Trabajo Fin de Máster

Un problema de organización de la producción con prioridades y afinidad geográfica en los pedidos

Estela Quintáns Fernández

Máster en Técnicas Estadísticas

Curso 2017-2018

Propuesta de Trabajo Fin de Máster

Título en galego: Un problema de organización da producción con prioridades e afinidade xeográfica nos pedidos

Título en español: Un problema de organización de la producción con prioridades y afinidad geográfica en los pedidos

English title: A production scheduling problem with priorities and geographical affinity in the orders

Modalidad: Modalidad A

Autor/a: Estela Quintáns Fernández, Universidad de Vigo

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

Breve resumen del trabajo: Se trata de planificar la producción de una fábrica de piensos cuyos pedidos vienen con una ventana temporal en la que necesitan ser atendidos. Los pedidos constan de una o varias fórmulas y la cantidad necesaria de cada uno. La fábrica necesita producir también una cantidad especificada de fórmulas genéricas. Las máquinas de producción necesitan un tiempo de inicio. Además, también es necesario contar con un tiempo de limpieza cada vez que se cambia la fórmula a producir. Finalmente, hay que tener en cuenta que la distribución de los pedidos también corre a cargo de la fábrica, por lo que es ventajoso producir pedidos de una misma zona geográfica el mismo día. Como esquema orientativo, el trabajo debe contemplar:

1. Formulación del modelo matemático asociado al problema.
2. Estudio de las técnicas de solución.
3. Aplicación a un conjunto de datos reales.

Recomendaciones: Aunque no es obligatorio, se recomienda haber cursado algunas de las siguientes materias del máster en Técnicas Estadísticas: Programación Lineal y Entera y Modelos Interactivos de la Investigación Operativa.

Otras observaciones: Bibliografía: Ahumada, O., & Villalobos, J. R. (2009). Application of planning models in the agri-food supply chain: A review. *European Journal of Operational Research,196*(1), 1-20.

Doña María Luisa Carpente Rodríguez, Titular de Universidad de la Universidad de Coruña, informa que el Trabajo Fin de Máster titulado

Un problema de organización de la producción con prioridades y afinidad geográfica en los pedidos

fue realizado bajo su dirección por doña Estela Quintáns Fernández para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, da su conformidad para su presentación y defensa ante un tribunal.

En A Coruña, a 27 de junio de 2018.

La directora:

Doña María Luisa Carpente Rodríguez

La autora:

Doña Estela Quintáns Fernández

Índice general

Resumen

Resumen en español

El objetivo de este trabajo será el estudio del caso donde una fábrica de pienso debe producir y distribuir diferentes productos a demanda de un conjunto de granjas, durante un horizonte temporal dado, así como producir fórmulas genéricas de pienso que se fabrica en los tiempos muertos de la fábrica, y que serán envasadas y almacenadas. Para la producción, contaremos con máquinas con límite de capacidad de fabricación y un máximo de horas laborables por jornada, que deberán someterse a un proceso de limpieza cada vez que se cambie la fórmula a producir; para la distribución, emplearemos una flota homogénea de vehículos compartimentados en tolvas y con límite de capacidad. La finalidad será tanto la reducción de costes en el transporte, como el aumento de ganancias gracias a la producción de pienso genérico, mediante la coordinación de ambas áreas. Dividiremos el problema en tres fases. En la primera de ellas, buscaremos una buena combinación de rutas para la distribución de los productos, tratándolo como un problema de rutas de vehículos, y empleando técnicas metaheurísticas como la Búsqueda Tabú o Recocido Simulado. Analizaremos el funcionamiento de las técnicas empleadas y presentaremos los principales resultados, así como las rutas obtenidas. A continuación, asignaremos estas rutas a los diferentes vehículos, tratando de minimizar el número de camiones a emplear. Por último, optimizaremos la producción, obteniendo el orden de fabricación de las diferentes fórmulas de pienso, de tal forma que nos permita reducir los tiempos de limpieza y así aumentar la producción de pienso genérico. Para llevar a cabo esta búsqueda de solución, programaremos los algoritmos necesarios en R.

English abstract

The objective of this work is to study the problem where a animal feed factory must produce and distribute different products on demand from a number of farms over a period of time as well as to produce generic feed formulas that are manufactured in the dead times of the production process and that can be packaged and stored. For the production, we will have machines with a limit of manufacturing capacity and a maximum number of working hours per day, which should be cleaned each time the formula to be produced is changed.

The factory has a fleet of homogeneus vehicles that are subdivided into hoppers that have a capacity limit. The goal is both the reduction of transport costs, as well as the increase in profits given by the production of generic feed. We will divide the problem into three phases. In the first of them, we will look for a good combination of routes for the distribution of the products, treating it as a problem of vehicle

routes, and employing metaheuristic techniques such as the Tabu Search or Simulated Annealing. Once the routes are obtained, we will assign these routes to the different vehicles, trying to minimize the number of trucks in use.

Finally, we will organize the production, obtaining the manufacturing order of the different formulas in a way that allows us to reduce the cleaning times and increase the production of the generic feed.

All the algorithms are programmed in R.

Capítulo 1

Introducción

Este trabajo se basará en el estudio de la cadena de suministro de una fábrica, que produce y distribuye pienso a granel al por mayor, a demanda de sus clientes (o granjas). El objetivo consistirá en la reducción de costes tanto en la producción como en la distribución, mediante la coordinación de ambos sectores. Considerando esta situación como un problema de optimización, buscaremos una buena combinación de rutas así como el orden recomendable de producción de los diferentes tipos de pienso.

Consideraremos el problema definido en un número de jornadas específico, donde se conocen las cantidades demandadas de cada fórmula de pienso por cada cliente, así como la urgencia de éstos para ser atendidos. Además, como parámetros fijos de este problema concreto, entrarán en juego las distancias y tiempos entre los clientes y la fábrica, el tiempo de carga y descarga de los productos, el límite de capacidad de la fábrica (tanto en toneladas como en tiempo de producción), el tiempo necesario para fabricar cada tonelada de producto, el tiempo de limpieza de las máquinas con cada cambio de fórmula, y el límite en el tiempo de viaje de cada vehículo por jornada.

Cabe destacar que en el tiempo sobrante de fabricación de cada jornada, se utilizarán las máquinas para producir pienso genérico que aporta una ganancia más a la fábrica, por lo tanto será interesante reducir el tiempo de limpieza, y aprovecharlo en la producción de este pienso.

La importancia del problema de optimización de rutas radica en su amplia aplicación en diversas situaciones (principalmente en el campo de la logística, donde se busca encontrar la ruta óptima para, por ejemplo, entrega de mercancías, recogida de basuras, diseño de rutas de metro o autobuses, etc.), así como por su gran capacidad de ahorro de costes de transporte. Además, con la cooperación entre producción y distribución, evitaremos incurrir en pérdidas producidas por gastos en inventario y retrasos en los pedidos. Esta cooperación puede darse a tres niveles:

- Estratégico: engloba decisiones tales como localización de la fábrica, su capacidad y el canal de transporte (aéreo, marítimo, en camiones, trenes, etc);

- Táctico: incluye la cantidad a producir, a inventariar o a enviar, la longitud del ciclo de producción y distribución, etc;

- Operacional: trata problemas sobre cuándo y qué producir/inventariar/enviar, qué granjas visitar, qué productos enviarles, qué vehículos emplear, etc.

Sabiendo que en nuestro caso no es necesario tomar decisiones estratégicas (la localización de la fábrica es fija, su capacidad viene dada de antemano, y la distribución se hará mediante camiones), estaremos interesados en estos dos últimos niveles: táctico (qué, cuánto y cuándo producir) y operacional (cuándo y a quién enviar, y qué rutas seguir).

Existen tres formas de abordar este problema de coordinación:

- La primera de ellas consiste en organizar la producción de forma óptima como primer paso, por ejemplo elaborando los productos de tal forma que se minimice el cambio de fórmula para evitar perder tiempo en limpieza. A continuación, se resolvería el problema de distribución para cada jornada (minimizar el coste de rutas de vehículos), sabiendo que los productos previamente fabricados son los que se deben enviar a los clientes.

- La segunda forma sería otorgarle prioridad a la distribución, minimizando el coste de transporte y respetando las urgencias en los pedidos. Una vez resuelto este primer problema de optimización, organizaremos la producción de lo que se va a enviar en cada jornada, tratando igualmente de reducir el tiempo de limpieza (o el número de veces que cambiamos de fórmula).

- El tercer enfoque es la optimización conjunta de producción y distribución.

Aunque la forma más completa de acercarnos al óptimo del problema global es esta última, nosotros estudiaremos el caso de primero-distribución-segundo-producción, tratando de reducir la carga computacional del problema.

Además, añadiremos una fase intermedia de asignación de rutas a vehículos. Es decir, dividiremos el problema en tres fases. En la primera de ellas, estudiaremos la forma de satisfacer la demanda de los clientes vía terrestre, de forma que minimicemos los costes de transporte asociados. Afrontaremos la primera fase como un problema de rutas de vehículos con capacidad limitada, dada por la capacidad de fabricación y de distribución de los vehículos. Además, puesto que se trata de un problema NP-duro, emplearemos técnicas heurísticas para encontrar una buena solución. En la segunda fase, partiendo de las rutas que se van a recorrer en cada jornada, las repartiremos entre la flota de vehículos homogéneos, tratando de minimizar el número de camiones a emplear por jornada, sin sobrepasar el tiempo máximo de viaje de cada vehículo en cada jornada. La tercera fase consiste en la organización de la producción, tratando de minimizar el tiempo de limpieza de las máquinas, para así poder fabricar más producto genérico.

Este trabajo se divide en dos grandes capítulos, posteriores a la introducción. En el Capítulo 2, nos centraremos en la parte más teórica de definición de algoritmos y obtención de soluciones. Se divide en las siguientes secciones:

- En la primera sección, definiremos el problema de rutas de vehículos, analizaremos las principales variantes a este problema y sus características, y veremos una breve revisión bibliográfica de problemas similares al que vamos a estudiar. Además, describiremos el modelo asociado al problema de rutas, así como sus particularidades en cuanto a capacidad de fabricación (limitada por la fábrica), tipos de productos (demandados o genéricos) y demás variables. Comentaremos los diversos métodos de solución (construcción y mejora de rutas, y técnicas metaheurísticas).

- Seguidamente, en la segunda sección, estudiaremos una buena forma de repartir las rutas en vehículos, tratando de minimizar éstos en cada jornada.

- En la tercera sección, organizaremos la producción que saldrá cada día, tratando de minimizar el tiempo de limpieza.

Una vez vistos los diferentes métodos de obtención de solución, veremos en el Capítulo 3 su aplicación a datos, todo implementado en R. Empezaremos por describir los datos de este estudio; a continuación, comentaremos los métodos de solución aplicados a nuestro caso y presentaremos los resultados obtenidos.

Capítulo 2

Fases para resolver el problema

Índice

2.1. Problema de rutas de vehículos

Los Problemas de Rutas de Vehículos (Vehicle Routing Problem - VRP), se basan en encontrar la combinación óptima de rutas tales que optimicen cierta función (como minimizar el coste o la distancia recorrida), sujeto a una serie de restricciones (como por ejemplo, visitar cada nodo una única vez, o no sobrepasar un tiempo de viaje por cada ruta, o bien no sobrepasar la capacidad del vehículo), y volviendo al punto de partida al final de cada ruta.

Pongamos un ejemplo: pensemos en una empresa que distribuye productos según la demanda de un conjunto de clientes. Esta empresa, que cuenta con una flota de vehículos con capacidad limitada, debe encontrar las rutas más cortas para visitarlos a todos exactamente una vez, de forma que cada vehículo empiece y termine su ruta en la empresa.

Existen multitud de variantes a este problema, aunque los más comunes se pueden diferenciar según las siguientes características:

- **Horizonte temporal:** La cantidad de períodos temporales para los que obtener las rutas: pueden ser un sólo período o un conjunto.

- **Ventanas temporales:** Cada cliente puede pedir ser atendido en una cierta ventana temporal. En este caso, puede que no esté permitido realizar la entrega fuera de la ventana temporal, o bien que esto conlleve penalizaciones, o incluso que, en caso de llegar antes de tiempo, el vehículo deba esperar hasta el inicio de la ventana. Además, será necesario conocer el tiempo que se tarda en descargar la mercancía en cada cliente, así como el tiempo de viajar de uno a otro. Este tipo de problemas es conocido como VRP con ventanas temporales (VRP with Time Windows - VRPTW).

- **Productos:** El problema puede considerar la distribución de un sólo producto o de varios diferentes.

- **Clientes:** Cada uno de ellos tiene una cierta demanda asociada a uno o varios productos. Además, pueden querer ser atendidos con una cierta ventana temporal, como veíamos previamente.

- **Demanda:** Puede ser conocida (determinista) o desconocida (estocástica). En este segundo caso, el problema toma el nombre de VRP Dinámico (Dynamic VRP - DVRP).

- **Servicios:** Existen diferentes servicios, como pueden ser solamente distribución, o envío de un producto y recogida del envase, o recogida de productos de diferentes fábricas y entrega a los clientes, etc. Éste último se conoce como VRP con Recogida y Reparto (VRP with Pick-up and Delivery - VRPPD).

- **Cantidad de visitas:** Hay problemas en los que existe la opción de decidir a qué clientes servir. En otros, cada cliente necesita ser visitado un número de veces dado durante todo el horizonte temporal, por varios vehículos o siempre por el mismo, como en el problema VRP Periódico (Periodic VRP - PVRP). También existe la posibilidad de visitar a cada cliente más de una vez, entregándole parte de la mercancía en cada visita (Split Delivery VRP - SDVRP).

- **Inventario:** Otros problemas tienen en cuenta la capacidad de almacenaje por parte de los clientes o de la fábrica, incurriendo en un cierto coste, como en el Problema de Rutas e Inventario (Inventory Routing Problem - IRP).

• **Vehículos:** Puede tratarse de un sólo vehículo o de varios, que pueden ser de diferentes tipos, capacidades, costes, o incluso con compartimentos para los diferentes productos, o con limitaciones en la cantidad de kilómetros recorridos en cada jornada. El problema más común en estos casos, es el VRP con límite de Capacidad (Capacited VRP - CVRP), donde todos los vehículos de la flota son iguales, es decir, tienen la misma capacidad. En este caso, la suma de las demandas de los clientes que visita cada vehículo no puede sobrepasar su capacidad. Otro tipo de problemas trata sobre vehículos de diferente capacidad, o VRP Heterogéneo (Heterogeneous VRP - HVRP), o bien vehículos compartimentados, donde generalmente no se pueden mezclar productos en un mismo compartimento (Multi-Compartment VRP - MCVRP).

• **Fábrica:** Puede tratarse de una o de varias, con o sin límite en la capacidad de producción. Existen problemas en los que el objetivo es encontrar el lugar óptimo donde ubicar la fábrica.

• **Objetivo:** Los diferentes problemas según el objetivo a considerar pueden ser, por ejemplo, los de minimizar costes o maximizar beneficios, minimizar distancias recorridas, o minimizar el número de vehículos a utilizar.

En nuestro caso, el problema a tratar será la distribución desde una sola fábrica, con un horizonte temporal de unos pocos días. Consideraremos una amplia ventana temporal para cada cliente, que se corresponderá con uno o varios días enteros, en lugar de tramos horarios concretos. Esta ventana se corresponde con la urgencia de los pedidos, y no se permitirá realizar entregas de producto fuera de esta ventana de tiempo.

En cuanto a la cantidad de productos, tendremos un elevado número de fórmulas disponibles, cuya demanda es determinista, y vendrá dada al principio del horizonte temporal. El servicio que realizaremos será el de producción y reparto de la mercancía, donde cada cliente puede ser visitado en varios viajes, aunque con una restricción: cada fórmula de producto de cada cliente deberá ser satisfecha de una sola vez.

La fábrica dispone de una flota de vehículos compartimentados en tolvas iguales con límite de capacidad, donde cada una sólo puede almacenar la demanda de un producto de un cliente. Además, cada vehículo puede realizar varias rutas en cada jornada, sin sobrepasar el tiempo máximo de trabajo del conductor.

Por otra parte, la fábrica dispone de grandes silos (no supondremos límite de capacidad) donde puede almacenar temporalmente el producto una vez fabricado, sin necesidad de que haya siempre un vehículo disponible en el que cargar la mercancía. Al final de cada jornada, estos silos deberán quedar vacíos, es decir, todo el producto fabricado en una jornada deberá ser distribuido en el mismo día. No incurriremos en coste por almacenar el producto en la fábrica.

2.1.1. Breve revisión bibliográfica

El Problema de Rutas de Vehículos nace como variación al Problema del Viajero (Travelling Salesman Problem, TSP). El TSP, propuesto por Flood, 1956, consiste en encontrar la ruta más corta que debe seguir un viajero para que, partiendo y terminando en la misma ciudad, pase por un conjunto de ciudades, de forma que visite una y sólo una vez cada ciudad.

Posteriormente, este problema fue extendido al Multiple Travelling Salesmen Problem (MTSP), donde en lugar de un único viajero, contamos con varios que empiezan y terminan en la misma ciudad.

Dantzing y Ramser, 1959, en el artículo de título *The truck dispatching problem*, introducen el ahora conocido como Problema de Rutas de Vehículos, basándose en el envío de combustible a estaciones de servicio. Lo proponen como una generalización del TSP, donde, una vez visitadas un cierto número de estaciones de servicio, se debe volver al punto de partida. Tratan este problema como la búsqueda de bucles (rutas) con un punto en común, tales que la distancia total de estos bucles sea mínima, y denominan este método como *Clover Leaf Problem* (o Problema de las Hojas de Trébol). Para resolverlo, tratan la creación de clusters de puntos, que unirán mediante ensayo-error hasta encontrar la ruta más corta. Para cuando el número de puntos es demasiado elevado o no es sencillo definir los clusters, proponen otros métodos en este artículo.

Posteriormente, Clarke y Wright, 1964, tratan el caso de vehículos con límite de capacidad, donde desarrollan un proceso iterativo para encontrar una buena ruta.

La primera vez que aparecen las palabras *Vehicle routing* en el título de un artículo fue en Golden et al., 1972. A partir de entonces, se empiezan a tratar diversas variantes de este problema. Por ejemplo, seis años más tarde, en Cook y Russell, 1978, y en Golden y Stewart, 1978, se introducen las demandas estocásticas; en Solomon, 1983, se trata por primera vez de los Problemas de rutas de vehículos con ventana temporal.

Con la llegada de los años 90, se produce un avance tecnológico que ayuda a desarrollar nuevos métodos de búsqueda de solución más complejos. Es entonces cuando se tratan las técnicas metaheurísticas, tales como Recocido Simulado o Búsqueda Tabú, en Gendreau et al., 1998.

Los problemas más recientes tratan sobre la reducción de emisiones o del consumo de combustible (Emissions Vehicle Routing problem - EVRP), donde en ocasiones se tienen en cuenta ciertas estaciones de servicio en las que repostar lo vehículos durante su recorrido. Entre los autores más destacados se encuentran Bektas y Laporte, 2011, Xiao et al., 2012, y Erdogan and Miller-Hooks, 2012. Además, artículos recientes tienen en cuenta el tráfico en las ciudades, como en Renaud et al., 2018.

2.1.2. Particularidades

A continuación, realizaremos la formulación matemática del modelo asociado al problema de rutas de vehículos, aunque primero veremos de una forma más detallada las asunciones y particularidades a considerar en este caso.

Consideremos el caso en el que queremos planificar la producción y distribución de pienso de animales, desde la fábrica hasta las granjas (o socios), durante un determinado horizonte temporal.

Productos

La fábrica tiene disponibles una gran cantidad de productos (o fórmulas) de pienso, que produce y distribuye a demanda de un conjunto de granjas para que alimenten a sus animales.

Fábrica

La fábrica dispone de un límite en la producción diaria, y un límite de tiempo de trabajo, que se emplea tanto para la fabricación de los productos, como para la limpieza de las máquinas cada vez que se cambia la fórmula a producir.

Clientes

Sea el grafo completo dirigido $G = (V, A)$, donde

- $V = \{0, 1, \ldots, N\}$ es el conjunto de vértices que definen la situación de las granjas, siendo 0 la fábrica,

- y $A = \{(i, j)|i, j \in V, i \neq j\}$ es el conjunto de arcos que los unen, que se corresponderán con las carreteras.

Cada uno de estos arcos lleva asociado un coste (o distancia) c_{ij} positivo por viajar de i a j. Supondremos que es un grafo no dirigido, es decir, que la matriz de costes es simétrica ($c_{ij} = c_{ji}$). Por otra parte, también disponemos de la matriz de tiempos de viaje, donde e_{ij} es el tiempo que lleva a un vehículo viajar de i a j. Además, cada una de las granjas, demanda cierta cantidad de una o varias fórmulas, e indicarán si el pedido debe ser atendido en un intervalo de jornadas en concreto o si puede ser satisfecho durante todo el horizonte temporal considerado.

Transporte

Puesto que es la fábrica quien se encarga de realizar el envío de los pedidos, será necesario minimizar el coste de transporte asociado, tratando de visitar en cada ruta a granjas próximas entre sí. Para el transporte, se utilizará una flota de camiones iguales con límite de capacidad y compartimentados (todos los compartimentos serán de igual tamaño). En cada compartimento (o tolva), no podrán mezclarse diferentes fórmulas ni productos de varios clientes, es decir, cada tolva se corresponderá con un producto de un cliente. Cada granja podrá ser visitada más una vez durante todo el horizonte temporal, aunque será necesario que la demanda de cada producto sea entregado por completo en una sola visita.

Cada vehículo saldrá de la fábrica una vez esté cargado con los productos, y volverá al punto de partida cuando haya entregado los pedidos correspondientes. Cabe destacar que tanto la carga como la descarga llevan un determinado tiempo dependiendo del producto y de la cantidad, aparte del tiempo de viaje entre cada dos clientes de la ruta. Cada uno de estos vehículos, realizará el mismo proceso de carga, entrega y descarga (que llamaremos rutas) sin sobrepasar el tiempo máximo de horas de trabajo.

En nuestro caso, realizaremos una previa transformación de los datos:

- Puesto que cada cliente puede ser atendido en diversas ocasiones, donde cada producto debe ser entregado de una sola vez, podremos considerar tratar cada conjunto cliente-producto como un único cliente, otorgándole a cada uno un nuevo identificador de cliente, sin variar su posición ni sus tiempos o costes de viaje. Comentaremos ambos casos (tanto el de partida como el transformado) en la siguiente sección, aunque finalmente estudiaremos el segundo de ellos.

- Obtendremos los datos de tiempos por cliente-producto, con el tiempo de carga y descarga de cada producto demandado por cada cliente. Para ello, basta con considerar el tiempo de carga más el tiempo de descarga por unidad de producto, y multiplicarlo por la cantidad que demanda cada granja.

• Por otra parte, sabemos que cada tolva sólo puede contener el envío de un cliente-producto. Entonces, conociendo la cantidad de producto que demanda cada cliente, y sabiendo la capacidad máxima de cada tolva, podremos calcular cuántas tolvas serán necesarias para satisfacer la demanda de cada cliente-producto. Así, podremos obviar la capacidad de las tolvas y centrarnos solamente en no superar la cantidad de compartimentos de cada vehículo.

Tipo de problema

Nos encontramos ante un problema multi-producto, multi-período, con límites en capacidad en los vehículos, que son compartimentados, límite de tiempo por jornada, límite en la capacidad de fabricación, y con urgencias en los pedidos. Lo consideraremos como una variante del problema de ruta de vehículos con múltiples compartimentos (Multi-Compartment Vehicle Rounting Problem - MCVRP).

Para poder encontrar una buena solución a este problema de planificación, empezaremos por definir los parámetros y las variables asociadas. A continuación formularemos el modelo matemático asociado, y por último nos centraremos en las diversas técnicas de búsqueda de solución.

2.1.3. Formulación matemática del modelo

En esta sección, propondremos el modelo matemático para primera fase de la distribución (obtención de una buena combinación de rutas), cuya función objetivo será minimizar costes, sujeto a restricciones tales como la capacidad de producción y la urgencia en los pedidos, además de las asociadas al problema de rutas de vehículos.

Parámetros

Los parámetros que vendrán dados para cada situación serán los siguientes:

T = número de períodos de tiempo (jornadas) del horizonte temporal considerado. Cada jornada se denotará con el subíndice t. El conjunto de todos los períodos de tiempo se denominará $\mathcal{T}$.

N = número de granjas que realizan los pedidos. Cada una se denotará con los subíndices i, j o l. La fábrica de piensos será $i = 0$. El conjunto de todas las granjas se denominará $\mathcal{N}$, y el conjunto de granjas más la fábrica será $\mathcal{N}^+$.

K = número de fórmulas diferentes disponibles. Cada una se denotará con el subíndice k. El conjunto de todas las fórmulas se denominará $\mathcal{K}$.

C = número de compartimentos de cada vehículo.

Cap = capacidad de cada tolva. Recordemos que todas las tolvas de todos los vehículos tienen la misma capacidad, por tratarse de un problema con vehículos homogéneos.

$\mathbf{R_t}$ = número de rutas que se realizan en cada jornada. Se denotarán con el subíndice r, y el conjunto de todas las rutas será $\mathcal{R}$. Este parámetro no vendrá dado de antemano, sino que podremos disponer de la cantidad de rutas que necesitemos, siempre que no se sobrepase el límite de horas disponibles para el reparto.

$\mathbf{g_{ik}}$ = demanda del producto k por el cliente i ($k \in \mathcal{K}$, $i \in \mathcal{N}$).

$\mathbf{d_{ik}}$ = número de tolvas necesarias para satisfacer la demanda del producto k para el cliente i. Es decir, partimos de las unidades (o peso) demandadas de la fórmula k ($k \in \mathcal{K}$) por parte de la granja i ($i \in \mathcal{N}$), g_{ik}, y, conociendo la capacidad máxima de cada tolva, Cap, calculamos cuántas serán necesarias para transportar el producto de la siguiente forma:

$$d_{ik} = \frac{g_{ik}}{Cap}.$$

De esta forma, para resolver el problema de rutas, utilizaremos este último parámetro en lugar la capacidad de las tolvas y la demanda exacta de los productos.

$\mathbf{c_{ij}}$ = coste de transporte (distancia) asociado a viajar de i a j, que satisfará las desigualdades triangulares ($c_{ij} \leq c_{il} + c_{lj}$, $i, j, l \in \mathcal{N}^+$). El coste de viajar de i a i será nulo ($c_{ii} = 0$, $i \in \mathcal{N}^+$), y se cumple que $c_{ij} = c_{ji}$, $i, j \in \mathcal{N}^+$. En este caso, consideraremos que el coste es proporcional a la distancia recorrida.

$\mathbf{e_{ij}}$ = tiempo de transporte asociado a viajar de i a j, que satisfará las desigualdades triangulares ($e_{ij} \leq e_{il} + e_{lj}$, $i, j, l \in \mathcal{N}^+$). El tiempo de viajar de i a i será nulo ($e_{ii} = 0$, $i \in \mathcal{N}^+$), y se cumple que $e_{ij} = e_{ji}$, $i, j \in \mathcal{N}^+$.

