Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado de Ingeniería en Organización Industrial

Algoritmos heurísticos y metaheurísticos basados en búsqueda local aplicados a Problemas de Rutas de Vehículos

Autora:

Fernández Hernández, Alba

Tutor:

Sáez Aguado, Jesús

Departamento de Estadística
e Investigación Operativa

Valladolid, Julio 2016

RESUMEN

Los Problemas de Rutas de Vehículos son uno de los problemas de optimización más estudiados y utilizados. Aunque existen muchas variantes, el que se desarrolla en este trabajo es el Problema de Rutas de Vehículos Capacitado con flota homogénea. Estos tipos de problemas tienen como objeto establecer la mejor combinación de rutas a realizar por un conjunto de vehículos para poder dar servicio a una serie de clientes.

Desde la primera formulación realizada, muchos han sido los métodos propuestos para la resolución de estos problemas. El presente documento se centra en los métodos heurísticos y metaheurísticos, los cuales son capaces de proporcionar soluciones satisfactorias en un tiempo de cálculo razonable.

No solo se expondrán los diferentes algoritmos, sino que también se llevará a cabo la correspondiente programación para su posterior implementación, así como el análisis de los resultados y de los diferentes métodos de aplicación.

Palabras clave:

Rutas de vehículos, heurísticas, metaheurísticas, Inserción Secuencial, GRASP.

ABSTRACT

The Vehicle Routing Problems are one of the most studied and used optimization problems. Although there are many variants, the only one which this project covers is the Capacitated Vehicle Routing Problem with homogeneous fleet. These sorts of problems are aimed to establish the optimal set of routes by a fleet in order to serve a set of customers.

Since the first formulation was made, a lot of new methods have been proposed to solve these problems. The actual document is focused on heuristics and metaheuristics methods, which are capable of offering satisfactory solutions within acceptable computing times.

We will propose not only several algorithms, but also we will carry out the convenient programming and its subsequent implementation, besides analyzing the results and the various techniques.

Key words:

Vehicle routing, heuristics, metaheuristics, Sequential Insertion, GRASP.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS, TABLAS Y GRÁFICOS

Lista de Figuras

Lista de Tablas

Lista de Gráficos

CAPÍTULO 1. INTRODUCCIÓN

1.1. Estado del arte

Los métodos que se van a estudiar en este Trabajo Fin de Grado reciben el nombre de algoritmos heurísticos y metaheurísticos. La palabra *heurística* proviene del griego, *heurisken,* que significa <hallar, descubrir>, y se emplea en el entorno de la optimización para la resolución de problemas.

La optimización es el proceso de búsqueda de la mejor solución posible para un determinado problema. De una forma más técnica, estos problemas de optimización tratan de encontrar los valores de las variables de decisión que maximicen o minimicen una función objetivo, pudiendo estar sujetos a una serie de restricciones.

Podemos encontrar una gran cantidad de problemas de optimización, tanto en la industria como en la ciencia. Dentro de la optimización, los problemas lineales pueden ser resueltos eficientemente mediante el método Simplex, pero incluso en este caso cuando intervienen muchas variables (del orden de millones), dicho método no resulta efectivo debido a que su tiempo de respuesta no es operativo.

Otro tipo específico de problemas de optimización son los *problemas de optimización combinatoria,* donde podemos destacar el problema de la mochila (*Knapsack Problem*), el del viajante (*Travelling Salesman Problem*, TSP) y el de rutas de vehículos, entre otros. En dichos problemas un método de resolución elemental estaría basado en explorar exhaustivamente el conjunto de soluciones y obtener el mejor resultado que se haya calculado. Es por ello que se les considera problemas computacionalmente difíciles de resolver, ya que se produce una explosión del tiempo de cálculo al aumentar el tamaño del problema.

El término científico *NP-hard* empleado en el contexto de la complejidad algorítmica, recoge el concepto de aquellos problemas que presentan una elevada dificultad para ser resueltos. Este tipo de problemas no garantiza la obtención de la solución óptima en un tiempo computacional razonable.

Con motivo del elevado número y variantes de problemas difíciles de resolver que aparecen en la práctica, surge la necesidad de desarrollar procedimientos eficientes que encuentren soluciones aceptables - no necesariamente las óptimas - dentro de un tiempo de cálculo moderado. Es entonces donde comienzan a surgir los algoritmos heurísticos.

Aunque al principio estos nuevos métodos no fueron recibidos con gran aceptación, poco a poco fueron adquiriendo una mayor importancia, sobre todo a partir de la mitad de los años setenta, con la propagación de resultados en el ámbito de la complejidad computacional y el uso intensivo de computadoras.

En Diaz y otros (1996) se recoge la siguiente definición de un algoritmo heurístico:

"Un método heurístico es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución".

En los últimos años se ha producido un elevado crecimiento en cuanto al desarrollo de estos procedimientos como forma de resolución a los problemas de optimización, reflejándose así en el número de publicaciones en revistas especializadas. Fue en 1995 cuando se edita el primer número de la revista *Journal of Heuristics* dedicada exclusivamente a la divulgación de los métodos heurísticos.

Algunas de las razones por las que resulta muy interesante la implementación de los métodos heurísticos son las siguientes:

- ✓ Capacidad para obtener una solución, aunque esta solo sea aceptable, cuando no existe un método exacto de resolución, o bien este requiere mucho tiempo de cálculo o de memoria, lo cual imposibilita tan si quiera la opción de hallar una solución.
- ✓ No existe la necesidad de alcanzar la solución óptima. Por ejemplo, cuando los valores de la función objetivo son relativamente pequeños y no existe una variación significativa entre el óptimo y un sub-óptimo.
- ✓ Cuando el tiempo dedicado a la resolución del problema presenta una importancia considerable y se precisa de una respuesta rápida.
- ✓ Se prescinde de la utilización de un *solver* para su resolución. Esto implica un gran ahorro económico al evitar la adquisición de una licencia comercial. Se trata así de eliminar la dependencia con respecto a programas externos y además, poder emplear cualquier lenguaje de programación.
- ✓ Cuando los datos de los que se parte son poco fiables o pertenecen a una simplificación de la realidad, puede resultar ineficaz encontrar la solución exacta al tratarse de datos que no son reales.

- ✓ Proporcionan más de una solución en muchos de los casos, lo cual permite ampliar el número de opciones del que decide, en especial cuando existe algún factor no cuantificable que merezca la pena ser considerado.

A partir de los años 80 comenzaron a desarrollarse otros procedimientos heurísticos, denominados metaheurísticos para poder dar soporte a aquellos problemas que mediante una heurística específica o un método exacto no ofrecían soluciones satisfactorias.

1.2. Objetivos

El objetivo global del presente Trabajo Fin de Grado reside en la elaboración y programación de diferentes métodos heurísticos con objeto de resolver los problemas de rutas de vehículos, así como obtener resultados de una alta calidad.

Para ello será necesario cumplir con unos objetivos más concretos:

- ➢ Presentar y definir el Problema de Rutas de Vehículos y sus variantes, centrándonos en el Problema de Rutas de Vehículos Capacitado (CVRP).
- ➢ Conocer los diferentes métodos exactos y aproximados para la resolución de los CVRP.
- ➢ Señalar la importancia que tiene la optimización de rutas de vehículos desde un ámbito económico.
- ➢ Exponer algunos de los métodos aproximados que permitan resolver los problemas de enrutamiento de vehículos.
- ➢ Programar los diferentes métodos usando Xpress-Mossel como soporte informático.
- ➢ Desarrollar una comparativa entre los distintos métodos aproximados analizados.
- ➢ Desarrollar una comparativa entre los métodos aproximados analizados y métodos exactos.
- ➢ Obtener conclusiones acerca de los métodos en relación con la calidad de la solución obtenida y el tiempo de cálculo destinado a ello.

1.3. Recursos utilizados

Con motivo de poder llevar a cabo la implementación de los métodos que se van a desarrollar en el presente documento, se va a hacer uso de un lenguaje de programación matemático llamado *Mosel* incluido en entorno de modelización-optimización Xpress-Optimizer, mediante el programa *Xpress IVE.*

Por otra parte, las librerías de datos de entrada al sistema que se van a utilizar provienen del *Operations Research Group of DEI, University of Bologna.*

Los datos vienen en formato euclídeo 2D, ya que la localización de los diferentes puntos aparece configurada con coordenadas, con el propósito de obtener las distancias euclídeas entre todas los puntos y obtener la correspondiente matriz de distancias.

En la práctica, las distancias se pueden recoger de las aplicaciones de mapas (por ejemplo, Google Maps) para que de la misma forma, se pueda configurar la matriz distancias.

Todas las figuras, tablas y gráficos serán de elaboración propia salvo que se haga constar lo contrario.

CAPÍTULO 2. PROBLEMAS DE RUTAS DE VEHÍCULOS (VRP)

2.1. Justificación

La trascendencia económica que supone la competitividad en una realidad presente y futura es tal que puede determinar la supervivencia de las empresas según la gestión que se lleve a cabo de variables clave como la flexibilidad, velocidad de llegada al mercado y la productividad. Por consiguiente, es preciso un manejo eficiente del flujo de bienes y servicios hacia los usuarios finales, por el hecho de que una correcta gestión de los sistemas logísticos y una adecuada planificación pueden suponer significativos ahorros.

Autores como Toth y Vigo (2002) justifican la utilización de técnicas de investigación operativa con la finalidad de consumar dichos ahorros, pudiendo ser estos desde el 5% hasta el 20% de los costes globales de transporte, dado que se estima que estos costes representan entre el 10% y el 20% del costo final de los bienes.

Drucker (1962) definió las actividades logísticas que se realizaban tras la fabricación como las "áreas peor realizadas y a la vez más prometedoras dentro del mundo industrial". Incluso una disminución aparentemente insignificante en los costos de distribución puede significar cuantiosos ahorros económicos, a la vez que se mejora notablemente la satisfacción de los clientes.

Así, el Problema de Rutas de Vehículos resulta potencialmente útil en una gran variedad de situaciones prácticas. A continuación, se destacan algunas de sus aplicaciones más importantes:

- Logística. Es en el ámbito de la logística donde predominan las aplicaciones del VRP. Entre ellas, se distinguen: rutas de vendedores, rutas escolares, y reparto de mercancías.
- Industria. No son tantas sus aplicaciones como en la logística, destacándose como la más importante la secuenciación de tareas.

2.2. Contextualización

El problema de rutas de vehículos (VRP, del inglés Vehicle Routing Problem) es uno de los problemas de optimización combinatoria y de programación entera más estudiados cuyo fin es determinar un diseño óptimo de rutas usado por una flota de vehículos para poder satisfacer la demanda de una serie de clientes.

Este problema fue introducido en el año 1959 por Dantzig y Ramser cuando diseñaron un modelo real de la entrega de gasolina a las diferentes estaciones de servicio. Cinco años más tarde, Clarke y Wright presentaron el primer algoritmo que resultó efectivo para resolverlo, comenzando así una amplia línea de investigación en el tema de ruteo de vehículos que cobraría mucha importancia.

Para describir el modelo en cuestión, se dispone de un grafo completo G=(V,A), donde $V=\{1,\ldots,n\}$ son todos los vértices, y A el conjunto de arcos.

Los vértices del grafo hacen referencia a los clientes, siendo el vértice *1* el correspondiente con la base o el depósito donde está almacenada la mercancía y desde el punto que salen los vehículos.

El conjunto de arcos representan las posibles rutas factibles que se pueden dar en el sistema. Cada una de estas rutas tiene asociado un determinado costo c_{ij}, siendo i el punto de origen y j el punto destino. El conjunto de todos estos costos forma la matriz de costos. El término de costo puede hacer referencia a la variable que más interese en casa situación, pudiendo ser la distancia entre los clientes, tiempo de desplazamiento, etc.

