más sofisticados que los algoritmos heurísticos clásicos que se basan en la combinación de diferentes técnicas heurísticas para conseguir una exploración más completa de la región de búsqueda. Son conocidos como algoritmos metaheurísticos o *metaheurísticas*, término procedente del griego combina el prefijo *meta* (μετά), que significa «más allá» con el término *heurística* (εὑρίσκειν), que significa «encontrar, descubrir». Comenzaron a desarrollarse a partir de los años sesenta y, como su nombre indica, pretenden ir más allá que los algoritmos heurísticos intentando "escapar" de *óptimos locales* para llegar hasta el *óptimo global* del problema.

Las **metaheurísticas** *son una clase de métodos aproximados, que están diseñados para atacar problemas de optimización combinatoria difíciles, para los que las heurísticas clásicas fracasaron en ser efectivas y eficientes. Las metaheurísticas proporcionan marcos generales que permiten crear nuevos híbridos combinando diferentes conceptos derivados de heurísticas clásicas, inteligencia artificial, evolución biológica, sistemas neuronales y mecánica estadística.* (Osman y Kelly, 1996)

Al igual que los algoritmos heurísticos, las metaheurísticas no pueden garantizar la solución óptima a un problema puesto que no realizan una búsqueda exhaustiva (ni siquiera implícitamente) de la región factible del problema, pero se espera que devuelvan "buenas" soluciones en un tiempo razonable. Por ello, se dice que estas técnicas son "ciegas", no saben si han llegado a la solución óptima, de forma que es necesario establecer un criterio de parada, habitualmente un número máximo de iteraciones o un tiempo máximo de ejecución del algoritmo.

No son algoritmos rígidos, son generales y deben adaptarse al problema particular que se esté estudiando. Tienen un gran número de aplicaciones para resolver problemas complejos en campos como la logística, la electrónica, la medicina, las telecomunicaciones, la informática...

4.1 Definiciones previas

Se comenzará definiendo formalmente el concepto de problema de optimización:

Definición: Un *problema de optimización* es una terna $(f, \mathcal{X}, \mathcal{F})$ donde $\mathcal{X}$ es el espacio de soluciones del problema y $\mathcal{F} \subseteq \mathcal{X}$ representa el conjunto de soluciones factibles. Además, f es una función real llamada *función objetivo*.

$$f : \mathcal{X} \to \mathbb{R}.$$

Resolver el problema de optimización (en el caso de minimización) consiste en encontrar una solución $S^* \in \mathcal{F}$ de forma que

$$f(S^*) \leq f(S) \ \forall S \in \mathcal{F},$$

Muchas metaheurísticas, las llamadas "metaheurísticas de búsqueda por entornos", siguen la siguiente metodología: parten de una solución inicial (posiblemente aleatoria) y le aplican una ligera modificación, llegando a una nueva solución. Este proceso se repite de forma que el algoritmo va visitando diferentes soluciones hasta que se alcanza algún criterio de parada. Al final del algoritmo, se devuelve la mejor solución encontrada.

Algo común a todas las metaheurísticas (no solo a las de búsqueda por entornos) es que a veces se permite, bajo ciertas condiciones, aceptar cambios a peor, es decir, trabajar con soluciones peores que la mejor solución conocida. El objetivo de esto es poder "escapar" de los óptimos locales en la búsqueda del óptimo global.

Definición: Sea $(f, \mathcal{X}, \mathcal{F})$ un problema de optimización y sea $S \in \mathcal{X}$ una solución. Definimos el *vecindario de* S como el conjunto de soluciones próximas a S, en el sentido de que se pueden alcanzar con una pequeña modificación de S.

$$V(S) = \{S' \in \mathcal{X} : S' = m_i^{\alpha}(S) \text{ para ciertos } i, \alpha\}$$

donde m_i^{α} representa el movimiento de tipo i bajo las condiciones α.

De esta forma, el vecindario de una solución quedará determinado una vez que se hayan definido los tipos de movimientos a considerar m_i. Un ejemplo de movimiento para el *GVRP-MTPR* puede ser intercambiar el orden de dos clientes en una misma ruta y, en ese caso, las condiciones α deberían determinar qué dos clientes de la ruta serían intercambiados. En la sección 4.3 se definirá con más exactitud cuáles serán los movimientos considerados para el *GVRP-MTPR* de forma que quedará establecido el vecindario de una solución.

Para que los algoritmos sean eficaces y eficientes es necesario considerar una serie de propiedades con las que deberían contar:

- ◇ **Comunicación**: Es deseable que desde cualquier solución se pueda alcanzar cualquier otra solución de $\mathcal{X}$, realizando movimientos y pasando de una solución vecina a otra, preferiblemente en pocos pasos.
- ◇ **Diversificación:** Se debe facilitar los movimientos entre áreas muy diferentes del espacio de soluciones.
- ◇ **Intensificación:** Está relacionada con el esfuerzo empleado en la búsqueda dentro de una región concreta del espacio, con la mejora de una solución con pequeñas variaciones.

Los dos últimos conceptos parecen contradictorios y de alguna forma lo son. Para que el algoritmo sea bueno, éste debe establecer un balance adecuado entre ambos, por ejemplo dándole más importancia a uno u otro en función de la etapa del algoritmo.

Una forma de mejorar la comunicación y la diversificación es que los algoritmos nos se muevan sólo por soluciones factibles, que puedan hacerlo también por soluciones infactibles, con un coste.

Definición: Dado el problema de optimización $(f, \mathcal{X}, \mathcal{F})$, podemos considerar un problema similar $(F, \mathcal{X}, \mathcal{X})$ y se define la *función de penalización* F (en minimización) , como

$$F(S) = f(S) + \beta g(S)$$

donde $\beta > 0$ y $g(S)$ es una función que mide el grado de infactibilidad de S, es decir,

$$g(S) \begin{cases} = 0 & \text{si } S \in \mathcal{F} \\ > 0 & \text{en caso contrario.} \end{cases}$$

Además, $g(S)$, tomará valores más grandes cuanto mayor sea la infactibilidad de S.

El parámetro β puede ser dinámico, de forma que haya etapas en el algoritmo en las que se penalice menos la infactibilidad (para favorecer la diversificación) y otras en las que se penalice más (para favorecer la intensificación y recuperar la factibilidad).

4.2 Descripción de algunas metaheurísticas

En este trabajo se han implementado tres algoritmos metaheurísticos. Los dos primeros, el *Recocido Simulado* y la *Búsqueda Tabú* han sido modificados a partir de los desarrollados en el TFG [2] adaptándolos al nuevo problema. El tercero, la *Búsqueda por Entornos Variables* es, sin embargo, propio de este trabajo.

4.2.1 Recocido simulado

El algoritmo de *Recocido Simulado* o *Simulated Annealing (SA)* fue formulado por Scott Kirkpatrick, C. Daniel Gelatt y Mario P. Vecchi en 1983 [16] y fue uno de los primeros algoritmos metaheurísticos.

La idea bajo el algoritmo es la imitación del proceso de recocido del acero, una técnica que consiste en calentar el metal para luego enfriarlo lentamente de manera que se modifiquen sus propiedades. El calor hace que los átomos aumenten su energía de forma que puedan moverse y abandonar sus posiciones iniciales (saliendo posiblemente de óptimos locales). El lento enfriamiento permite que se alcancen configuraciones estables (deseablemente de menor energía y por lo tanto más cercanas al óptimo global). Es importante que el enfriamiento sea suficientemente lento para que las partículas puedan reconfigurarse adecuadamente.