$\mathbf{h_{ik}}$ = tiempo de carga (en la fábrica) y descarga (en el cliente i) del producto k. Este tiempo se obtiene de multiplicar tanto el tiempo de carga por unidad de producto, como el de descarga, por la cantidad demandada.

M= capacidad máxima de fabricación en cada jornada.

L= límite de tiempo de viaje más carga y descarga de cada vehículo en cada jornada. Puesto que un vehículo puede realizar varias rutas, este límite en el tiempo de viaje se aplicará también a cada ruta.

$[\mathbf{a_{ik}}, \mathbf{b_{ik}}]$ = intervalo temporal (jornadas) en las que debe ser entregado el producto k al cliente i, $k \in \mathcal{K}$, $i \in \mathcal{N}$.

Variables

Las variables de interés son las siguientes:

$$\mathbf{x_{ijrt}} = \begin{cases} 1 & \text{si la ruta } r \text{ viaja de } i \text{ a } j, \text{ en el período } t; \\ 0 & \text{en otro caso; } i, j \in \mathcal{N}^+;\ r \in \mathcal{R}_t;\ t \in \mathcal{T} \end{cases}$$

$$\mathbf{y_{ikrt}} = \begin{cases} 1 & \text{si en la ruta } r \text{ se entrega el producto } k \text{ al cliente } i, \text{ en el período } t; \\ 0 & \text{en otro caso; } i \in \mathcal{N}^+;\ k \in \mathcal{K};\ r \in \mathcal{R}_t;\ t \in \mathcal{T} \end{cases}$$

Modelo

El objetivo será minimizar el coste de transporte (equivalente al coste de los trayectos entre cada dos granjas, multiplicado por el número de veces que se recorre), sujeto a restricciones de capacidad de fabricación, de capacidad del vehículo, de tiempo de viaje por cada jornada y de urgencia en los pedidos, principalmente, como vemos a continuación:

$$
\begin{aligned}
\textit{min} \quad & \sum_{i,j\in\mathcal{N}^+}\sum_{r\in\mathcal{R}_t}\sum_{t\in\mathcal{T}} c_{ij}x_{ijrt} \\
\textit{sujeto a} \quad & \sum_{i\in\mathcal{N}^+} x_{ijrt} \leq 1, \ \ \forall j \in \mathcal{N}^+, r \in \mathcal{R}_t, t \in \mathcal{T} && (2.1)\\
& \sum_{r\in\mathcal{R}_t}\sum_{t\in\mathcal{T}} y_{ikrt} = \text{ind}(d_{ik}), \ \ \forall i \in \mathcal{N}, k \in \mathcal{K} && (2.2)\\
& \sum_{j\in\mathcal{N}} x_{ijrt} = \max_k(y_{ikrt}), \ \ \forall i \in \mathcal{N}, r \in \mathcal{R}_t, t \in \mathcal{T} && (2.3)\\
& \sum_{i\in\mathcal{N}^+} x_{ijrt} = \sum_{i\in\mathcal{N}^+} x_{jirt}, \ \ \forall j \in \mathcal{N}^+, r \in \mathcal{R}_t, t \in \mathcal{T} && (2.4)\\
& \sum_{i,j\in\mathcal{S}} x_{ijrt} \leq |\mathcal{S}| - 1, \ \ \forall \mathcal{S} \subseteq \mathcal{N}, \ \ |\mathcal{S}| \geq 2, \ \ \forall r \in \mathcal{R}_t, t \in \mathcal{T} && (2.5)\\
& \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}} y_{ikrt}d_{ik} \leq C, \ \ \forall r \in \mathcal{R}_t, t \in \mathcal{T} && (2.6)\\
& \sum_{i,j\in\mathcal{N}^+} x_{ijrt}e_{ij} + \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}} y_{ikrt}h_{ik} \leq L, \ \ \forall r \in \mathcal{R}_t, t \in \mathcal{T} && (2.7)\\
& \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_t} g_{ik}y_{ikrt} \leq M, \ \ \forall t \in \mathcal{T} && (2.8)\\
& \sum_{t=a_i}^{b_i}\sum_{r\in\mathcal{R}_t} y_{ikrt} = \text{ind}(d_{ik}), \ \ \forall i \in \mathcal{N}, k \in \mathcal{K} && (2.9)\\
& x_{ijrt} \in \{0,1\}, \ \ \forall i,j \in \mathcal{N}^+, r \in \mathcal{R}_t, t \in \mathcal{T} \\
& y_{ikrt} \in \{0,1\}, \ \ \forall i \in \mathcal{N}^+, k \in \mathcal{K}, r \in \mathcal{R}_t, t \in \mathcal{T}
\end{aligned}
$$

La restricción (2.1) garantiza que cada granja es visitada a lo sumo una vez en cada ruta.

Además, la demanda completa de cada producto k de un cliente i se satisface por completo en una misma visita (el número de veces que se entrega k a i es igual a 1, si éste fue demandado por dicho cliente, es decir, si el índice de d_{ik} es igual a 1), como podemos ver en la restricción (2.2). Ésta, a su vez garantiza que todas las demandas de los clientes deben ser satisfechas (todos los productos deben ser entregados, si fueron solicitados).

En la restricción (2.3) tenemos la relación entre las variables x e y: si algún producto k ha sido entregado a la granja i (por la ruta r en la jornada t), ésta ha sido visitada (por la misma ruta r en la misma jornada t), y viceversa (si ha sido visitado, se le entrega al menos un producto).

Por otra parte, la restricción (2.4) marca que todas las rutas que llegan a una granja, salen de dicha granja, y que todas terminan en el punto de partida, la fábrica (conservación de flujo).

Además, para evitar subcircuitos, es necesario definir la ecuación (2.5) para cada ruta en cada jornada: si tomamos un subconjunto de granjas (sin incluir la fábrica), llamado $\mathcal{S}$, el número de arcos que las unen debe ser menor o igual que la cantidad de granjas en el subconjunto, menos 1.

La restricción (2.6) impide sobrepasar el límite de capacidad de la ruta, es decir, el número de tolvas necesarias para transportar los productos en cada viaje, no debe superar la cantidad de tolvas que posee un vehículo.

Además, la restricción (2.7) también añade un límite en las rutas: evita que el tiempo de carga y descarga de cada producto en cada cliente, más el tiempo de viaje, no supere el límite de tiempo de la ruta, para así garantizar que no se sobrepasará el tiempo de recorrido total de cada vehículo (recordemos que cada vehículo puede realizar varias rutas con un límite de tiempo).

De la misma forma, la restricción (2.8) hace referencia al límite máximo de fabricación, es decir, a la capacidad de producción de la fábrica, que no debe ser sobrepasado por la cantidad de producto repartido en cada jornada.

Por último, vemos en la restricción (2.9) que la totalidad de los productos demandados por cada cliente deben ser entregados dentro del rango temporal establecido por cada granja. Esta restricción vemos que es exactamente igual que la (2.2) pero sumando en el intervalo temporal establecido, en vez de en todo el rango. De esta forma, fuera de los días establecidos por cada granja, no se entregará ningún pedido.

Otro enfoque del problema

Para mayor simplicidad a la hora de resolver el problema, podemos considerar la dupla (cliente, producto) como un único cliente, es decir, si una granja solicita dos productos, será tratada como dos granjas independientes (con distancia y tiempo entre ellas igual a cero). De esta forma, los parámetros, variables y modelo quedarían de la siguiente forma:

Parámetros

T = número de períodos de tiempo (jornadas) del horizonte temporal considerado. Cada jornada se denotará con el subíndice t. El conjunto de todos los períodos de tiempo se denominará $\mathcal{T}$.

N = número de granjas que realizan los pedidos, donde cada granja se refiere a la dupla (granja,producto), es decir, cada cliente aquí considerado demanda un único producto. Cada uno se denotará con los subíndices i, j o l. La fábrica de piensos será $i = 0$. El conjunto de todas las granjas se denominará $\mathcal{N}$, y el conjunto de granjas más la fábrica será $\mathcal{N}^+$.

C = número de compartimentos de cada vehículo.

Cap = capacidad de cada tolva. Recordemos que todas las tolvas de todos los vehículos tienen la misma capacidad, por tratarse de un problema con vehículos homogéneos.

$\mathbf{R_t}$ = número de rutas que se realizan en cada jornada. Se denotarán con el subíndice r, y el conjunto de todas las rutas será $\mathcal{R}_t$. Este parámetro no vendrá dado de antemano, sino que podremos disponer de la cantidad de rutas que necesitemos, siempre que no se sobrepase el límite de horas disponibles para el reparto.

$\mathbf{g_i}$ = demanda del cliente i ($i \in \mathcal{N}$).

$\mathbf{d_i}$ = número de tolvas necesarias para satisfacer la demanda del cliente i. Es decir, partimos de las unidades (o peso) demandadas por parte de la granja i ($i \in \mathcal{N}$), y, conociendo la capacidad

máxima de cada tolva, calculamos cuántas serán necesarias para transportar su demanda.

$$d_i = \frac{g_i}{Cap}.$$

De esta forma, para resolver el problema de rutas, utilizaremos este último parámetro en lugar la capacidad de las tolvas y la demanda exacta de los productos.

$\mathbf{c_{ij}}$ = coste de transporte (distancia) asociado a viajar de i a j, que satisfará las desigualdades triangulares ($c_{ij} \leq c_{il} + c_{lj}$, $i, j, l \in \mathcal{N}^+$). El coste de viajar de i a i será nulo ($c_{ii} = 0$, $i \in \mathcal{N}^+$), y se cumple que $c_{ij} = c_{ji}$, $i, j \in \mathcal{N}^+$. En este caso, consideraremos que el coste es proporcional a la distancia recorrida.

$\mathbf{e_{ij}}$ = tiempo de transporte asociado a viajar de i a j, que satisfará las desigualdades triangulares ($e_{ij} \leq e_{il} + e_{lj}$, $i, j, l \in \mathcal{N}^+$). El tiempo de viajar de i a i será nulo ($e_{ii} = 0$, $i \in \mathcal{N}^+$), y se cumple que $e_{ij} = e_{ji}$, $i, j \in \mathcal{N}^+$.

$\mathbf{h_i}$ = tiempo de carga (en la fábrica) y descarga (en el cliente i).

M= capacidad máxima de fabricación en cada jornada.

L= límite de tiempo de viaje más carga y descarga de cada vehículo en cada jornada. Puesto que un vehículo puede realizar varias rutas, este límite en el tiempo de viaje se aplicará también a cada ruta.

$[\mathbf{a_i}, \mathbf{b_i}]$ = intervalo temporal (jornadas) en las que debe ser entregada la cantidad demandada al cliente i, $i \in \mathcal{N}$.

Variables

La variable de interés es la siguiente:

$$\mathbf{x_{ijrt}} = \begin{cases} 1 & \text{si la ruta } r \text{ viaja de } i \text{ a } j, \text{ en el período } t; \\ 0 & \text{en otro caso; } i, j \in \mathcal{N}^+;\ r \in \mathcal{R}_t;\ t \in \mathcal{T} \end{cases}$$

Modelo

El objetivo será minimizar el coste de transporte (equivalente al coste de los trayectos entre cada dos granjas, multiplicado por el número de veces que se recorre), sujeto a restricciones de capacidad de las tolvas, de tiempo de viaje por cada jornada y de urgencia en los pedidos, principalmente, como

vemos a continuación:

$$
\begin{aligned}
\textit{min} \quad & \sum_{i,j\in\mathcal{N}^+}\sum_{r\in\mathcal{R}_t}\sum_{t\in\mathcal{T}} c_{ij}x_{ijrt} \\
\textit{sujeto a} \quad & \sum_{i\in\mathcal{N}^+} x_{ijrt} \leq 1, \ \ \forall j \in \mathcal{N}^+, r \in \mathcal{R}_t, t \in \mathcal{T} && (2.10)\\
& \sum_{j\in\mathcal{N}^+}\sum_{r\in\mathcal{R}_t}\sum_{t\in\mathcal{T}} x_{ijrt} = 1, \ \ \forall i \in \mathcal{N} && (2.11)\\
& \sum_{i\in\mathcal{N}^+} x_{ijrt} = \sum_{i\in\mathcal{N}^+} x_{jirt}, \ \ \forall j \in \mathcal{N}^+, r \in \mathcal{R}_t, t \in \mathcal{T} && (2.12)\\
& \sum_{i,j\in\mathcal{S}} x_{ijrt} \leq |\mathcal{S}| - 1, \ \ \forall \mathcal{S} \subseteq \mathcal{N}, \ \ |\mathcal{S}| \geq 2, \ \ \forall r \in \mathcal{R}_t, t \in \mathcal{T} && (2.13)\\
& \sum_{i,j\in\mathcal{N}} x_{ijrt} d_i \leq C, \ \ \forall r \in \mathcal{R}_t, t \in \mathcal{T} && (2.14)\\
& \sum_{i,j\in\mathcal{N}^+} x_{ijrt} e_{ij} + \sum_{i,j\in\mathcal{N}} x_{ijrt} h_i \leq L, \ \ \forall r \in \mathcal{R}_t, t \in \mathcal{T} && (2.15)\\
& \sum_{i,j\in\mathcal{N}}\sum_{r\in\mathcal{R}_t} g_i x_{ijrt} \leq M, \ \ \forall t \in \mathcal{T} && (2.16)\\
& \sum_{t=a_i}^{b_i}\sum_{j\in\mathcal{N}^+}\sum_{r\in\mathcal{R}_t} x_{ijrt} = 1, \ \ \forall i \in \mathcal{N} && (2.17)\\
& x_{ijrt} \in \{0,1\}, \ \ \forall i,j \in \mathcal{N}^+, r \in \mathcal{R}_t, t \in \mathcal{T}
\end{aligned}
$$

Vemos que el problema es similar al primer enfoque realizado, salvo que no se consideran productos, es decir, gracias a la transformación de los datos que hemos hecho, en esta situación cada cliente demanda un único producto.

Es sencillo ver que este problema es una generalización del Problema de Ruta de Vehículos con Límite de Capacidad (Capacited Vehicle Routing Problem - CVRP), puesto que bastaría considerar una única jornada ($T = 1$), un único producto disponible ($K = 1$), número de compartimentos ilimitado (de esta forma podrían compartir vehículo todos los clientes, siempre que no sobrepasen su capacidad). Además, en el CVRP, no consideraríamos límite de tiempo de viaje, ni de tiempo de fabricación ni urgencia en los pedidos (por lo tanto omitiríamos las restricciones (2.15), (2.16) y (2.17)).

A continuación estudiaremos las técnicas de solución para este problema. En primer lugar, nos centraremos en el caso donde la poca cantidad de granjas y el gran número de restricciones nos permiten obtener un número de rutas factibles limitado, que enumeraremos, y posteriormente será sencillo encontrar el óptimo a partir de estas rutas factibles. En segundo lugar, veremos técnicas para cuando la cantidad de rutas posibles es elevada, o no es posible analizarlas todas debido al elevado tiempo computacional que esto supondría. En particular, estudiaremos diferentes técnicas heurísticas para aproximarnos a la solución óptima.

2.1.4. Búsqueda de solución

Antes de centrarnos en los diferentes métodos para encontrar una buena solución al problema de optimización de rutas, vamos a analizar su complejidad, relacionada con la cantidad de recursos necesarios para resolverlo. Para ello, recordemos que los problemas de decisión de clase $\mathcal{P}$ son aquellos

que pueden resolverse mediante un algoritmo determinístico en tiempo polinómico. Los de clase $\mathcal{NP}$, corresponden a los problemas de decisión que pueden ser resueltos por una máquina de Turing no determinística en tiempo polinómico. De esta forma, un problema será $\mathcal{NP}$-Duro si es al menos tan duro como un problema $\mathcal{NP}$. Está probado que el problema del viajante (TSP) es $\mathcal{NP}$-Duro, y puesto que el Problema de Rutas de Vehículos es una extensión de éste, también será $\mathcal{NP}$-Duro.

Para encontrar la solución a este problema, existen diversas formas, entre las que destacan los métodos exactos (Branch and Bound, Branch and Cut, Branch and Price y Enumeración Implícita) y los métodos heurísticos (basados en construcción de rutas iniciales, mejora de rutas mediante intercambio de clientes intra-rutas o inter-rutas, y técnicas metaheurísticas), como veremos a continuación.

Métodos exactos

Los métodos exactos son útiles para buscar la solución óptima cuando el problema es de una dimensión no demasiado grande. A continuación comentaremos brevemente cada uno de ellos.

Branch and Bound

La idea principal del método B&B radica en dividir el espacio de búsqueda en subespacios (branch), y evaluar los límites superior e inferior en estos subespacios (bound). Si vemos que uno de estos subespacios no contiene la solución óptima, se descarta. En caso contrario, vuelve a realizarse el proceso.

Por ejemplo, si una de las variables x es entera, podemos dividir el espacio de posibles soluciones en dos subespacios de la siguiente forma: uno de ellos será considerando la restricción $x \leq a$, y otro $x \geq a+1$, siendo a un número entero dentro del espacio de búsqueda. Una forma de obtener un valor de a apropiado, sería resolver el problema sin la restricción de que x sea una variable entera (también llamado problema relajado). De esta forma, x será un número racional situado entre dos enteros, que se corresponderán con a y $a+1$.

Otra forma de dividir el espacio de búsqueda, por ejemplo, si una variable x debe ser binaria (como ocurre en nuestro caso particular), sería dividir el espacio en los dos siguientes: uno de ellos considerando $x = 0$ y el otro tomando $x = 1$. Podríamos considerar, por ejemplo, dos subproblemas: en uno de ellos una determinada ruta pasa por un arco, y en el otro subproblema no pasa por dicho arco.

A continuación, una vez dividido el espacio de búsqueda, buscaremos un límite inferior del coste. Para ello, se resuelve un problema relajado, es decir, eliminaremos o relajaremos algunas de las restricciones asociadas, y buscaremos el óptimo de este nuevo problema, que se corresponderá con el límite inferior del coste. Si una de las soluciones encontradas para algún problema verifica todas las restricciones del problema inicial, ésta será un límite superior de la solución óptima del problema inicial (llamémosla X^*). De esta forma, cualquier subproblema relajado cuya solución óptima sea mayor que X^*, será descartado del espacio de búsqueda.

Existen otros métodos para obtener un problema relajado, entre los cuales destaca la Relajación de Lagrange (Fisher, 1994). Esta técnica consiste en añadir restricciones a la función objetivo, con una penalización por violarlas.

Repetiremos el proceso de división del espacio y búsqueda de límites inferiores y superiores hasta hallar la solución óptima. Podemos encontrar ejemplos de aplicación a datos en Winston, 2004, y Hillier y Lieberman, 2001.

Branch and Cut

Similar al método anterior, partimos de un problema relajado para obtener un límite inferior de la solución óptima. En cada iteración, añadiremos una nueva restricción válida que haya sido violada por la solución actual. Repetiremos el proceso hasta haber encontrado una solución óptima que satisfaga todas las restricciones del problema inicial (Baldacci et al., 2004). Un ejemplo aplicado a nuestro problema sería empezar considerando las restricciones de que todos los pedidos deben ser entregados y de que las variables sean binarias. Posteriormente, añadir secuencialmente las siguientes restricciones: que deban ser entregados en la ventana temporal, límite en la capacidad de producción, límite de capacidad de los vehículos, y por último límite de tiempo de rutas.

Branch and Price

Este método es análogo a la técnica Branch and Bound, con la diferencia de que en cada nodo del árbol de búsqueda se resuelve un problema relajado utilizando el método de Generación de Columnas (Dantzig y Wolfe, 1960), y de esta forma obtener los límites inferiores de la solución óptima.

Enumeración implícita

Es obvio ver que un problema de grandes dimensiones como el que estamos tratando no puede resolverse analizando todas las posibles opciones. Sin embargo, puesto que muchas de las rutas no son factibles debido a la capacidad de los vehículos, es posible enumerar las rutas que se pueden llevar a cabo, para posteriormente tomar aquellas que son óptimas, y que visiten a cada granja una única vez. Basándonos en Ruiz et al, 2003, modificaremos su algoritmo de una sola jornada y un único producto a nuestro caso particular, multi-jornada, multi-producto.

Supongamos que tenemos contabilizadas las rutas factibles (es decir, las que no sobrepasan el límite de capacidad del vehículo ni el límite de tiempo de ruta), y sabemos a qué clientes visitan, qué productos les entregan, y cuál es el coste de transporte asociado a cada una. Entonces, el problema se reduce a elegir qué rutas se llevan a cabo y en qué jornada se realizan, cumpliendo las restricciones de urgencia y garantizando que se entreguen todos los pedidos.

Tendremos entonces el siguiente problema de minimización, una vez hayamos enumerado las rutas factibles, y calculado su coste y tiempo asociados:

Parámetros

$\mathbf{R}$ = número de rutas factibles, de las que conocemos su coste, a qué clientes visitan y qué productos entregan. Cada una se denotará con el subíndice r, y el conjunto de todas será $\mathcal{R}$.

$\mathbf{T}$ = número de períodos de tiempo (jornadas) del horizonte temporal considerado. Cada jornada se denotará con el subíndice t. El conjunto de todos los períodos de tiempo se denominará $\mathcal{T}$.

$\mathbf{N}$ = número de granjas que realizan los pedidos (recordemos que cada una representa un cliente-producto). Cada una se denotará con los subíndices i, j o l. La fábrica de piensos será $i = 0$. El conjunto de todas las granjas se denominará $\mathcal{N}$, y el conjunto de granjas más la fábrica será $\mathcal{N}^+$.

$\mathbf{d_i}$ = número de tolvas necesarias para satisfacer la demanda del cliente i.

$[\mathbf{a_i}, \mathbf{b_i}]$ = intervalo temporal (jornadas) en las que debe ser atendido el cliente i, $i \in \mathcal{N}$.

$\mathbf{Coste_r}$ = coste de transporte de la ruta r, $r \in \mathcal{R}_t$.

$$\mathbf{w_{ir}} = \begin{cases} 1 & \text{si la ruta } r \text{ visita la granja } i;\ i \in \mathcal{N}, r \in \mathcal{R}_t \\ 0 & \text{en otro caso} \end{cases}$$

Variables

$$\mathbf{PP_{rt}} = \begin{cases} 1 & \text{si se recorre la ruta } r \text{ en el periodo } t;\ r \in \mathcal{R}_t, t \in \mathcal{T} \\ 0 & \text{en otro caso} \end{cases}$$

Modelo

El objetivo será minimizar el coste de transporte, que ya es conocido, de las rutas que se lleven a cabo, sujeto a la restricción de que se satisfagan las demandas de los clientes en el intervalo temporal marcado por éstos:

$$\begin{aligned} min \quad & \sum_{r \in \mathcal{R}_t} \sum_{t \in \mathcal{T}} PP_{rt} Coste_r \\ sujeto\ a \quad & \sum_{r \in \mathcal{R}_t} \sum_{t \in \mathcal{T}} PP_{rt} w_{ir} = \text{ind}(d_i), \quad \forall i \in \mathcal{N} \qquad (2.18) \\ & \sum_{r \in \mathcal{R}_t} \sum_{t=a_i}^{b_i} PP_{rt} w_{ir} = \text{ind}(d_i), \quad \forall i \in \mathcal{N} \qquad (2.19) \\ & PP_{rt} \in \{0,1\}, \quad \forall r \in \mathcal{R}_t, t \in \mathcal{T} \end{aligned}$$

La restricción (2.18), obliga a satisfacer la demanda de todos los productos de cada cliente, y la restricción (2.19), a que se haga durante el tiempo establecido por cada granja.

Pongamos un ejemplo sencillo. Supongamos que el problema consta de 4 clientes y una fábrica, representados en la Figura 2.1.

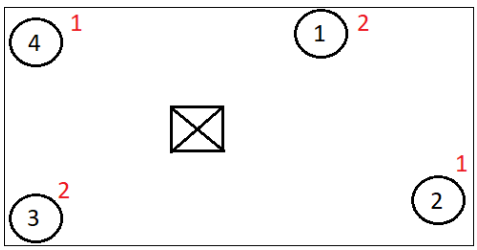


Figura 2.1: Posición de los clientes respecto a la fábrica y demanda asociada medida en tolvas (en rojo)

Cada ruta tiene un límite de tiempo de viaje de 10 horas, y un límite de capacidad de 3 tolvas. Las matrices de tiempos y costes son las siguientes (por simplicidad supondremos costes = tiempos):

	Matriz de tiempos					Matriz de costes				
	0	1	2	3	4	0	1	2	3	4
0	0	2	5	3	2	0	2	5	3	2
1	2	0	6	5	3	2	0	6	5	3
2	5	6	0	9	7	5	6	0	9	7
3	3	5	9	0	4	3	5	9	0	4
4	2	3	7	4	0	2	3	7	4	0

Puesto que el número de clientes es reducido, y que la restricción de capacidad del vehículo impide visitar a más de dos clientes, y que la restricción en tiempo obliga además a que el cliente 2 se visite solo, la cantidad de rutas factibles es limitada, y se reduce a las siete representadas en la Figura 2.2.

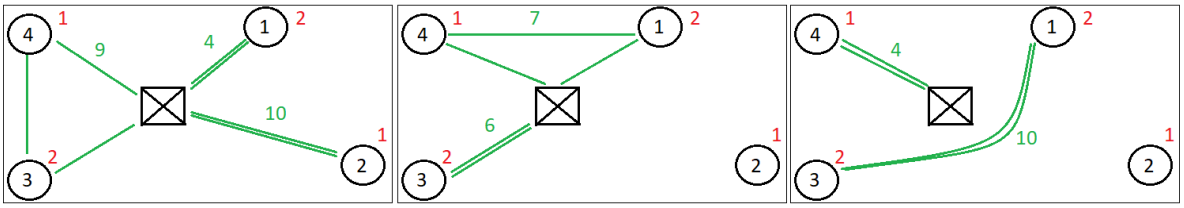


Figura 2.2: Rutas factibles y coste y tiempo asociados (en verde)

A continuación, elegiremos aquella combinación de rutas que garantice que todos los productos son enviados a los clientes. Para el cliente 2, vemos que sólo hay una ruta que pase por él (la ruta 0-2-0, con un coste de 10), con lo cual ésta deberá recorrerse. Por otra parte, nos quedan cuatro opciones:

- Rutas 0-1-0, 0-3-0 y 0-4-0, con un coste total de 14.
- Rutas 0-1-3-0 y 0-4-0, con un coste total de 14.
- Rutas 0-1-0 y 0-3-4-0, con un coste total de 13.
- Rutas 0-3-0 y 0-2-4-0, con un coste total de 13.

Es decir, hemos reducido el problema a elegir entre cuatro combinaciones de rutas, donde cualquiera de las dos últimas opciones es óptima. El coste total es de 23.

Métodos heurísticos

Si el problema es de alta dimensión, en torno a más de 100 clientes (granjas), los métodos exactos no son eficientes para encontrar la solución en un tiempo razonable. En general, los problemas encontrados en la vida real son de mayor tamaño y necesitan ser resueltos de forma rápida, por lo que se utilizan técnicas que proporcionan una buena solución (próxima al óptimo), mediante métodos heurísticos. Éstos, pueden dividirse en tres clases: construcción de rutas, mejora de rutas, y metaheurísticos, según la división hecha por Laporte y Semet, 2002.