Cada uno de los clientes a los que se tiene que dar servicio, tiene asociado su correspondiente demanda d, bien sea de distribución (demanda negativa) o de recepción (demanda positiva). Además, dichos clientes tienen que ser abastecidos mediante una flota de K vehículos con una capacidad C.

El objetivo es conseguir K rutas correspondientes a cada uno de los vehículos haciendo que el coste sea mínimo. Para ello:

- Cada ruta debe comenzar y terminar en el depósito.

- Cada cliente debe ser servido por una única ruta.

- La suma de las demandas de todos los puntos de servicio de cada ruta debe poder ser suministrado por la capacidad C del vehículo.

En la figura 1 se muestra un esquema básico de un Problema de Rutas de Vehículos:

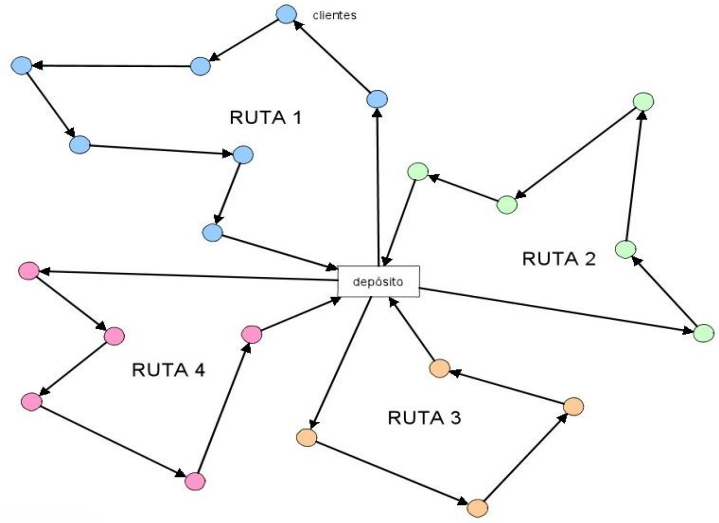

Figura 1. Esquema de un VRP
[Fuente: Wikipedia]

Consideraciones previas

- No se puede ir de un punto a sí mismo, para ello se le impone un coste infinito $c_{ii} = +\infty$.

- Cada cliente *i*=1,...,*n* tiene una demanda $d_i \geq 0$ y a la base o depósito se le asigna una demanda ficticia $d_1 = 0$.

- Para una flota homogénea, los *K* vehículos son idénticos en cuanto a la capacidad *C* que pueden transportar.

- Todas las demandas de los clientes deben ser satisfechas $d_i \leq C$.

- El problema debe ser factible, es decir que los K vehículos sean capaces de cumplir con las demandas $K \geq K_{min}$, siendo K_{min} el número de vehículos mínimo necesarios determinado por el problema de Bin Packing.

2.3. Tipos de VRP

2.3.1. Según las distancias

- **VRP Asimétrico**: se dice de aquel problema en el que el valor del coste de los arcos depende de la dirección de estos entre los dos nodos que unen. Se denomina grafo dirigido (figura 2) y la matriz de costes es asimétrica $c_{ij} \neq c_{ji}$.

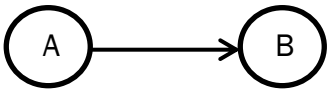

Figura 2. Grafo dirigido

- **VRP Simétrico:** el valor de los costes de los arcos no dependen de la dirección que toman. Se denomina grafo no dirigido (figura 3) y la matriz de costes es simétrica $c_{ij} = c_{ji}$.

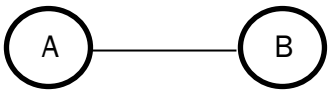

Figura 3. Grafo no dirigido

2.3.2. Según tipo de flota

- **Flota homogénea**: es aquel conjunto de vehículos que tienen una capacidad idéntica entre ellos, no existiendo por tanto diferencias entre los vehículos. Se denominan modelos con dos índices.

- **Flota heterogénea**: es aquel conjunto de vehículos que difieren en las características técnicas, como por ejemplo la capacidad, y es necesario especificar qué tipo de vehículo circula por cada ruta. Se denominan modelos con tres índices, puesto que incluye el índice correspondiente al vehículo que se debe usar para ese recorrido.

2.4. Variantes del VRP

- **Problema de Rutas de Vehículos Capacitados (Capacitated VRP) (CVRP):** las demandas de cada uno de los puntos de entrega se conocen con antelación, existe una flota homogénea de vehículos y solo existe la restricción de la capacidad de los vehículos. La función objetivo es la de minimizar el coste total a servir a todos los clientes. Esta variante es la que se focaliza el presente documento.

- **Problema de Rutas de Vehículos con Ventanas de Tiempo (VRP with Time Windows) (VRPTW):** es una extensión del CVRP, en el que a cada cliente se le asocia un intervalo de tiempo $[a_i, b_i]$, llamado ventana de tiempo. El inicio del servicio a cada cliente debe estar comprendido dentro de dicho intervalo y debe permanecer en ese punto durante el determinado tiempo de servicio. En el caso en el que el vehículo llegue antes de lo establecido, a este se le permite esperar hasta el instante a_i para que pueda comenzar el servicio. El objetivo es minimizar el tiempo de viaje total, la flota de vehículos y el tiempo de espera necesario para abastecer a todos los clientes dentro de su tiempo requerido.

- **Problema de Rutas de Vehículos con Retornos (VRP with Backhauls) (VRPB):** los clientes se dividen en dos grupos, unos denominados *Linehaul,* a los que se les debe entregar su debido pedido, y otros conocidos como *Backhaul* a los que se les tiene que recoger su determinada mercancía. Cuando en una misma ruta existen ambos tipos de clientes, es necesario que se proceda en primer lugar con todas las entregas antes de comenzar con las recogidas. El objetivo es encontrar un conjunto de rutas que minimice la distancia total realizada.

- **Problema de Rutas de Vehículos con Recogidas y Entregas (VRP with Pick-Up and Delivering) (VRPPD):** existe la posibilidad de que cada cliente pueda entregar cierta mercancía. La restricción de la capacidad implica dificultar la planificación, pudiendo llevar a aumentar el tiempo de viaje, la flota de vehículos o hacer un mal uso de la capacidad.

- **Problema de Rutas de Vehículos con multi-depósito (Multiple Depot VRP) (MDVRP):** Una compañía está compuesta por varios depósitos desde donde poder servir a sus clientes. Cada vehículo está asignado a uno de sus depósitos y se desplaza hacia los clientes que están asociados a ese mismo depósito. El objetivo es el de servir a todos sus clientes mientras que se minimiza la flota de vehículos y la distancia realizada.

- **Problema de Rutas de Vehículos periódico (Periodic VRP) (PVRP):** la prestación del servicio a los clientes se realiza de forma periódica, cada M días.

- **Problema de Rutas de Vehículos con entregas divididas (Split Delivery VRP) (SDVRP):** se permite que un mismo cliente pueda ser servido por diferentes vehículos si esto supone una reducción en los costes generales. Esta opción es muy útil cuando el tamaño de las órdenes de los clientes son tan grandes como la capacidad del propio vehículo.

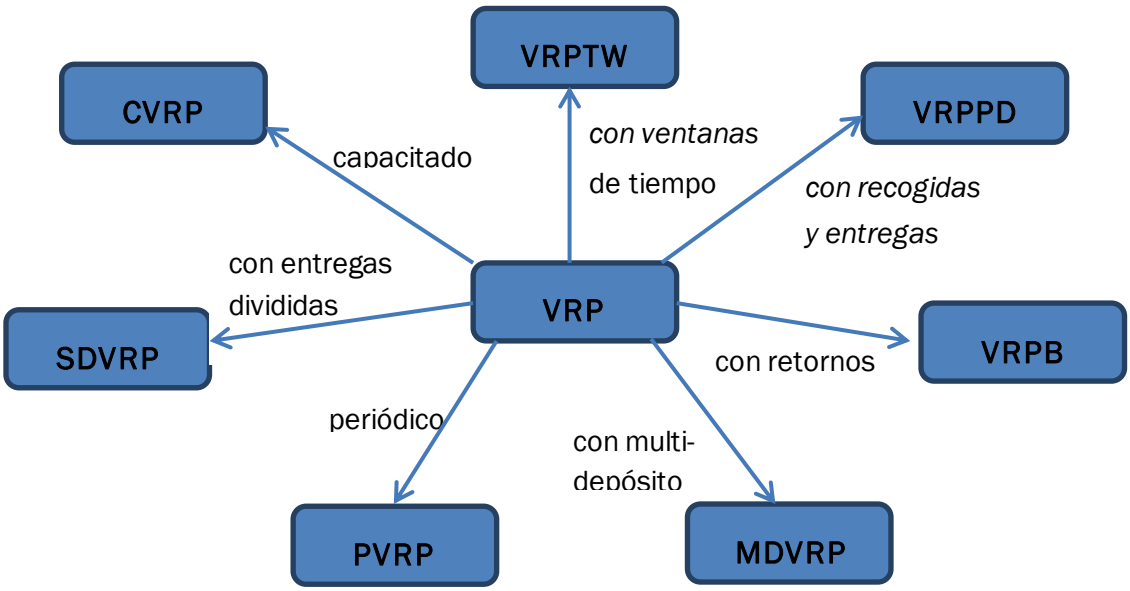

Figura 4. Variantes del *VRP*

2.5. El Problema del Agente Viajero (TSP)

El Problema del Agente Viajero (TSP, del inglés Travelling Salesman Problem) es una particularidad del Problema de Rutas de Vehículos, que se define como una situación de partida para luego poder resolver problemas combinatorios más complejos y más prácticos en la vida real.

Este modelo consiste en minimizar el coste total, contando con un solo vehículo que debe recorrer todos los puntos de demanda en una sola ruta, no haciendo distinción entre cliente y almacén e independientemente de la capacidad del vehículo. Es un modelo básico en el que la capacidad no es relevante debido al pequeño tamaño de los productos y la carga total puede transportarse fácilmente con un solo vehículo.

Se puede formular el problema como un problema de programación lineal entera donde las variables binarias x_{ij} toman el valor de 1 si pertenecen a la solución del problema, siendo este el trayecto desde el origen i hasta el destino j.

$$Función\ objetivo: \qquad Min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$Sujeto\ a: \qquad \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 2, \dots, n \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 2, \dots, n \qquad (2)$$

$$x_{ij} \in \{ 0, 1 \} \quad i, j = 1, \dots, n$$

Las dos restricciones de entrada y salida indican que:

(1) De cada nodo solo puede salir un arco.

(2) A cada nodo solo puede llegarle un arco.

Es probable que se puedan generar *subtours* (figura 5), formación de un subconjunto de nodos, ya que suelen adoptarse como la solución más económica.

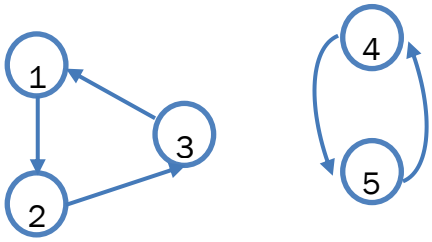

Figura 5. Subtours

Para poder eliminar estos subtours, es necesario incorporar restricciones que impidan su formación, las cuales se mencionan en el apartado 4.2.4.1.