El algoritmo parte de una solución inicial (aleatoria u obtenida mediante algún procedimiento) y en cada iteración se elige una solución vecina aleatoriamente. La vecina reemplazará a la primera con probabilidad 1 si tiene mejor valor objetivo y con cierta probabilidad (estrictamente entre 0 y 1) en caso contrario.

La probabilidad de aceptar cambios a peor es proporcional a la temperatura del sistema en ese momento y a lo mala que sea la nueva solución. Cuanto peor sea la posible nueva solución, menos probable será de ser aceptada. Al mismo tiempo, cuanto mayor sea la temperatura del

sistema, mayor probabilidad habrá de aceptar cambios a peor.

Con el avance del algoritmo, la temperatura va descendiendo y con ella, la probabilidad de aceptar cambios a peor, de forma que si el número de iteraciones es suficiente, se podrá escapar de los óptimos locales y el proceso convergerá.

A continuación se presenta el pseudocódigo de una posible implementación del algoritmo:

Entrada: Una solución inicial S, una temperatura inicial T, un factor de enfriamiento r, siendo $0 < r < 1$ y un número natural L.
Salida: Una solución final "razonablemente buena".

mientras el sistema no esté congelado ($T > \varepsilon$) **hacer**
  para $i = 1$ **hasta** L **hacer**
    Elegir aleatoriamente $S' \in V(S)$.
    $\Delta \leftarrow F(S') - F(S)$.
    si $\Delta \leq 0$ **entonces**
      $S \leftarrow S'$.
    si no
      Generar $u \sim U(0,1)$. (Distribución uniforme)
      si $u \leq e^{\frac{-\Delta}{T}}$ **entonces**
        $S \leftarrow S'$.
      fin si
    fin si
  fin para
  $T \leftarrow rT$. (Se reduce la temperatura)
fin mientras
devolver la mejor solución encontrada.

Algoritmo 6: Algoritmo del recocido simulado.

Al principio del algoritmo, cuando la temperatura es "alta", se aceptan con frecuencia cambios a peor (fase de diversificación), mientras que cuando $T \approx 0$, prácticamente sólo se aceptan cambios a mejor (fase de intensificación).

4.2.2 Búsqueda tabú

Este algoritmo fue propuesto por Fred Glover en 1986 [11]. Sus principios básicos fueron elaborados en una serie de artículos de finales de los años 80 y posteriormente unificados en el libro "Tabu Search" en 1997 [12].

La idea básica del algoritmo, y la que le da nombre, es la consideración de ciertas soluciones prohibidas, *tabú*. Uno de los objetivos de esta idea es el de evitar ciclos. También se trata de una metaheurística de búsqueda por entornos, de forma que se comienza con una solución inicial y el algoritmo se va moviendo a la *mejor* solución de su vecindario hasta alcanzar un criterio de parada. Este proceso es sensible a la creación de ciclos, a volver a la misma solución en pocos pasos:

$$S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_0 \rightarrow S_1 \rightarrow \dots$$

El hecho de considerar soluciones tabú (en la versión más simple del algoritmo aquéllas que han sido visitadas recientemente) hace que no se produzcan estos ciclos y que se mejore la diversificación.

Existen versiones del algoritmo bastante complejas en las que se consideran varias listas tabú (memorias a corto y largo plazo), pero en este trabajo se ha desarrollado una versión más sencilla, aunque bastante competitiva. A continuación se presenta el pseudocódigo del algoritmo:

Entrada: Una solución inicial S.
Salida: Una solución final "razonablemente buena".

Se inicializa una lista tabú vacía.

mientras el criterio de parada no se satisfaga **hacer**
Elegir un subconjunto $V'(S) \subseteq V(S)$ de soluciones *no tabú*.
Sea $S' = \text{argmin}\{F(W) : W \in V'(S)\}$.
$S \leftarrow S'$ y se actualiza la lista tabú.

fin mientras
devolver la mejor solución encontrada.

Algoritmo 7: Algoritmo de la búsqueda tabú.

Se trata de una metaheurística que tuvo mucho éxito en la resolución de problemas de optimización y especialmente en aquéllos con aplicaciones en el mundo real. Su principio fundamental es la *acumulación de información* durante la búsqueda.

4.2.3 Búsqueda por Entornos Variables

La *Búsqueda por Entornos Variables* o *Variable Neighbourhood Searh (VNS)* es una metaheurística bastante reciente para resolver problemas de optimización cuya idea básica es el cambio sistemático de entorno dentro de una búsqueda local [13].

Se define un *entorno* de S como un subconjunto de $V(S)$ que se puede alcanzar a partir de S mediante uno de los movimientos definidos m_i. Por lo tanto, cada movimiento definido en nuestro problema dará lugar a un tipo de entorno. La metaheurística *VNS* consiste básicamente en cambiar la estructura de entornos de forma sistemática, para aprovechar los siguientes hechos:

- ◇ Un mínimo local con una estructura de entornos no tiene por qué serlo necesariamente con otra.
- ◇ Un mínimo global es mínimo local con todas las posibles estructuras de entornos.
- ◇ En muchos problemas los mínimos locales están relativamente próximos entre sí.

Atendiendo a los dos primeros hechos, es razonable utilizar varias estructuras de entornos para abordar un problema de optimización. Además, el último hecho (comprobado empíricamente) indica que los óptimos locales proporcionan información sobre el óptimo global, por ejemplo es posible que tengan características comunes, aunque no se pueda saber cuáles.

Existen numerosas variantes de este algoritmo, algunas de ellas son: VNS Descendente, VNS Reducida, VNS Básica, VNS General, VNS Anidada... En este documento se trabajará con la VNS Descendente, descrita a continuación.

Una búsqueda local descendente consiste esencialmente en determinar una solución mejor que la actual (de alguna forma) y elegirla para la siguiente iteración. Existen varias formas de hacer esta elección:

* Búsqueda exhaustiva: consiste en reemplazar la solución actual por la mejor de todas las soluciones del entorno considerado.
* Búsqueda ansiosa: es el extremo opuesto, acepta el primer movimiento de mejora encontrado.
* Estrategias intermedias: que no examinan exhaustivamente el entorno de la solución actual, pero tampoco aceptan la primera mejora.

La *Búsqueda por Entornos Variables Descendente* es una búsqueda local descendente en la que se cambia la estructura de entornos cada vez que se llega a un mínimo local. Dada una solución S, se establece que $V_i(S)$ es el entorno de S correspondiente al movimiento m_i. A continuación se presenta un posible pseudocódigo del algoritmo:

Entrada: Una solución inicial S.
Salida: Una solución final "razonablemente buena".
  $\Delta \leftarrow -1$.
  mientras $\Delta < 0$ (mientras se haya mejorado) **hacer**
    $i \leftarrow 1$.
    mientras $i \leq N$ (para cada tipo de entorno) **hacer**
      Encontrar la mejor solución S' de $V_i(S)$. (Exploración del entorno).
      $\Delta \leftarrow F(S') - F(S)$.
      si $\Delta < 0$ **entonces**
        $S \leftarrow S'$.
        $i \leftarrow 1$.
      si no
        $i \leftarrow i + 1$.
      fin si
    fin mientras
  fin mientras
  devolver S (la mejor solución encontrada)

Algoritmo 8: Algoritmo de la VNS descendente.

Nótese que, puesto que se vuelve a iniciar el recorrido de las estructuras de entornos cuando se produce un cambio de solución, cuando se está trabajando con el i-ésimo entorno de la solución actual S, dicha solución no ha podido ser mejorada con ninguno de los $(i-1)$ entornos anteriores. La solución final proporcionada por el algoritmo es un mínimo local con respecto a todos los entornos definidos.