Construcción de rutas

Aunque las técnicas metaheurísitcas, que veremos más adelante, son capaces de partir de una solución inicial aleatoria no factible y encontrar una buena solución, exploraremos soluciones iniciales factibles, con el fin de hacer más rápida la convergencia. Esta solución inicial se obtendrá mediante técnicas heurísticas de contrucción de rutas. En nuestro caso particular, puesto que tenemos varias jornadas, con un límite en la capacidad de fabricación en cada período de tiempo, empezaremos por dividir los clientes en jornadas, de la siguiente forma:

> Ordenando los clientes de mayor a menor urgencia, asignamos los primeros clientes a la primera jornada, hasta llegar al límite de capacidad de fabricación. Los siguientes, por orden, serán agregados a la segunda jornada, sin sobrepasas dicho límite. Se continúa el proceso hasta haber asignado todos los clientes a una jornada. De esta forma, se respetarán las urgencias en los pedidos.

Una vez que tengamos los clientes organizados en jornadas, utilizaremos técnicas heurísitcas de construcción de rutas para cada jornada, entre las cuales destacan las siguientes:

- **Rutas de ida y vuelta.** Cada cliente i es visitado por una ruta del tipo 0-i-0, es decir, de ida y vuelta desde la fábrica hasta el cliente.

- **Selección aleatoria.** Cada cliente se selecciona aleatoriamente para cada ruta, teniendo en cuenta las restricciones de capacidad del vehículo y tiempo de viaje.

- **Heurísticas de ahorro.** Otra forma de obtener una solución inicial factible, es mediante el algoritmo propuesto por Clarke y Wright, 1964:

  Paso 1. Partimos de n rutas de la forma $(0, i, 0)$, $\forall i \in \mathcal{N}$. Es decir, para cada cliente, se realiza un viaje de ida y vuelta desde la fábrica.

  Paso 2. Computamos los ahorros de añadir a cada ruta un nuevo cliente, uniendo i y j, $\forall i, j \in \mathcal{N}$, siendo 0 la fábrica:

  $$s_{ij} = c_{i0} + c_{0j} - c_{ij},\ i \neq j$$

  Paso 3. Ordenamos los ahorros de forma decreciente.

  Paso 4. Consideramos las dos rutas con mayor ahorro s_{ij} positivo y que contengan los arcos $(0, i)$ y $(j, 0)$. Es decir, serán rutas de la forma $(0, i, \ldots, 0)$ y $(0, \ldots, j, 0)$. En caso de que la unión de ambas rutas mediante el arco (i, j) sea factible (es decir, que no sobrepase el límite de capacidad del vehículo ni el de tiempo de ruta), las unimos, obteniendo la ruta $(0, \ldots, i, j, \ldots, 0)$.

  Paso 5. Repetir el paso anterior hasta que no existan más uniones posibles con ahorro positivo.

Pondremos pequeño un ejemplo de una sola jornada. Supongamos que tenemos una fábrica que sirve a nueve socios, representados en la Figura 2.3, donde su tamaño es proporcional a la cantidad de compartimentos que necesita para ser servido. Además, disponemos de vehículos homogéneos para realizar las rutas, con 6 compartimentos cada uno, de capacidad igual a 3 toneladas por tolva, y con límite de recorrido de 800 minutos. Recordemos que todos los clientes deben ser visitados exactamente una vez, y que en esa visita toda su demanda debe ser satisfecha.

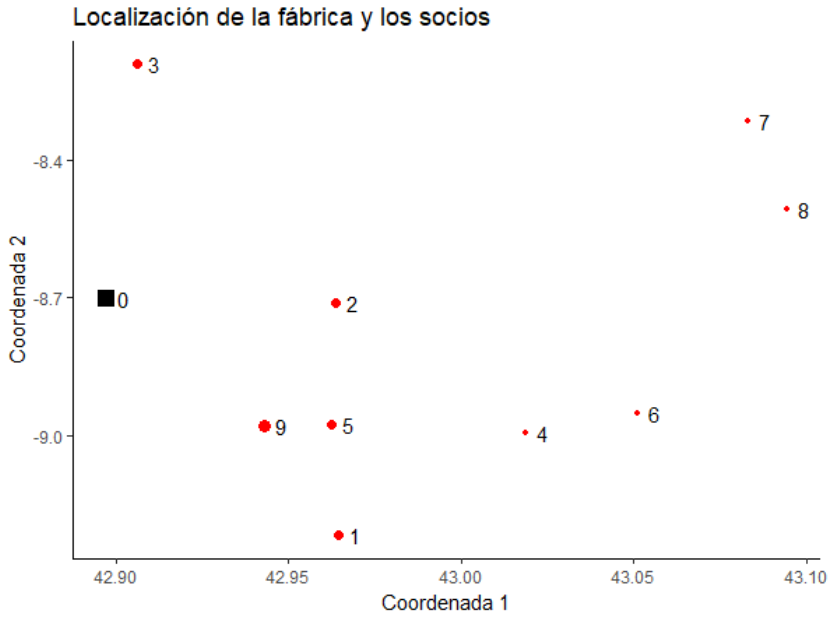


Figura 2.3: Posición de la fábrica y los socios para este ejemplo

Podemos ver las demandas, posiciones y tiempos de carga y descarga en la siguiente tabla. Además, las distancias (o costes) y tiempos serán proporcionales a las distancias euclídeas entre cada dos puntos.

id_granja	Coordenada1	Coordenada2	ton	compartimentos_necesarios	tiempo_CD
1	42.96458	-9.217867	5.1	2	51.0
2	42.96351	-8.709952	4.1	2	49.2
3	42.90616	-8.192431	4.1	2	28.7
4	43.01865	-8.993463	2.0	1	20.0
5	42.96239	-8.975249	5.1	2	35.7
6	43.05120	-8.950567	1.5	1	12.0
7	43.08283	-8.315068	2.6	1	31.2
8	43.09426	-8.506472	1.2	1	9.6
9	42.94307	-8.978493	7.1	3	56.8

Partimos de las rutas del tipo 0-i-0, para cada cliente i, y computamos los ahorros (s_{ij}) de unir cada dos clientes i y j, obteniendo la siguiente tabla de ahorros, ordenados de forma decreciente:

i	j	s_ij
3	7	213.6648585
1	4	178.5236204
9	1	165.2502635
1	5	164.7631615
4	6	163.4981263
9	5	159.7606584
4	5	158.5564576
1	6	156.8242880
9	4	153.0797031
8	7	152.0178266
5	6	141.9097959
9	6	135.9642250
8	3	124.3460879
8	6	36.4884101
6	2	30.3252852
8	2	30.2076631
8	4	29.6153376
4	2	27.5604042
6	7	25.1278966
2	7	24.5902378
5	2	24.2263852
1	2	23.2604277
9	2	22.8519487
8	1	22.0102065
8	5	21.0663680
4	7	18.7003840
8	9	18.3183285
3	2	16.4423014
5	7	11.4775973
1	7	11.4512590
9	7	9.2498648
3	6	8.8852902
3	4	4.8577095
3	5	1.6812983
9	3	0.8753202
1	3	0.8195168

En primer lugar, unimos los clientes 3 y 7, ya que son los que más ahorro aportan, y cumplen las restricciones. A continuación, unimos los clientes 1 y 4, y los clientes 1 y 9. A continuación, uniríamos los clientes 1 y 5 si no incumplieran la restricción de capacidad del vehículo. Continuamos el proceso, cuya secuencia podemos ver en la Figura 2.4, de izquierda a derecha, y de arriba a abajo.

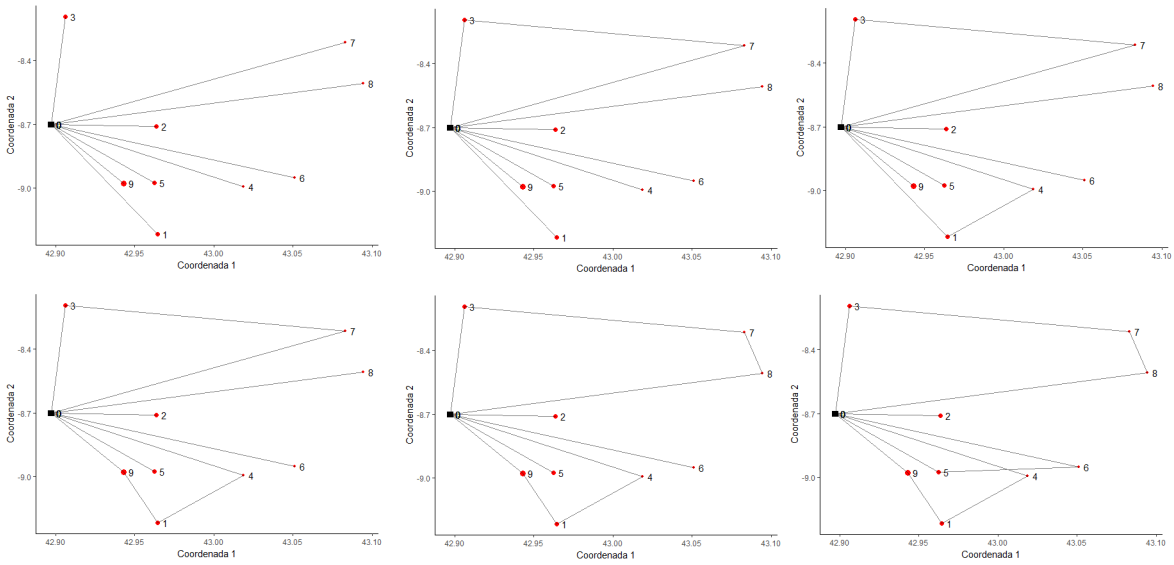


Figura 2.4: Pasos del algoritmo de Clarke and Wright

Finalmente, obtenemos las rutas representadas en la Figura 2.5, con un coste total de 872,7.

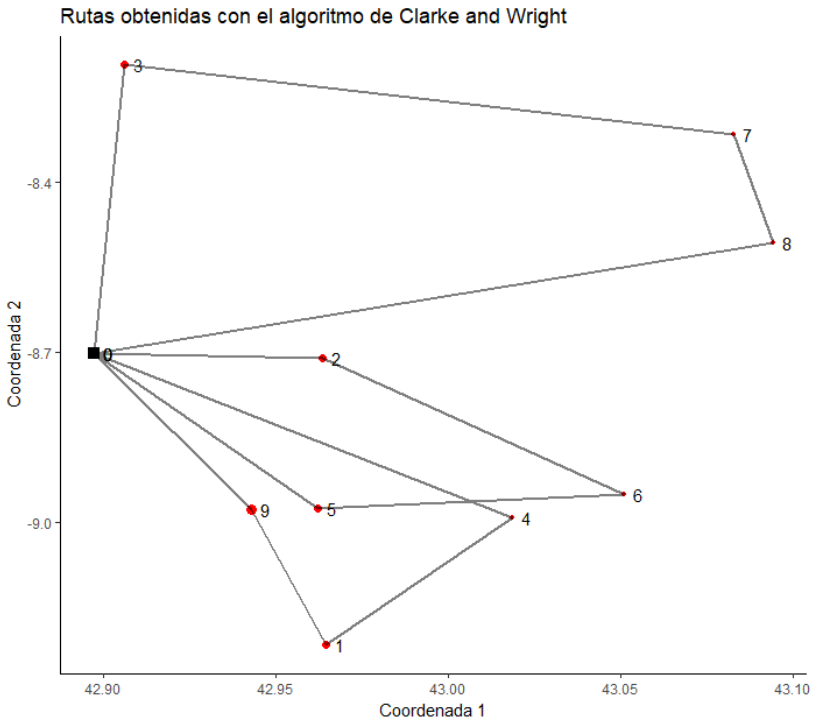


Figura 2.5: Rutas obtenidas mediante el algoritmo de Clarke and Wright

• **Heurísticas de inserción.** Para realizar la construcción de las rutas mediante este método, se sigue el siguiente mecanismo, propuesto por Mole y Jameson, 1976:

Paso 1. Empezamos con todos los clientes sin rutas.

Paso 2. Creamos una ruta $(0, i, 0)$, donde $i \in \mathcal{N}$ es el cliente con menor distancia a la fábrica.

Paso 3. Para cada ruta, calculamos el coste de agregar el cliente l (entre los que aún no disponen de ruta) entre los clientes i y j (que ya estén en una ruta común):

$$h_{ilj} = c_{il} + c_{jl} - c_{ij}.$$

Paso 4. El menor coste corresponderá con el mejor lugar para colocar el cliente l, en caso de que la ruta sea factible (no sobrepase la capacidad ni tiempo del vehículo). En caso contrario, dividimos la ruta en dos. Si a pesar de haber dividido la ruta, sigue sin ser factible añadir dicho cliente, lo dejamos con una ruta individual.

Paso 5. Repetimos los pasos 3 y 4 hasta que todos los clientes tengan una ruta asignada.

• **Vecino más cercano.** Este método consiste en:

Paso 1. Partimos de todos los clientes sin ruta.

Paso 2. Creamos la ruta compuesta por el arco $(0, i)$, donde $i \in \mathcal{N}$ es el cliente con menor distancia a la fábrica, que aún no pertenece a ninguna ruta.

Paso 3. A continuación, visitamos el cliente j más cercano a i entre los que aún no pertenecen a ninguna ruta y siempre que no sobrepase el límite de capacidad ni tiempo del vehículo. Tendremos entonces: $(0, i, j)$.

Paso 4. Repetimos el proceso hasta que no sea factible añadir ningún nuevo cliente a la ruta, y la cerraremos volviendo a la fábrica. Creamos una nueva ruta como en el paso 2.

Paso 5. Repetimos hasta que todos los clientes pertenezcan a alguna ruta.

• **Heurísticas de dos fases.** El algoritmo de Fisher y Jaikumar, 1981, propone dividir el problema en dos fases. En la primera, realizaremos clusters de clientes, mientras que en la segunda resolveremos un Problema del Viajante:

Paso 1. Elegimos aleatoriamente T vértices diferentes, que se corresponderán con los centroides de las T rutas o clusters. Denominaremos i_t a cada cliente centroide, y Cl_t a cada cluster, $t \in \mathcal{T}$.

Paso 2. Computamos el coste de añadir a cada cluster Cl_t un nuevo cliente j de los que aún no están asignados:

$$h_{Cl_t j} = c_{0j} + c_{ji_t} + c_{i_t 0} - c_{0i_t},\ i \neq j.$$

Paso 3. Resolvemos el problema de asignación generalizada con costes dados por $h_{Cl_t j}$, demandas d_{kj} y capacidad del vehículo C.

Paso 4. Una vez que los clientes estén clasificados en los diferentes clusters, resolveremos un Problema del Viajante para cada cluster.

Existen multitud de métodos aparte de los mencionados, como pueden ser los algoritmos de pétalo (Foster y Ryan, 1976), que genera el conjunto de rutas factibles con sus costes asociados y las combina para obtener una buena solución, o el algoritmo propuesto por Gillet y Miller, 1974, denominado algoritmo de barrido, que divide los clientes en clusters rotando en el sentido de las agujas del reloj, con la fábrica como centro.

Mejora de rutas

Una vez que tengamos un conjunto de rutas iniciales, podremos mejorar la solución ya existente intercambiando nodos o arcos dentro de una misma ruta o entre varias rutas, con los mecanismos que podemos ver a continuación:

Movimientos intra-ruta: En los operadores intra-ruta interviene una única ruta. La intención sería resolver un Problema del Viajante para cada una, pero debido a que es un problema NP-Duro, necesitaríamos utilizar métodos heurísticos para cada ruta. De esta forma, los operadores diseñados para mejorar las rutas en el Problema del Viajante pueden ser empleados aquí.

El operador λ-opt, propuesto por Lin, 1965, consiste en eliminar λ arcos de una ruta, y volver a conectar los nodos mediante λ arcos de una forma diferente, como podemos ver en la Figura 2.6. Lo más habitual es tomar $\lambda = 2$ o 3, ya que un mayor valor supondría demasiada carga computacional.

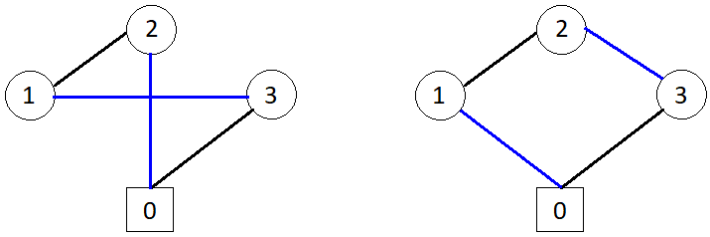


Figura 2.6: Operador λ-opt con $\lambda = 2$, donde los ejes (0,2) y (1,3) son reemplazados por (0,1) y (2,3)

Movimientos inter-ruta: A la hora de clasificar los operadores intre-ruta, Van Breedam, 1994, propuso las siguientes categorías:

String cross (SC) o cruce de arcos, donde se intercambian dos nodos o cadenas de nodos mediante el cruce de dos arcos (Figura 2.7).

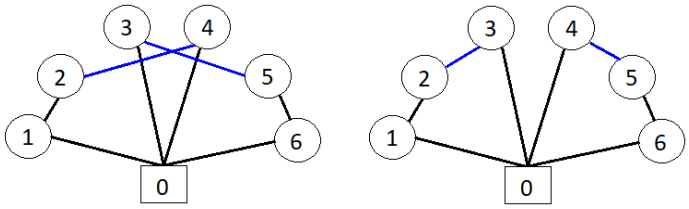


Figura 2.7: String cross, intercambiando los nodos 3 y 4 entre ambas rutas

String exchange (SE) o intercambio de arcos. Se intercambian cadenas de vértices entre rutas (Figura 2.8).

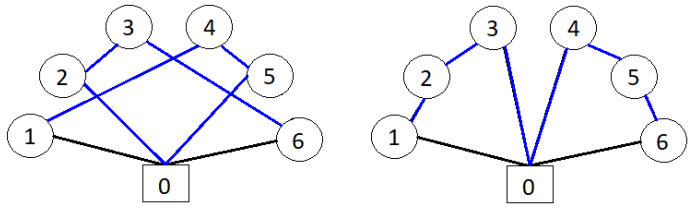


Figura 2.8: String exchange, intercambiando las cadenas (2,3) y (4,5) de una ruta a otra

String relocation (SR) o reubicación de arcos. Una cadena de k clientes consecutivos se traslada de una ruta a otra, normalmente compuesta por 1 o 2 nodos (Figura 2.9).

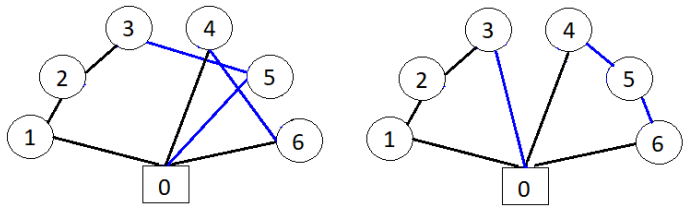


Figura 2.9: String relocation, trasladando el nodo 5 de una ruta a otra

String mix (SM) o mezcla de arcos, consiste en tomar el mejor movimiento entre los dos anteriores.

Movimientos b-ciclos,k-transfer, propuestos por Thompson y Psaraftis, 1993. En este movimiento consiste en una permutación circular de b rutas, donde k clientes de cada ruta se trasladan a la siguiente, como podemos ver en la Figura 2.10.

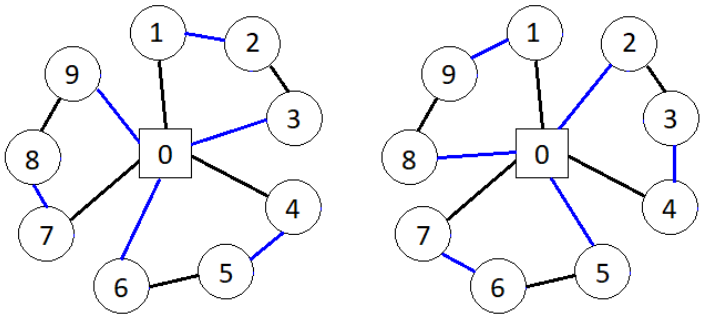


Figura 2.10: 3-ciclos,1-transfer: Se cambia un cliente de cada ruta entre tres rutas.

Metaheurísticos

Lo métodos metaheurísticos pueden dividirse, principalmente, en tres grandes grupos:

• **Búsqueda local.** Estos métodos exploran el espacio de soluciones moviéndose de iteración en iteración desde una solución a otra, entre sus vecindades. Destacan Recocido Simulado (SImulated Annelaing, SA) y Búsqueda Tabú (Tabu Search, TS)

• **Búsqueda de población.** Consideramos una población de soluciones, que serán combinadas entre sí con el fin de generar soluciones mejores. El método más importante es el Algoritmo Genético (Genetic Algorithm, GA), propuesto por Holland, 1975.

• **Mecanismos de aprendizaje:** Algoritmos con memoria, donde en cada iteración utilizan la información aprendida previamente. Destacamos el método de Optimización de Colonia de Hormigas (Ant Colony Optimization, ACO), propuesto por Dorigo, 1992.

A continuación, comentaremos los métodos de búsqueda local de forma más detallada, puesto que serán los que empleemos para resolver nuestro problema de rutas.

Recocido Simulado (Simulated Annealing)

Este método, inspirado en las analogías con los fenómenos físicos, fue propuesto por Kirkpatrick et al, 1983. Su principal aplicación es a la hora de resolver problemas relacionados con circuitos eléctricos, como por ejemplo, el problema de obtener la colocación óptima de las piezas para reducir el tamaño de los cables que las unen.

La idea de este método proviene de la similitud con el cambio en el estado de los elementos, en concreto el paso de líquido a sólido, mediante el cambio en la temperatura. Imaginémonos un vidrio en estado líquido (es decir, se encuentra a una alta temperatura) que queremos convertir en sólido, y para ello bajaremos la temperatura. Si este descenso es demasiado rápido, podrían producirse defectos en el cristal (mínimo local), debido a un enfriamiento demasiado brusco. Por lo tanto, para prevenir este hecho, el descenso deberá ser relativamente lento, creándose progresivamente un estado sólido y estable (balance termodinámico), correspondiente a un mínimo global.

De esta forma, la manera de encontrar una buena solución a nuestro problema, será considerando un parámetro de control, que juega el papel de la temperatura.

Para simular el comportamiento de un sistema físico, utilizaremos el algoritmo de Metropolis et al., 1953. Partimos de una solución factible inicial x_{it} (donde it se corresponderá con el número de iteración), y nos movemos a otra solución factible, x, en su vecindad (por ejemplo mediante reubicación o intercambio de arcos). Este movimiento causa un incremento en la función objetivo, dado por $\Delta E = f(x) - f(x_{it})$. Si mejora la función objetivo, es decir, si el incremento es negativo en el caso de minimizar, $f(x) - f(x_{it}) < 0$, se acepta la solución ($x_{it+1} = x$) y se sigue iterando partiendo de esta nueva solución. Si empeora la función objetivo ($f(x) - f(x_{it}) \geq 0$), se aceptará la nueva solución con cierta probabilidad p_{it}, que dependerá de la diferencia $f(x_{it}) - f(x_{it+1})$ y de θ_{it}. La forma más habitual de definir p_{it} es

$$p_{it} = \exp\left(\frac{-[f(x) - f(x_{it})]}{\theta_{it}}\right),$$

donde θ_{it}, llamada temperatura, decrece a medida que aumentan las iteraciones (it), como veremos más adelante.

Una forma de ver si aceptamos o no una solución que produzca un empeoramiento de la función objetivo, sería tomar aleatoriamente un número entre 0 y 1 (es decir, elegir de forma aleatoria una observación de una distribución uniforme en el intervalo $[0, 1]$, $U(0, 1)$). Si éste es menor que p_{it}, se acepta esta solución, y en caso contrario, se rechaza.

En la Figura 2.11 podemos ver cómo se distribuye esta función, es decir, cómo varía la probabilidad de aceptar una solución no factible dependiendo del incremento que produzca en la función objetivo, a cada temperatura.

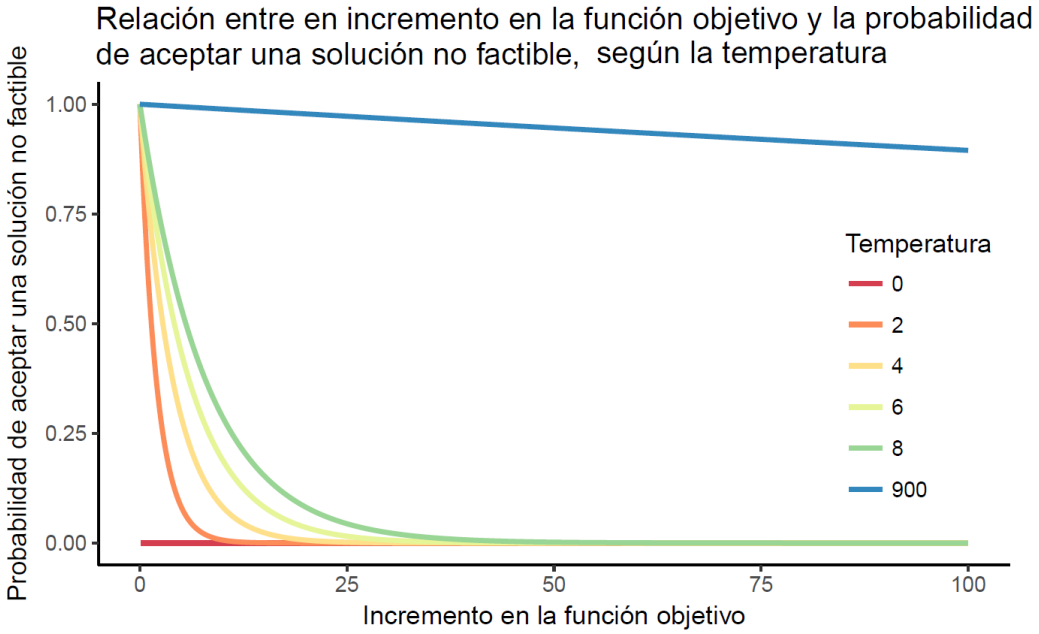


Figura 2.11: Relación entre ΔE y la probabilidad de aceptar la solución, según la temperatura

Podemos observar que, con una alta temperatura (por ejemplo $\theta_{it} = 900$), la probabilidad p_{it} de aceptar una solución que empeore la función objetivo será próxima a 1, aunque el incremento en la función objetivo sea considerablemente grande. Esto implica aceptar la mayoría de movimientos, es decir, supondría un paseo aleatorio por el espacio de soluciones factibles. De esta forma, escaparíamos de un mínimo local. A media que la temperatura decrece, p_{it} se acerca a 0, por lo tanto la mayoría de los movimientos que empeoran la función objetivo serían rechazados, es decir, sólo aceptaríamos movimientos que mejoren la función objetivo (nos acercaríamos al mínimo). De la misma forma, cuanto más se empeore la función objetivo (es decir, cuanto más negativo sea el numerador $-[f(x) - f(x_{it})]$), menos probabilidades habrá de aceptar dicha solución.

Como hemos visto, la temperatura debe disminuirse de forma progresiva para acercarnos al óptimo global. Esto se hará una vez que el sistema esté en balance termodinámico, a cada temperatura.