2.6. Modelos básicos para VRP

Existen tres diferentes acercamientos para modelar los VRP:

- **Modelos de flujo de vehículos (Vehicle flow formulations)**: usa variables enteras asociadas a cada uno de los arcos del gráfico, donde se cuenta el número de veces que cada arco es cubierto con un vehículo. Este modelo es el más usado para versiones básicas de VRPs, cuando el coste total se puede expresar como la suma de todos los costes de los arcos. Sin embargo, no puede utilizarse en muchas aplicaciones prácticas puesto que presenta problemas cuando el coste depende de la secuencia de vértices o del tipo de vehículo asignado a una ruta.
- **Modelos de flujo de bienes o Modelo basado en redes (Commodity flow formulations)**: se emplean variables enteras adicionales asociadas a los arcos, los cuales representan el flujo de mercancías de cada uno de los trayectos hechos por los vehículos. Se han empezado a utilizar recientemente solo para encontrar una solución exacta.
- **Set partitioning problem**: Tienen un número exponencial de variables binarias que estás asociadas a cada una de las posibles rutas del circuito. Este modelo trata de seleccionar un conjunto de circuitos con un mínimo coste, donde cada cliente es servido una vez a la vez que satisface otras posibles restricciones. Este modelo se adapta a una gran posibilidad de costes de rutas, el cual resulta así más potente con respecto a los dos anteriores.

Para el modelo de dos índices basado en Tucker-Miller-Zemlin, la formulación sería la siguiente:

$$Función\ objetivo: \qquad Min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$Sujeto\ a: \qquad \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 2, \dots, n \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 2, \dots, n \qquad (2)$$

$$\sum_{i=1}^{n} x_{i1} = K \qquad (3)$$

$$\sum_{j=1}^{n} x_{j1} = K \qquad (4)$$

$$u_i - u_j + C \cdot x_{i,j} \leq C - d_j \quad \forall\, i,j \in V - \{1\}/i \neq j, \quad d_i + d_j \leq C \qquad (5)$$

$$d_i \leq u_i \leq C \qquad i = 2, \dots, n \quad (6)$$

$$x_{ij} \in \{\, 0, 1 \,\} \quad i,j = 1, \dots, n$$

$$u_i \geq 0 \qquad i = 2, \dots, n$$

Donde la variable u_i es una variable continua adicional que representa la carga del vehículo después de haber visitado al cliente *i*

Las anteriores restricciones indican que:

(3) Del depósito tienen que salir *K* vehículos (si el número de vehículos está prefijado).

(4) Del depósito tienen que llegar *K* vehículos (si el número de vehículos está prefijado).

(5) y (6) Son generalizaciones de las restricciones de Tucker-Miller-Zemlin propuestas para el TSP, que se mencionan en el apartado 4.2.4.1.

2.7. Métodos de resolución de los VRP

Existen diversas técnicas para resolver los Problemas de Rutas de Vehículos, estas se podrían agrupar de forma general en dos grandes grupos: métodos exactos y métodos aproximados.

2.7.1. Métodos exactos

Los métodos de enumeración total tratan de calcular todas las posibles soluciones hasta que se logra alcanzar la solución óptima del problema, lo cual conlleva un tiempo de cálculo de $O(2^n)$. Estos métodos enumerativos solo son posibles en la teoría, porque en la práctica se utilizan métodos de enumeración parcial que logran hallar una solución óptima mediante un acotamiento del conjunto de todas las soluciones posibles.

Debido a su elevado tiempo de ejecución computacional, estos métodos son apropiados para problemas pequeños en los que no haya un gran número de nodos. Además, estos problemas están planteados con la formulación de programación lineal (entera).

Algunos de estos métodos son:

- Branch and bound
- Branch and cut

2.7.2. Métodos aproximados

Debido a la complejidad de los VRP, y por tanto, a su dificultad para encontrar una solución aceptable dentro de un tiempo computacional moderado, se han propuesto unos métodos de resolución alternativos que actúan sobre un espacio de búsqueda limitado y proporcionan soluciones de una buena calidad en un tiempo computacional admisible.

Estos métodos se pueden clasificar en dos grandes clases: heurísticas y metaheurísticas.

2.7.2.1. Heurísticas

Las heurísticas clásicas comenzaron a desarrollarse entre 1960 y 1990 y, según establecen Toth y Vigo en *The Vehicle Routing Problem* (2002), se pueden clasificar en tres amplias categorías:

- **Heurísticas constructivas**: van obteniendo la solución de forma gradual incorporando en cada iteración el elemento con mayor valor para un criterio establecido. Dentro de esta heurística destaca el algoritmo de ahorros de Clarke and Wright y las heurísticas de inserción.
- **Heurísticas de dos fases**: el problema se descompone en dos componentes naturales, agrupando los vértices en rutas posibles y rutas actuales con posibles retroalimentaciones de los dos campos.
- **Heurísticas de mejora**: intentan mejorar cada posible solución mediante la prueba de intercambiar la secuencia de vértices dentro o fuera de las rutas.

2.7.2.2. Metaheurísticas

La aplicación de las metaheurísticas ha crecido notablemente durante las últimas décadas. En este caso, se realiza una exploración profunda de las regiones más importantes del espacio de soluciones. Las soluciones producidas por estos métodos son de una calidad mucho mayor que las que se obtienen mediante las heurísticas clásicas, pero a cambio de un tiempo de ejecución mayor.

Algunas de las metaheurísticas más conocidas, son las siguientes:

- Algoritmos de hormigas
- GRASP
- Recocido simulado
- Recocido determinístico
- Búsqueda tabú
- Algoritmos genéticos
- Redes neuronales

CAPÍTULO 3. HEURÍSTICAS DE CONSTRUCCIÓN

3.1. Contextualización

Las heurísticas de construcción son métodos iterativos que van añadiendo elementos hasta completar una solución, de tal forma que para cada iteración se añade el elemento con la mejor evaluación.

Se dice que los métodos de construcción greedy son estrategias voraces, devoradoras, puesto que para cada iteración seleccionan el mejor valor dado por un criterio greedy o costo. También se les denomina algoritmos miopes, ya que en cada iteración el procedimiento emplea únicamente lo que puede ver en ese momento, sin importar qué puede haber en pasos posteriores. Los nodos que se van añadiendo a la solución, pueden seguir una estrategia greedy determinista o pura, si se inserta el mejor valor para el criterio en cuestión, o también una estrategia greedy aleatorizada eligiendo cada vez un nodo al azar entre todos los candidatos o entre una lista restringida de ellos.

Dentro de esta modalidad, destacan el algoritmo de Ahorros de Clarke-Wright y las heurísticas de Inserción, en las cuales se centrará el foco en el presente Trabajo Fin de Grado.

3.2. Algoritmo de Clarke & Wright

3.2.1. Descripción

El algoritmo de Ahorros de Clarke & Wright (C&W), en inglés *Savings*, es uno de los más conocidos utilizado para los modelos VRP. Fue desarrollado en 1964 por las dos personas que llevan el nombre del algoritmo, y se aplica para problemas en los que el número de vehículos no está prefijado.

Se halla calculando el ahorro, en términos de distancia, que supone la unión de dos puntos que no pertenecían a la misma ruta y que ahora supone una ruta más grande.

Si partimos de una ruta que contiene al nodo i y al depósito, y otra que contiene al nodo j y al depósito, el ahorro que supone incluir los nodos i y j en la misma ruta (figura 6) sería el siguiente:

$$s(i,j) = d_{1i} + d_{j1} - d_{ij}$$

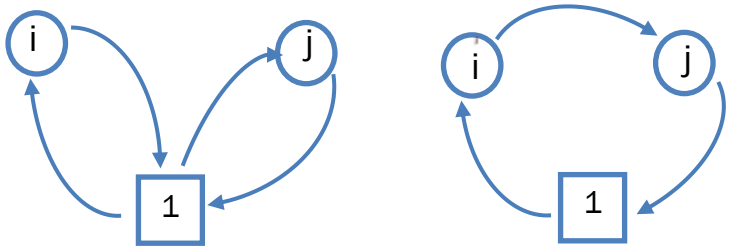

Figura 6. Unión de dos *rutas*

3.2.2. Procedimiento

Se parte de una solución inicial en la que cada ruta está compuesta por un único nodo, de tal forma que en el modelo de n clientes existan n-1 rutas (figura 7).

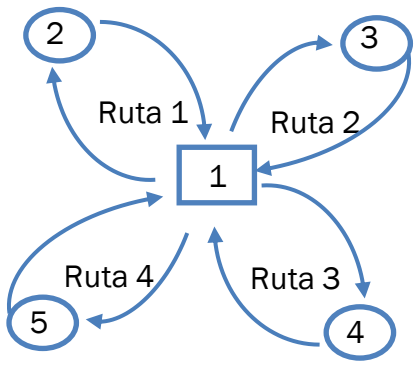

Figura 7. Solución inicial C&W

- Paso 1. Construir la matriz de ahorros para todos los nodos con la fórmula explicada anteriormente:

$$s(i,j) = d_{1i} + d_{j1} - d_{ij}$$

- Paso 2. Ordenar de mayor a menor los ahorros calculados.

- Paso 3. Se va recorriendo dicha lista y se van uniendo las rutas, conservando en todo caso la capacidad del vehículo. Para que dos rutas se puedan agrupar y por tanto se produzca una unión de dos nodos, el primero de ellos tiene que corresponder con el último nodo de su anterior ruta, así como que el segundo nodo sea el primero de su ruta, nunca pudiendo ambos pertenecer a la misma.

Para la experimentación computacional correspondiente, se ha tomado de referencia el Trabajo Fin de Grado *Problemas de rutas de vehículos: modelos, aplicaciones logísticas y métodos de resolución* de Ana Benito Quintanilla (2015).

3.3. Heurísticas de Inserción

Las heurísticas de inserción han sido probadas como métodos muy populares para resolver problemas de rutas de vehículos (*routing*) y de programación de tareas (*scheduling*), ya que son rápidas y producen soluciones buenas que son fáciles de implementar y de extender a las diferentes variantes que se puedan dar. Algunas de estas restricciones a las que se pueden adaptar sirven para resolver los problemas de ventanas de tiempo (Time Windows), VRP asimétrico y capacitado o el problema del tamaño de la flota de vehículos entre otros.

Estas heurísticas son métodos constructivos que crean una solución factible mediante la inserción iterativa en cada una de las rutas de los diferentes clientes que no han sido visitados. Las soluciones dependerán de la variante que se escoja, las cuales determinan el criterio a seguir a la hora de decidir qué clientes insertar y dónde insertarlos en la ruta parcial.

Las heurísticas de inserción se clasifican principalmente en secuencial y paralelo según se considere una única ruta para la inserción de los clientes o todas las rutas activas. La principal desventaja de las primeras con respecto a las del método paralelo, es que como los nodos se van incorporando en una ruta hasta que no es posible ninguna otra inserción, es posible que los últimos nodos introducidos se encuentren a una distancia considerable entre sí, y por tanto, se empeore la calidad de las rutas. Es por ello, por lo que se diseñaron algoritmos de inserción en paralelo para enmendar esta deficiencia al poder realizar el acoplamiento en la mejor ruta para ese nodo.

Hoy en día, existe un creciente interés por actuar debidamente ante cualquier tipo de demanda exigida por los clientes. Los desarrollos tecnológicos han creado una situación en el que en cada momento somos capaces de saber dónde está exactamente nuestro transporte y poder comunicarnos. En estos casos, las decisiones deben hacerse en cuestión de muy poco tiempo y es por ello, que las heurísticas de inserción pueden proporcionar una de las pocas viables opciones para la toma de decisiones.

3.3.1. Inserción Secuencial de Mole y Jameson

3.3.1.1. Descripción

La heurística de Mole y Jameson (1976) se basa en ir insertando los clientes en la ruta actual hasta que esta se completa y se pueda comenzar con otra nueva ruta.