4.3 Movimientos utilizados en el *GVRP-MTPR*

En esta sección se definirán con exactitud los diferentes movimientos que han sido considerados en el desarrollo de las técnicas metaheurísticas para resolver el *GVRP-MTPR*. Formalmente, una solución S del problema abordado será un conjunto de n circuitos (n es el número máximo de vehículos permitidos):

$$S = \{\mu_1, \mu_2, \ldots, \mu_n\}$$

Además, cada circuito μ_k está caracterizado por una sucesión *ordenada* de nodos que comienza por el depósito. Es necesario que cada cliente pertenezca a un circuito y sólo a uno.

La información relativa a las cantidades recargadas y la tecnología utilizada en cada caso debe ser almacenada de alguna forma, sin embargo no se tendrá en cuenta en esta sección.

En los apartados siguientes se ilustrará cada tipo de movimiento definido sobre el grafo de ejemplo de la figura 4.1. En ese caso se cuenta con $n = 3$ vehículos y la solución representada $S = \{\mu_1, \mu_2, \mu_3\}$ es la siguiente:

$$\mu_1 = (\mathcal{D}, 10, 3, 6, 5) \qquad \mu_2 = (\mathcal{D}, 8, 9, 2, 1, R1) \qquad \mu_3 = (\mathcal{D}, 4, 7, 11)$$

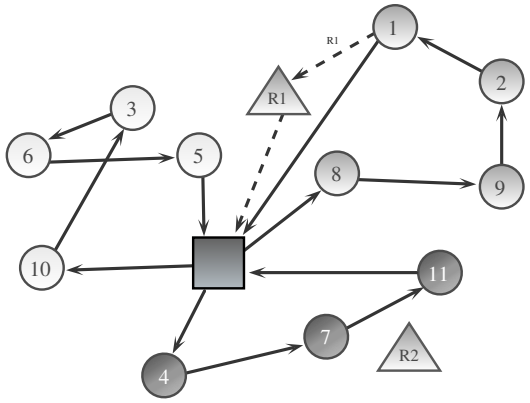

Figura 4.1: Grafo de ejemplo.

4.3.1 Intercambio 2-óptimo

Este movimiento es el mismo que se realiza en el algoritmo heurístico de mejora visto en 3.3.2. Dada una ruta μ_k se seleccionan dos arcos no consecutivos, se eliminan y se reconecta

el circuito de otra forma (sólo existe otra forma posible). Los arcos eliminados están caracterizados por dos índices $i, j \in \{1, \ldots, \#\mu_k - 1\}$ y para que no sean consecutivos debe ser $i+2 \leq j$.

La nueva solución es idéntica a la anterior salvo en μ'_k, que queda:

$$\mu'_k = \{\mu_{k(1,\ldots,i)},\ \underbrace{\mu_{k(j)},\ \mu_{k(j-1)},\ \mu_{k(j-2)}, \ldots,\ \mu_{k(i+1)}}_{\text{camino } \mu_{k(i+1,\ldots,j)} \text{ invertido}},\ \mu_{k(j+1,\ldots,\#\mu_k)}\}$$

En la notación utilizada, $\mu_{k(a,\ldots,b)}$ representa los elementos del circuito k-ésimo que van desde la posición a hasta la b, ambas inclusive. Si $a > b$, se trata del conjunto vacío y el caso $a = b$ se abrevia como $\mu_{k(a)}$.

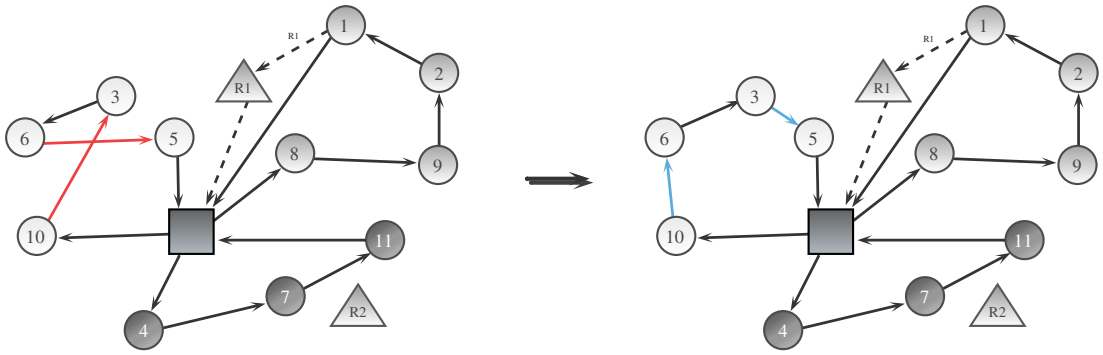

Figura 4.2: Intercambio 2-óptimo de los arcos $(10, 3)$ y $(6, 5)$.

En el ejemplo se ha seleccionado el circuito $k = 1$ con $i = 2$ y $j = 4$. La nueva solución vecina es $S' = \{\mu'_1, \mu'_2, \mu'_3\}$ donde

$$\mu_1 = (\mathcal{D}, 10, 6, 3, 5) \qquad \mu_2 = (\mathcal{D}, 8, 9, 2, 1, R1) \qquad \mu_3 = (\mathcal{D}, 4, 7, 11)$$

4.3.2 Transferencia de un cliente

En este caso, uno de los vehículos deja de visitar a un cliente y es otro el que se encarga de él. Los parámetros que determinan este movimiento son tres: el cliente c que será transferido (sea $c \in \mu_k$), el vehículo l de destino (debe ser $l \neq k$) y la posición p en la que irá el cliente ($p \in \{1, \ldots, \#\mu_j\}$). Este movimiento sólo modifica las rutas μ_k y μ_l que pasan a ser:

$$\mu'_k = \mu_k \setminus c \qquad \mu'_l = (\mu_{l(1,\ldots,p)},\ c,\ \mu_{l(p+1,\ldots,\#\mu_l)})$$

En este caso, se ha seleccionado $c = 9$ y por lo tanto $k = 2$, además $l = 3$ y $p = 4$. La nueva solución contiene las siguientes rutas:

$$\mu_1 = (\mathcal{D}, 10, 3, 6, 5) \qquad \mu_2 = (\mathcal{D}, 8, 2, 1, R1) \qquad \mu_3 = (\mathcal{D}, 4, 7, 11, 9)$$

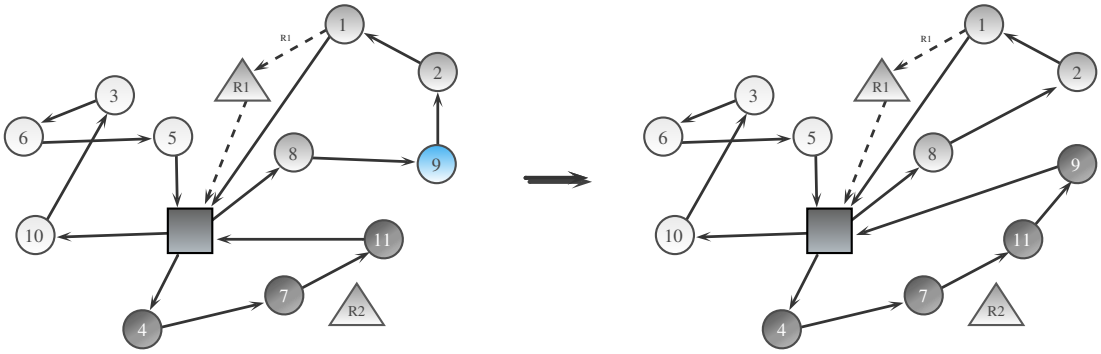

Figura 4.3: Transferencia del cliente 9 a la ruta 3.