Por último, el criterio de parada será que se verifique que, tras una serie de iteraciones, el sistema

esté solidificado, es decir, que la temperatura alcance el valor cero, o no se hayan aceptado más movimientos que empeoren la función objetivo en las últimas iteraciones.

La gran ventaja de este método es que, tras una cantidad suficiente de iteraciones, el algoritmo converge a un óptimo global, es decir, se aproxima al balance termodinámico. Su mayor desventaja es la selección de los parámetros:

- la temperatura inicial;
- el criterio para elegir cuándo cambiar la temperatura;
- cuánto bajar la temperatura en cada cambio.

A continuación, veremos cómo podemos elegir estos parámetros de forma inicial, aunque lo ideal es hacer un estudio previo a la implantación del modelo sobre la selección de los parámetros.

Para elegir una temperatura inicial, Dréo et al, 2003, proponen el siguiente método: partimos de una solución inicial y obtenemos aleatoriamente un número dado (por ejemplo, 1.000) soluciones vecinas. Para cada una de ellas, evaluaremos la variación en la función objetivo respecto a la solución de partida. Además, elegimos un ratio de aceptación inicial, τ_0 (por ejemplo, 20 % para una alta temperatura, o 5 % para una baja), y obtendremos la temperatura inicial como aquella que acepta el τ_0 % de las soluciones vecinas elegidas.

Además, Dréo et al, 2003 también proponen la forma de elegir cuándo cambiar la temperatura, que será si se cumple alguna de las dos condiciones:

- se aceptaron $12 \cdot G$ movimientos,
- se intentaron $100 \cdot G$ movimientos,

siendo G el número de grados de libertad (parámetros) del problema. Nosotros testearemos cuál es la mejor cantidad de movimientos aceptados e intentados para bajar la temperatura.

Por otra parte, para controlar el descenso de la temperatura, el método más simple es hacerlo de forma geométrica, es decir: $\theta_{it+1} = \alpha \cdot \theta_{it}$, siendo α una constante (habitualmente, $\alpha = 0{,}9$). Otra alternativa más efectiva es hacer que el descenso sea función de la temperatura: $\theta_{it+1} = \alpha(\theta_k) \cdot \theta_t$, por ejemplo de la siguiente forma:

$$\theta_{it+1} = \left(1 - \theta_k \cdot \frac{\Delta(\theta_k)}{\sigma^2(\theta_k)}\right) \cdot \theta_{it},$$

donde $\sigma^2(\theta_k) = (f^2_{\theta_k}) - (f_{\theta_k})^2$, siendo f la función objetivo, y $\Delta(\theta_k)$ una función que depende del método elegido, aunque lo más sencillo es tomar $\Delta(\theta_k)$ como una constante.

Por último, los posibles criterios de parada son cuando, después de tres cambios sucesivos de temperatura, no se haya aceptado ninguna nueva solución (o bien no haya variado el coste), o cuando la temperatura sea muy próxima a cero.

En la Figura 2.12, podemos ver el esquema que utilizaremos para aplicar este método a nuestro caso. Se lee de arriba a abajo y de izquierda a derecha, a no ser que las flechas indiquen lo contrario. En este caso, la forma de buscar un vecino será intercambiando clientes entre dos rutas de forma aleatoria (se intercambiarán 0, 1 o 2 clientes de cada ruta).

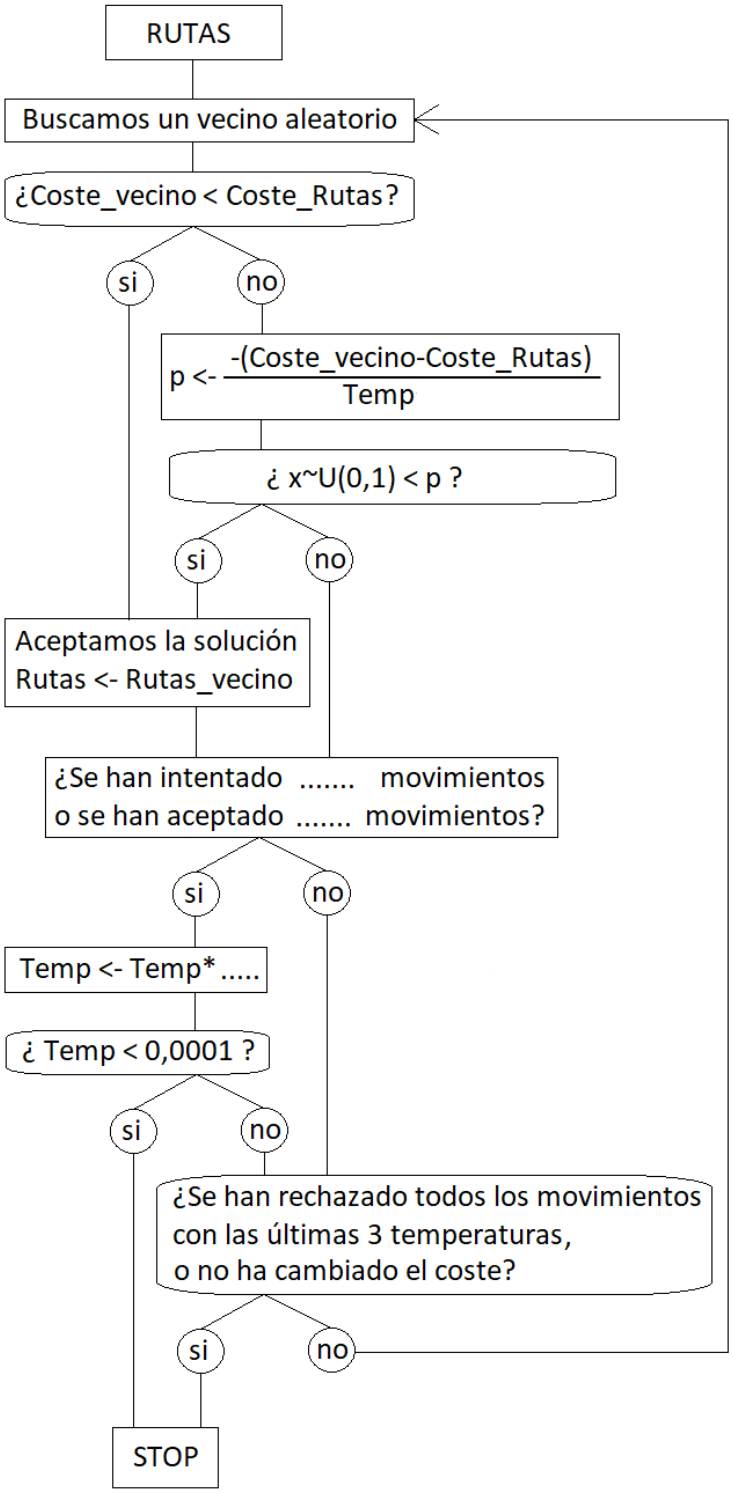


Figura 2.12: Esquema de la aplicación de Simulated Annealing para nuestro Problema de Ruta de Vehículos (los puntos suspensivos indican que el parámetro aún está por seleccionar)

Para ilustrar el funcionamiento de este algoritmo, continuaremos con el ejemplo que empleamos para probar el algoritmo de Clarke and Wright, es decir, partiremos de las rutas de la Figura 2.13.

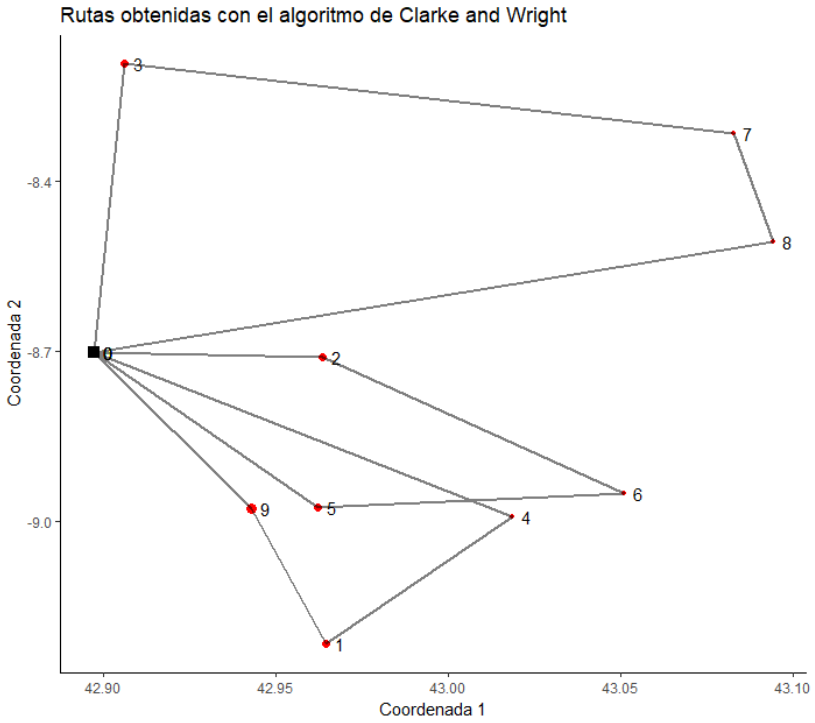


Figura 2.13: Rutas obtenidas mediante el algoritmo de Clarke and Wright

Supongamos que la temperatura actual es de 5, y un movimiento aleatorio podría ser cambiar el cliente 2, que actualmente se encuentra en la ruta 2 - 6 - 5, a la ruta 3 - 7 - 8, obteniendo así las dos nuevas rutas: 6 - 5, y 3 - 7 - 8 - 2, ambas factibles. De esta forma, el coste pasaría de 872,7 a 872,8. Puesto que aumentamos el coste, aceptaremos esta nueva solución con probabilidad:

$$p = \exp\left(\frac{-[872,8 - 872,7]}{5}\right) = \exp\left(\frac{-0,1}{5}\right) = 0,98,$$

es decir, con una muy alta probabilidad aceptaríamos esta nueva solución, obteniendo así las rutas de la Figura 2.14.

Procediendo como hemos descrito, llegamos finalmente a las rutas obtenidas en la Figura 2.15, con un coste de 863.4, es decir, hemos conseguido un ahorro en coste del 1,07 %, sobre la solución obtenida con el algoritmo de Clarke and Wright.

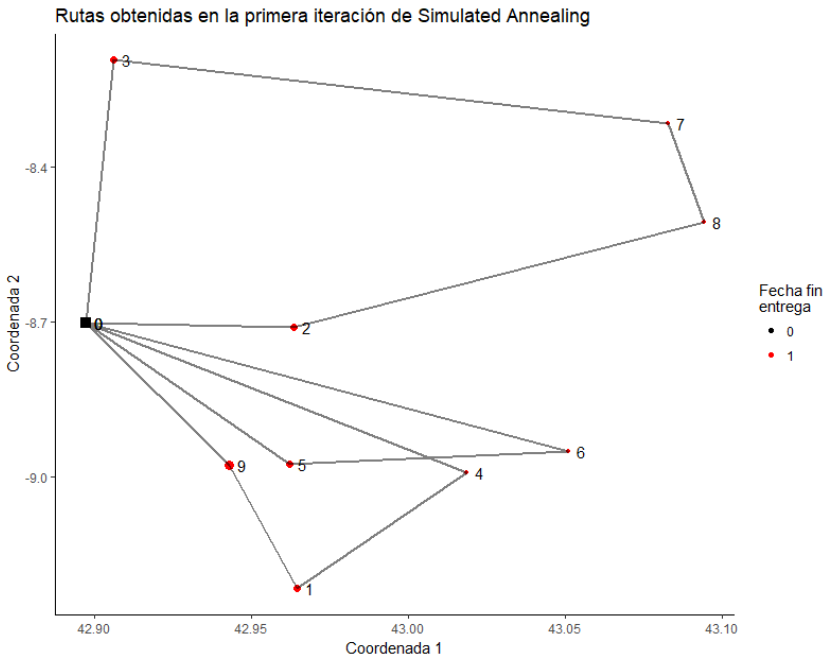


Figura 2.14: Rutas obtenidas en la primera iteración de Simulated Annealing

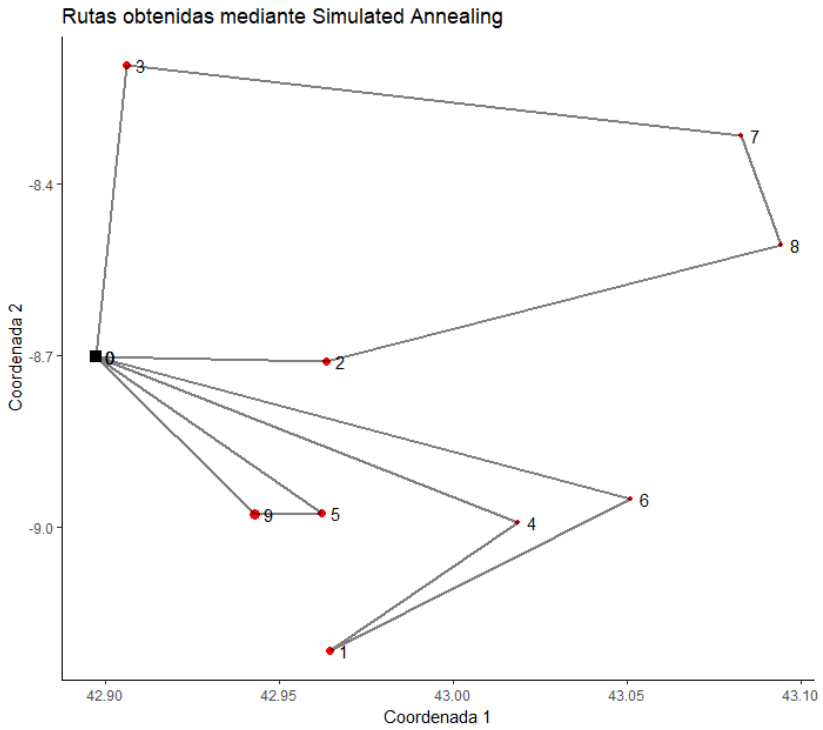


Figura 2.15: Rutas obtenidas mediante la técnica de Simulated Annealing

Búsqueda Tabú (Tabu Search)

Este método fue propuesto por Glover, 1986, inspirándose en ideas desarrolladas durante los años sesenta. Su principal diferencia respecto al método anterior, radica en que la Búsqueda Tabú posee memoria. Aunque tiene multitud de aplicaciones, una de las más importantes es el problema de asignación cuadrática, donde, partiendo de un conjunto de objetos y los flujos entre cada dos elementos, debemos obtener la colocación óptima de éstos en un conjunto de puntos (separados por ciertas distancaias), de forma que se minimice la suma del flujo por la distancia. Este problema puede encontrarse en numerosos casos, tales como la ubicación de las áreas en un hospital o una universidad, las puertas de embarque en un aeropuerto, o incluso las teclas en un teclado de ordenador.

El algoritmo se basa, igual que el Recocido Simulado, en partir de una solución inicial y proceder mediante búsqueda local, realizando cambios en las rutas tales como intercambio o reubicación de arcos, vistos anteriormente.

Dada una solución inicial (no necesariamente factible, como veremos a continuación), analizaremos todos (o una parte de) los movimientos posibles, y elegiremos aquél que genere el mayor descenso en la función objetivo. Puesto que en la mayoría de problemas el número de movimientos puede ser elevado, para tomar una parte de éstos existen diversas opciones:

- la opción más sencilla es una selección aleatoria de dichos movimientos, que será la que empleemos para nuestro caso particular;
- también podemos limitar el tipo de movimiento a realizar (por ejemplo considerando sólo intercambio de dos arcos, o, si se ha hecho un movimiento determinado, en la siguientes iteraciones evitaremos realizar el movimiento inverso);
- una tercera opción, sería considerar todos los movimientos, ordenados de mayor a menor calidad, como una lista de candidatos, y efectuar los mejores cambios en las siguientes iteraciones, sin necesidad de volver a calcular los vecinos en cada una de éstas. Una vez que hayamos efectuado los mejores movimientos, será necesario volver a obtener la lista entera de candidatos.

Por otra parte, comentábamos que este método tiene memoria. Ésta se utiliza para no caer en mínimos locales, tratando de evitar soluciones por las que ya habíamos pasado. Es decir, guardaremos una *Lista Tabú* con las soluciones obtenidas previamente para no volver sobre ellas. Surgen dos problemas importantes a raíz de esta lista, como veremos a continuación.

El primero de ellos es que esta lista crecerá linealmente a medida que aumentan las ieraciones, ocupando así gran parte de la memoria.

Una posible solución, sería almacenar memoria mediante una *Tabla de Discusión*, es decir, memorizar solamente el valor obtenido por la función objetivo, en vez de todas las variables. De esta forma, descartaríamos cada solución vecina que obtenga el mismo valor en la función objetivo que otra solución previa. Además, únicamente memorizaríamos las últimas it iteraciones, siendo it un número constante o variable, para prevenir la aparición de ciclos de longitud it o menor. Si lo tomamos como constante, y lo elegimos demasiado pequeño (memoria a corto plazo), existe el riesgo de visitar una y otra vez las mismas soluciones en bucle. Si es demasiado grande, será

más difícil encontrar una buena solución, debido a la ausencia de movimientos permitidos. Dada esta situación, tenemos varias opciones:

- una buena opción sería modificar *it* durante las iteraciones (podríamos elegir aleatoriamente *it* entre un mínimo y un máximo, y utilizar este *it* durante una o varias iteraciones antes de generar un nuevo valor);
- otra opción es añadir una penalización proporcional a la frecuencia de cada solución (o a la frecuencia al cuadrado), obteniendo así una memoria a largo plazo;
- por último, podemos obligarnos a elegir un movimiento que no se ha elegido todavía (o no se ha elegido en las últimas iteraciones), aunque se obtenga un empeoramiento en la función objetivo, para así escapar de los mínimos locales.

Por lo tanto, podemos combinar ambos tipos de memoria: utilizaremos la memoria a largo plazo para diversificar la búsqueda, es decir, para movernos ampliamente por el espacio de soluciones, y usaremos la memoria a corto plazo (*Lista Tabú* con poca memoria) para intensificar la búsqueda en una región en concreto.

Además, pueden presentarse ocasiones en las que un movimiento tabú sea con el que se obtenga la mejor solución de las últimas iteraciones, y no tenga sentido prohibirlo. En casos como estos, ese movimiento será permitido (es decir, *aspirado*).

El segundo problema de la presencia de memoria, es la posibilidad de quedar 'desconectados' de la solución óptima, debido a que la única forma de acercarnos al óptimo sea pasando por una solución de la lista.

Para evitar esto, podemos considerar pasar por soluciones no factibles (Gendreau et al, 1994), añadiendo a la función objetivo una penalización por incumplir alguna de las restricciones. En el caso del Problema de Rutas de Vehículos (VRP), podríamos añadir penalizaciones por no entregar todos los pedidos. Por ejemplo, por cada cliente que no sea visitado, se incurrirá en una penalización de dos veces la distancia que lo separa de la fábrica. De esta forma, podríamos partir de una solución inicial no factible.

En nuestro problema, podríamos considerar una penalización por cada cliente urgente que no sea servido dentro del intervalo que ha solicitado, otra por el hecho de sobrepasar la capacidad de los vehículos, otra por sobrepasar el límite de tiempo de rutas, y una carta por sobrepasar la capacidad de producción de la fábrica en cada jornada. Además, esta penalización iría incrementándose si se incumplen las restricciones (multiplicándola por $1 + \delta$, $\delta > 0$), o reduciéndose si no se incumplen (dividiéndola entre $1 + \delta$, $\delta > 0$), tal y como proponen Cordeau et al, 2001. Sin embargo, en nuestro caso no consideraremos penalizaciones ni la posibilidad de soluciones no factibles, pues al tener un elevado número de clientes y de rutas posibles, no quedaremos desconectados del resto de soluciones.

Por último, tomaremos como criterio de parada del algoritmo un número máximo de iteraciones. Una vez alcanzado, nos quedaremos con la mejor solución factible encontrada hasta ese momento. Para ello, será necesario ir almacenando las mejores soluciones factibles encontradas durante el algoritmo.

En la Figura 2.16, podemos ver el esquema que utilizaremos para aplicar este método a nuestro caso. Se lee de arriba a abajo y de izquierda a derecha, a no ser que las flechas indiquen lo

contrario. En este caso, la forma de buscar un vecino será intercambiando clientes entre dos rutas de forma aleatoria (se intercambiarán 0, 1 o 2 clientes de cada ruta).

Cabe destacar que el $Coste\_vecino^*$ es el coste de la solución encontrada más una penalización por la frecuencia al cuadrado de dicha solución.

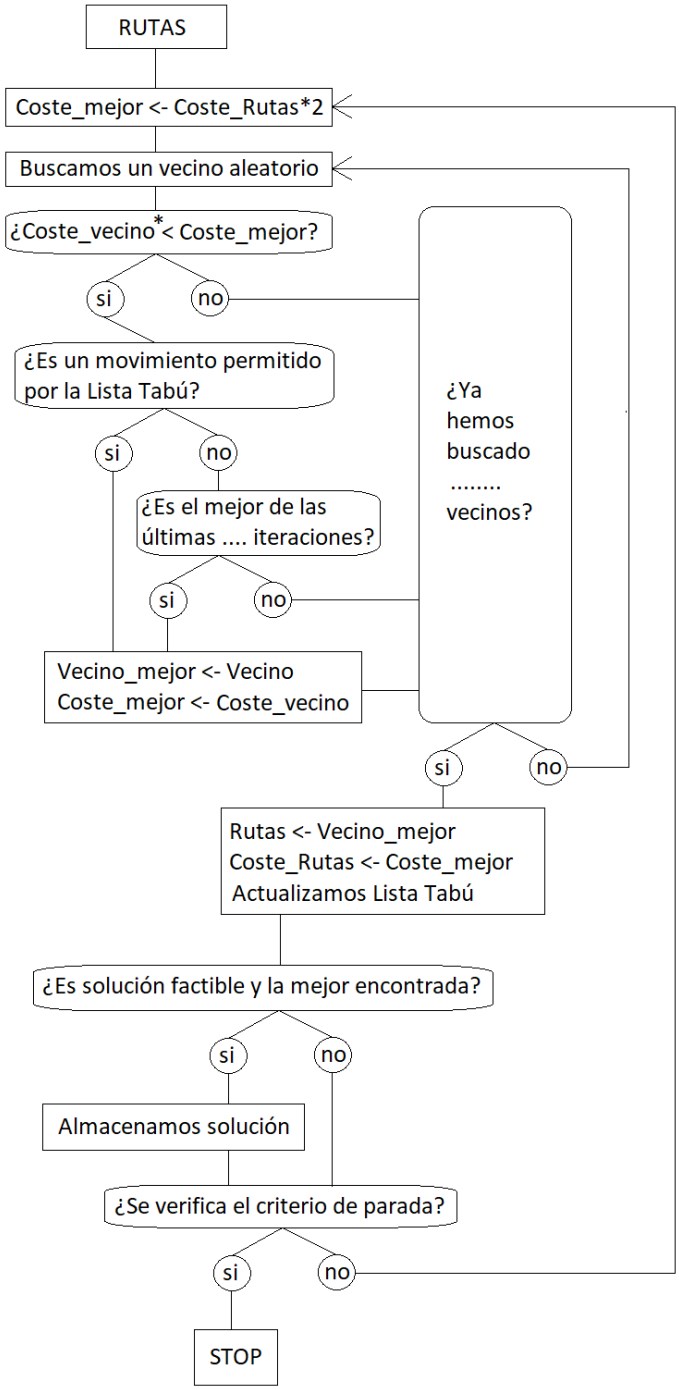


Figura 2.16: Esquema de la aplicación de Tabu Search para nuestro Problema de Ruta de Vehículos

2.2. Problema de asignación de rutas a vehículos

A continuación, propondremos el modelo matemático para la segunda fase de la distribución (asignación de rutas a vehículos), cuya función objetivo será minimizar la cantidad de vehículos empleados en cada jornada. Es decir, resolveremos un problema para cada jornada, donde partiremos de las rutas ya organizadas para distribuirlas en vehículos.

2.2.1. Particularidades

Parámetros

Los parámetros que vendrán dados para cada situación serán los siguientes:

V = número de vehículos utilizados (mayor o igual que 1). Se denotarán con el subíndice v, y el conjunto de todos los vehículos será $\mathcal{V}$. Este parámetro no vendrá dado de antemano, sino que trataremos de minimizarlo. Un límite superior de este valor será la cantidad de rutas realizadas, pues no deberá haber más vehículos que rutas.

R = número de rutas que se van a realizar. Se denotarán con el subíndice r, y el conjunto de todas las rutas será $\mathcal{R}$.

$\mathbf{Tiempo_r}$ = tiempo de transporte de la ruta r, $r \in \mathcal{R}$, contando con el tiempo de viaje más tiempo de carga y descarga del producto.

L = límite de tiempo (carga, descarga y viaje) de cada vehículo en cada jornada.

Variables

Las variables de interés son las siguientes:

$$\mathbf{z_{rv}} = \begin{cases} 1 & \text{si el vehículo } v \text{ realiza la ruta } r;\ r \in \mathcal{R}, v \in \mathcal{V} \\ 0 & \text{en otro caso} \end{cases}$$

$$\mathbf{u_v} = \begin{cases} 1 & \text{si se utiliza el vehículo } v;\ v \in \mathcal{V} \\ 0 & \text{en otro caso} \end{cases}$$

2.2.2. Formulación matemática del modelo

El objetivo será minimizar el número de vehículos utilizados en cada jornada ($t \in \mathcal{T}$), como vemos a continuación:

$$
\begin{aligned}
min \quad & \sum_{v \in \mathcal{V}} u_v \\
& u_v = \max_r z_{rv}, \quad \forall v \in \mathcal{V} && (2.20)\\
& \sum_{v \in \mathcal{V}} z_{rv} = 1, \quad \forall r \in \mathcal{R} && (2.21)\\
& \sum_{r \in \mathcal{R}} Tiempo_r z_{rv} \leq L, \quad \forall v \in \mathcal{V} && (2.22)\\
& z_{rv} \in \{0,1\}, \quad \forall r \in \mathcal{R}, v \in \mathcal{V} \\
& u_v \in \{0,1\}, \quad \forall v \in \mathcal{V}
\end{aligned}
$$

En la restricción (2.20) vemos que, si un vehículo es utilizado, recorrerá al menos una ruta, y viceversa. La restricción (2.21) garantiza que cada ruta es recorrida exactamente por un vehículo.

La restricción (2.22) hace referencia al límite máximo de tiempo que un vehículo puede viajar en cada jornada laboral. Éste no debe ser superado por la suma de tiempo de todas las rutas se que realizan (donde el tiempo de cada ruta viene dado por el tiempo de viajar entre cada dos clientes, más el tiempo de carga y descarga de cada producto en la fábrica y en cada granja):

$$Tiempo_r = \sum_{i,j \in \mathcal{N}^+} x_{ijrt} e_{ij} + \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{K}} y_{ikrt} h_{ik},$$

y será calculado previamente a resolver este segundo problema de optimización.

Para entender esta formulación, pondremos un pequeño ejemplo: pensemos en 5 rutas (representadas en las filas) a repartir en 3 vehículos (en las columnas). A continuación, podemos ver dos posibles repartos de estas rutas.