Esta heurística utiliza dos medidas para decidir el cliente a insertar en la ruta parcial. Para todos aquellos clientes que aún no han sido visitados, se calcula la mejor posición para colocar este cliente en la ruta parcial, teniendo en cuenta las distancias y las demandas y sin volver a reordenar los nodos que actualmente están en la ruta.

El algoritmo utiliza dos parámetros $\boldsymbol{\lambda}, \mu$ para modificar los criterios de los clientes a insertar. Si el parámetro $\boldsymbol{\lambda}$ crece se favorece la inserción de clientes entre nodos lejanos, y si es μ el que crece se prioriza a los clientes lejanos del depósito.

En una iteración cualquiera, se tiene una ruta $(v_1, \dots, v_t, v_{t+1})$ donde $(v_1 = v_{t+1} = 1)$ es el depósito. Entonces, si w es un nodo que no ha sido visitado, el coste de incorporar w entre los nodos v_i y v_{i+1} $(1 \leq i \leq t)$ es el siguiente:

$$C1(v_i, w) = \begin{cases} c_{v_i,w} + c_{w,v_i+1} - \lambda c_{v_i,v_i+1} & \text{si } (v_1, \dots, v_i, w, v_{i+1}, \dots, v_{t+1}) \text{ es factible} \\ \infty & \text{si no es factible} \end{cases}$$

De forma gráfica vendría a significar la suma de los costes del nuevo nodo a insertar (w) con respecto a los dos nodos contiguos (v_1y v_{i+1}) en la ruta actual menos el coste de estos dos (figura 8):

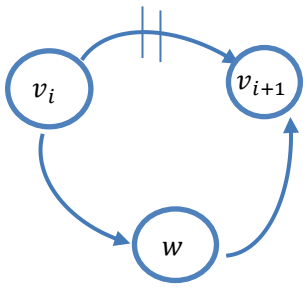

Figura 8. Inserción de un elemento

De tal forma que la mejor posición para incorporar al cliente w en la ruta a construir será:

$$i(w) = \arg min_{\ i=1..t}\ C1(v_i, w)$$

El problema de utilizar esta metodología es que los clientes más cercanos a la ruta son los que se van insertando, mientras que los que se sitúan más lejos sólo son tenidos en cuenta en las últimas iteraciones. Para evitar este hecho, se utiliza el segundo criterio de esta heurística, llamado criterio de urgencia C2, para cada cliente no visitado w:

$$C2(v_i, w) = \mu\, c_{1,w} - C1(v_i, w)$$

En cada una de las iteraciones se busca el cliente que maximice el valor C2 y se le inserta en la posición para el valor mínimo de C1 de ese mismo nodo.

Varias reglas estándar de inserción están gobernadas por los dos parámetros λ y μ. Por ejemplo, cuando $\lambda = 1$ y $\mu = 0$ el algoritmo inserta el vértice que supone la mínima distancia extra. Si $\lambda = \mu = 0$ el vértice a insertar corresponde con el de la mínima suma de distancias con los dos vecinos. Si $\lambda > 0$ y $\mu = \infty$ se inserta el vértice más alejado al depósito.

Para comprobar si una posible inserción de un cliente es factible, es necesario calcular la carga restante del vehículo así como la demanda de ese cliente. Cuando se da el caso de que la inserción no es factible, ya que su correspondiente demanda es mayor que la carga restante disponible del vehículo, se da por terminada la ruta y si aún quedan nodos por visitar, se comienza una nueva ruta.

Pueden utilizarse diferentes alternativas para seleccionar el cliente que iniciará la ruta, como seleccionar el más lejano o el más cercano al depósito o también de forma aleatoria.

Por lo general, los últimos clientes que se insertan suelen estar alejados entre sí y provocan que la calidad de las rutas no sea de las mejores. Por ello, se propone utilizar un procedimiento para intercambiar los clientes cuando el algoritmo haya acabado como se verá en el capítulo 4 con los métodos de mejora.

3.3.1.2. Procedimiento

El pseudocódigo de este método es el siguiente:

```
- Inicializaciones: nvis=0   [nº nodos visitados]

while (nvis<n) do
        - Se inicializa ruta con nodo inicial:      nvis=nvis+1
                                                    ruta_activa=1
        while (ruta_activa=1) do
                forall (cliente w no visitado) do
                        if (carga(r)+dem(w)<C) then
                                - calcular C1, C2,
                        else (no existe ningún cliente que se pueda insertar) then
                                - ruta finalizada:   ruta_activa=0
                                - opción de mejorar la ruta
                        end-if
                end-do
                if (algún cliente no visitado puede ser insertado) then
                        - se inserta el nodo con C2 max en la posición con C1 mín:
                          nvis=nvis+1
                end-if
        end-do
end-do
```

3.3.1.3. Ejemplo numérico

A continuación se muestra un ejemplo del procedimiento a seguir para la formación de las rutas siguiendo este método.

DATOS

Se requiere de la matriz de distancias entre los clientes (tabla 1) y de las demandas de los mismos (tabla 2).

	1	2	3	4	5	6	7
1	0	12,042	8,062	29,155	18,110	24,739	22,023
2	12,042	0	14,213	40,311	30,017	25,710	10
3	8,063	14,213	0	26,926	21,472	16,763	29,682
4	29,156	40,311	26,926	0	18,385	32,28	49,649
5	18,110	30,017	21,471	18,385	0	35,384	40,013
6	24,739	25,710	16,763	32,280	35,384	0	29,682
7	22,023	10	29,682	49,649	40,013	29,682	0

Tabla 1. Matriz distancias

1	2	3	4	5	6	7
0	10	20	15	12	17	15

Tabla 2. Demandas clientes 1

Capacidad de los vehículos:

Capacidad	50

Valores escogidos para λ y μ:

λ	1
μ	1

En primer lugar, establecemos una semilla para inicializar la primera ruta. Para este ejemplo vamos a seleccionar el nodo según el criterio del nodo más alejado del depósito.

Nodo inicial	Ruta	Carga	Distancia
4	1-4-1	22	58,310

Para todos los nodos que no han sido visitados se calcula el coste de inserción después de los nodos presentes en la ruta. Se escoge el cliente con el mayor valor para el criterio de urgencia C2 y se le inserta después del nodo con el menor valor de C1 para ese mismo nodo.

Ruta 1									
$v_i = 1$					$v_{i+1} = 4$				
$C1(v_i, w)$	$c_{v_i,w}$	c_{w,v_i+1}	$\lambda c_{v_i,v_i+1}$	$C1(v_i, w)$	$C2(v_i, w)$	$\mu c_{1,w}$	$C1(v_i, w)$	$C2(v_i, w)$	Carga
C1(1,2)	12,042	40,311	29,155	23,198	C2(1,2)	12,042	23,198	-11,157	25
C1(1,3)	8,062	26,926	29,155	5,833	C2(1,3)	8,062	5,833	2,229	35
C1(1,5)	18,111	18,385	29,155	7,342	C2(1,5)	18,112	7,3419	10,770	27
C1(1,6)	24,739	32,280	29,155	27,864	C2(1,6)	24,739	27,864	-3,125	32
C1(1,7)	22,023	49,649	29,155	42,517	C2(1,7)	22,023	42,517	-20,494	30
$v_i = 4$					$v_{i+1} = 1$				
$C1(v_i, w)$	$c_{v_i,w}$	c_{w,v_i+1}	$\lambda c_{v_i,v_i+1}$	$C1(v_i, w)$	$C2(v_i, w)$	$\mu c_{1,w}$	$C1(v_i, w)$	$C2(v_i, w)$	Carga
C1(4,2)	40,311	12,042	29,155	23,198	C2(4,2)	1,042	23,198	-11,157	25
C1(4,3)	26,626	8,062	29,548	5,833	C2(4,3)	8,063	5,833	2,229	35
C1(4,5)	18,386	18,111	29,155	7,341	C2(4,5)	18,111	7,341	10,770	27
C1(4,6)	32,280	24,739	29,155	27,864	C2(4,6)	24,739	27,865	-3,125	32
C1(4,7)	49,649	22,023	29,155	42,517	C2(4,7)	22,023	42,517	-20,494	30
Carga:									**27**

Nodo insertado	Ruta	Carga	Distancia
5	1-5-4-1	27	65,650

Se busca insertar un nodo entre los que constituyen la solución parcial de la ruta 1.

Ruta 1									
$v_i = 1$					$v_{i+1} = 5$				
$C1(v_i, w)$	$c_{v_i,w}$	$c_{w,\, v_i+1}$	$\lambda c_{v_i,\, v_i+1}$	$C1(v_i, w)$	$C2(v_i, w)$	$\mu c_{1,w}$	$C1(v_i, w)$	$C2(v_i, w)$	Carga
C1(1,2)	12,042	30,017	18,111	23,948	C2(1,2)	12,042	23,948	-11,905	37
C1(1,3)	8,062	21,471	18,111	11,422	C2(1,3)	8,062	11,422	-3,360	47
C1(1,6)	24,739	35,384	18,111	42,012	C2(1,6)	24,739	42,012	-17,273	44
C1(1,7)	22,023	40,013	18,111	43,924	C2(1,7)	22,023	43,924	-21,902	42
$v_i = 5$					$v_{i+1} = 4$				
$C1(v_i, w)$	$c_{v_i,w}$	$c_{w,\, v_i+1}$	$\lambda c_{v_i,\, v_i+1}$	$C1(v_i, w)$	$C2(v_i, w)$	$\mu c_{1,w}$	$C1(v_i, w)$	$C2(v_i, w)$	Carga
C1(5,2)	30,017	40,311	18,385	51,943	C2(5,2)	12,042	51,943	-39,902	37
C1(5,3)	21,471	26,928	18,385	30,012	C2(5,3)	8,062	30,012	-21,950	47
C1(5,6)	35,384	32,280	18,385	49,279	C2(5,6)	24,739	49,279	-24,540	44
C1(5,7)	40,013	49,649	18,385	71,277	C2(5,7)	22,023	71,277	-49,254	42
$v_i = 4$					$v_{i+1} = 1$				
$C1(v_i, w)$	$c_{v_i,w}$	$c_{w,\, v_i+1}$	$\lambda c_{v_i,\, v_i+1}$	$C1(v_i, w)$	$C2(v_i, w)$	$\mu c_{1,w}$	$C1(v_i, w)$	$C2(v_i, w)$	Carga
C1(4,2)	40,311	12,042	29,155	23,198	C2(4,2)	12,042	23,198	-11,1576	37
C1(4,3)	26,926	8,062	29,155	5,833	C2(4,3)	8,062	5,833	2,229	47
C1(4,6)	32,280	24,739	29,155	27,864	C2(4,6)	24,739	27,864	-3,125	44
C1(4,7)	49,649	22,023	29,155	42,517	C2(4,7)	22,023	42,517	-20,494	42
Carga:									47

El nodo 3 resulta ser el de mayor valor para el criterio C2, por lo que es el siguiente en incorporarse, y lo podría hacer detrás de 1, 5 o 4. El mínimo valor de C1 es el que determina la posición del nodo 3 a insertar (C1(1,3), C1(5,3), C1(4,3)), cumpliéndose este mínimo para el nodo 4.