4.3.3 Intercambio de dos clientes

Este movimiento consiste en intercambiar los papeles de dos clientes i y j, de forma que i será visitado en la misma ruta y posición que j y viceversa. Nótese que si los clientes pertenecen a rutas diferentes se produce un intercambio entre rutas (y el movimiento equivale a dos "transferencias" consecutivas). Existen dos casos diferenciados:

* Los clientes pertenecen a rutas diferentes.
  Sean μ_i y μ_j los circuitos a los que pertenecen i y j respectivamente. Sea p_i la posición del cliente i en μ_i y sea p_j la posición de j en μ_j. La nueva solución es igual a la anterior salvo en los nuevos circuitos μ_i' y μ_j':

$$\mu_i' = (\mu_{i(1,\ldots,p_i-1)},\ j,\ \mu_{i(p_i+1,\ldots,\#\mu_i)}) \qquad \text{y} \qquad \mu_j' = (\mu_{j(1,\ldots,p_j-1)},\ i,\ \mu_{j(p_j+1,\ldots,\#\mu_j)}).$$

* Los clientes pertenecen a la misma ruta.
  Sea μ_k el circuito común a ambos y sean p_i y p_j las posiciones de i y j respectivamente en μ_k. Se puede suponer sin pérdida de generalidad que $p_i < p_j$. La nueva solución sólo se diferencia de la anterior en el circuito μ_k':

$$\mu_k' = (\mu_{k(1,\ldots,p_i-1)},\ j,\ \mu_{k(p_i+1,\ldots,p_j-1)},\ i,\ \mu_{k(p_j+1,\ldots,\#\mu_k)})$$

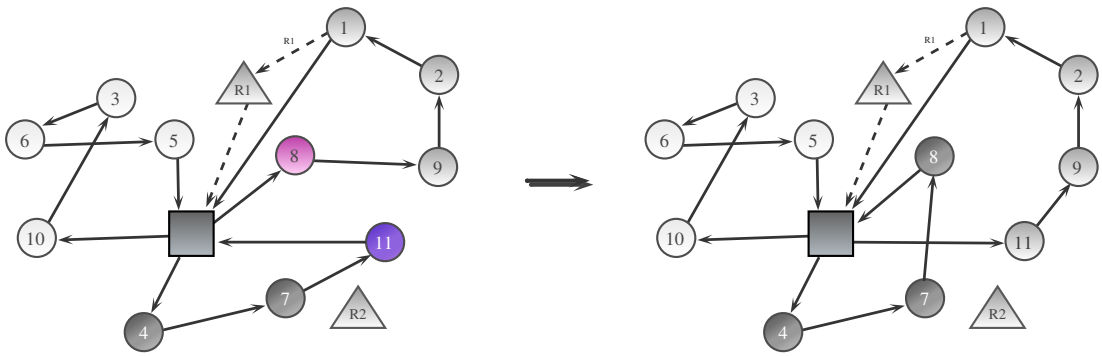

Figura 4.4: Intercambio de los clientes 8 y 11.

Se ha elegido $i = 8$, $j = 11$ para ser intercambiados y la nueva solución está formada por:

$$\mu_1 = (\mathcal{D}, 10, 3, 6, 5) \qquad \mu_2 = (\mathcal{D}, 11, 9, 2, 1, R1) \qquad \mu_3 = (\mathcal{D}, 4, 7, 8)$$

4.3.4 Movimientos de estación de recarga

Es necesario contemplar algunos cambios en la configuración de recargas de la solución para poder alcanzar toda la región de soluciones. A continuación se proponen tres tipos de movimiento:

Inclusión de una estación de recarga en una ruta:

Consiste en determinar una estación de recarga r, un vehículo k, una posición p y una tecnología $t \in \mathcal{T}_r$ disponible. Al incluir estaciones de recarga se supondrá que en éstas se recarga al máximo, es decir, E unidades de energía. Para ajustar de forma óptima las cantidades a recargar se puede utilizar el algoritmo 3.3.1 en el momento adecuado. La nueva ruta sólo se diferenciaría de la primera en el circuito μ_k:

$$\mu'_k = (\mu_{k(1,\ldots,p)},\ r,\ \mu_{k(p+1,\ldots,\#\mu_k)}).$$

Eliminación de una estación de recarga en una ruta:

A veces se puede eliminar las estaciones de recarga de algunas rutas. Dada una ruta μ_k y una posición en ella, p_r (correspondiente a una estación de recarga), la nueva solución es idéntica a la anterior salvo por el circuito μ'_k:

$$\mu'_k = (\mu_{k(1,\ldots,p_r-1)},\ \mu_{k(p_r+1,\ldots,\#\mu_k)})$$

Desplazamiento de una estación de recarga en una ruta:

Este movimiento permite adelantar o retrasar la recarga de energía en una ruta. Se podría conseguir mediante una eliminación y una inclusión consecutivas, pero es preferible considerarlo como movimiento a parte para favorecer la diversificación de los algoritmos.

Dada una ruta μ_k y una posición p_r correspondiente a una estación de recarga se consideran los dos casos:

◇ Se *adelanta* la recarga: la nueva solución se diferencia de la primera en el circuito μ'_k:

$$\mu'_k = (\mu_{k(1,\ldots,p-2)},\ \mu_{k(p)},\ \mu_{k(p-1)},\ \mu_{k(p+1,\ldots,\#\mu_k)})$$

◇ Se *retrasa* la recarga: el nuevo circuito μ'_k quedaría modificado de la siguiente forma:

$$\mu'_k = (\mu_{k(1,\ldots,p-1)},\ \mu_{k(p+1)},\ \mu_{k(p)},\ \mu_{k(p+2,\ldots,\#\mu_k)})$$

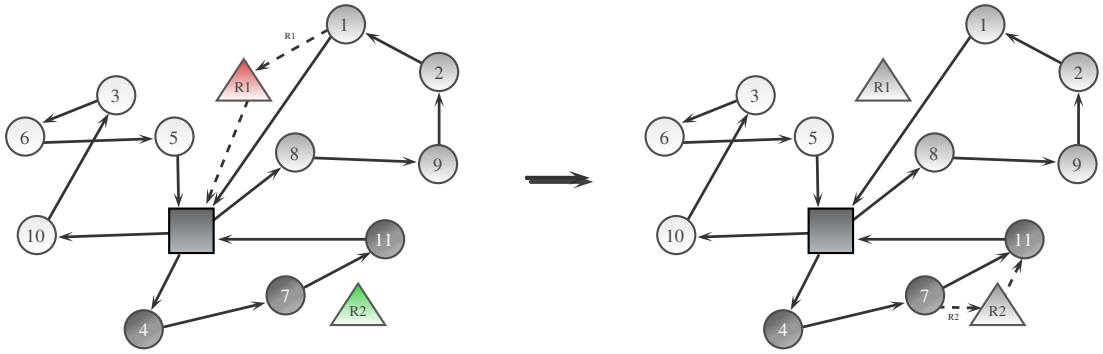

Figura 4.5: Eliminación e inclusión consecutivas de estaciones de recarga.