	1	2	3
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0
5	0	1	0
max	1	1	1

	1	2	3
1	1	0	0
2	0	1	0
3	0	1	0
4	1	0	0
5	0	1	0
max	1	1	0

En la tabla de la izquierda, los tres vehículos recorren alguna ruta, por lo tanto:

$$u_1 = 1, \quad u_2 = 1, \quad u_3 = 1 \quad \Rightarrow \quad \sum_{v \in \mathcal{V}} u_v = 3.$$

En la tabla de la derecha, el vehículo número 3 no se emplea para ninguna ruta, por lo tanto vemos que

$$u_1 = 1, \quad u_2 = 1, \quad u_3 = 0 \quad \Rightarrow \quad \sum_{v \in \mathcal{V}} u_v = 2.$$

De esta forma, minimizar $\sum_{v \in \mathcal{V}} u_v$ equivale a minimizar el número de vehículos a emplear en cada jornada.

2.2.3. Búsqueda de solución

Una vez que disponemos de todas las rutas, y en qué jornada se realizan, trataremos de minimizar la cantidad de vehículos empleados en cada jornada para recorrerlas, teniendo en cuenta que no se debe sobrepasar el tiempo máximo disponible de cada vehículo para viaje, carga y descarga.

El método heurístico que utilizaremos será el siguiente:

Para cada jornada, partiremos del número mínimo de vehículos que debemos emplear. Este número se obtendrá dividiendo el tiempo de todas las rutas de dicha jornada, entre el tiempo máximo por vehículo, y quedándonos con el mínimo entero mayor o igual que esta cifra. Por ejemplo, si la suma del tiempo de todas las rutas son 600 minutos (10 horas), y cada vehículo puede viajar a lo sumo 110 minutos, necesitaremos ($\frac{600}{110} = 5,45$) 6 vehículos como mínimo.

A continuación, ordenaremos las rutas de mayor a menor tiempo, e iremos asignándolas por orden al vehículo que menos tiempo de viaje lleve acumulado.

Una vez que hayamos asignado todas las rutas a vehículos, comprobaremos si se sobrepasa el tiempo permitido en algún vehículo. De ser así, añadiremos un vehículo más y repetiremos el proceso.

2.3. Problema de organización de la producción

Una vez obtenidas las rutas de cada jornada, y sabiendo qué vehículos las realizan, vamos a organizar la producción. Recordemos que en cada jornada se producen las cantidades exactas de los productos que se van a distribuir en dicha jornada, y que mientras no sean cargados en los vehículos, podrán ser almacenados sin incurrir en coste alguno. Al final de cada jornada, este depósito de productos deberá quedar vacío, es decir, no se almacenan productos de un día para otro. Además, cada vez que se cambie la fórmula a producir, las máquinas deberán someterse a un proceso de limpieza. Por lo tanto, produciremos cada fórmula de una sola vez, con el fin de reducir los tiempos de limpieza, y poder fabricar más producto genérico.

2.3.1. Búsqueda de solución

Para encontrar una buena solución, emplearemos el siguiente método heurístico para cada jornada de forma independiente:

- Paso 1. Encendemos las máquinas.

- Paso 2. Elegimos el vehículo con más horas de viaje pendientes, y elegimos su ruta más larga, entre las que aún no se han realizado.

- Paso 3. De esta ruta, vemos qué clientes visita y qué productos se entregan.

- Paso 4. De cada una de estas fórmulas, vemos si ya está fabricada. De no ser así, vemos qué cantidad total hay que fabricar en la jornada para satisfacer la demanda de todos los clientes, y la fabricamos. Se producirá cada fórmula de una sola vez, limpiando las máquinas en cada cambio. Los productos recién fabricados se almacenarán en silos, separados por fórmulas.

• Paso 5. Si el vehículo elegido en el paso 2 se encuentra en la fábrica, lo cargamos, y realizará la ruta seleccionada. Si no se está en la fábrica (por estar haciendo otra ruta), se cargará a su vuelta.

• Paso 6. Si hay suficiente producto para cargar otro vehículo que se encuentre en la fábrica, se cargará y realizará la ruta correspondiente.

• Paso 7. Se repite todo el proceso desde el paso 2, hasta haber fabricado todos los productos y realizado todas las rutas.

• Paso 8. El tiempo sobrante se dedica a la fabricación de producto genérico, guardando tiempo al final de la jornada para la limpieza de las máquinas.

Capítulo 3

Aplicación a un conjunto de datos

Índice

A continuación, veremos la aplicación de estas técnicas a un conjunto de datos para evaluar su funcionamiento. Para poner en práctica los algoritmos anteriormente descritos, se han programado en R desde cero, sin la ayuda de paquetes externos (salvo para lectura de los ficheros, transformación de datos, o para graficar los resultados). Ha sido necesario programar tanto la generación de soluciones iniciales y los movimientos a realizar en las rutas, como las propias técnicas metaheurísticas para encontrar la solución. Para mayor detalle sobre los ficheros en R, podemos consultar el Apéndice A.

3.1. Datos empleados en el estudio

Para poner a prueba los métodos descritos en este trabajo aplicándolos a nuestro problema, utilizaremos un fichero de datos, que contiene información sobre la demanda de 137 clientes para un período temporal de 3 jornadas.

La fábrica. Con una capacidad de producción diaria de 250 toneladas, la fábrica produce las fórmulas que se deben enviar cada día. Dispone de grandes silos donde almacenar el producto recién producido hasta que sea cargado por los vehículos, sin incurrir en gastos de inventario. Sin embargo, no es posible guardar el producto de una jornada a la siguiente, debiéndose enviar a las granjas correspondientes. Además, la fábrica produce pienso genérico durante sus ratos libres, que proporciona una ganancia extra.

El tiempo total de trabajo de la fábrica son 14 horas por jornada laboral (de 7:00h a 21:00h), es decir, 840 minutos. Para producir cada tonelada de producto, necesita 2 minutos y 15 segundos, y para limpieza por cambio de fórmula son necesarios 2 minutos. Por otra parte, el tiempo de carga por tonelada es de 2 minutos, pero no afecta a la fábrica, sino a las rutas, ya que el proceso de carga corre a cargo de los conductores.

Los vehículos. La fábrica cuenta con una flota homogénea de vehículos suficientes para cubrir las necesidades de transporte, aunque será necesario minimizar este número en cada jornada laboral. Con una capacidad total de 18 toneladas, están divididos en 6 tolvas de igual tamaño, es decir, cada tolva tendrá capacidad para 3 toneladas de producto. Cada vehículo dispone de dos conductores que se irán turnando la conducción, y haciendo los descansos pertinentes.

De esta forma, cada camión puede realizar una o varias rutas, siempre que no sobrepase el tiempo máximo establecido en 800 minutos (13 horas y 20 minutos), ya descontados los descansos de los conductores.

Los clientes. En la Figura 3.1 podemos ver dónde se encuentran las granjas respecto a la fábrica (los clientes representados con puntos, y la fábrica con el cuadrado negro); cuál es la urgencia en sus pedidos según el color (en rojo las que deben recibir en la siguiente jornada, el naranja las que pueden recibir hasta en la segunda jornada, y en verde las que tienen de margen hasta la tercera jornada); y cuántas tolvas necesitan para ser atendidas (según el tamaño del punto es proporcional a la cantidad de producto que demandan).

Los productos. La cantidad de fórmulas que intervienen son 138. En media, cada cliente demanda 1,26 productos diferentes, según el histograma que podemos ver en la Figura 3.2. Es decir, 109 clientes demandan solamente un producto; 23 granjas, solicitan 2 productos diferentes; etc.

Estos productos no necesariamente poseen la misma urgencia, sino que un cliente puede demandar cierto producto para recibirlo en la primera jornada, y otro producto con margen entre la primera y la tercera, por ejemplo.

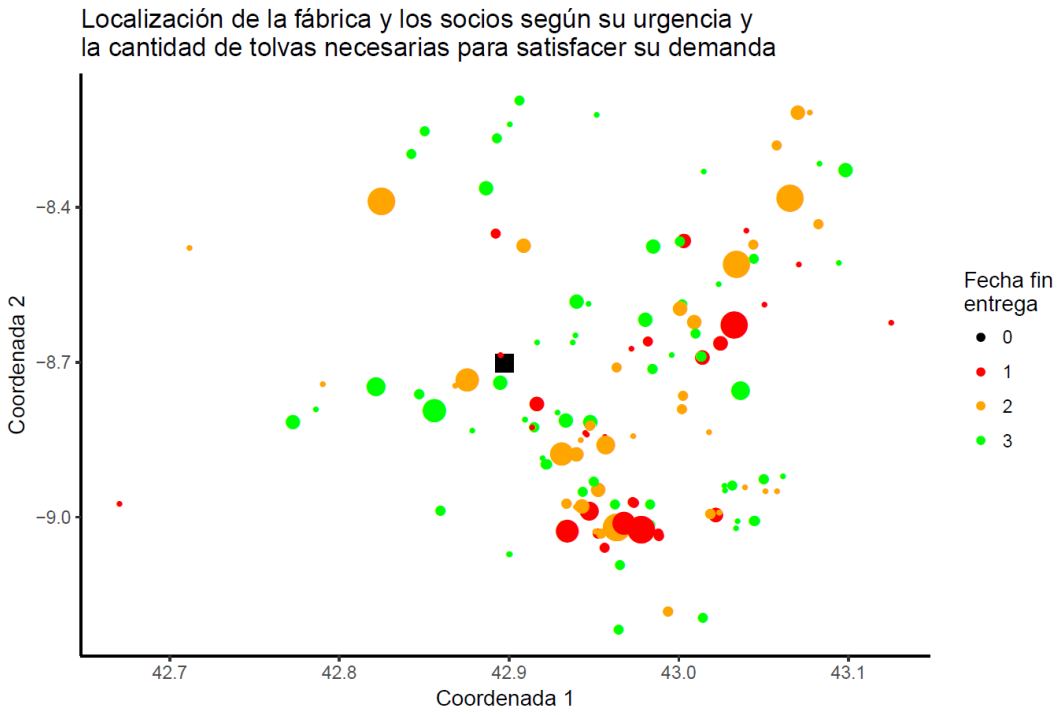


Figura 3.1: Localización de las granjas y urgencia en los pedidos

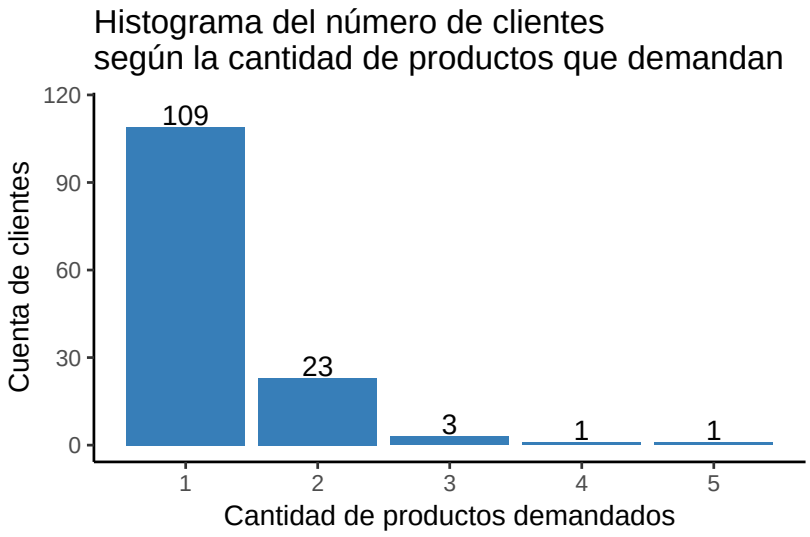


Figura 3.2: Histograma con la cantidad de clientes según el número de productos diferentes que demanda cada uno

Tal y como hemos comentado, si consideramos cada dupla (cliente, producto) como un sólo cliente, otorgándole un nuevo identificador de cliente, tendríamos un total de 173 granjas con demanda de un único producto cada una. Es decir, las rutas deben visitar una y sólo una vez a los 173 conjuntos

de cliente-producto. La demanda total medida en toneladas es igual a 603,62. El número de tolvas necesarias para satisfacer la demanda de las granjas supone un total de 289, cuya distribución puede verse representada en la Figura 3.3. Es decir, 101 clientes necesitan solamente una tolva para ser atendidos; 44 granjas, deberán ser servidas con 2 tolvas; etc.

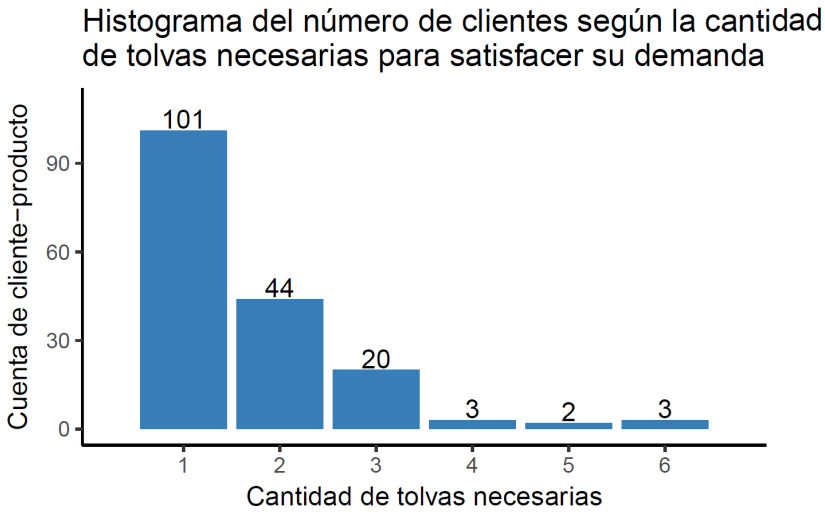


Figura 3.3: Histograma con la cantidad de clientes según el número de tolvas que necesitamos para satisfacer la demanda de cada uno

A continuación, comentaremos cómo se presentan los ficheros de datos, con la información que acabamos de ver:

- Tabla *Solicitudes*: contiene los siguientes campos:

  id_granja: identificador de la granja;

  id_pienso: identificador del pienso que solicita la granja;

  ind_generico: índice que indica si el pienso solicitado es genérico (1) o no lo es (0);

  ton: toneladas solicitadas de dicho pienso;

  tiempo_descarga_ton: tiempo necesario para la descarga de cada tonelada de producto en la granja;

  $fecha\_inicio$: primera jornada en la que puede ser atendido el cliente;

  $fecha\_fin$: última jornada en la que puede ser atendido el cliente;

- Tabla *Tiempos*: matriz del tiempo necesario para ir de una granja a otra (incluyendo la fábrica);

- Tabla *Distancias*: matriz de distancias entre cada dos granjas (incluyendo la fábrica);

- Tabla *Camiones*: contiene los siguientes campos:

  capacidad_vehiculo: capacidad máxima de cada vehículo en toneladas;

  numero_compartimentos: número de compartimentos de cada vehículo (todos del mismo tamaño);

tiempo_maximo_diario_por_vehiculo: tiempo máximo que cada vehículo puede circular en cada jornada (sumando el tiempo de viaje, tiempo de carga y de descarga);

- Tabla *Fábrica*: contiene los siguientes campos:

  produccion_maxima_diaria: capacidad de producción de la fábrica en cada jornada, medida en toneladas;

  tiempo_carga_por_ton: tiempo de carga de cada tonelada de producto en la fábrica;

  tiempo_limpieza: tiempo para la limpieza de las máquinas cada vez que se cambia la fórmula a producir;

  $tiempo\_fabricacion\_por\_ton$: tiempo necesario para la fabricación de cada tonelada de producto;

  tiempo_total_disponible: tiempo total de apertura de la fábrica en cada jornada, englobando la fabricación de los productos y la limpieza de las máquinas.

3.2. Optimización de rutas

En un primer intento por resolver el problema de 137 clientes mediante métodos exactos, vemos que éstos suponen una elevada carga computacional, al ser un problema de gran dimensión y tener incontables rutas posibles. Por lo tanto, nos enfrentaremos a él mediante técnicas heurísticas, vistas anteriormente. Es decir, utilizaremos Simulated Annealing y Tabu Search, partiendo de la solución inicial obtenida mediante el algoritmo de Clarke y Wright como buena aproximación, y de la solución inicial de rutas de ida y vuelta como peor aproximación. Para estos dos métodos, el primer paso será un análisis de los parámetros a elegir, tales como la temperatura inicial, o el número de vecinos a buscar, respectivamente, que dependerán de la solución inicial elegida (Clarke and Wright o rutas de ida y vuelta). Además, analizaremos cómo afecta la cantidad de clientes considerados a dichos parámetros. Una vez realizado el estudio de los parámetros, ejecutaremos el algoritmo para el problema de 137 clientes con los parámetros adecuados, y analizaremos las soluciones obtenidas.

En primer lugar, buscamos una solución factible inicial implementando en R el algoritmo de Clarke y Wright. Para ello, en un primer paso dividimos los clientes en jornadas (sin sobrepasar la capacidad de fabricación de cada jornada), según el intervalo donde cada uno debe recibir el pienso, y repartiéndolos entre las jornadas de forma equitativa (para evitar que una jornada posea demasiados clientes y otra casi ninguno). A continuación, procedemos como hemos descrito anteriormente para aplicar el algoritmo de Clarke y Wright, sin sobrepasar los límites de capacidad y tiempo del vehículo. Podemos ver en la Figura 3.4 cómo quedan las rutas de cada jornada en esta primera aproximación.

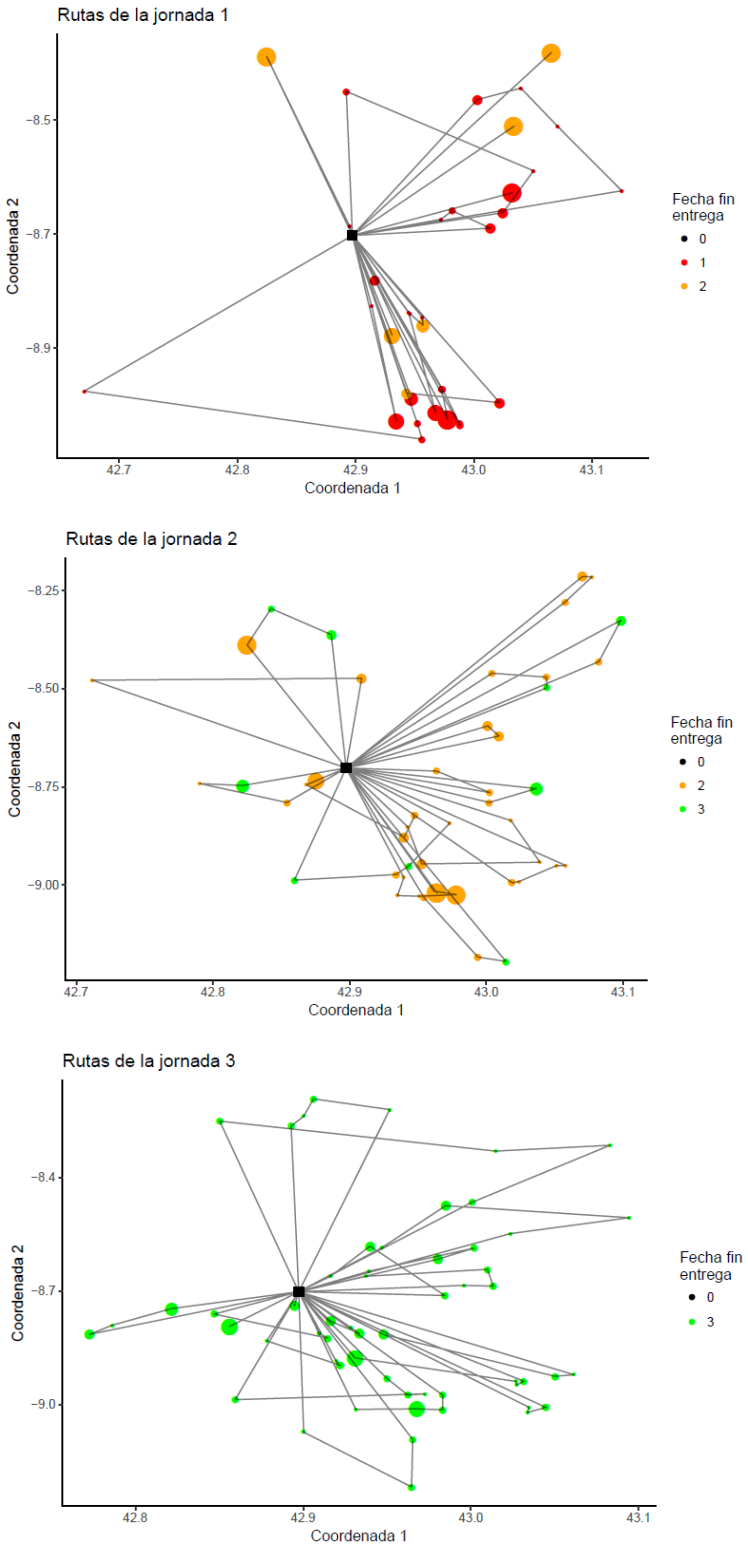


Figura 3.4: Rutas iniciales obtenidas mediante el algoritmo de Clarke y Wright para las tres jornadas. El tamaño de los puntos representa la cantidad de compartimentos necesarios para satisfacer su demanda.

Una vez obtenidas las soluciones iniciales (con coste igual a 8.589 para el algoritmo de Clarke y Wright, y de 24.366 para las rutas de ida y vuelta), trataremos de mejorarlas mediante técnicas metaheurísticas, como Simulated Annealing y Tabu Search.

Para la búsqueda de soluciones vecinas, en ambos algoritmos, los movimientos que emplearemos será el intercambio de clientes entre dos rutas. Para ello, en R se programó este intercambio como definimos a continuación:

> Se eligen aleatoriamente dos rutas (la jornada a la que pertenezcan será indiferente). Una de las rutas puede ser incluso una ruta vacía (con el fin de permitir la partición de dos rutas).
>
> Se seleccionará también de forma aleatoria cuántos clientes de cada ruta van a pasar a formar parte de la otra (sin sobrepasar la cantidad de clientes de la ruta, es decir, no podremos elegir dos clientes si la ruta sólo consta de uno). Para cada ruta, este número será 0, 1 o 2 (sin necesidad de escoger el mismo número para las dos rutas). Por ejemplo, podemos elegir 2 clientes de una ruta que se intercambiarán por 1 cliente de otra. También existe la posibilidad de dejar estos números fijos (por ejemplo que los intercambios sean solamente de un cliente por otro), aunque en nuestro caso será aleatorio, ya que en las pruebas hechas considerando estos números como fijos, obteníamos peores costes y convergencia más lenta.
>
> A continuación, el programa verifica que no se sobrepasa la capacidad de fabricación de cada jornada, ni la capacidad y tiempo de cada ruta, y que respeta las urgencias en los pedidos. En caso de que no se cumpla, se vuelve a repetir el proceso eligiendo dos nuevas rutas.
>
> Además, existe la posibilidad de rutas no factibles, incurriendo en un posible coste por penalización de las restricciones. En este caso no sería necesario comprobar la factibilidad de la solución, pues la aceptaríamos, sea factible o no. El algoritmo programado en R está preparado para ambos casos: bastará indicarle que aceptamos soluciones no factibles, y otorgar una penalización. De todas formas, en este estudio no consideraremos la posibilidad de rutas no factibles.

3.2.1. Simulated Annealing

Para la totalidad de los clientes (137), ejecutamos el algoritmo anteriormente explicado con diferentes valores de los parámetros, y realizando varias pruebas para cada parámetro, con el fin de ver cómo fluctúa la solución encontrada en cada caso. En concreto, los parámetros sobre los que vamos a experimentar son:

- La temperatura inicial, con las siguientes posibilidades: 2, 5 y 15;
- La cantidad de descenso de la temperatura: 0,7, 0,8, 0,9;
- Los movimientos intentados antes de cambiar de temperatura: 50, 75;
- Los movimientos aceptados antes de cambiar de temperatura: 200, 500;
- La solución inicial: algoritmo de Clarke y Wright (con un coste inicial de 8.589), o rutas de ida y vuelta a la fábrica (con un coste inicial de 24.366), con el fin de ver las diferencias entre partir de una solución inicial buena, y de una mala.

Como criterio de parada, utilizaremos el que antes se cumpla entre los dos siguientes: o bien que la temperatura llegue a 0,0001, o bien que se hayan realizado 50.000 iteraciones.

Realizaremos 5 pruebas por cada combinación de parámetros, ya que en cada ejecución del algoritmo con unos parámetros dados, se obtienen soluciones diferentes, por tratarse de técnicas con una componente aleatoria.

Para estas pruebas, utilizaremos un ordenador portátil marca Asus con 16 GB de RAM, Intel Core i7, de 8 procesadores de hasta 3.5GHz de velocidad (estará a medio rendimiento por ejecutar a la vez las pruebas de Simulated Annealing y Tabu Search, que veremos más adelante).

Cabe destacar que la selección de estos parámetros no es de forma aleatoria, sino que se realizó un breve estudio previo con pruebas de varios parámetros, y se han elegido aquellos que aportan mejor solución en un tiempo razonable. Por ejemplo, para temperaturas altas la convergencia era excesivamente lenta y los resultados obtenidos eran similares a los que se consiguen empezando con temperaturas más bajas. Por lo tanto en este estudio solamente entrarán en juego las temperaturas de 2, 5 y 15.

Veamos los resultados obtenidos. En primer lugar, en la Figura 3.5, podemos observar cómo se distribuye el coste según la solución inicial obtenida. Aunque pudiera parecer que siguen una distribución normal, aplicando el Test de Shapiro en R, obtenemos que ninguna de las dos es normal. Vemos también que el coste es menor si partimos de una buena solución inicial (la diferencia es significativa, contrastado con el Test de Wilcoxon Mann Whitney), y que además la varianza es menor.

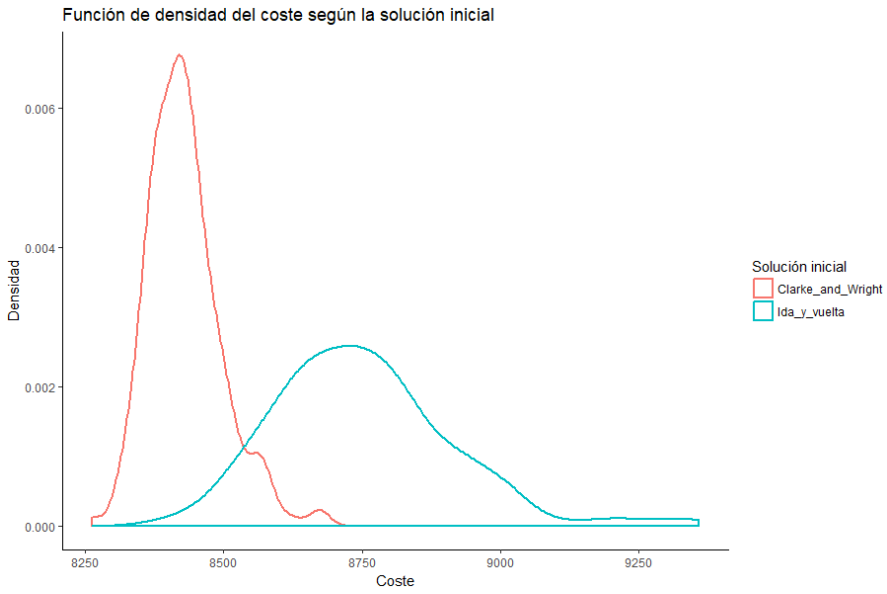


Figura 3.5: Función de densidad del coste según la solución inicial

En la Figura 3.6 podemos ver la distribución del coste según cada uno de los parámetros (cantidad de descenso de la temperatura, temperatura inicial, número de movimientos aceptados y número de movimientos intentados para bajar la temperatura).