Nodo insertado	Ruta	Carga	Distancia
3	1-5-4-3-1	47	71,484

Como no existe ningún nodo que pueda incorporarse a la ruta debido a la restricción de la capacidad, se crea una nueva ruta con una semilla para inicializarla:

Nodo inicial	Ruta	Carga	Distancia
6	1-6-1	17	49,477

Ruta 2									
$v_i = 1$					$v_{i+1} = 6$				
$C1(v_i, w)$	$c_{v_i,w}$	$c_{w,\, v_i+1}$	$\lambda c_{v_i,\, v_i+1}$	$C1(v_i, w)$	$C2(v_i, w)$	$\mu c_{1,w}$	$C1(v_i, w)$	$C2(v_i, w)$	Carga
C1(1,2)	12,042	25,712	24,739	13,013	C2(1,2)	12,042	13,013	–0,971	27
C1(1,7)	22,023	29,682	24,740	26,966	C2(1,7)	22,023	26,966	-4,943	32
$v_i = 6$					$v_{i+1} = 1$				
$C1(v_i, w)$	$c_{v_i,w}$	$c_{w,\, v_i+1}$	$\lambda c_{v_i,\, v_i+1}$	$C1(v_i, w)$	$C2(v_i, w)$	$\mu c_{1,w}$	$C1(v_i, w)$	$C2(v_i, w)$	Carga
C1(6,2)	25,710	12,042	24,739	13,013	C2(6,2)	12,042	13,013	-0,971	27
C1(6,7)	29,682	22,023	24,739	26,966	C2(6,7)	22,023	26,966	-4,943	32
Carga:									27

Nodo insertado	Ruta	Carga	Distancia
2	1-2-6-1	27	62,490

Ruta 2									
$v_i = 1$					$v_{i+1} = 2$				
$C1(v_i,w)$	$c_{v_i,w}$	c_{w,v_i+1}	$\lambda c_{v_i,v_i+1}$	$C1(v_i,w)$	$C2(v_i,w)$	$\mu c_{1,w}$	$C1(v_i,w)$	$C2(v_i,w)$	Carga
C1(1,7)	22,023	10	12,042	19,981	C2(1,7)	22,023	19,981	2,0416	42
$v_i = 2$					$v_{i+1} = 6$				
$C1(v_i,w)$	$c_{v_i,w}$	c_{w,v_i+1}	$\lambda c_{v_i,v_i+1}$	$C1(v_i,w)$	$C2(v_i,w)$	$\mu c_{1,w}$	$C1(v_i,w)$	$C2(v_i,w)$	Carga
C1(2,7)	10	22,023	25,710	13,972	C2(2,7)	22,023	13,972	8,052	42
$v_i = 6$					$v_{i+1} = 1$				
$C1(v_i,w)$	$c_{v_i,w}$	c_{w,v_i+1}	$\lambda c_{v_i,v_i+1}$	$C1(v_i,w)$	$C2(v_i,w)$	$\mu c_{1,w}$	$C1(v_i,w)$	$C2(v_i,w)$	Carga
C1(6,7)	29,682	22,023	24,739	26,966	C2(6,7)	22,023	26,966	-4,943	42
Carga:									**42**

Nodo insertado	Ruta	Carga	Distancia
7	1-2-7-6-1	42	76,462

Todos los nodos han sido insertados, por lo que la solución final es la siguiente:

Ruta	Carga	Distancia
1-5-4-3-1	47	71,484
1-2-7-6-1	42	76,462
Distancia total:		**147,946**

3.3.2. Inserción en Paralelo

3.3.2.1. Descripción

La heurística de inserción en paralelo se fundamenta en ir construyendo las rutas simultáneamente, introduciendo cada cliente en la ruta con el coste de inserción menor. A la hora de crear una nueva ruta, no es necesario definir una semilla, sino que cuando sea mejor insertar un nuevo cliente en una ruta vacía que insertarlo en una de las rutas existentes, este cliente va a ser el primer nodo de la nueva ruta.

El coste de insertar el cliente *w* después del nodo v_i y antes del nodo v_{i+1} de la ruta *r* es el siguiente:

$$C(v_i, w, r) = c_{v_i,w,r} + c_{w,\, v_i+1,r} - c_{\,v_i,\, v_i+1,r}$$

Para el caso de insertar un cliente *w* en una nueva ruta *r'*, su costo es:

$$C(1, w, r') = c_{1,w} + c_{w,1}$$

3.3.2.2. Procedimiento

El pseudocódigo de este método es el siguiente:

- Se inicializa la primera ruta con un nodo inicial: *nvis=1* *[nº nodos visitados]*
nra=1 *[nº de rutas activas]*
ruta_activa=1
while (nvis<n) do
forall (cliente w no visitado) do
forall (ruta r| carga(r)+dem(w)<C) do
forall (posición dentro de la ruta r) do
calcular min $C(v_i, w, r)$
end-do
end-do
if $(c_{1,w} + c_{w,1} <$ *min* $C(v_i, w, r))$ *then*
- Se crea nueva ruta: *nueva_ruta:=1*
else
nueva_ruta:=0
end-if
end-do
if (nueva_ruta = 0) then
- se inserta nodo w en la ruta r después del nodo v_i*:* *nvis=nvis+1*
else
- se inicializa ruta con nodo inicial w: *nra:=nra+1*
nvis=nvis+1
end-if
end-do

3.3.2.3. Ejemplo numérico

A continuación se muestra un ejemplo del procedimiento a seguir para la formación de las rutas siguiendo este método.

DATOS

Se utiliza la misma matriz de distancias entre los clientes del ejemplo anterior (tabla 1) y la demanda de los clientes es la que se muestra en la tabla 3.

1	2	3	4	5	6	7
0	10	20	15	5	17	15

Tabla 3. Demandas clientes 2

Capacidad de los vehículos:

Capacidad	50

En primer lugar, establecemos una semilla para inicializar la primera ruta. Para este ejemplo vamos a seleccionar el nodo según el criterio del nodo más cerca del depósito.

Nodo inicial	Ruta	Carga	Distancia
3	1-3-1	20	16,125

Para todos los nodos que no han sido visitados se calcula el coste de inserción después de los nodos presentes en las diferentes rutas inicializadas y el coste de inserción en una nueva ruta. Se escoge el cliente con el menor coste de inserción.

	Ruta 1			Nueva ruta	
w	$c_{v_i,w}$		Carga	$c_{v_i,w}$	Carga
	$v_i = 1$	$v_i = 3$	20	1	0
2	18,192	18,192	30	24,083	10
4	48,018	48,018	35	58,310	15
5	31,519	31,519	5	36,222	5
6	33,439	33,439	37	49,477	17
7	36,808	36,808	35	44,045	15

	Ruta 1				Nueva ruta	
w	$c_{v_i,w}$			Carga	$c_{v_i,w}$	Carga
	$v_i = 1$	$v_i = 2$	$v_i = 3$	30	1	0
4	57,425	53,024	48,018	45	58,310	15
5	36,086	37,275	31,519	35	36,222	5
6	38,407	28,260	33,439	47	49,477	17
7	19,981	18,635	36,808	45	44,045	15

Nodo insertado	Ruta	Carga	Distancia
2	1-2-3-1	30	34,317

Nodo insertado	Ruta	Carga	Distancia
7	1-2-7-3-1	45	52,951

	Ruta 1					Nueva ruta	
w	$c_{v_i,w}$				Carga	$c_{v_i,w}$	Carga
	$v_i = 1$	$v_i = 2$	$v_i = 7$	$v_i = 3$	45	1	0
4	-	-	-	-	Sobrepasa	58,310	15
5	36,086	60,029	38,636	31,519	50	36,222	5
6	-	-	-	-	Sobrepasa	49,477	17

Nodo insertado	Ruta	Carga	Distancia
5	1-2-7-3-5-1	50	84,470

La carga acumulada es la misma de la capacidad del vehículo, por lo que la ruta 1 quedaría completa, y se busca ahora el cliente para inicializar la siguiente ruta.

	Ruta 1		Nueva ruta	
w	$c_{v_i,w}$	Carga	$c_{v_i,w}$	Carga
	Ruta completa	50	1	0
4	-	-	58,310	15
6	-	-	49,477	17

Nodo inicial	Ruta	Carga	Distancia
6	1-6-1	17	49,477

Y se vuelve a hacer el mismo proceso para las rutas activas, en este caso solo sería con la ruta 2, y para una nueva ruta.

	Ruta 2			Nueva ruta	
w	$c_{v_i,w}$		Carga	$c_{v_i,w}$	Carga
	$v_i = 1$	$v_i = 6$	17	1	0
4	29,430	29,030	32	58,310	15

Nodo inicial	Ruta	Carga	Distancia
4	1-4-6-1	32	86,173

Todos los nodos han sido insertados, por lo que la solución final es la siguiente:

Ruta	Carga	Distancia
1-2-7-3-5-1	50	84,477
1-4-6-1	32	86,173
Distancia total:		**170,650**

3.3.3. Inserción en Paralelo de Christofides, Mingozzi y Toth

3.3.3.1. Descripción

Según se establece en *Heurísticas para Problemas de Ruteo de Vehículos* de Alfredo Olivera (2004), la heurística propuesta por Christofides, Mingozzi y Toth (1979) opera en dos fases. En la primera fase se lleva a cabo una inserción secuencial para determinar el número de rutas necesarias, así como el nodo inicial de cada una de ellas. Es en la segunda fase cuando se realiza la inserción de los restantes nodos en función de la urgencia o diferencia que supone insertar el nodo en su ruta asociada y en la segunda mejor ruta.

Para inicializar una ruta en la primera fase se selecciona un cliente v_k dentro de los no visitados y se halla el coste de insertar un cliente no visitado en esa ruta como:

$$\delta_{w,v_k} = \begin{cases} c_{1,w} + \lambda\, c_{w,\,v_k} & \text{si es factible} \\ \infty & \text{si no es factible} \end{cases}$$

El parámetro λ al aumentar penaliza la inserción de nodos lejanos entre sí.

3.3.3.2. Procedimiento

FASE 1:

En esta fase se va a realizar un algoritmo de inserción secuencial para insertar los clientes con menor coste de inserción en la ruta hasta que esta se completa y no pueden añadirse más clientes, y por tanto se inicializa otra ruta. En este procedimiento no interesa la posición en la que se deberían colocarse los nodos, sino que lo importante es el nodo inicial de cada una de las rutas y el número de ellas.

- Paso 1 (primera ruta). Se inicializa la primera ruta con k=1.

- Paso 2 (cliente inicial). Se selecciona un cliente inicial v_k dentro de los clientes no visitados para insertar en la ruta. Para cada cliente w no visitado, se calcula su coste de inserción en la ruta como δ_{w,v_k}.

- Paso 3 (inserción). Se halla $w* = \arg min_{\,w}\ \delta_{w,v_k}$ sobre los clientes no visitados y se inserta en la ruta al ser el cliente con el menor coste de inserción en la ruta. Si existen clientes que puedan ser insertados en esta misma ruta, volver a hacer el paso 3, sino pasar al siguiente paso.

- Paso 4 (nueva ruta). Si faltan clientes por visitar, ir al paso 2 con *k=k+1*, y si no terminar fase 1 y realizar la fase 2.

Para la siguiente fase se van a crear las rutas obtenidas anteriormente junto con su cliente inicial. En primer lugar se realiza una asociación de todos los clientes no visitados con la ruta más cercana. Y para decidir el orden de inserción de estos clientes en su ruta, se define una urgencia de inserción como la diferencia del coste entre la ruta en la que está asociado y la segunda mejor ruta. Cuanto mayor sea esta diferencia, mayor es la urgencia por introducir el nodo en la ruta en la que está asociado.

FASE 2

- Paso 5 (inicialización). Se crean las *k* rutas obtenidas anteriormente $r_t = (1, v_t, 1)$ con $t = 1, \dots, k$. El conjunto de rutas es $J = \{ r_t, \dots, r_k \}$.