En la figura 4.5 se han producido dos movimientos consecutivos, uno de ellos de creación de una estación de recarga y otro de eliminación. En el primero de ellos los parámetros han sido $r = 2$, $k = 3$ y $p = 3$ (la t es indiferente en este ejemplo). Por otro lado, en el segundo, los parámetros han sido $k = 2$, $p_r = 6$, llegándose a la siguiente solución vecina:

$$\mu_1 = (\mathcal{D}, 10, 3, 6, 5) \qquad \mu_2 = (\mathcal{D}, 8, 9, 2, 1) \qquad \mu_3 = (\mathcal{D}, 4, 7, R2, 11)$$

4.4 Detalles de las metaheurísticas desarrolladas

En esta sección se incluyen algunos detalles relativos a la implementación de los algoritmos metaheurísticos desarrollados: recocido simulado (Algoritmo 6), búsqueda tabú (Algoritmo 7) y búsqueda por entornos variables (Algoritmo 8).

4.4.1 Función de penalización

En la sección 4.1 se menciona que una forma de mejorar la diversificación de los algoritmos es permitir que se muevan por soluciones infactibles con una penalización. En las metaheurísticas desarrolladas para el *GVRP-MTPR* se trata de minimizar la función $F(S) = f(S) + g(S)$ donde $f(S)$ es la función objetivo del *GVRP-MTPR* y $g(S)$ es una función de infactibilidad.

Sea S una solución del problema y sea t_S el instante de llegada al depósito más tardío en S, e_S la menor cantidad de energía alcanzada por algún vehículo en S (se permite $e_S < 0$) y m_S la carga de mercancía del vehículo más cargado de S. La función g que se ha utilizado es la siguiente:

$$g(S) = K\left(\gamma_1 \frac{t_S - T_{max}}{T_{max}} + \gamma_2 \frac{0 - e_S}{E} + \gamma_3 \frac{c_S - Q}{Q}\right)$$

donde γ_i vale 1 si el problema es infactible por el motivo i ($i = 1$ corresponde a tiempo, $i = 2$ a energía e $i = 3$ a carga de mercancía) y 0 en caso contrario. Cada tipo de infactibilidad está normalizada y va multiplicada por un factor K de penalización que idealmente es variable. Debería ser "alto" cuando aún no se ha conseguido factibilidad y "bajo" cuando el algoritmo se encuentra en la región factible (de forma que se facilita la pérdida temporal de factibilidad y con ello una exploración más amplia del espacio de soluciones).

4.4.2 Detalles del recocido simulado

La elección de la temperatura inicial T, el factor de enfriamiento r o la longitud del bucle L es determinante en el algoritmo. Lo ideal es probar diferentes configuraciones de estos parámetros teniendo en cuenta su significado en el algoritmo. Se ha comprobado que si se elige una temperatura inicial que permita aceptar aproximadamente el 80 % de los cambios a peor al comienzo del algoritmo y un factor de enfriamiento r de forma que en las últimas iteraciones sólo se acepten mejoras (intensificación) se consiguen buenos resultados.

En este algoritmo se han considerado dos criterios de parada: tiempo máximo de ejecución o congelamiento del sistema (cuando la temperatura es menor que una cota). A la hora de generar una solución de $V(S)$ se elige un tipo de movimiento al azar (y una vez elegido el tipo, también se eligen al azar las condiciones del movimiento), con las siguientes probabilidades elegidas empíricamente:

	Transf. c.	Cambio 2-ópt	Intercambio c.	Crea. est.	Elim. est.	Mov. est.
Probabilidad	0.35	0.1	0.25	0.05	0.05	0.2

4.4.3 Detalles de la búsqueda tabú

En una etapa de este algoritmo no se explora toda la vecindad $V(S)$ de una solución, sino un subconjunto $V'(S)$ de la misma. La cardinalidad de este subconjunto es un parámetro del algoritmo. Si tiene pocos elementos, el algoritmo no convergerá adecuadamente, pero si tiene muchos el algoritmo realizará pocas iteraciones (aunque esto puede ser útil para fomentar la intensificación). El número de soluciones tabú no debe ser ni muy alto ni muy elevado. Algunos autores afirman que 7 soluciones tabú es una buena elección.

En este caso, se seleccionan soluciones vecinas con las mismas probabilidades descritas anteriormente en el *recocido simulado*. Además, en este algoritmo se consideran también dos criterios de parada: número máximo de iteraciones o un tiempo máximo de ejecución del algoritmo.

4.4.4 Detalles de la Búsqueda por Entornos Variables

Se ha probado este algoritmo considerando como entornos los diferentes tipos de movimiento, sin embargo éste se quedaba atascado relativamente rápido en óptimos locales. Por eso, se ha pasado a considerar también entornos dobles, es decir, conjuntos de posiciones que se pueden alcanzar realizando dos movimientos seguidos. Un ejemplo podría ser el entorno de "*transferencia de un cliente e inclusión de una estación*". Esta nueva estructura de entornos más compleja permite obtener mejores resultados, no obstante, como esta versión del algoritmo no contempla empeorar la solución, es precisio ejecutarlo varias veces porque es propenso a quedarse atascado en óptimos locales. El algoritmo finaliza en un tiempo máximo de ejecución o si no consigue mejorar la solución.

Algo común a las tres metaheurísticas es que cuando se produce una modificación en la solución, se llama al Algoritmo 3.3.1 para ajustar las cantidades recargadas en cada estación. Los tres algoritmos programados han conseguido encontrar la misma solución al problema modelo de la sección 3.1, con un coste de 25,25€. Además, esta solución coincide con la que proporcionó el algoritmo de la sección 3.2.3 y que está representada en la figura 3.4.

Capítulo 5

Experiencia computacional

En este último capítulo se compararán los algoritmos desarrollados a lo largo de este documento pensados para resolver el *GVRP-MTPR*. Se resolverán distintos casos generados aleatoriamente, anotando la solución obtenida con cada algoritmo y haciendo una comparativa de los tiempos medios de ejecución.

Los algoritmos aproximados han sido implementados en *Matlab R2012a* y recogidos en una interfaz gráfica con la herramienta *Guide*. Han sido ejecutados en un procesador *Intel Core i7-3630QM*, con cuatro núcleos y ocho subprocesos, funcionando a *2.4Hz*, con *8GB* de memoria *RAM* y sobre el s.o. *Ubuntu 14.04 LTS*.

Además, algunos problemas "pequeños" han sido resueltos de forma exacta en el mismo ordenador, sobre *Windows 8*. Para ello se ha utilizado el Modelo II de programación lineal entera propuesto en 2.2.2, implementado en *GAMS* y con el optimizador *CPLEX 12.2*.

5.1 Problemas resueltos de forma exacta

En problemas pequeños se puede utilizar el modelo de programación matemática para encontrar la solución exacta, lo cual resulta muy útil para comprobar cómo de cerca se quedan los algoritmos aproximados.

En esta sección se han resuelto problemas de 10 clientes, facilitados por los autores de [8]. El problema estudiado en este trabajo es exactamente el mismo que en su artículo y los parámetros de dichos problemas toman valores acordes con la realidad, como los siguientes:

- ◇ Capacidad de carga de los vehículos, Q: 2300 Kg
- ◇ Capacidad de energía de las baterías, E: 16000 Wh.
- ◇ Consumo de energía por unidad de distancia, g: 125 Wh/Km.
- ◇ Velocidad media de los vehículos: 25 Km/h.
- ◇ Tiempo máximo permitido, T_{max}: 8 horas.
- ◇ Coste por recarga, c_f: 2.27€.