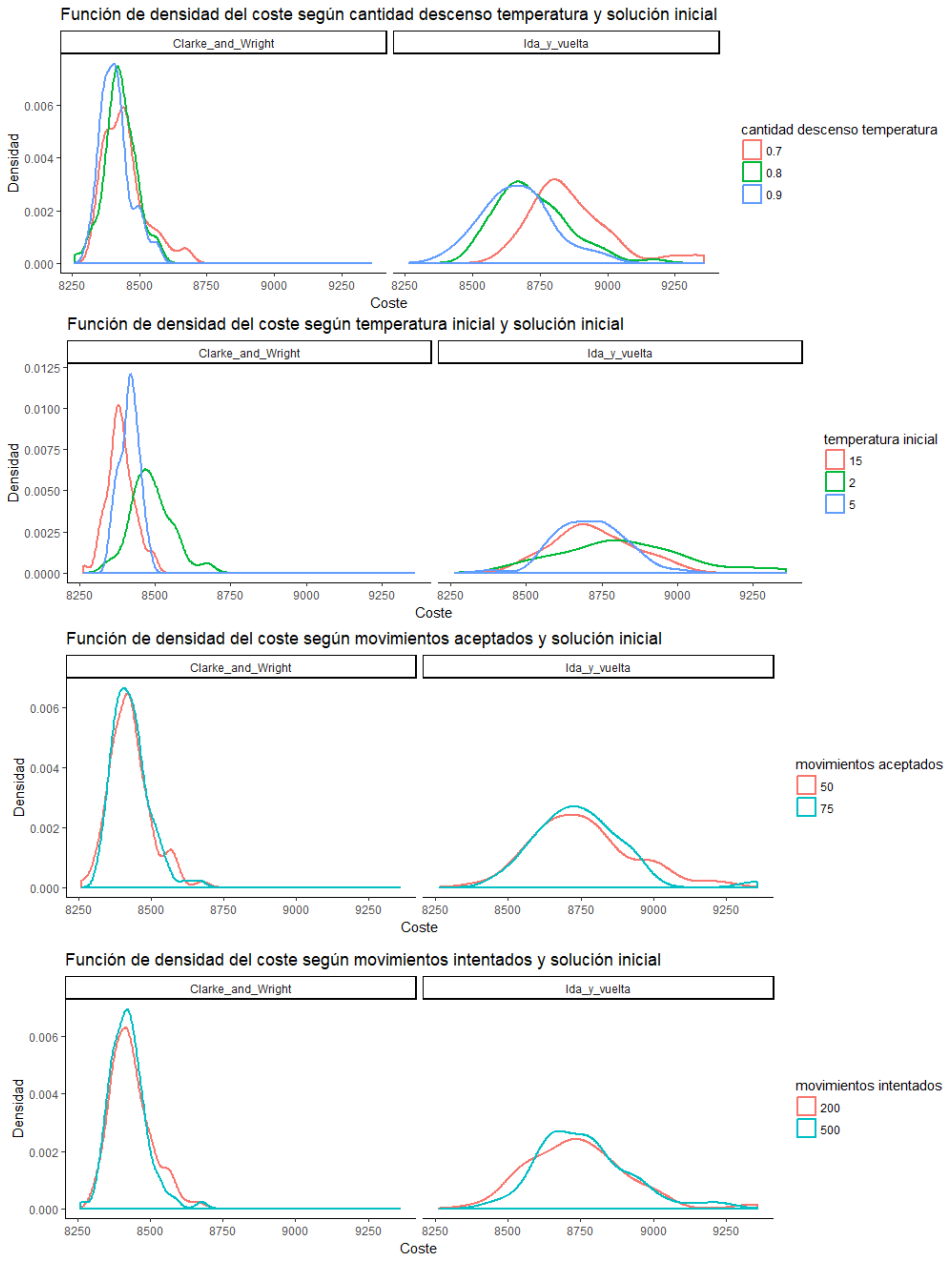


Figura 3.6: Función de densidad del coste según la solución inicial y resto de variables

Vemos que la temperatura inicial, la cantidad de descenso de ésta, y la solución de partida afectan en gran medida al coste obtenido: se obtiene menor coste con una temperatura inicial elevada y con un descenso lento. Es decir, para obtener un buen coste se recomienda empezar con una alta temperatura e ir bajándola poco a poco (por ejemplo, con un descenso del 0.9). Vemos también que el número de movimientos aceptados o intentados para bajar la temperatura no afectan al coste.

Si nos fijamos en la Figura 3.7, observamos que, al partir de la solución inicial obtenida con el algoritmo de Clarke and Wright, la temperatura inicial tiene más peso en el coste que la cantidad de

descenso aplicado. Por el contrario, si la solución de partida se construye con rutas de ida y vuelta, la cantidad de descenso de la temperatura afecta más al coste final que la temperatura inicial.

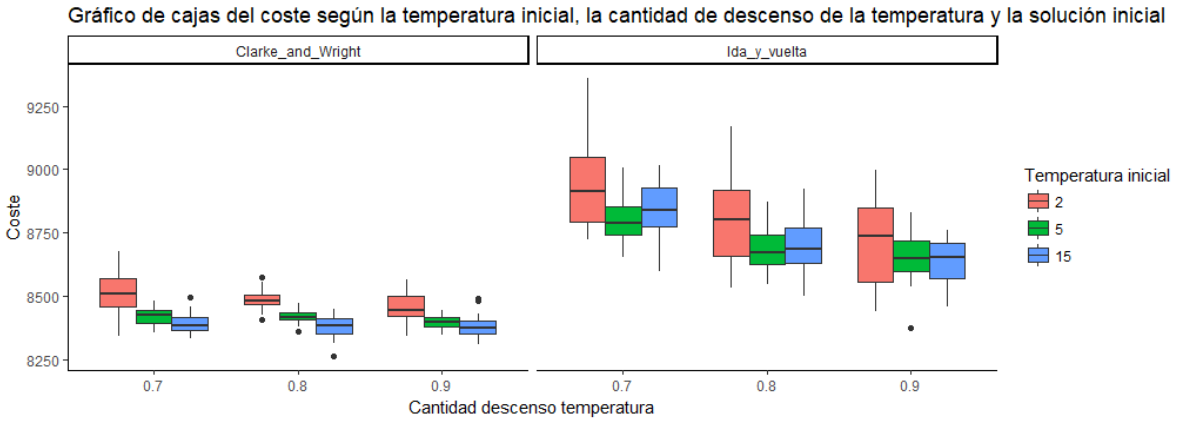


Figura 3.7: Gráfico de cajas del coste en función de la temperatura inicial, la cantidad de descenso de la temperatura, y la solución inicial

Es decir, si la solución inicial es buena, sería recomendable partir de la temperatura alta para permitir movimientos que empeoren la función objetivo, ya que si partimos de una temperatura baja y una solución buena, no habrá casi movimientos que aceptar, y apenas nos moveremos por el espacio de soluciones. Sin embargo, si partimos de una solución mala, como las rutas de ida y vuelta, puesto que la mayoría de las soluciones vecinas serán mejores que ésta, se aceptarán prácticamente todas independientemente de la temperatura inicial (por eso vemos que la temperatura inicial afecta menos que si la solución de partida fuera buena). Sin embargo, es importante que el descenso de la temperatura sea muy lento, pues un enfriamiento brusco causaría que nos quedásemos en una mala solución. Por eso, vemos que la cantidad de descenso de la temperatura es más importante en caso de partir de una mala solución.

Ahora ya sabemos qué parámetros elegir para obtener una buena solución, sin embargo, es importante también tener en cuenta el tiempo de ejecución. Veamos cómo se ve afectado por los diferentes parámetros.

Si observamos las Figuras 3.8 y 3.9, vemos que no hay una relación directa entre coste y tiempo (en minutos). Los parámetros que más afectan al tiempo de ejecución son la cantidad de descenso de la temperatura y la cantida de movimientos intentados. Es decir, un descenso lento de la temperatura provoca un gran aumento en el tiempo de ejecución del algoritmo.

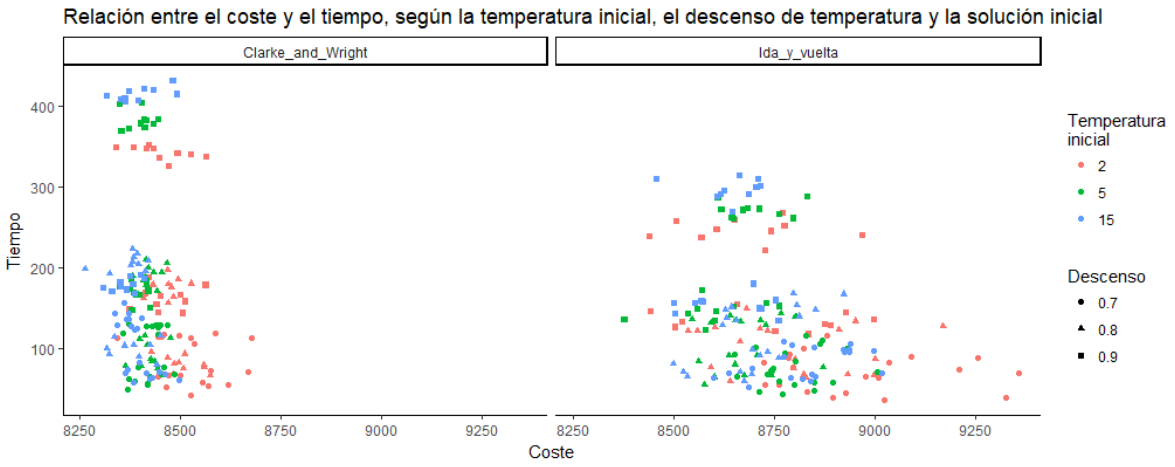


Figura 3.8: Relación entre el coste y el tiempo de ejecución (en minutos), según la temperatura inicial y el descenso

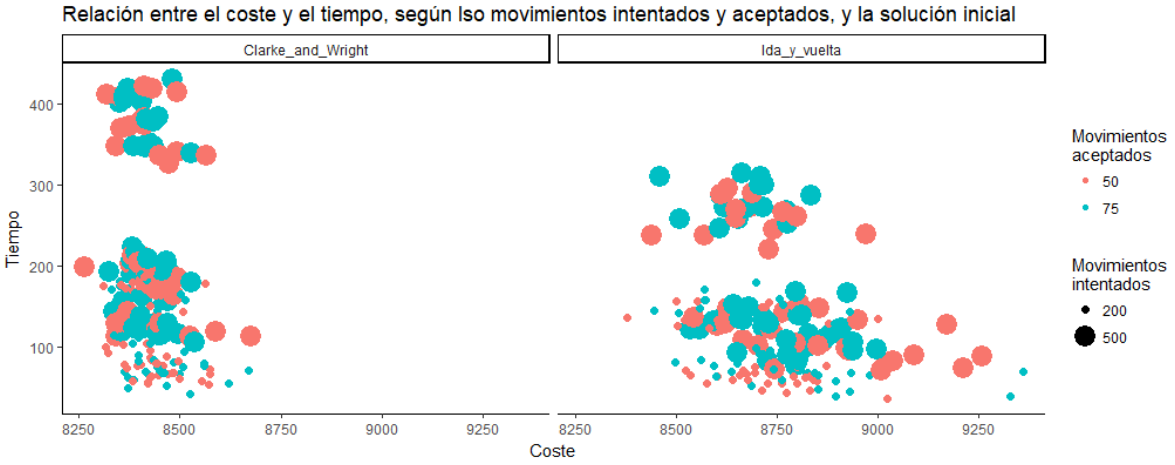


Figura 3.9: Relación entre el coste y el tiempo de ejecución (en minutos), según la cantidad de movimientos aceptados e intentados

Es obvio pensar que si esperamos a tener 500 movimientos intentados antes de pasar a la siguiente temperatura, el algoritmo convergerá de forma más lenta que si simplemente esperamso a tener 200 intentados.

Por otra parte, la explicación de por qué afecta la cantidad de descenso de la temperatura la encontramos en la Figura 3.10. En el eje Y, está representada la temperatura inicial; cada color se corresponde con un porcentaje de descenso; y en el eje X vemos cuántas bajadas de temperatura habrá en cada caso, hasta que el algoritmo finalice. De esta forma, observamos que el parámetro que más afecta es la cantidad de descenso.

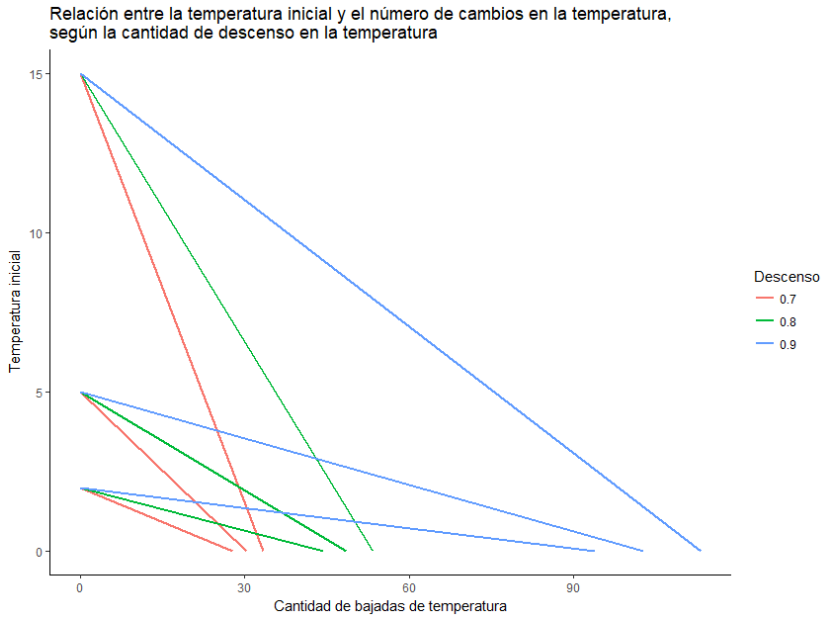


Figura 3.10: Cantidad de bajadas en la temperatura según la temperatura inicial y la cantidad de descenso

Por lo tanto, una buena opción se selección de parámetros podría ser partir de una solución inicial buena (Clarke and Wright) y una temperatura media-alta (5 o 15). Descenderemos la temperatura con un ratio de 0.8 cuando se hayan aceptado 50 movimientos o se hayan intentado 200. De esta forma, obtendríamos una muy buena solución en un tiempo razonable. También se podría descender con un ratio de 0.9, pero la convergencia sería más lenta.

En este caso, la mejor solución encontrada en todas estas pruebas tiene un coste de 8.231 y se ha obtenido con los siguientes parámetros, en un tiempo aproximado de 3 horas y 20 minutos:

- Temperatura inicial: 5;
- Tasa de descenso de la temperatura: 0.9;
- Cantidad de movimientos aceptados: 75;
- Cantidad de movimientos intentados: 200;
- Solución inicial: obtenida mediante el algoritmo de Clarke and Wright.

Con esta solución, pasamos de un coste inicial de 8.589, a un coste final igual a 8.231, lo que supone un ahorro del coste de rutas de un $4,17\,\%$.

El algoritmo devuelve una tabla con todas las rutas, indicando en qué jornada se realizan, cuál es su coste y tiempo de transporte, así como cuántas toneladas son necesarias para satisfacer la demanda de los clientes, y cuántos compartimentos necesita cada ruta:

jornada	V2	V3	V4	V5	V6	V7	V8	V9	coste	compartimentos	ton
1	0	126	103	121	159	0	0	0	219.12427	6	13.90
2	0	37	134	135	0	0	0	0	307.95519	6	12.30
2	0	156	5	6	78	0	0	0	301.11065	6	11.80
2	0	96	95	52	44	105	70	0	198.56945	6	9.30
2	0	160	119	125	0	0	0	0	172.27018	6	14.90
3	0	61	115	50	155	0	0	0	309.65623	6	11.50
3	0	66	99	117	100	0	0	0	208.35288	6	10.40
3	0	87	114	113	145	79	0	0	203.52330	6	8.80
3	0	75	74	118	71	104	0	0	172.57467	6	10.40
3	0	17	26	25	64	0	0	0	165.33360	6	12.30
3	0	1	39	38	30	0	0	0	92.09430	6	13.10
1	0	36	73	139	165	0	0	0	302.26704	6	12.00
1	0	46	123	124	0	0	0	0	262.55529	6	14.40
2	0	158	29	151	0	0	0	0	99.28793	6	12.80
1	0	60	129	131	0	0	0	0	172.67005	6	14.60
1	0	62	97	28	27	12	0	0	92.66744	6	10.20
3	0	122	77	132	136	0	0	0	279.27071	6	11.40
3	0	8	7	172	173	171	0	0	71.76052	5	8.40
3	0	82	98	65	68	0	0	0	204.61377	6	15.20
2	0	102	162	161	163	0	0	0	30.52420	6	10.50
1	0	152	112	58	59	0	0	0	204.98897	6	9.40
1	0	53	22	21	33	0	0	0	78.57642	6	12.40
1	0	116	101	143	0	0	0	0	299.34641	5	12.30
2	0	107	23	24	0	0	0	0	165.05963	5	12.30
2	0	72	144	48	49	69	0	0	194.67671	6	10.08
1	0	43	42	40	16	0	0	0	193.27342	6	11.00
1	0	170	169	138	164	0	0	0	84.52760	6	12.20
1	0	108	109	168	167	0	0	0	77.76511	6	11.70
1	0	19	94	47	81	0	0	0	194.35515	6	10.60
1	0	9	149	10	0	0	0	0	99.45507	6	14.00
3	0	11	88	4	140	0	0	0	313.02868	6	11.80
2	0	80	120	92	91	93	0	0	201.55825	6	10.70
2	0	41	90	89	153	45	0	0	201.03227	6	11.60
3	0	13	111	110	54	55	106	0	116.00611	6	9.48
3	0	32	14	51	31	0	0	0	82.66982	6	10.44
3	0	2	83	84	85	86	0	0	62.77636	6	9.80
3	0	3	34	20	150	0	0	0	100.78228	6	10.92
2	0	142	141	0	0	0	0	0	189.79339	6	15.70
1	0	67	133	0	0	0	0	0	150.67712	5	10.10
1	0	57	147	0	0	0	0	0	171.32947	6	14.30
1	0	56	148	0	0	0	0	0	194.32692	6	15.10
2	0	137	157	0	0	0	0	0	98.71333	6	14.10
2	0	35	127	0	0	0	0	0	92.83061	6	12.20
2	0	128	0	0	0	0	0	0	138.69309	6	16.00
2	0	63	15	0	0	0	0	0	48.37966	5	13.60
1	0	146	154	0	0	0	0	0	196.52146	6	13.30
1	0	166	0	0	0	0	0	0	91.12378	6	17.20
2	0	18	76	0	0	0	0	0	108.73876	6	17.00
2	0	130	0	0	0	0	0	0	213.57033	6	16.10

Podemos ver en la Figura 3.11 las rutas finales encontradas para cada jornada, respectivamente.

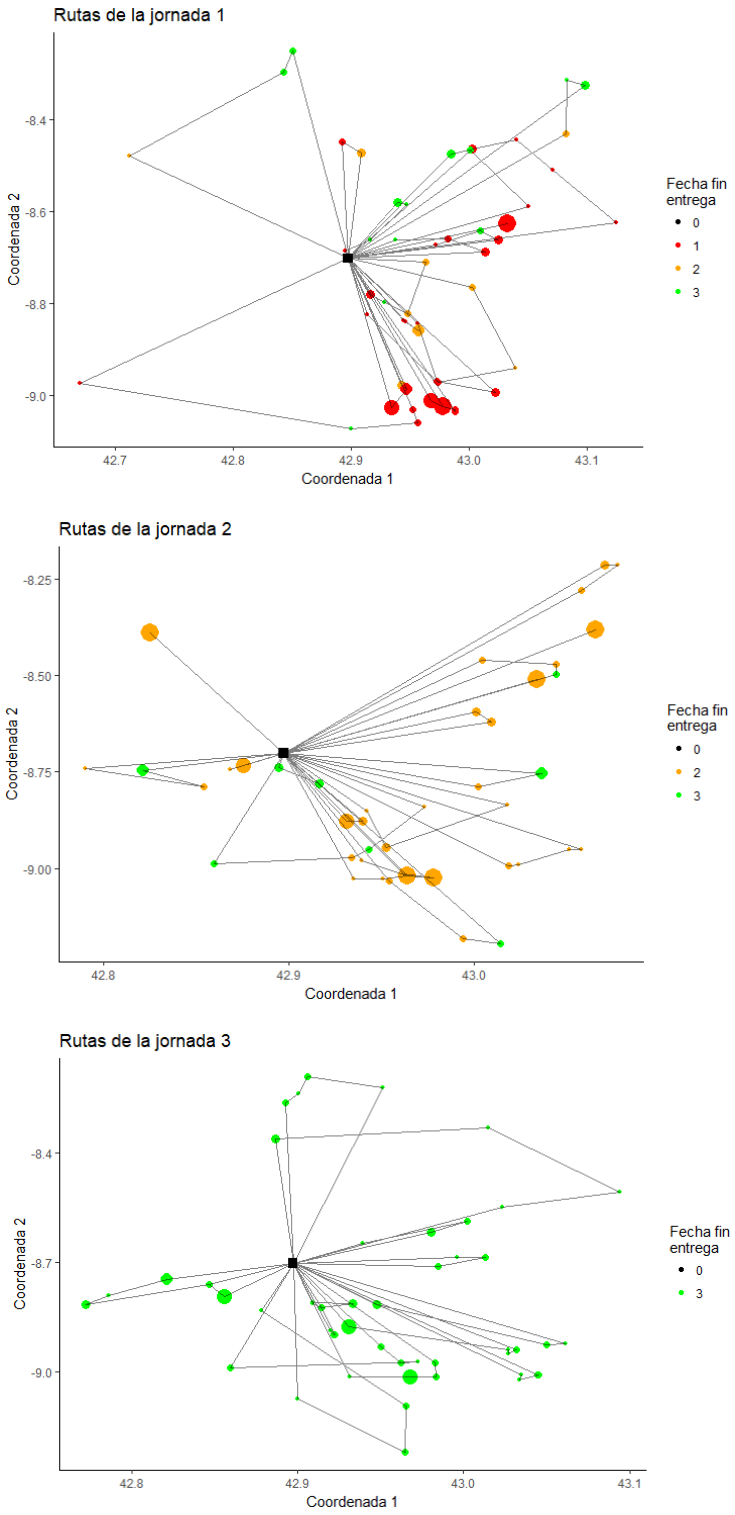


Figura 3.11: Rutas finales obtenidas mediante Simulated Annealing para las tres jornadas. El tamaño de los puntos representa la cantidad de compartimentos necesarios para satisfacer su demanda; y el color, su urgencia.

En la Figura 3.12, tenemos la evolución del coste y de la temperatura durante el algoritmo. Vemos que partiendo de una temperatura inicial de 5, en las primeras iteraciones se aceptan bastantes soluciones que empeoran la función objetivo para movernos por el espacio de soluciones. A medida que la temperatura baja, la cantidad de soluciones aceptadas que empeoran la función objetivo disminuye, acercándonos así al óptimo global del problema.

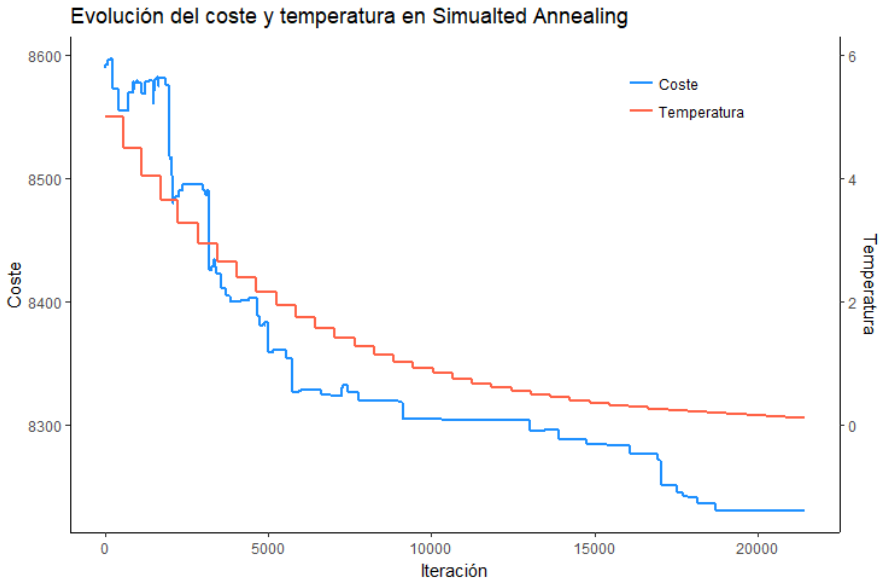


Figura 3.12: Evolución del coste (azul) y temperatura (rojo) en Simulated Annealing

Ahora observemos qué ocurre con los parámetros si en lugar de 137 clientes, consideramos solamente 10 elegidos de forma aleatoria. Repetiremos las pruebas con los mismos parámetros, pero esta vez realizaremos 2 ejecuciones por cada combinación en lugar de 5. En la Figura 3.13 vemos la distribución del coste según la solución inicial considerada.

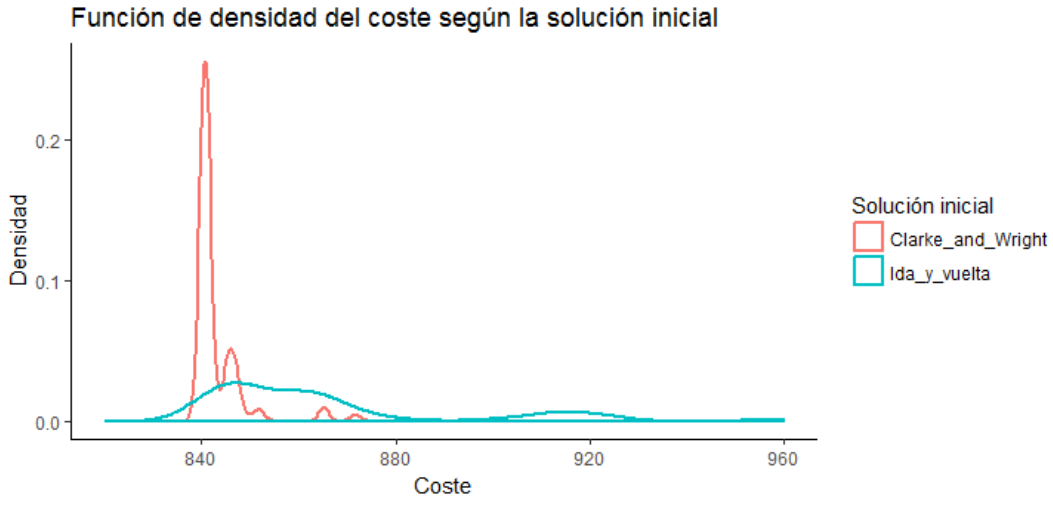


Figura 3.13: Distribución del coste según la solución inicial

En la Figura 3.14 vemos el gráfico de cajas del coste en función de los diferentes parámetros. Con estas dos imágenes, deducimos que partir de una buena solución es muy importante para obtener un buen coste final. De todas formas, en caso de partir de una solución mala, si elegimos una temperatura alta, tenemos garantizado que nos acercaremos al óptimo.