- Paso 6 (asociación). Para cada cliente no visitado w, calcular $t_w = \arg min_{\, t|r_t \in J} \ \delta_{w,v_t}$ para asociarle la ruta t con el menor coste de inserción.

- Paso 7 (urgencias). Se selecciona una ruta cualquiera $r_t \in J$ y se hace $J := J \setminus \{r_t\}$. Para cada cliente *w* tal que su ruta asociada coincida con la ruta seleccionada $t_w = t$, se calcula la ruta *t'* que supone el segundo mejor coste de inserción $t'_w = \arg min_{\, t|r_t \in J} \ \delta_{w,v_t}$ y se halla la diferencia entre el coste de la mejor y la segunda mejor ruta $\tau_w = t'_w - t_w$.

- Paso 8 (inserción). Se halla el nodo con la mayor diferencia $w* = \arg max_{\, w|t_w = t} \ \tau_w$, lo que supone una mayor urgencia por insertar el nodo en la ruta en la que está asociada. Se introduce el nodo $w*$ en la mejor posición dentro de la ruta. Si quedan clientes asociados a la ruta r_t que no se han insertado, volver a realizar el paso 8.

- Paso 9 (finalización). Si quedan clientes asociados a alguna ruta y no se han insertado, ir al paso 6. Y para el caso de que todos los clientes hayan sido visitados, entonces se termina el algoritmo, y si no, volver a realizar todo el proceso comenzando por las fase 1 para los clientes no visitados.

El pseudocódigo de este método es el siguiente:

```
- Inicializaciones:    nvis=0  y nvis_total=0      [nº nodos visitados]
                       nra=0                       [nº de rutas activas]
while (nvis_total<n) do
        +FASE 1:
        while (nvis<n) do
                - Se inicializa ruta k con nodo inicial v_k: nvis=nvis+1; nra=nra+1
                while (se puedan insertar clientes en la ruta k) do
                        forall (cliente w no visitado| carga(k)+dem(w)<C) do
                                - calcular δ_{w,v_k}, w*
                        end-do
                        - insertar el nodo w*:   nvis=nvis+1
                end-do
        end-do
        +FASE 2:
        nvis_total=nra
        k=nra           [nº de rutas]
        while (existan nodos no visitados que puedan ser insertados) do
                - Asociación de cada nodo a una ruta:
                forall (cliente w no visitado) do
                        forall (ruta t | carga(t)+dem(w)<C) do
                                - calcular t_w
                        end-do
                end-do
                - Urgencias:
                - seleccionar ruta t
                while (existan clientes no visitados asociados a ruta t) do
                        forall (cliente w | t_w = t) do
                                - calcular t'_w, τ_w, w*
                        end-do
                        - insertar w* en la mejor posición de la ruta
                end-do
        end-do
end-do
```

3.3.3.3. Ejemplo numérico

A continuación se muestra un ejemplo del procedimiento a seguir para la formación de las rutas siguiendo este método.

DATOS

Se utiliza la misma matriz de distancias entre los clientes de los ejemplos anteriores (tabla 1) y la demanda de los clientes es la que se muestra en la tabla 4.

1	2	3	4	5	6	7
0	10	20	15	12	37	15

Tabla 4. Demandas clientes 3

Capacidad de los vehículos:

Capacidad	50

En la Fase 1 se lleva a cabo una inserción secuencial sin atender a la ubicación que tienen los clientes dentro de la ruta. De esta fase se recogerá el número de rutas necesarias, así como el nodo inicial de cada una de ellas.

Para este ejemplo vamos a seleccionar el nodo que inicializa cada ruta según el criterio del nodo más cercano al depósito.

Nodo inicial	Ruta	Carga
3	1-3-1	20
7	1-7-1	15
6	1-6-1	37

FASE 2

- Asociación de cada nodo no visitado a su ruta más cercana

	Ruta 1		Ruta 2		Ruta 3		
w	δ_{w,v_k}	Carga	δ_{w,v_k}	Carga	δ_{w,v_k}	Carga	t_w
	3	20	7	15	6	37	
2	26,254	30	22,042	25	52,353	47	2
4	56,081	35	78,804	30	-	Sobrepasa	1
5	39,581	32	58,123	27	53,494	49	1

- Urgencias. Para los nodos asociados a la ruta 1 (nodos 4 y 5) se calcula la diferencia con respecto a la segunda mejor ruta para insertar.

$t = 1$

	Ruta 1	Ruta 2		Ruta 3			
w	$\delta_{w,v_k}(t_w = 1)$	δ_{w,v_k}	Carga	δ_{w,v_k}	Carga	t'_w	τ_w
	3	7	15	6	37		
4	56,081	78,804		∞	Sobrepasa	2	22,723
5	39,581	58,123		53,494		3	13,913

Inserto el nodo 4 en la ruta 1 (w*=4), puesto que es aquel con una mayor urgencia por ser insertado en esa ruta, y busco la mejor posición dentro de ella para colocarle.

Nodo insertado	Ruta	Carga	Distancia
4	1-4-3-1	35	64,142
-	1-7-1	15	44,045
-	1-6-1	37	49,477

Como existe otro nodo asociado a esta misma ruta, buscamos el siguiente w*.

	Ruta 1	Ruta 2		Ruta 3			
w	$\delta_{w,v_k}(t_w = 1)$	δ_{w,v_k}	Carga	δ_{w,v_k}	Carga	t'_w	τ_w
	3	7	15	6	37		

5	39,581	58,123		53,494		3	13,913

Inserto el nodo 5 en la ruta 1 y busco la mejor posición dentro de la ruta para colocarle.

Nodo insertado	Ruta	Carga	Distancia
5	1-5-4-3-1	47	71,483
-	1-7-1	15	44,045
-	1-6-1	37	49,477

Al no quedar clientes asociados a la ruta 1 que puedan ser insertados, volvemos al paso de asociación.

- Asociación

	Ruta 1		Ruta 2		Ruta 3		
w	δ_{w,v_k}	Carga	δ_{w,v_k}	Carga	δ_{w,v_k}	Carga	t_w
	3	47	7	15	6	37	
2	26,254	Sobrepasa	22,042	25	37,752	47	2

- Urgencias

Ningún nodo asociado a la primera ruta

$t = 2$

	Ruta 2	Ruta 1		Ruta 3			
w	$\delta_{w,v_k}(t_w = 2)$	δ_{w,v_k}	Carga	δ_{w,v_k}	Carga	t'_w	τ_w
	7	3	47	6	37		
2	22,042	∞	Sobrepasa	37,752		3	15,710

Inserto el nodo 2 en la ruta 2 y busco la mejor posición dentro de la ruta para colocarle.

Nodo insertado	Ruta	Carga	Distancia
-	1-3-4-5-1	47	71,483
2	1-2-7-1	25	44,064
-	1-6-1	37	49,477
Distancia total:			**165,024**

3.3.4. Métodos de construcción greedy aleatorizados

Con el pretexto de conseguir una mayor diversificación y poder visitar otras posibles soluciones, se insta a emplear un método greedy aleatorizado, que permita que el proceso se dirija a regiones no exploradas.

Existen diferentes estrategias greedy aleatorizadas, como por ejemplo la introducción de cada uno de los nodos de tal forma que se seleccione al azar entre todos los candidatos o entre una lista restringida de ellos.

Estos métodos se denominan a su vez Multi-Start (Multi-Arranque), dado que de cada iteración se obtiene una solución y la mejor de todas ellas es la salida final del algoritmo.

Lista Restringida de Candidatos (RCL)

Una forma de introducir aleatoriedad a los algoritmos es mediante la utilización de una lista restringida de candidatos (RCL: Restricted Candidate List), en la cual se incluyen los mejores aspirantes a ser partícipes de la solución. Es de esta lista desde donde se selecciona de forma aleatoria el nodo a insertar en vez de considerar todo el conjunto.

A su vez, esta lista puede tener un número fijo de elementos (restricción por cardinalidad) o estar constituida por los elementos candidatos *e,* cuyo valor *c(e)* está dentro de la función miope (restricción por valor). Esta restricción por valor delimita a los candidatos cuyo valor está dentro del rango entre el mejor valor c^* y el peor c_*, tal que $c(e) \leq c_* + \alpha\,(c^* - c_*)$, y $0 \leq \alpha \leq 1$.

En el caso de la restricción por cardinalidad, la lista restringida viene delimitada por un parámetro K a establecer que determina su tamaño, de tal forma que si toma el valor de 1 equivaldría a un greedy puro donde se selecciona siempre el mejor valor y no existe aleatoriedad, o la opción opuesta de establecer un valor muy alto para este parámetro y considerar un elevado número de candidatos provocando así una aleatoriedad casi completa.

3.3.5. Experimentación computacional

Se comenzará la experimentación de forma individual con cada uno de los algoritmos descritos anteriormente analizando cada una de sus diferentes variantes, para que posteriormente se establezcan unas conclusiones generales comparando las soluciones ofrecidas de los tres procedimientos, así como el tiempo computacional requerido para ello.

- *Métodos de construcción greedy determinista*

Análisis de los parámetros λ y μ

- **Inserción Secuencial de Mole & Jameson**

En primer lugar, se desea conocer cómo varían los resultados finales en función de los valores que se asignen a los dos parámetros λ y μ que contiene este método, situando los mejores resultados para unos valores de los parámetros de $0{,}1 \leq \lambda \leq 2$ y $0 \leq \mu \leq 2$ según Toth y Vigo (2002). Para ello, se van a diferenciar los resultados computacionales según se seleccione el nodo más alejado (tabla 5) o más cercano (tabla 7) como nodo inicial de cada una de las rutas, y se va a observar la mejora que supone modificar el valor de los parámetros λ y μ con respecto a la solución con unos valores fijos e intermedios de λ=1 y μ=1.

- Nodo inicial: lejano

Modelo	Mejor combinación			Solución (Km) λ=1 y μ=1	Variación	Tiempo ejecución (s)
	λ	μ	Solución (Km)			
E021-06m	1	1	469,12	469,12	0,00%	0,00
E022-06m	1	1	542,57	542,57	0,00%	0,00
E023-05S	1,1	1,1	624,92	746,66	16,31%	0,00
E026-08m	1,1	1,2	704,45	725,08	2,85%	0,00
E030-04S	1	1,4	591,19	638,04	7,34%	0,00
E031-09h	1	1	613,44	613,44	0,00%	0,02
E033-05s	1,2	1,3	895,51	912,80	1,89%	0,02
E036-11h	1,2	1,4	723,80	771,18	6,14%	0,02
E041-14h	1	1	954,55	954,55	0,00%	0,02
E045-04f	1,1	1,2	898,19	972,68	7,66%	0,02
E051-05e	1,1	1,2	576,73	592,18	2,61%	0,02
E072-04f	1	1,1	339,80	339,80	0,00%	0,03
E076C09r	1,1	1,4	1419,60	1567,85	9,46%	0,02
E101-14s	1	1,2	1272,61	1327,87	4,16%	0,02
E121-07c	1	1,2	1107,20	1117,07	0,88%	0,05
E151-12c	1,5	1	1242,22	1272,54	2,38%	0,06
E200-17c	1,5	1,1	1558,38	1612,17	3,34%	0,10

Tabla 5. Resultados obtenidos de la heurística de Inserción Secuencial de Mole & Jameson determinista con el nodo inicial más alejado en función de la modificación de los parámetros λ y μ.

Se comprueba que para el 70,59% de los casos, se produce una mejora en la solución final tras la modificación de los parámetros λ y/o μ, siendo esta mejora potencial de un 3,82% (gráfico 1).