Además, la recarga en el depósito cuesta 0.016 €/Kwh y existen dos tecnologías de recarga alternativas: una de 0,176€/Kwh con una potencia de recarga de 20000W y otra de 0,192€/Kwh con una potencia de 45000W.

A continuación se presentan los resultados de dos problemas resueltos de forma exacta, resumidos las siguientes tablas:

Problema 10N-10	Vec.C.	Agrup.	Heur.P.M.	R.Sim.	B.Tabú	VNS	Exacto
Coste (€)	29.27	24.57	22.92	22.99	22.92	22.99	22.92
Distancia (Km)	644	546	461	464	461	464	461
Tiempo (min)	1716.1	1413.3	1225.1	1233.5	1225.1	1233.5	1225.1
T.ejecución (s)	0.00	0.00	42.9	41.4	10.4	20.33	12418.4

Problema 29N-10	Vec.C.	Agrup.	Heur.P.M.	R.Sim.	B.Tabú	VNS	Exacto
Coste (€)	30.94	26.71	22.26	22.26	22.26	22.31	22.26
Distancia (Km)	628	535	425	425	425	427	425
Tiempo (min)	1636.8	1408.2	1145.2	1145.2	1145.2	1150.3	1145.2
T.ejecución (s)	0.00	0.00	57.1	10.30	6.4	24.20	14572.1

El tiempo de ejecución es el tiempo de proceso hasta que el algoritmo ha llegado a alcanzar su solución final. Nótese que algunas metaheurísticas han sido capaces de llegar a la solución óptima.

5.2 Comparativa en problemas más complejos

Para estudiar cómo se comportan los algoritmos desarrollados, se han probado sobre una muestra de problemas generados aleatoriamente, con condiciones similares a los de la sección anterior, pero más clientes. En concreto se han resuelto casos de 25, 100 y 200 clientes, cuyos resultados se muestran a continuación:

Problemas de 25 clientes:

En la siguiente tabla se muestra el valor objetivo (en euros) de la mejor solución obtenida con cada algoritmo en una muestra de 10 problemas de 25 clientes:

25 clientes	Vec.C.	Agrup.	Heur. P.M.	R.Sim.	B.Tabú	VNS
Problema 1	20.93	19.72	18.83	18.67	18.77	18.77
Problema 2	17.55	16.28	16.04	15.90	15.90	15.90
Problema 3	20.81	19.75	19.34	19.29	19.22	20.11
Problema 4	26.32	24.34	21.91	21.91	21.91	22.26
Problema 5	27.52	30.20	26.93	26.93	27.02	26.93
Problema 6	34.74	31.63	25.75	26.64	26.61	26.80
Problema 7	28.32	23.60	22.40	22.38	22.40	24.67
Problema 8	27.18	25.93	22.39	22.29	21.93	22.22
Problema 9	23.59	25.74	20.15	19.56	19.59	20.84
Problema 10	19.93	21.96	19.40	19.08	19.18	19.40
Coste medio	24.69	23.91	21.31	21.27	21.25	21.79
T. medio ejec. (s)	0.01	0.19	705.08	20.53	26.04	25.01

En la figura 5.1 se presenta una comparativa entre coste medio de cada algoritmo y tiempo medio de ejecución.

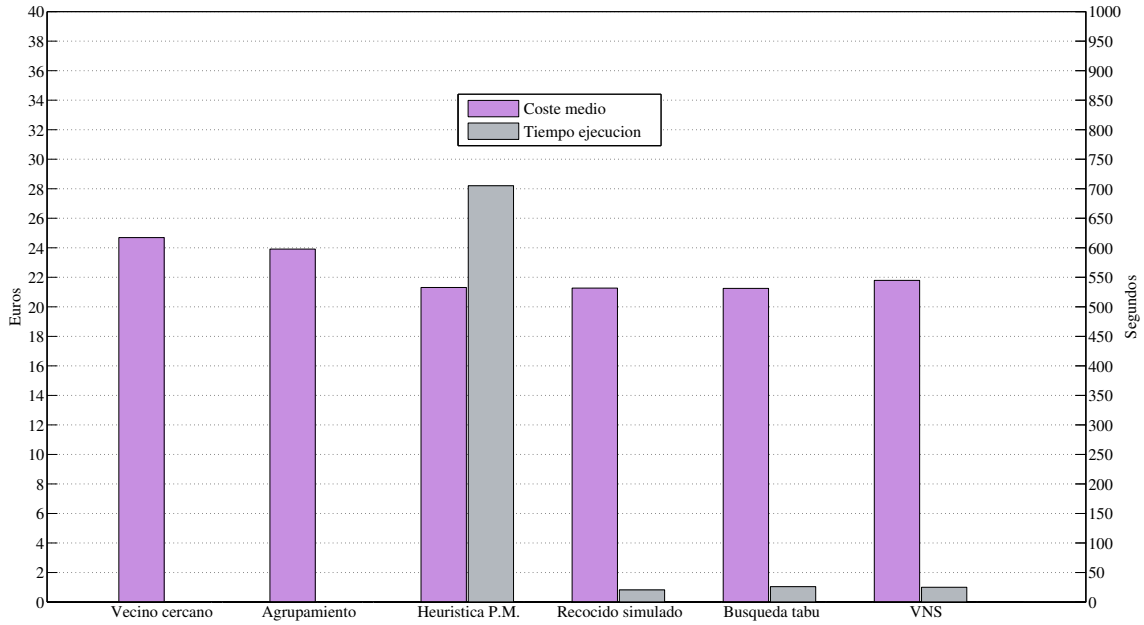

Figura 5.1: Comparativa en problemas con 25 clientes.

Se puede apreciar que la *heurística de programación matemática* necesita significativamente más tiempo que el resto de algoritmos, esto es debido a que el modelo de programación lineal entera es difícil de resolver, a pesar de que se ha eliminado la mayor parte de las variables binarias. Se ha utilizado $K = 7$ en este algoritmo (véase la sección 3.2.3). También es relevante que la *VNS* ha dado lugar a resultados ligeramente peores que las otras metaheurísticas, es más propensa a "atascarse en óptimos locales".

Problemas de 100 clientes:

A continuación se presentan los resultados en problemas de 100 clientes:

100 clientes	Vec.C.	Agrup.	R.Sim.	B.Tabú	VNS
Problema 11	49.80	45.05	40.74	40.26	39.34
Problema 12	54.83	42.01	38.52	38.17	37.81
Problema 13	51.36	43.90	43.58	43.58	43.62
Problema 14	56.34	46.41	44.14	42.36	42.99
Problema 15	52.68	41.40	37.45	36.82	36.72
Problema 16	43.60	42.58	36.98	36.51	36.44
Problema 17	40.53	43.97	36.94	37.61	37.45
Problema 18	39.57	45.38	37.68	38.09	38.35
Problema 19	40.39	41.51	37.93	38.19	37.26
Problema 20	46.82	40.23	33.23	33.58	33.17
Coste medio	47.60	43.24	38.72	38.52	35.00
T. medio ejec. (s)	0.03	1.81	100.5	118.9	177.4