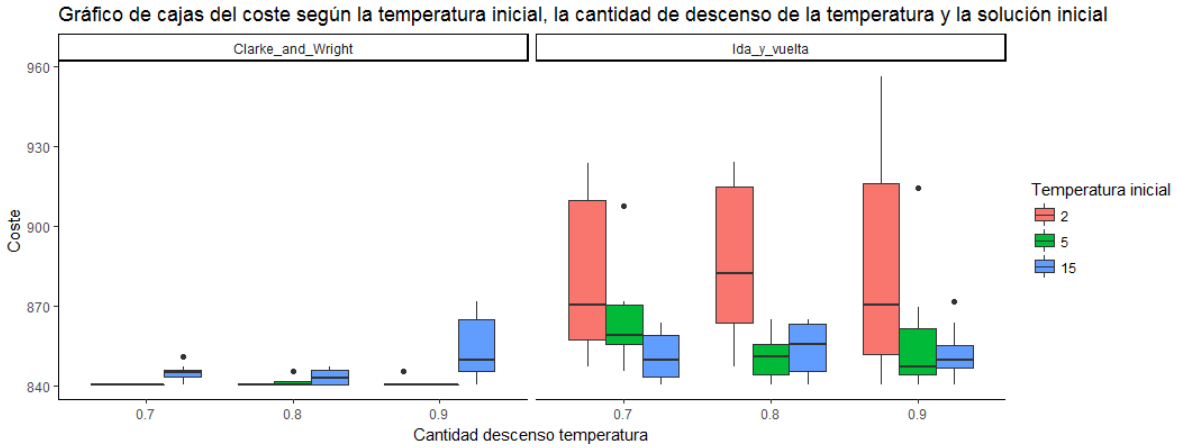


Figura 3.14: Gráfico de cajas del coste según la solución inicial, la temperatura, y el descenso de ésta

3.2.2. Tabu Search

A continuación, analizaremos los parámetros del algoritmo Tabu Search descrito anteriormente. Los parámetros que veremos son los siguientes:

- La cantidad de vecinos que analizamos en cada iteración, con las siguientes posibilidades: 50, 100 y 500;
- Supondremos el valor de la penalización por repetir soluciones ya visitadas, igual a 0;
- Para el número de iteraciones en las que un movimiento es considerado tabú, veremos tres casos: la solución permanece tabú durante 10 iteraciones, durante 50, o durante un número aleatorio entre 10 y 50, que varía de una iteración a otra;
- Supondremos que el número de iteraciones para la aspiración de una solución es 5, es decir, si una solución tabú obtenida es la mejor de las últimas 5, la aceptaremos;
- La solución inicial: algoritmo de Clarke y Wright, o rutas de ida y vuelta a la fábrica, con el fin de ver las diferencias entre partir de una solución inicial buena, y de una mala.

El criterio de parada será haber alcanzado un total de 500 iteraciones.

Igual que para Simulated Annealing, realizaremos 5 pruebas por cada combinación de parámetros, y analizaremos los resultados obtenidos. Además, en este caso la selección de los parámetros a evaluar tampoco se hizo de forma aleatoria: se realizó un breve estudio previo sobre varios parámetros, y se han escogido los que aportan mejor solución en un tiempo razonable. Además, puesto que es muy poco probable volver a una solución anteriormente visitada debido a la gran cantidad de soluciones

existentes, podremos considerar que no se repetirán soluciones. De esta forma, el valor de la penalización por repetir una combinación de rutas pierde significancia, y de ahí que lo consideremos igual a 0. Sucede lo mismo con el número de iteraciones para la aspiración de una solución (consideraremos un único valor), y con la cantidad de iteraciones tabú (consideraremos varios valores pero veremos que no afectan a la solución).

Empezaremos observando la distribución del coste según la solución inicial (Figura 3.15), donde vemos que la solución de partida afecta al coste (contrastado con el test de Wilcoxon Mann Whitney).

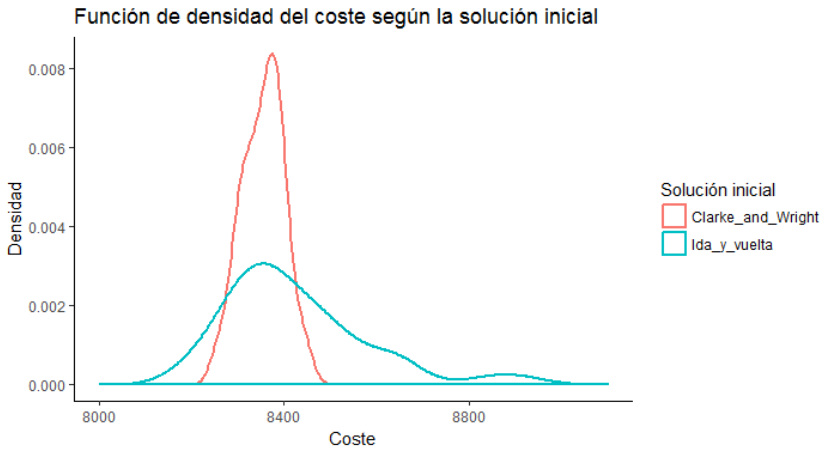


Figura 3.15: Función de densidad del coste según la solución inicial

En la Figura 3.16 tenemos la distribución del coste según la cantidad de vecinos considerados en cada iteración, y el número de iteraciones tabú.

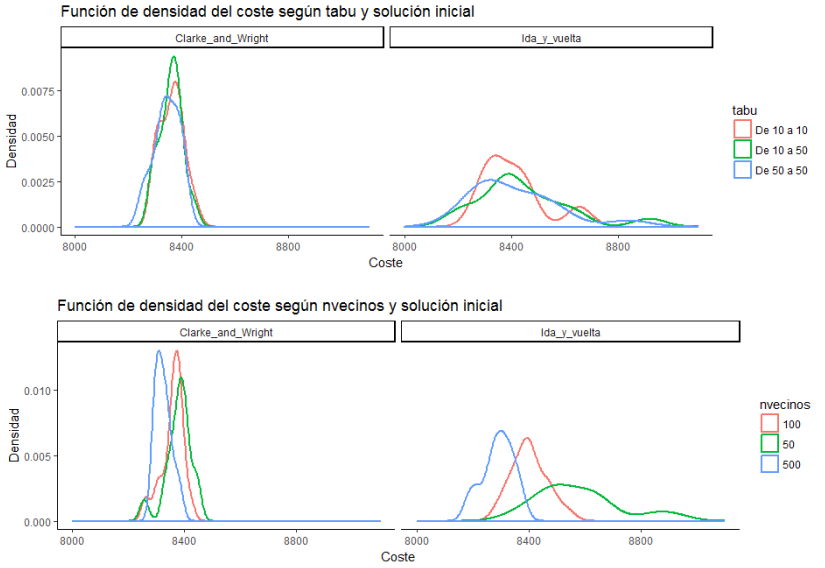


Figura 3.16: Función de densidad del coste según la solución inicial y resto de variables

Si nos fijamos en la Figura 3.17, vemos que la cantidad de iteraciones tabú no afecta al coste. Esta

falta de relación entre ambos parámetros se debe a la cantidad de clientes que estamos considerando: al haber multitud de rutas posibles, es muy difícil volver a una solución ya considerada, y por ello no entraría en juego la Lista Tabú. Sí podemos observar en cambio una relación entre la cantidad de vecinos y el coste, tal y como indica el modelo.

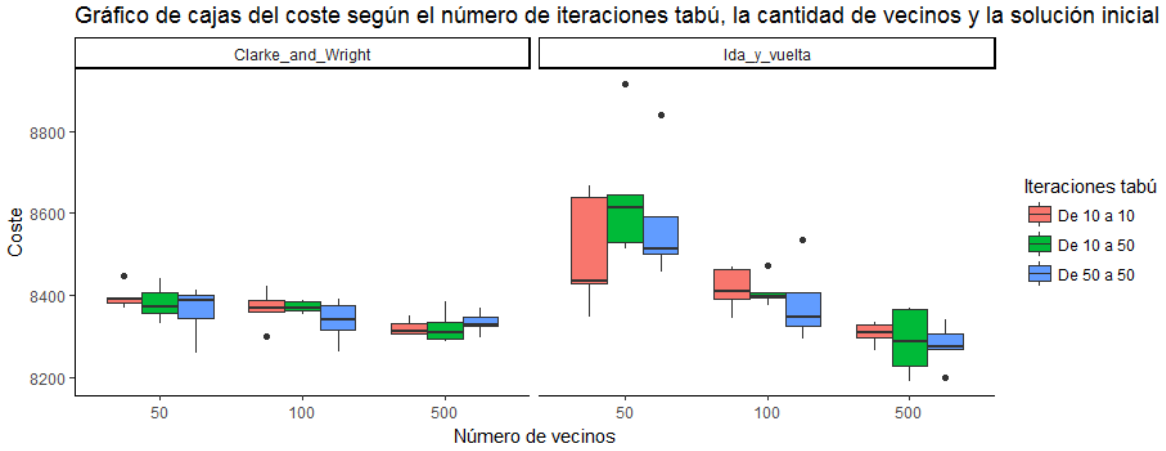


Figura 3.17: Gráfico de cajas del coste en función de la cantidad de vecinos, las iteraciones tabú, y la solución inicial

Si nos fijamos la Figura 3.18, observamos que, al partir de la solución inicial obtenida con el algoritmo de Clarke and Wright, la cantidad de vecinos tiene más peso cuando la solución de partida se obtiene mediante rutas de ida y vuelta.

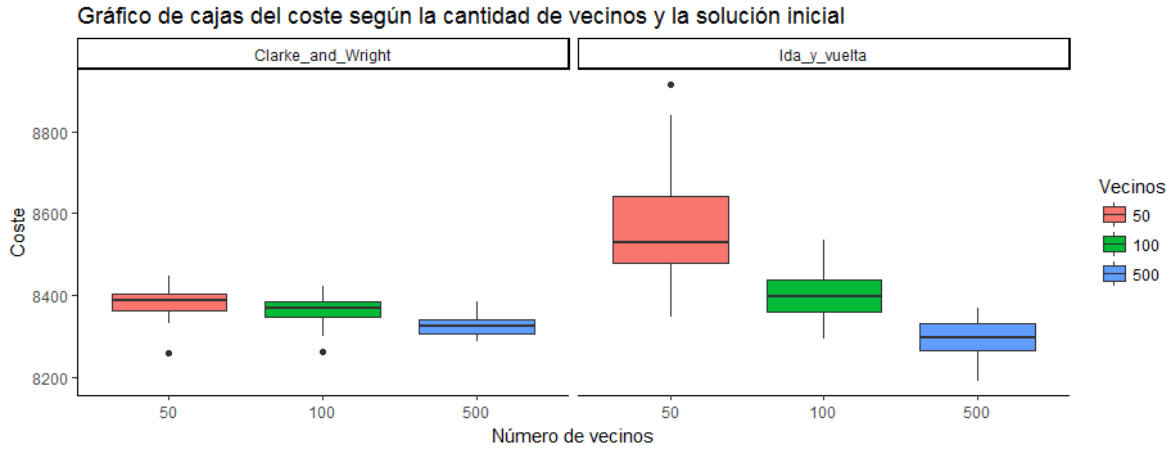


Figura 3.18: Gráfico de cajas del coste en función de la cantidad de vecinos y la solución inicial

Es decir, la solución inicial afecta a la relación entre el coste y la cantidad de vecinos considerados. De esta forma, vemos que con una buena solución inicial no es demasiado importante la cantidad de vecinos. Sin embargo, si partimos de una solución mala, este parámetro sí es importante, y debe ser elevado, con el fin de que se exploren gran cantidad de soluciones vecinas en cada iteración y poder elegir la mejor.

El problema es que a medida que aumenta la cantidad de vecinos, aumenta el tiempo de ejecución, como vemos en la Figura 3.19. Además, observamos que no hay una relación directa entre coste y tiempo (en minutos). Es decir, no necesariamente encontrar una buena solución supone mucho tiempo de ejecución.

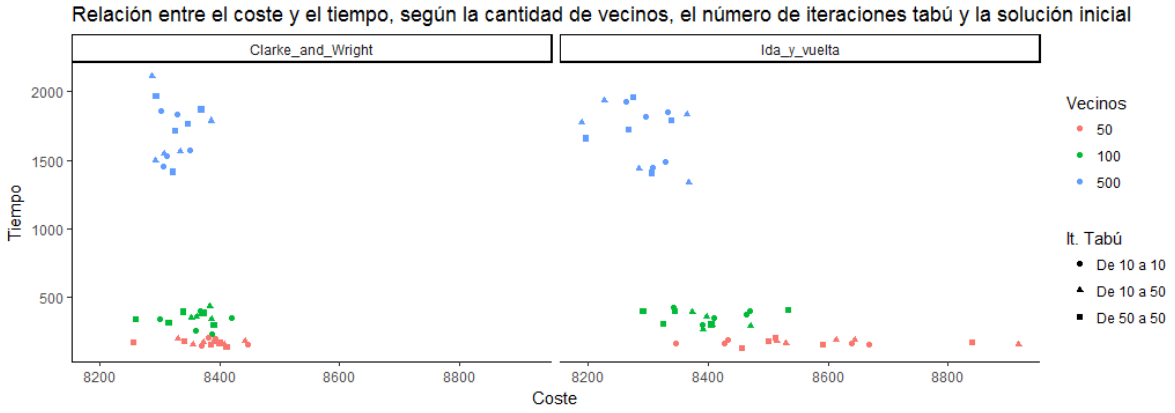


Figura 3.19: Relación entre el coste y el tiempo de ejecución (en minutos), según la cantidad de vecinos y las iteraciones tabú

Por lo tanto, una buena opción se selección de parámetros podría ser partir de una solución inicial buena (Clarke and Wright) y considerar 100 vecinos en cada iteración. De esta forma, obtendremos buenas soluciones en, relativamente, poco tiempo.

La mejor solución encontrada tiene un coste de 8.190 (conseguimos reducir el coste en un 4,65 % respecto a la solución de Clarke and Wright, con un tiempo de ejecución aproximado de 1 día y 5 horas) y se ha obtenido con los parámetros:

- Cantidad de vecinos en cada iteración: 500;
- Iteraciones Tabú: aleatorio entre 10 y 50;
- Solución inicial: rutas de ida y vuelta a la fábrica.

La tabla de las rutas es la que sigue:

jornada	V2	V3	V4	V5	V6	V7	V8	V9	coste	compartimentos	ton
2	0	6	4	5	156	0	0	0	327.739538	6	12.20
2	0	134	37	46	0	0	0	0	316.094125	6	13.40
2	0	12	137	81	0	0	0	0	89.376453	6	12.80
2	0	76	74	75	107	0	0	0	134.043248	6	13.30
1	0	47	118	71	57	0	0	0	200.739768	6	12.40
2	0	142	143	0	0	0	0	0	223.331355	6	17.30
2	0	128	0	0	0	0	0	0	138.693095	6	16.00
1	0	138	159	170	169	0	0	0	143.161651	6	11.10
2	0	123	135	124	0	0	0	0	283.737161	6	13.30
2	0	122	141	101	0	0	0	0	268.708003	6	14.20
1	0	126	125	103	0	0	0	0	176.609543	6	15.60
1	0	38	39	136	67	0	0	0	166.853572	6	10.40
1	0	119	121	160	131	0	0	0	186.771323	6	13.80
3	0	115	155	50	77	132	0	0	359.493855	6	10.70
1	0	61	116	133	0	0	0	0	276.732373	6	11.70
3	0	157	8	7	0	0	0	0	89.548195	6	13.30
3	0	63	25	26	104	0	0	0	168.897782	6	12.70
2	0	35	127	0	0	0	0	0	92.830612	6	12.20
2	0	130	0	0	0	0	0	0	213.570326	6	16.10
3	0	32	31	106	54	111	110	0	116.485282	6	11.54
3	0	15	0	0	0	0	0	0	21.773231	3	7.60
3	0	144	98	87	0	0	0	0	174.346520	6	13.80
1	0	171	167	172	173	168	0	0	70.940501	6	11.60
1	0	164	109	30	0	0	0	0	76.761212	6	13.00
3	0	22	60	129	0	0	0	0	153.408808	6	15.00
2	0	21	1	53	108	0	0	0	81.063483	6	13.50
1	0	56	148	0	0	0	0	0	194.326921	6	15.10
2	0	97	51	14	0	0	0	0	77.379277	5	7.94
3	0	68	69	139	140	82	0	0	226.360958	6	11.98
2	0	48	41	78	120	80	0	0	201.891558	6	13.40
1	0	166	0	0	0	0	0	0	91.123785	6	17.20
1	0	9	149	16	0	0	0	0	99.603194	6	14.00
2	0	49	45	92	93	90	0	0	192.993374	6	9.80
2	0	147	79	89	91	0	0	0	199.046440	6	10.80
1	0	33	0	0	0	0	0	0	9.854264	1	1.10
2	0	84	85	2	3	29	0	0	77.458118	6	10.62
3	0	34	83	86	0	0	0	0	104.758423	6	12.70
1	0	17	27	62	0	0	0	0	78.157026	5	10.30
1	0	88	73	36	0	0	0	0	234.143676	6	12.30
1	0	44	42	52	95	59	58	0	212.977930	6	9.20
1	0	10	65	40	94	0	0	0	166.466251	6	11.60
1	0	11	145	146	0	0	0	0	191.686378	6	11.90
1	0	28	18	24	19	0	0	0	111.359691	6	13.70
2	0	13	55	23	72	0	0	0	153.342799	6	11.54
1	0	64	70	117	100	43	0	0	214.872950	6	10.40
2	0	105	66	99	96	0	0	0	205.210023	6	9.80
2	0	102	151	158	150	0	0	0	73.762119	6	11.30
2	0	161	162	163	0	0	0	0	22.973157	5	8.40
1	0	113	114	112	154	0	0	0	202.674269	6	9.60
1	0	20	165	153	152	0	0	0	296.142151	6	10.40

Podemos ver en la Figura 3.20 las rutas finales encontradas para cada jornada.

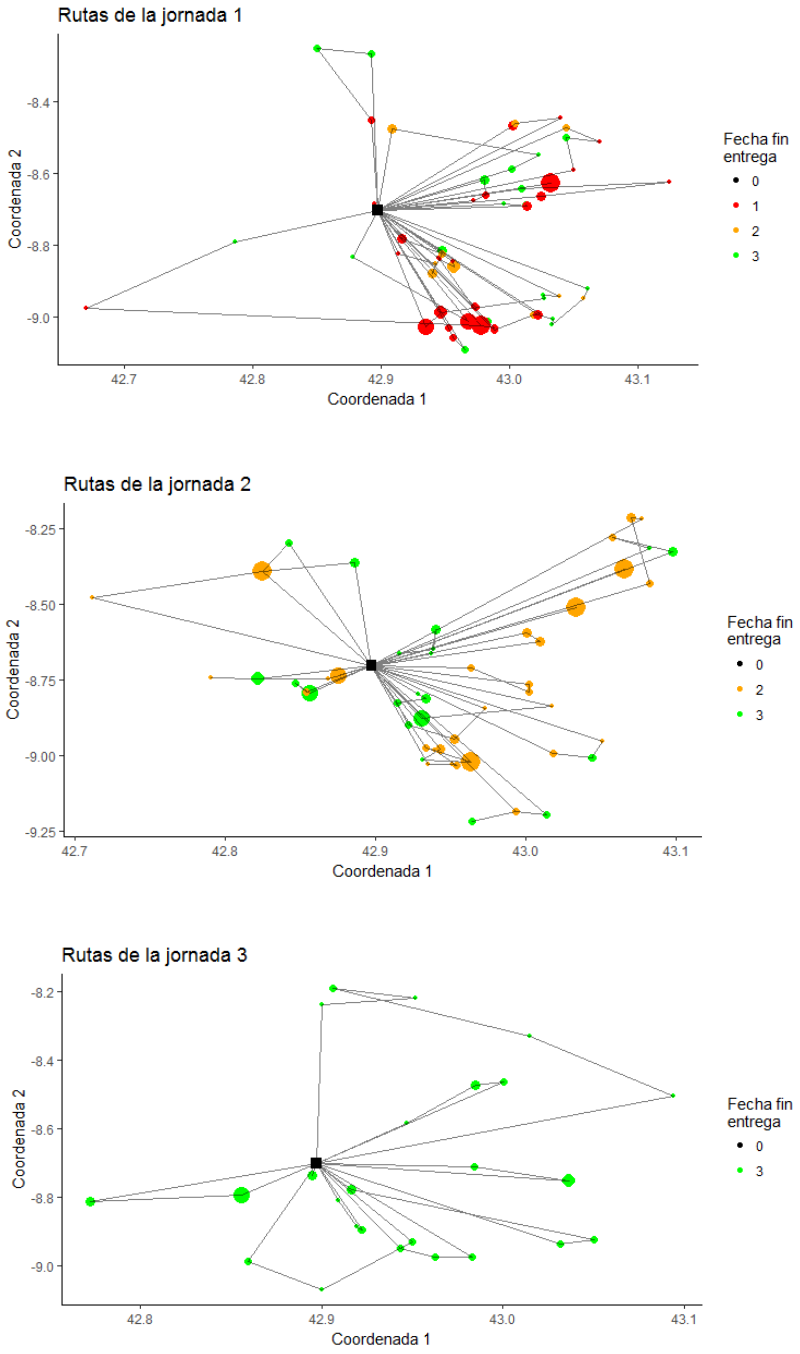


Figura 3.20: Rutas finales obtenidas mediante Tabu Search para las tres jornadas. El tamaño de los puntos representa la cantidad de compartimentos necesarios para satisfacer su demanda; y el color, su urgencia.

En la Figura 3.21, tenemos la evolución del coste. Vemos que en las primeras iteraciones se reduce de forma considerable, pues partimos de una mala solución fácilmente mejorable.

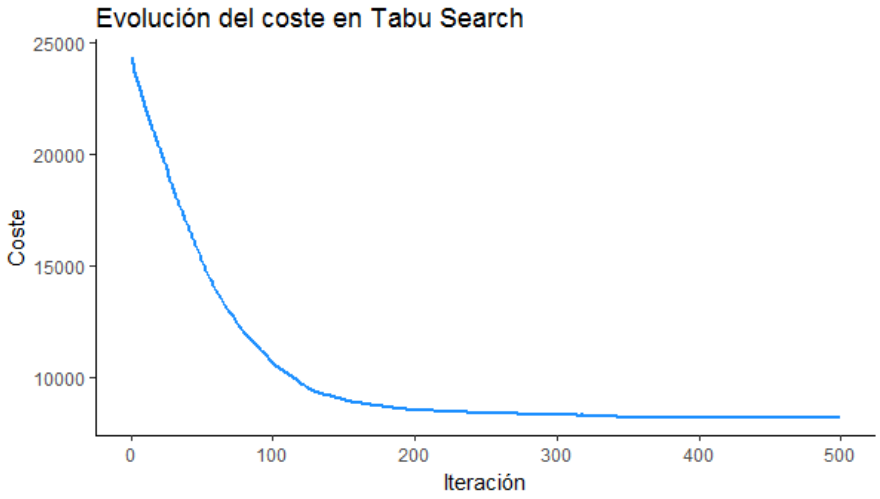


Figura 3.21: Evolución del coste en Tabu Search

Observemos en la Figura 3.22 cómo varía este descenso del coste según la cantidad de vecinos que consideremos. Vemos que a medida que aumentamos los vecinos, la convergencia es más rápida.

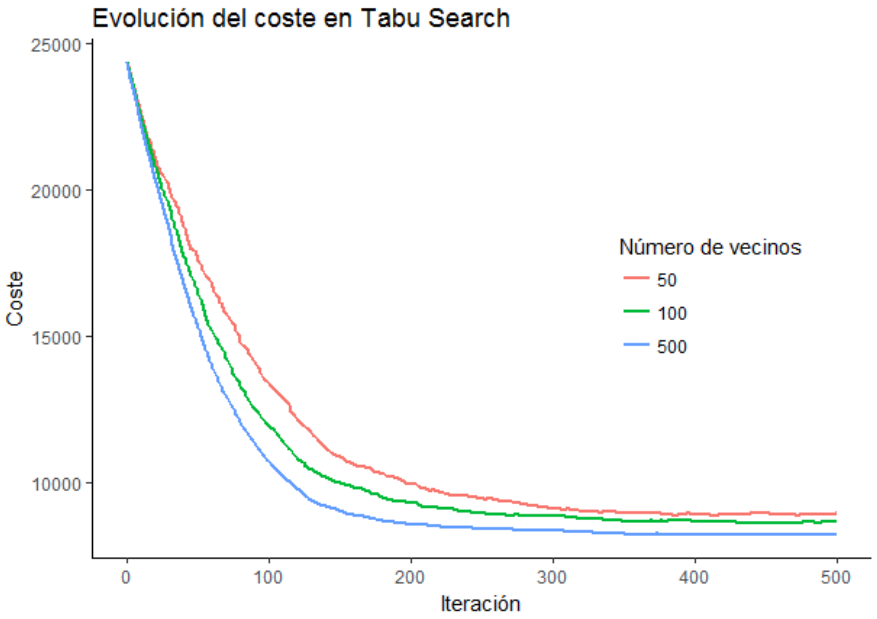


Figura 3.22: Evolución del coste en Tabu Search según la cantidad de vecinos

Ahora observemos qué ocurre con los parámetros si en lugar de 137 clientes, consideramos solamente 10 elegidos de forma aleatoria. Repetiremos las pruebas con los mismos parámetros, pero esta vez realizaremos 2 ejecuciones por cada combinación en lugar de 5, y además no consideraremos la opción de 500 vecinos. En la Figura 3.23 vemos la distribución del coste según la solución inicial considerada.