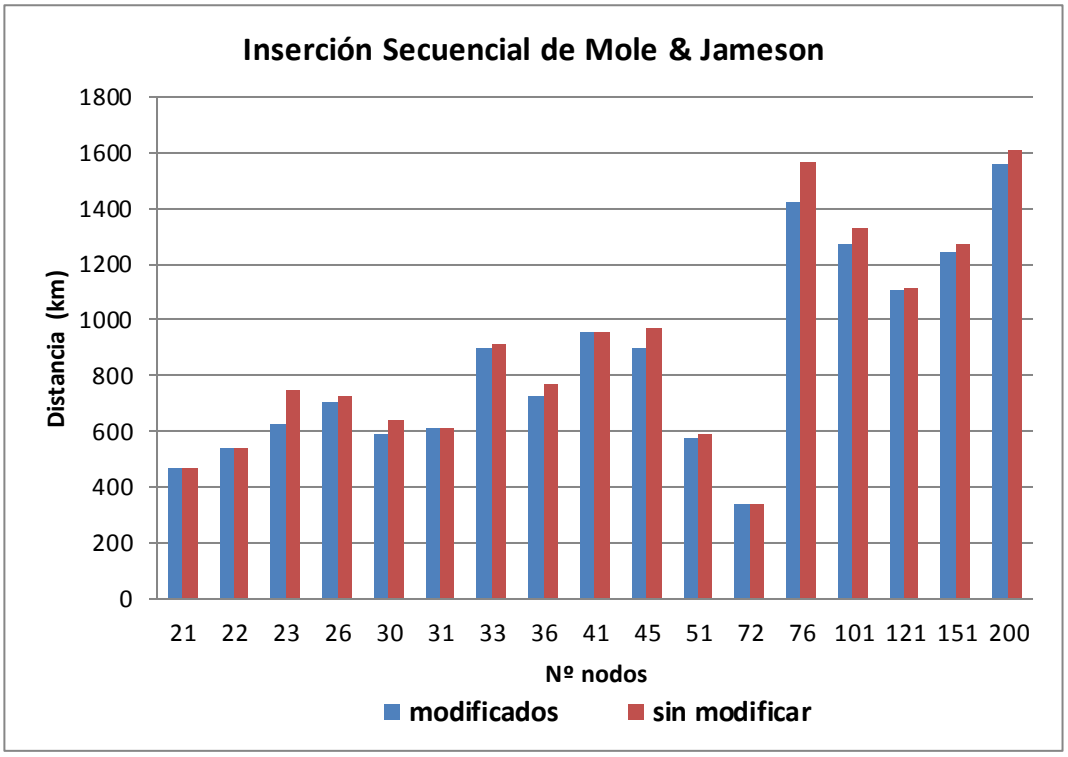

Gráfico 1. Distancia recorrida de la heurística de Inserción Secuencial de Mole & Jameson con el nodo inicial más lejano al depósito en función de la modificación de los parámetros λ y μ.

Si se recopila el número de veces que se repite cada uno de las diferentes combinaciones de los parámetros que permiten mejorar la solución, se obtienen los datos mostrados en la tabla 6.

λ	μ	N° veces repetidas
1	1	5
1	1,2	2
1	1,4	1
1,1	1,1	1
1,1	1,2	3
1,1	1,4	1
1,2	1,3	1
1,2	1,4	1
1,5	1	1
1,5	1,1	1

Tabla 6. Frecuencia las diferentes posibles combinaciones de los parámetros λ y μ que reproducen la mejor solución para la heurística de Inserción Secuencial de Mole & Jameson con el nodo inicial más lejano al depósito.

Tras analizar las diferentes combinaciones de los parámetros, no se aprecia ningún claro patrón para establecer cuál de ellas puede ser la mejor posibilidad, pero sí destacar que los mejores valores para λ y μ se sitúan entre 1 y 1,2.

- Nodo inicial: cercano

Modelo	Mejor combinación			Solución (Km) λ=1 y μ=1	Variación	Tiempo ejecución (s)
	λ	μ	Solución (Km)			
E021-06m	1	1,2	520,93	634,34	17,88%	0,00
E022-06m	1,1	1	734,51	737,61	0,42%	0,00
E023-05S	1,3	1,5	676,55	824,47	17,94%	0,00
E026-08m	1,1	1	701,08	702,31	0,18%	0,00
E030-04S	1	1	577,45	577,45	0,00%	0,00
E031-09h	1,1	1	687,552	747,81	8,06%	0,02
E033-05s	1	1,4	908,405	967,16	6,07%	0,02
E036-11h	1,2	1,4	821,216	822,29	0,13%	0,02
E041-14h	1	1	1027,74	1027,74	0,00%	0,02
E045-04f	1,1	1	1190,49	1197,81	0,61%	0,02
E051-05e	1,1	1	596,384	673,90	11,50%	0,02
E072-04f	1,3	1	357,056	421,21	15,23%	0,05
E076C09r	1,2	1	1774,92	1930,17	8,04%	0,03
E101-14s	1	1	1447,99	1447,99	0,00%	0,02
E121-07c	1,2	1	1381,77	1543,98	10,51%	0,05
E151-12c	1,1	1,1	1362,34	1583,00	13,94%	0,06
E200-17c	1,1	1	1688,55	1790,72	5,71%	0,11

Tabla 7. Resultados obtenidos de la heurística de Inserción Secuencial de Mole & Jameson determinista con el nodo inicial más cercano en función de la modificación de los parámetros λ y μ.

Para el 82,35% de los casos, se comprueba que existe una mejora al modificar los parámetros λ y/o μ, siendo esta mejora media potencial de un 6,84% (gráfico 2).

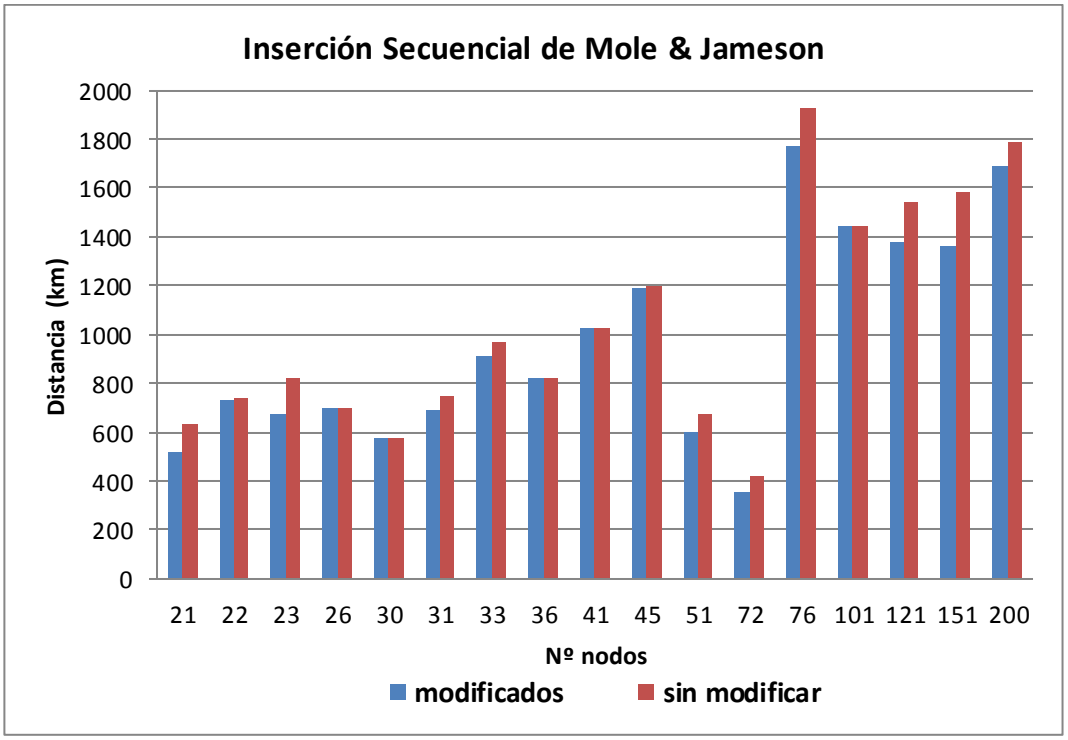

Gráfico 2. Distancia recorrida de la heurística de Inserción Secuencial de Mole & Jameson con el nodo inicial más cercano al depósito en función de la modificación de los parámetros λ y μ.

Si se recopila el número de veces que se repite cada uno de los diferentes valores de los parámetros con objeto de mejorar la solución, se puede observar en la tabla 8 que la mejora suele producirse cuando λ toma valores ligeramente superiores a 1 (1,1 y 1,2).

λ	μ	Nº veces repetidas
1	1	3
1	1,2	1
1	1,4	1
1,1	1	6
1,1	1,1	1
1,2	1	2
1,2	1,4	1
1,3	1	1
1,3	1,5	1

Tabla 8. Frecuencia de las diferentes posibles combinaciones de los parámetros λ y μ que reproducen la mejor solución para la heurística de Inserción Secuencial de Mole & Jameson con el nodo inicial más cercano al depósito.

Análisis del nodo inicial a insertar

Para el siguiente análisis, se van a comparar los resultados obtenidos de las tres heurísticas de inserción estableciéndose un valor fijo para cada nodo inicial a insertar en función de que este sea el nodo más alejado al depósito o el más cercano y para unos valores concretos de λ=1 y μ=1 (tabla 9, tabla 10 y tabla 11).

- Inserción Secuencial de Mole & Jameson

Modelo	Nodo inicial lejano	Nodo inicial cercano	Variación
	Solución (Km)	Solución (Km)	
E021-06m	469,12	634,34	26,05%
E022-06m	542,57	737,61	26,44%
E023-05S	746,66	824,47	9,44%
E026-08m	725,08	702,31	3,24%
E030-04S	638,04	577,45	10,49%
E031-09h	613,44	747,81	17,97%
E033-05s	912,80	967,16	5,62%
E036-11h	771,18	822,29	6,22%
E041-14h	954,55	1027,74	7,12%
E045-04f	972,68	1197,81	18,80%
E051-05e	592,18	673,90	12,41%
E072-04f	339,80	421,21	12,13%
E076C09r	1567,85	1930,17	19,33%
E101-14s	1327,87	1447,99	18,77%
E121-07c	1117,07	1543,98	8,30%
E151-12c	1272,54	1583,00	27,65%
E200-17c	1612,17	1790,72	19,61%

Tabla 9. Resultados obtenidos de la heurística de Inserción Secuencial de Mole & Jameson simple en función del criterio a seleccionar el nodo inicial de cada ruta.

Tras los resultados obtenidos, se puede afirmar que para el 88,24% de los casos, existe una importante capacidad de mejora (12,92% de media) al seleccionar el criterio de comenzar cada ruta con el cliente que se encuentre a una distancia mayor del depósito (Gráfico 3), siendo este criterio más importante que la variación del valor de los parámetros λ y μ.