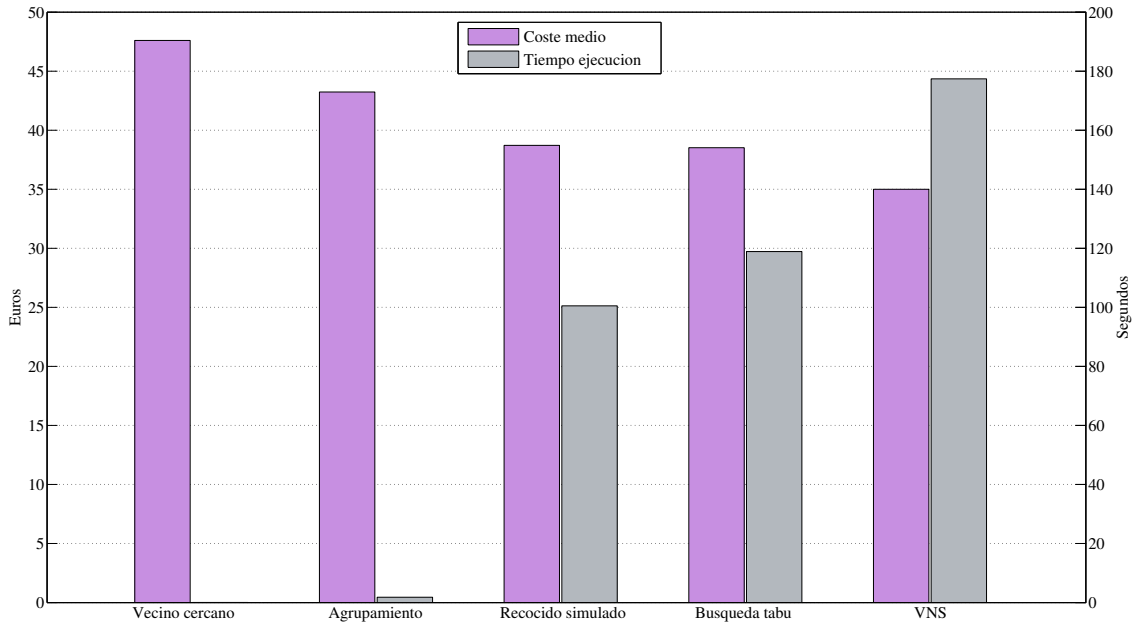

Figura 5.2: Comparativa en problemas con 100 clientes.

En este caso no es posible utilizar la heurística de programación matemática puesto que se trata de problemas con demasiados nodos. Como se puede apreciar en la gráfica, "cuesta" poco tiempo mejorar las soluciones proporcionadas por el algoritmo *vecinos más cercanos* si se utiliza el algoritmo de *agrupamiento*. Sin embargo, el tiempo necesario para obtener aun mejores soluciones (mediante las metaheurísticas) se incrementa bastante.

Hay una diferencia cualitativa entre estos problemas y los de 25 clientes y es que ahora la *VNS* ha sido la que mejores resultados ha conseguido. En los problemas con tantos clientes no se queda atascada con tanta facilidad, lo que es acorde a su tiempo de ejecución, que es el más alto.

Problemas de 200 clientes:

200 clientes	Vec.C.	Agrup.	R.Sim.	B.Tabú	VNS
Problema 21	65.95	61.41	60.51	60.71	60.26
Problema 22	66.95	63.33	60.90	60.04	59.20
Problema 23	71.42	76.87	65.15	65.01	64.81
Problema 24	65.70	60.16	61.63	61.18	59.61
Problema 25	75.43	75.77	66.60	67.48	67.36
Problema 26	67.51	62.54	61.43	61.50	61.09
Problema 27	73.34	73.43	65.18	63.41	62.84
Problema 28	71.57	70.92	69.93	70.70	70.04
Problema 29	67.63	70.01	64.30	64.37	65.11
Problema 30	68.32	66.52	62.24	62.50	62.14
Coste medio	69.38	68.09	63.79	63.69	63.24
T. medio ejec. (s)	0.05	2.93	139.7	165.4	288.2

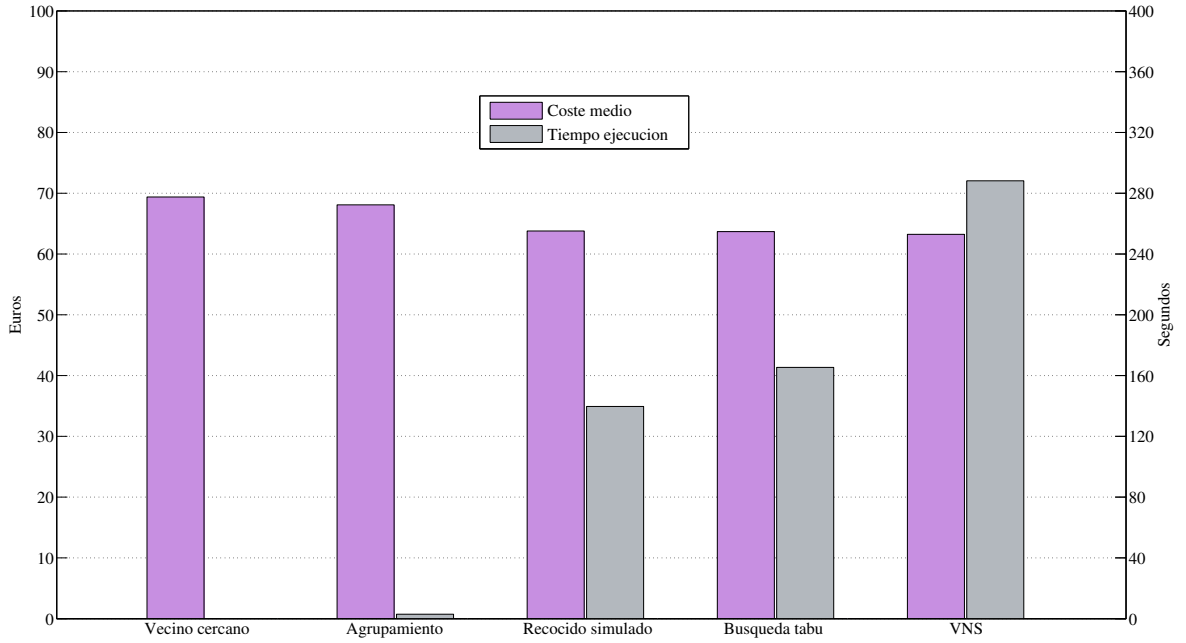

Figura 5.3: Comparativa en problemas con 200 clientes.

En problemas tan grandes, parece que no hay un algoritmo claramente mejor que el resto. Bien es cierto que, de nuevo, el algoritmo *VNS* ha dado lugar a mejores soluciones (en media). Se podría decir que el *recocido simulado* es el algoritmo más recomendable en este tipo de problemas ya que es relativamente rápido y las otras metaheurísticas no ofrecen significativamente mejores resultados. Sin embargo, es adecuado contar con diferentes algoritmos debido a que en un problema particular algunos pueden llegar a mejores soluciones.

5.3 Conclusiones y futuras ampliaciones de este trabajo

El *Problema de Rutas Ecológico con Múltiples Tecnologías y Recargas Parciales* es un problema de optimización combinatoria de la clase de problemas $\mathcal{NP}$-completos, de forma que no se conoce ningún algoritmo que garantice la solución óptima a este problema en tiempo polinomial.

Los modelos de programación lineal entera desarrollados en este trabajo son una buena alternativa para resolver de forma exacta problemas pequeños, algo que se ha conseguido hasta problemas de 10 clientes. Además, el segundo modelo (desarrollado en 2.2.2) tiene menos variables binarias, por lo que cabe esperar que sea más eficiente.