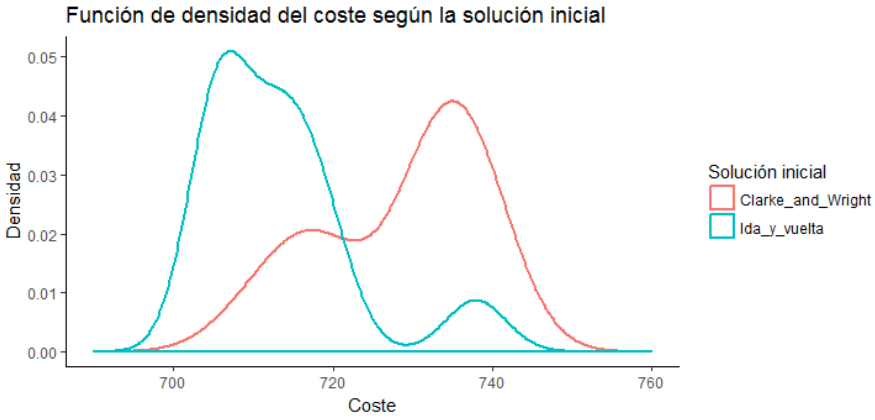


Figura 3.23: Distribución del coste según la solución inicial

En la Figura 3.24 vemos el gráfico de cajas del coste en función de los diferentes parámetros. Con estas dos imágenes, deducimos que partir de una buena solución es muy importante para obtener un buen coste final. Además, deberemos bajar la cantidad de iteraciones tabú, pues al tener menos clientes, los movimientos son más limitados, y considerar un movimiento tabú durante demasiadas iteraciones puede hacer que nos alejemos del óptimo.

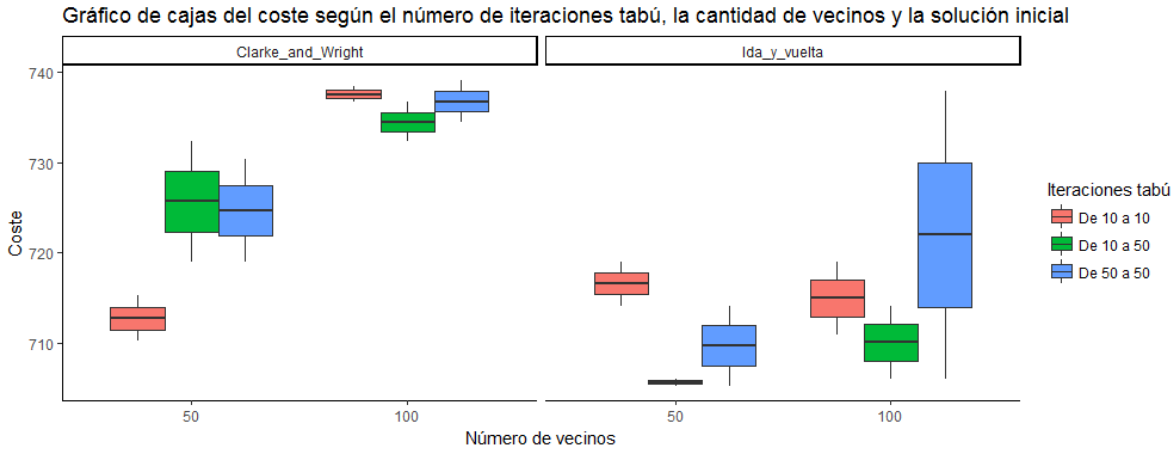


Figura 3.24: Gráfico de cajas del coste según la solución inicial, la temperatura, y el descenso de ésta

Simulated Annealing vs Tabu Search

Por último, podemos pensar en qué técnica elegir entre las dos anteriormente vistas: Simulated Annealing o Tabu Search. Para decidir cuál consideramos mejor, empezaremos por quedarnos con los resultados y tiempos de cada técnica con los parámetros que hemos considerado mejores:

- Simulated Annealing: temperaturas de 5 o 15, bajada de temperatura de 0.8, y solución inicial obtenida mediante el algoritmo de Clarke and Wright;
- Tabu Search: 100 vecinos y solución inicial obtenida mediante el algoritmo de Clarke and Wright

Comparamos el coste obtenido con cada una de estas técnicas en la Figura 3.25. Podemos observar que con Tabu Search se obtienen mejores resultados, pues el coste es menor.

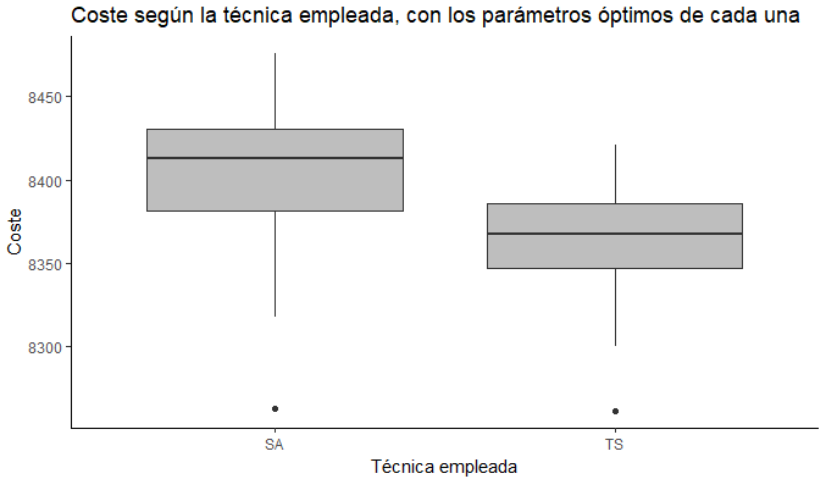


Figura 3.25: Gráfico de cajas del coste según la técnica elegida: Simulated Annealing o Tabu Search

Sin embargo, este mejor coste con Tabu Search tiene la penalización del incremento en el tiempo de ejecución, como podemos observar en la Figura 3.26: Tabu Search encuentra una buena solución en, aproximadamente, el doble o triple de tiempo que con Simulated Annealing.

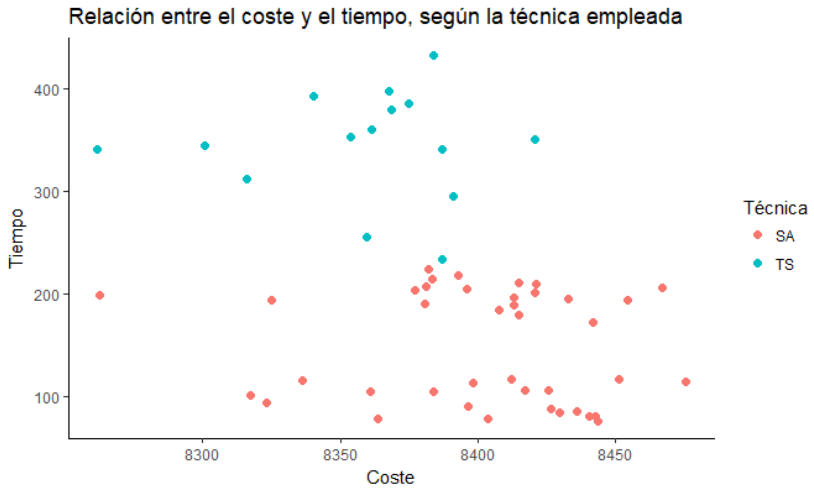


Figura 3.26: Gráfico de puntos del tiempo de ejecución en relación al coste, según la técnica elegida: Simulated Annealing o Tabu Search

En base a estas imágenes, aunque las dos encuentran muy buenas soluciones si elegimos adecuadamente los parámetros, Simulated Annealing tiene una convergencia más rápida. Por lo tanto, para buscar buenas soluciones de forma rápida, lo mejor sería utilizar Simulated Annealing.

3.3. Asignación de rutas a vehículos

Partiendo de la solución obtenida con Simulated Annealing, repartiremos las rutas entre los vehículos, empezando por las de mayor duración, y asignándolas al vehículo que menos tiempo de recorrido lleve, tal y como hemos comentado en la Sección 2.2. Empezaremos con el mínimo número de vehículos necesarios, y en caso de pasarnos de tiempo, añadiremos más camiones.

El algoritmo implementado en R para este problema devuelve una imagen y una tabla, que veremos a continuación. Obtendremos cada uno de los vehículos relacionado con las rutas que realiza (cada ruta tendrá un identificador), así como el tiempo de viaje por ruta y el total de cada jornada.

En la Figura 3.27 podemos ver cómo se asignan finalmente las rutas según su duración a los diferentes vehículos, en cada una de las jornadas. Cada barra vertical representa un vehículo, y cada bloque dentro de las barras equivale a una ruta (representadas con el identificador de ruta), donde su tamaño y color es proporcional a su duración en minutos.

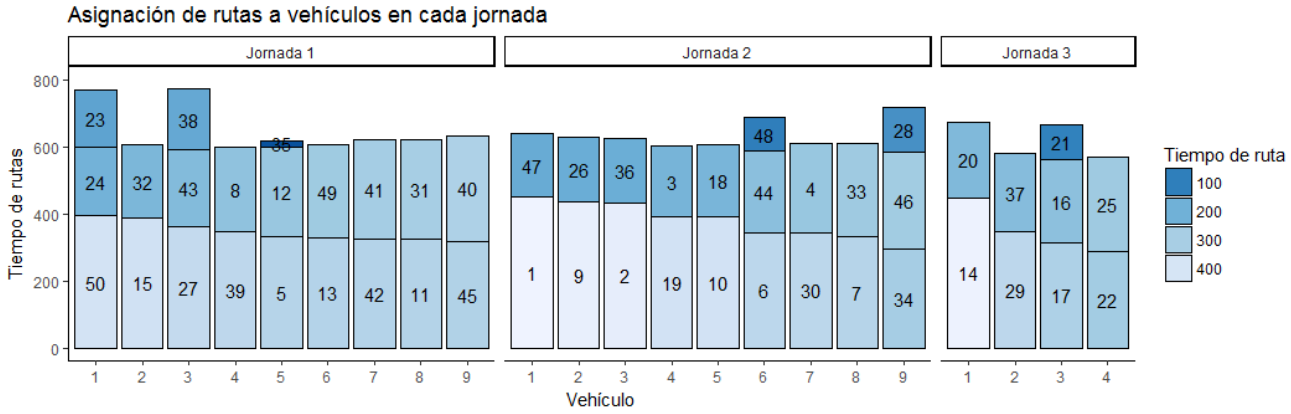


Figura 3.27: Asignación de rutas a vehículos para cada jornada

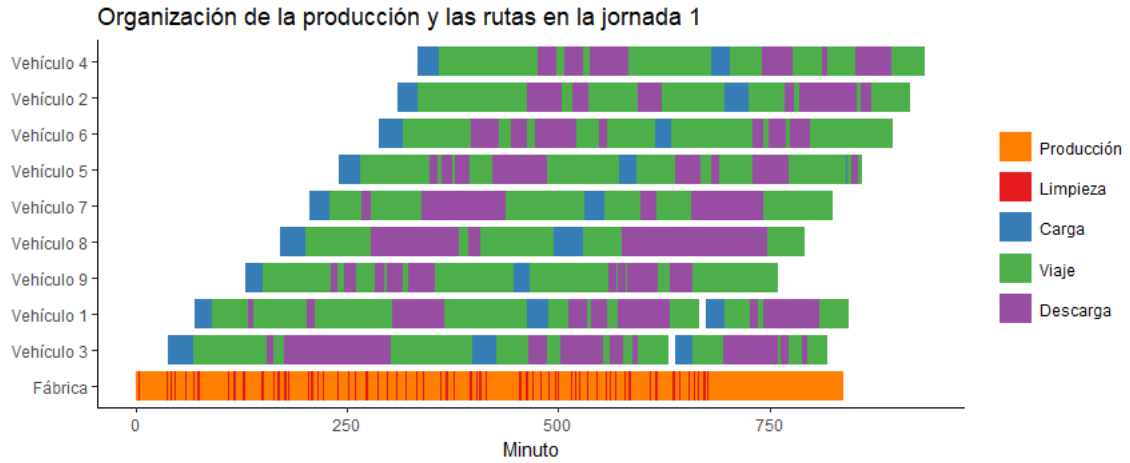
Vemos que en la primera y segunda jornada se emplean 9 vehículos; y en la tercera, necesitaremos 4 vehículos para recorrer las rutas. Los costes y tiempos totales, así como el número de rutas de cada vehículo, son, por jornada y vehículo, los siguientes:

jornada	vehiculo	coste_total	tiempo_total	rutas
1	1	443.84	768.44	3
1	2	376.34	606.04	2
1	3	383.84	773.14	3
1	4	377.31	600.81	2
1	5	377.45	618.55	3
1	6	389.45	608.45	2
1	7	358.15	620.75	2
1	8	267.73	622.53	2
1	9	427.85	631.95	2
2	1	401.50	638.60	2
2	2	364.80	627.40	2
2	3	393.55	624.17	2
2	4	302.95	602.15	2
2	5	361.54	605.34	2
2	6	399.65	687.69	3
2	7	335.93	611.93	2
2	8	331.69	611.39	2
2	9	481.64	716.78	3
3	1	475.98	674.82	2
3	2	331.12	581.82	2
3	3	280.22	666.52	3
3	4	327.76	569.36	2

3.4. Organización de la producción

Realizando el método definido en la Sección 2.3 para la organización de la producción, obtenemos la programación para cada jornada, que podemos ver en la Figura 3.28.

El eje X representa el minuto de la jornada, y en el Y tenemos tanto la fábrica como los diferentes vehículos. Mediante colores, diferenciamos los momentos de producción y limpieza de la fábrica, así como la carga, el viaje, y la descarga de los vehículos. Además, en cada jornada, el último tramo de fabricación se corresponde con la producción de fórmulas genéricas. Por simplicidad, supondremos que en cada jornada se produce una única fórmula de pienso genérico. Por ejemplo, un día se fabrica pienso para pollos, otro día pienso para ovejas, etc.



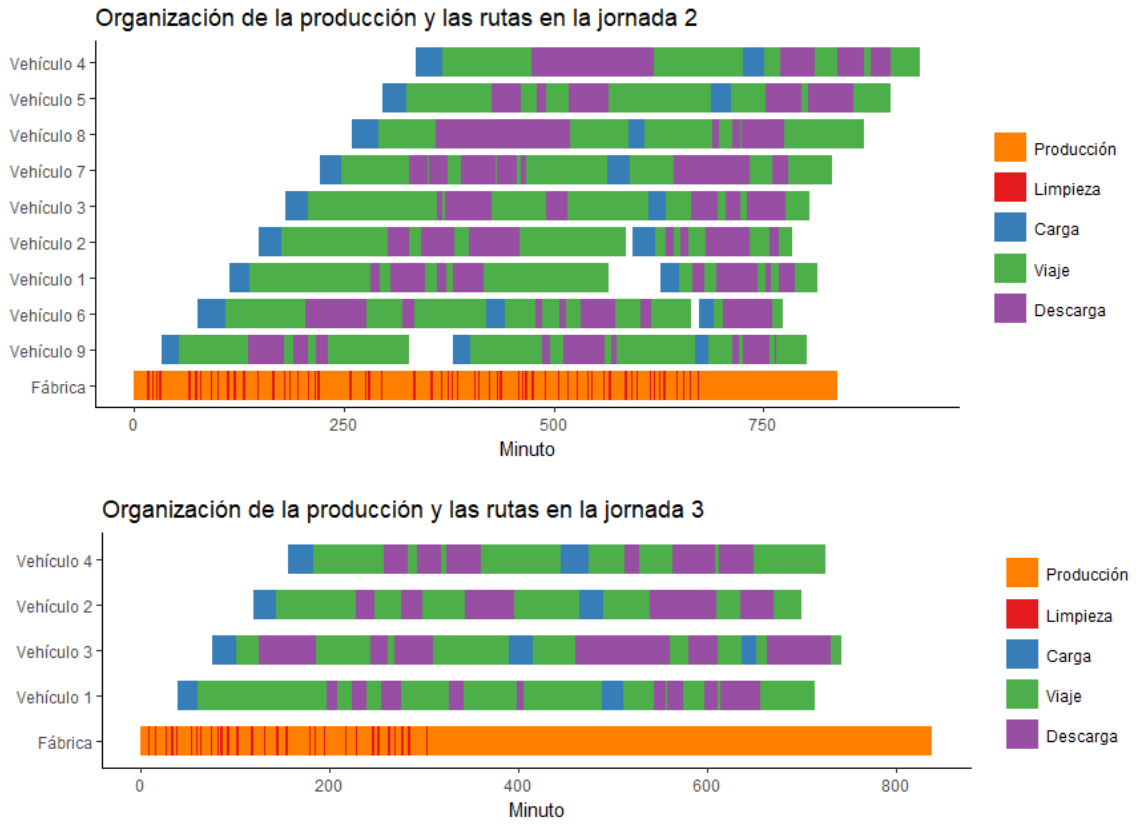


Figura 3.28: Organización de la producción y las rutas para las tres jornadas

El tiempo de producción total es de 2.216 minutos (es decir, 37 horas en tres jornadas), de los cuales aproximadamente 858 minutos (el 39 % del tiempo total) se dedican a la producción de pienso genérico, que equivale a 381,27 toneladas. Además, el tiempo dedicado a limpieza es de 304 minutos (5 horas repartidas en las tres jornadas). En la siguiente tabla podemos ver cómo se reparten los tiempos por jornada, cuántos productos se fabrican, cuántos clientes son visitados, y qué cantidad de producto genérico obtenemos:

	Jornada 1	Jornada 2	Jornada 3
Tiempo Producción	714	722	780
Tiempo Limpieza	126	118	60
Tiempo Carga	492.80	495.80	218.64
Tiempo Viaje	3401.96	3373.24	1415.07
Tiempo Descarga	1955.9	1856.4	858.8
Fórmulas	63	59	30
Clientes	66	58	28
Tiempo genérico	159.60	164.22	534.03
Toneladas genérico	70.93	72.99	237.35

Con esto, ya hemos organizado la producción y las rutas para un período temporal de tres días, y tenemos las herramientas para seguir organizando las siguientes jornadas.

Capítulo 4

Conclusión

En este trabajo hemos visto cómo organizar la producción y distribución de pienso a demanda de un conjunto de granjas. Para ello, hemos dividido el problema en fases: primero-distribución-segundo-producción, y así conseguir reducir considerablemente los costes de transporte asociados. Esta reducción de coste se consigue gracias al empleo de técnicas metaheurísticas, tales como Simulated Annealing o Tabu Search, que parten de una solución inicial (obtenida con el algoritmo de Clarke and Wright o con rutas de ida y vuelta), y buscan una buena solución de forma iterativa. Programado todo el proceso en R, encontramos rutas que satisfacen las restricciones de urgencia en los pedidos, de capacidad de producción de la fábrica, y de límite de capacidad y tiempo en los vehículos. A continuación, desarrollamos algoritmos tanto para asignar las rutas a la flota de vehículos, tratando de minimizar en cada jornada la cantidad de camiones a utilizar, como para, posteriormente, organizar la producción de la fábrica, de forma que tratemos de minimizar la cantidad de cambios de fórmula, y así poder dedicar más tiempo a la producción de fórmulas genéricas.

De esta forma, empleando los ficheros programados en R, podremos organizar de forma rápida y precisa la producción y distribución para cualquier horizonte temporal, y para cualquier variación en la demanda de los clientes, o en la capacidad de la fábrica o vehículos, tan solo con cambiar los datos en los ficheros de Excel correspondientes.

Apéndice A

Indicaciones para la ejecución del código en R

Para poder experimentar con los diferentes métodos que se estudian en este trabajo, disponemos de una serie de archivos .xlsx y .R que comentaremos a continuación. Cabe destacar que todos los scripts están preparados para ser ejecutados sin necesidad de modificar nada, salvo el *directorio de trabajo*, definido en el script *01_Datos.R*.

• **Datos_entrada.xlsx**. Contiene las tablas con toda la información vista en el Capítulo 5: tabla de solicitudes, de camiones, de fábrica, y matrices de distancias y tiempos. Este fichero se puede actualizar según nuevas demandas de los clientes, cambiando únicamente la hoja de *Solicitudes*.

• **00_Tools.R**. Aparte de la instalación y carga de los paquetes necesarios, en este fichero encontramos una serie de funciones útiles, como por ejemplo el cálculo del coste, tiempo y compartimentos de una ruta. También tenemos definida la función de intercambio de clientes (o cadenas de clientes) entre dos rutas. Es imprescindible ejecutar este fichero para poder trabajar con el resto de archivos.

• **01_Datos.R**. El primer paso del trabajo será la lectura de datos. En este fichero R, de forma opcional se puede decidir cuántos clientes vamos a considerar (predeterminado a 137), elegidos de forma aleatoria, lo que será útil para comprobar el funcionamiento de las heurísticas con menor número de clientes. Además, se realiza una transformación de los datos previa a la ejecución de los algoritmos. En concreto, se considera cada dupla (cliente,fórmula) como un único socio, otorgándole un identificador único.

• **02_Solucion_inicial.R**. De forma previa a la búsqueda de soluciones iniciales, asignaremos los clientes a jornadas según su urgencia, sin sobrepasar la capacidad de producción de cada día. A continuación, para cada jornada, generamos las soluciones iniciales: mediante el algoritmo de Clarke and Wright, y considerando rutas de ida y vuelta a la fábrica, respectivamente.

• **03a_Simulated_Annealing_obtencion_temperatura_inicial.R**. Tal y como hemos comentado, una buena forma de obtener una temperatura inicial para Simulated Annealing puede ser definiendo un ratio de aceptación inicial, buscando un elevado número de vecinos desde la solución

de partida, y quedándonos con aquella temperatura para la que se acepten tantas soluciones como indique el ratio de aceptación. En este archivo, exploramos tres ratios de aceptación diferentes, aunque queda a disposición del usuario cambiar estos números.

• **03b_Simulated_Annealing_pruebas.R**. En este script tenemos las ejecuciones del método Simulated Annealing para diferentes parámetros, en bucle, con el fin de probar cómo varía el coste encontrado con cada combinación de éstos. Tenemos también la opción de pasar por soluciones no factibles, con una determinada penalización. Obtendremos un fichero .csv con el resultado (coste y tiempo) de la combinación de todos los parámetros, que analizaremos en el siguiente script.

• **03c_Analisis_resultados_SA.R**. En este fichero tenemos el análisis realizado para identificar la mejor combinación de parámetros, con los que conseguir una buena solución en un tiempo razonable.

• **03d_Simulated_Annealing.R**. Una vez que sepamos qué parámetros debemos emplear, los introduciremos en este script y lo ejecutaremos para obtener una buena combinación de rutas mediante Simulated Annealing, que se guardará en formato .csv.

• **04a_Tabu_Search_pruebas.R**. De la misma forma que con Simulated Annealing, antes de decidir qué parámetros elegir para la ejecución del algoritmo, debemos testearlos. Para ello, este script contiene un bucle de ejecuciones Tabu Search con diferentes parámetros, que podremos cambiar según creamos conveniente. Obtendremos otro fichero .csv con el resultado (coste y tiempo) de la combinación de todos los parámetros.

• **04b_Analisis_resultados_TS.R**. Realizamos el análisis de los resultados obtenidos en las pruebas de Tabu Search para elegir los mejores parámetros: serán aquellos con los que obtengamos un buen coste en poco tiempo.

• **04c_Tabu_Search.R**. Cuando sepamos qué parámetros son buenos, podremos introducirlos en este script y ejecutar el algoritmo de búsqueda de solución con Tabu Search. Obtendremos las rutas finales: en qué jornada se realizan, qué clientes se visitan y en qué orden, el coste y tiempo de rutas, etc., que será guardado en un fichero .csv.

• **05_Comparacion_SA_TS.R**. En este archivo encontramos una breve comparación entre ambas técnicas metaheurísticas.

• **06_Asignacion_Rutas_a_Vehiculos.R**. Con las rutas ya definidas, las asignaremos a vehículos para cada una de las jornadas tal y como hemos visto anteriormente, obteniendo imágenes y una tabla con este reparto.

• **07_Organizacion_de_la_produccion.R**. Con los datos que salen de ejecutar el script anterior, organizaremos la producción en las diferentes jornadas, siguiendo el método visto. Obtendremos una tabla e imágenes de a qué hora debemos empezar y terminar cada tarea.

• **Carpeta: Datos**. Por último, los resultados obtenidos están guardados en esta carpeta: tanto las pruebas de los algoritmos con los diferentes parámetros, como las rutas obtenidas.

Bibliografía

[1] Baldacci R, Hadjiconstantinou E, Mingozzi A (2004) An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. Operations Research.

[2] Bektas I, Laporte G (2011) The Pollution-Routing Problem. Transportation Research Part B: Methodological, 45(8), 1232-1250.

[3] Clarke G, Wright JW (1964) Scheduling of vehicles from central depot to number of delivery points. Operations Research.

[4] Cordeau JF, Laporte G, Mercier A (2001) An unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society, 52.

[5] Cook TM, Russell RA (1978) A simulation and statistical analysis of stochastic vehicle routing with timing constraints. Decision Sciences.

[6] Dantzig G, Wolfe P (1960) Descomposition principle for linear-programs. Operations Research.

[7] Dorigo M (1992) Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy.

[8] Dréo J, Pétrowski A, Siarry P, Taillard E (2003) Metaheuristics for hard optimization. Springer.

[9] Erdogan S, Miller-Hooks E (2012) A Green Vehicle Routing Problem. Transportation Research Part E, 48(1), 100-114.

[10] Fisher ML (1994) Optimal solution of vehicle-routing problems using minimum k-trees. Operations Research.

[11] Fisher ML, Jaikumar R (1981) A generalized assignment heuristic for vehicle-routing. Networks.

[12] Flood MM (1956) The traveling salesman problem. Operations Research.

[13] Foster BA, Ryan DM (1976) Integer programming approach to vehicle scheduling problem. Operational Research Quarterly.

[14] Gendreau M, Herts A, Laporte G (1994) A tabu search heuristic for the vehicle routing problem. Management Science, 40.

[15] Gendreau M, Laporte G, Potvin JY (1998) Metaheuristics for the vehicle routing problem. Technical report.

[16] Glover F (1986) Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 13.

[17] Golden BL, Magnanti TL, Nguyan HQ (1972) Implementing vehicle routing algorithms. Networks.

[18] Golden BL, Stewart JW (1978) Vehicle routing with probabilistic demands. Computer science and statistics.

[19] Hillier FS, Lieberman GJ (2001) Introduction to operations research. McGraw-Hill.

[20] Holland JH (1975) Adaptation in Natural and Artificial systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. The University of Michigan Press.

[21] Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science, 220.

[22] Laporte G, Semet F (2002) Classical heuristics for the capacitated VRP. SIAM Monographs on Discrete Mathematics and Applications.

[23] Lin S (1965) Computer solutions of traveling salesman problem. Bell System Technical Journal.

[24] Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. J. Chem. Phys., 21.

[25] Mole RH, Jameson SR (1976) Sequential route-building algorithm employing a generalized savings criterion. Operational Research Quarterly.

[26] Renaud J, Heni H, Coelho LC (2018) Time-dependent Vehicle Routing Problem with emission and cost minimization considering dynamic paths. CIRRELT 2018-14.

[27] Ruiz R, Maroto C, Alcaraz J (2003) A decision support system for a real vehicle routing problem. Tilburg University.

[28] Solomon M (1983) Vehicle routing and scheduling with time window constraints: models and algorithms. Technical report.

[29] Thompson PM, Psarafitis HN (1993) Cyclic transfer algorithms for multivehicle routing and scheduling problems. Operations Research.

[30] Van Breedam A (1994) An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related, customerrelated, and time-related constraints. PhD thesis, University of Antwerp.

[31] Winston WL (2004) Rich vehicle routing problems and applications. Thomson Brooks/Cole.

[32] Xiao Y, Zhao Q, Kaku I, Xu Y (2012) Development of a fuel consumption optimization model for the capacitated vehicle routing problem. Computers and Operations Research, 39(7), 1419-1431.