En problemas grandes (con mayor número de clientes) las heurísticas permiten obtener buenos resultados, aunque no garanticen la solución óptima del problema. Además, las metaheurísticas han sido la mejor alternativa de aproximación. El *recocido simulado* y la *búsqueda tabú* son algoritmos que permiten cambios a peor en la solución y esto favorece la diversificación

de soluciones. Por el contrario, la *búsqueda por entornos variables* sólo acepta mejoras, lo que hace que sea un algoritmo muy bueno en intensificación. Los resultados computacionales muestran que en problemas medianos el *RS* y la *BT* encuentran mejores soluciones, pues realizan una búsqueda más exhaustiva en el espacio de soluciones que el resto de los algoritmos. Sin embargo, en problemas grandes ha sido superior la *VNS*. Una interpretación de este hecho es que en estos problemas la región de soluciones es tan grande que la labor de diversificación pierde importancia y prima la de intensificación.

No existe un algoritmo que sea la mejor alternativa en todos los casos, por lo tanto, la forma de obtener mejores resultados es contar con diferentes algoritmos, ejecutarlos desde numerosas soluciones iniciales aleatorias para llegar a múltiples óptimos locales y seleccionar el mejor de ellos. Esta circunstancia es frecuente en problemas de optimización combinatoria.

Una de las posibles ampliaciones de este trabajo es la mejora de los métodos exactos de resolución. Se podría desarrollar un algoritmo de generación de columnas utilizando programación dinámica para resolver los subprocesos asociados. Por otro lado, se podría trabajar en diferentes técnicas metaheurísticas, por ejemplo incluyendo la *Large Neighbourhood Search* [25], con cada vez más éxito en este tipo de problemas.

También se podrían considerar extensiones del problema tratado que lo hagan más realista, como las clásicas ventanas de tiempo. Otra posible mejora del modelo podría ser en el tiempo de recarga de las baterías, que ahora es una función lineal, pero en la realidad las baterías cargan mucho más rápido al principio de la carga que al final.

Otra mejora interesante y que está relacionada con una asignatura optativa del Máster de Ingeniería Matemática (*Simulación Numérica en Ordenadores Paralelos*) sería la de paralelizar los algoritmos implementados de forma que se puedan ejecutar con mejores resultados en el mismo tiempo.

Realizar este Trabajo de Fin de Máster me ha resultado muy interesante al haber tenido la oportunidad de aplicar diferentes conocimientos adquiridos durante mi formación, llegando a resolver problemas "reales" de moderada complejidad a partir de algoritmos competitivos en los que he podido aportar ideas propias. Ha sido gratificante ampliar el problema iniciado en mi Trabajo de Fin de Grado y mi intención es continuar investigando en este tipo de problemas.

Bibliografía

[1] Afifi, S., Guibadj y R.N., Moukrim, A. (2014), *"New Lower Bounds for the Vehicle Routing Problem with Time Windows"* Université de Technologie Compiègne, documento de trabajo.

[2] Ambrona Castellanos, M. (2013) *"El Problema de Rutas Ecológico (The Green Vehicle Routing Problem)"*, Trabajo de Fin de Grado.

[3] Blazinskas, A. y Misevicius, A. (2009) *"Combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the travelling salesman problem"*. Kaunas University of Technology, Department of Multimedia Engineering, Studentu St. 50-401, 416a, Kaunas, Lithuania.

[4] Byung-In, K., Jae-Ik, S. y Min, Z. (1998) *"Comparison of TSP algorithms"*, Proyect for Models in Facilities Planning and Materials Handling.

[5] Cook, W.J., Cunningham, W.H., Pulleyblank, W.R. y Schrijver, A. (1998) *"Combinatorial Optimization"*. Wiley-Interscience.

[6] Cook, W.J. (2012) *"In Pursuit of the Traveling Salesman"*. Princeton University Press.

[7] Erdogan, S. y Miller-Hooks, E. (2011) *"A green vehicle routing problem"*, Transportation Research Part E 48 (2012) 100-114.

[8] Felipe Ortega, Á., Ortuño Sánchez, M.T., Tirado Domínguez, G. y Righini, G. (2013) *Heuristics for the green vehicle routing problem*, Documento de trabajo.

[9] Felipe Ortega, Á. (2012) *"Programación matemática"*, Universidad Complutense de Madrid, Notas de clase.

[10] García Sánchez, A. (2007) *"Técnicas metaheurísticas"*, Documento de trabajo.

[11] Glover, F. (1986) *"Future paths for integer programming and links to artificial intelligence"*, Computers and Operations Research, 5. 533-549.

[12] Glover, F. y Laguna, M. (1997) *"Tabu Search"*, Kluwer Academic Publishers.

[13] Hansen, P y Mladenovic, N. (2001) *"Variable Neighbourhood Search: Principles and Applications"*, European Journal of Operational Research 130, 449-467.

[14] Hochbaum, D.S. (1997) *"Approximation Algorithms for NP-hard problems"*. International Thomson Publishing.

[15] Kaufmann, A. y Henry-Labordère, A. (1977) *Integer and Mixed Programming: Theory and Applications*. Academic Press.

[16] Kirkpatrick, S., Gelatt, C.D. y Vecchi, M.P. (1983) *Optimizacion by Simulated Annealing*. Science, New Series, Vol. 220, No 4598, 671-680.

[17] Korte, B. y Vygen, J. (2000) *"Combinatorial Optimization. Theory and Algorithms"*. Springer.

[18] Laporte G. (1992) *"The Vehicle Routing Problem: An overview of exact and approximate algorithms"*, European Journal of Operational Research 59 345-358.

[19] Lin, S. y Kernighan, B.W. (1973) *"An effective heuristic algorithm for the travelling salesman problem"*, Operations Research 21 498-516.

[20] Martello, S. y Toth, P. (1990) *"Knapsack problems: algorithms and computer implementations"*, Wiley, London.

[21] Melián Batista, B. y Glover, F. (2007) *"Introducción a la Búsqueda Tabú"*, Documento de trabajo.

[22] Moreno Pérez, J.A. y Mladenović, N. (2007) *Búsqueda por Entornos Variables para Planificación Logística*, Documento de trabajo.

[23] Nilsson, C. (2003) *"Heuristics for the Travelling Salesman Problem"*, Linköping University, Documento de trabajo.

[24] Rodríguez Ortiz, C. (2010) *"Algoritmos heurísticos y metaheurísticos para el problema de localización de regeneradores"*, Proyecto fin de carrera, Universidad Rey Juan Carlos.

[25] Ropke, S. y Pisinger, D. (2006) *"An adaptive large neighbourhood search heuristic for the pickupp and delivery problem with time windows"*, Transportation Science 40, 455-472.

[26] Schneider, M., Stenger, A. y Goeke, D. (2014) *"The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations"*, Transportation Science.

[27] Sóley Bjarnadóttir, Á. (2004) *"Solving the Vehicle Routing Problem with Genetic Algorithms"*, Technical University of Denmark, Documento de trabajo.

[28] Touati-Moungla, N. y Jost, V. (2012) *"Combinatorial optimization for electric vehicles management"*, Journal of Energy and Power Engineering 6 738-743.

[29] Uetz, M. (2010) *"Discrete Optimization"*, MSc course Mathematics.

[30] Wolsey, L.A. (1998) *"Integer Programming"*. Wiley-Interscience.

[31] Zanakis, S.H. y Evans, J.R. (1981) *"Heuristic optimization: Why, When and How to use it"*, Interfaces Vol. 11 N°. 5, 84-90.

[32] *Coches híbridos* (última visita el 23 de junio de 2014): http://www.coches.net/noticias/coches-hibridos-con-menor-consumo.

[33] *El vehículo eléctrico en España* (última visita el 23 de junio de 2014), http://www.coches.net/noticias/el-vehiculo-electrico-en-espana.