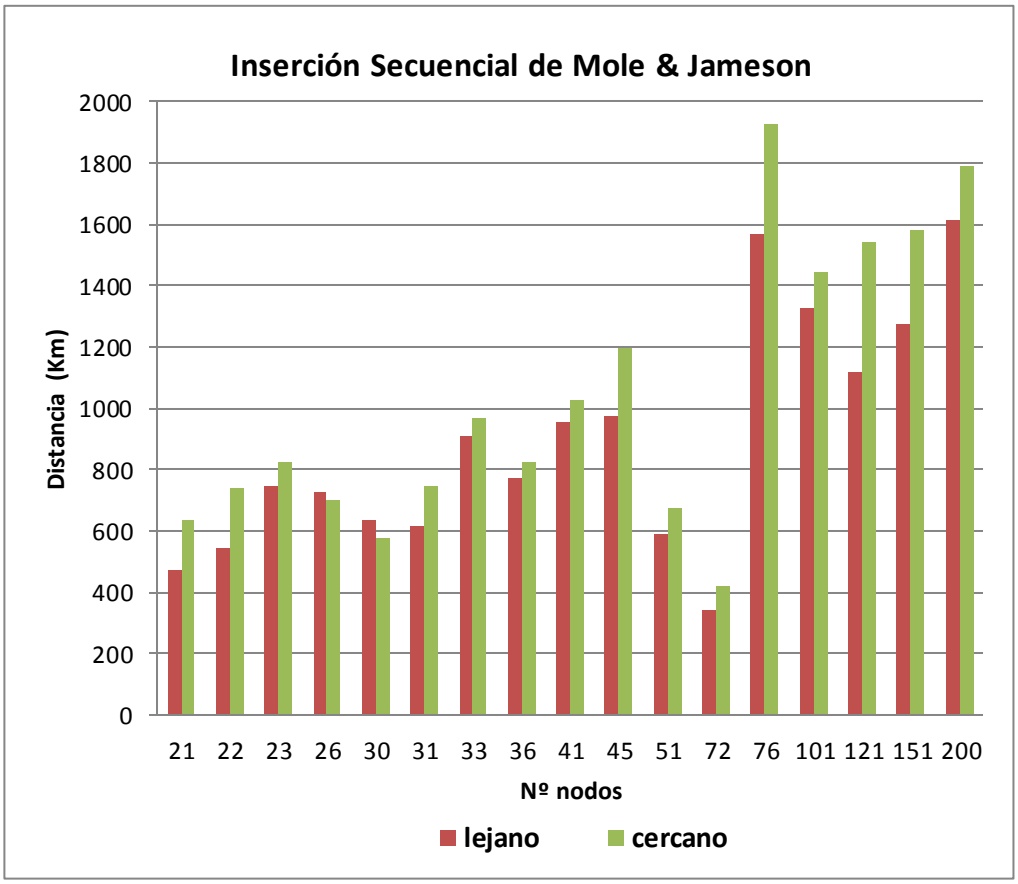

Gráfico 3. Distancia recorrida de la heurística de Inserción Secuencial de Mole & Jameson en función de la elección del nodo inicial según sea el más lejano o cercano para λ=1 y μ=1.

- Inserción en Paralelo

Modelo	Nodo inicial lejano	Nodo inicial cercano	Tiempo ejecución(s)	Variación
	Solución (Km)	Solución (Km)		
E021-06m	488,51	525,61	0	7,60%
E022-06m	514,34	608,46	0	18,30%
E023-05S	856,32	725,07	0	15,33%
E026-08m	765,74	745,36	0,01	2,66%
E030-04S	752,14	785,23	0,01	4,40%
E031-09h	746,62	679,09	0,01	9,04%
E033-05s	1077,24	952,37	0,01	11,59%
E036-11h	913,13	884,35	0,01	3,15%
E041-14h	1079,05	1080,21	0,01	0,11%
E045-04f	1268,80	1253,26	0,02	1,22%
E051-05e	703,31	653,34	0,02	7,11%
E072-04f	375,69	448,73	0,04	19,44%
E076C09r	2296,96	1975,77	0,03	13,98%
E101-14s	1573,74	1477,68	0,02	6,10%
E121-07c	1576,73	1464,39	0,06	7,12%
E151-12c	1559,64	1496,33	0,05	4,06%
E200-17c	1923,03	1869,82	0,1	2,77%

Tabla 10. Resultados obtenidos de la heurística de Inserción en Paralelo simple en función del criterio a seleccionar el nodo inicial de cada ruta.

Para el 70,59% de los casos, la mejor solución se obtiene seleccionando el nodo más cercano a la base como nodo inicial, lo que supone una mejora potencial media de 2,02%. (Gráfico 4)

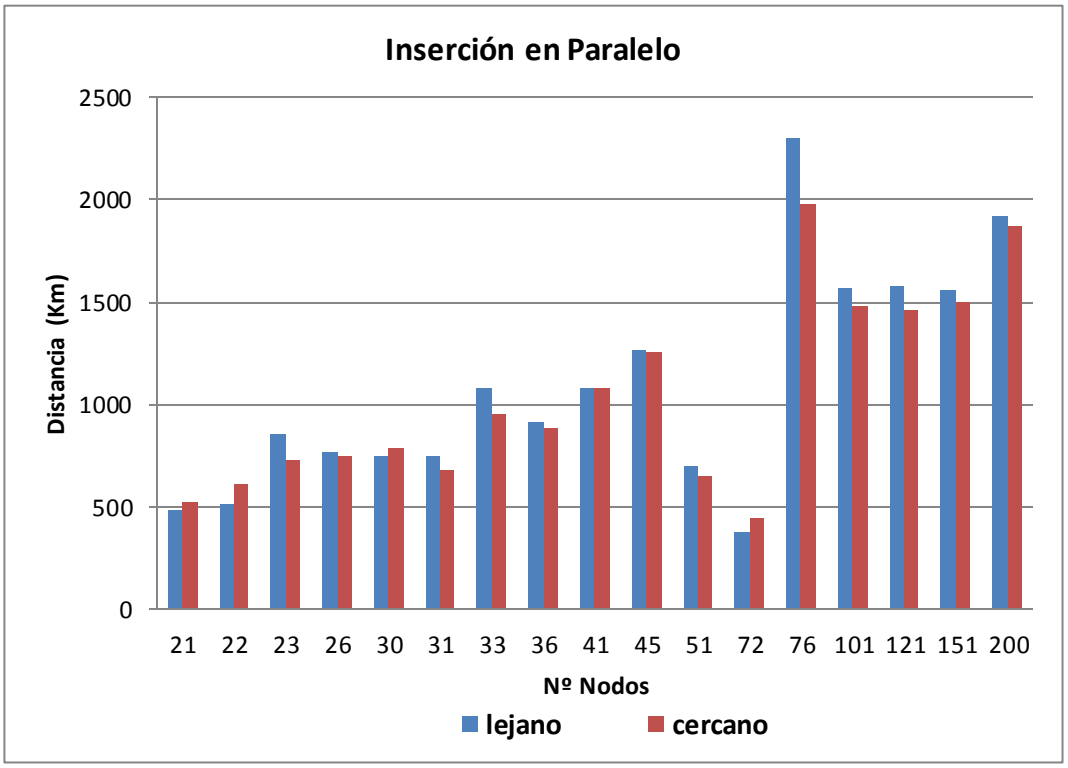

Gráfico 4. Distancia recorrida de la heurística de Inserción en Paralelo en función de la elección del nodo inicial según sea el más lejano o cercano para λ=1 y μ=1.

- Inserción en Paralelo de Christofides, Mingozzi y Toth

Modelo	Nodo inicial lejano	Nodo inicial cercano	Tiempo ejecución(s)	Variación
	Solución (Km)	Solución (Km)		
E021-06m	524,71	510,53	0	2,70%
E022-06m	511,24	549,67	0	7,52%
E023-05S	665,97	765,46	0,016	14,94%
E026-08m	708,24	751,02	0,015	6,04%
E030-04S	605,26	674,91	0,016	11,51%
E031-09h	708,62	667,50	0,016	5,80%
E033-05s	971,13	1037,05	0,015	6,79%
E036-11h	808,83	875,79	0,016	8,28%
E041-14h	987,62	1017,16	0,016	2,99%
E045-04f	935,73	836,12	0	10,64%
E051-05e	573,58	643,16	0,016	12,13%
E072-04f	268,14	314,59	0,016	17,32%
E076C09r	1740,05	1534,71	0,062	11,80%
E101-14s	1341,38	1275,19	0,124	4,93%
E121-07c	1439,09	1395,10	0,062	3,06%
E151-12c	1260,89	1337,30	0,281	6,06%
E200-17c	1647,74	1858,92	0,783	12,82%

Tabla 11. Resultados obtenidos de la heurística de Inserción en Paralelo de Christofides, Mingozzi y Toth simple en función del criterio a seleccionar el nodo inicial de cada ruta.

En este último método, los mejores resultados se obtienen seleccionando el nodo inicial más lejano a la base en el 64,71% de los modelos, así como también la mejora que se produce es mayor (8,57%). (Gráfico 5).

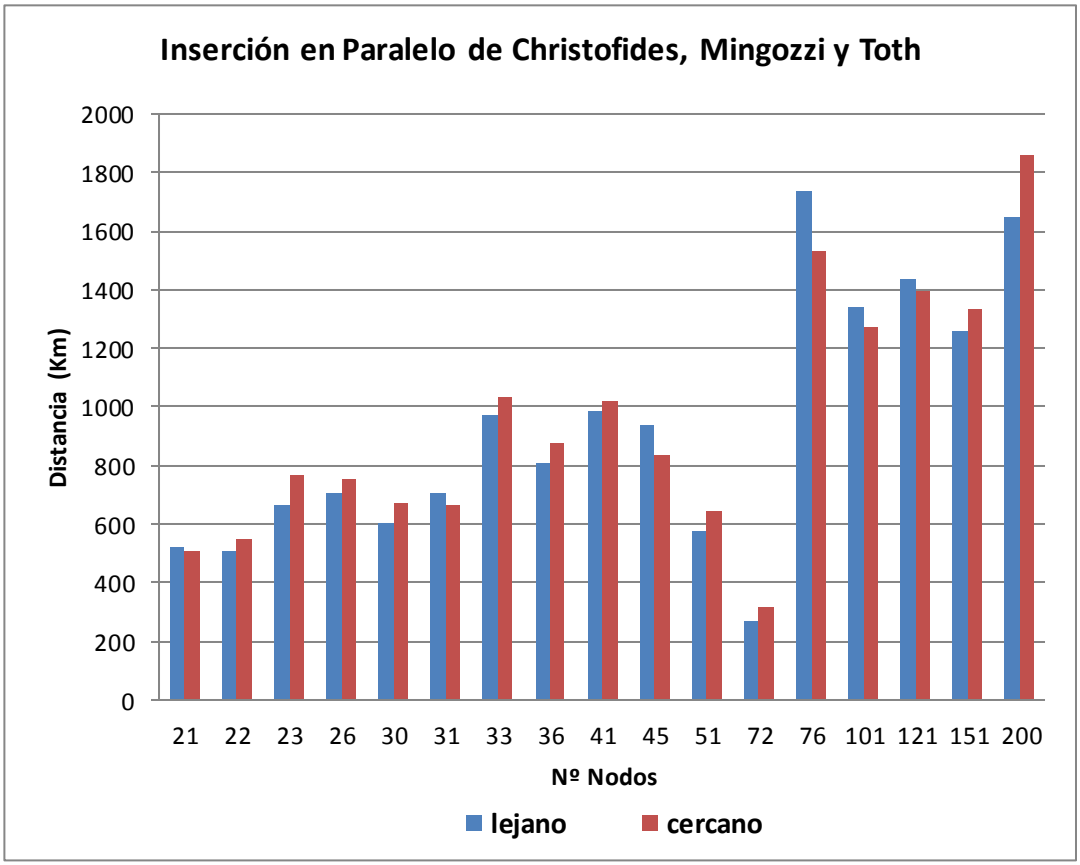

Gráfico 5. Distancia recorrida de la heurística de Inserción en Paralelo de Christofides, Mingozzi y Toth en función de la elección del nodo inicial según sea el más lejano o cercano para λ=1 y μ=1.

Resultados generales

Por último, se comparan en la Tabla 12 los tres métodos de inserción analizados, junto con el algoritmo de Clarke & Wright (C&W)[1], cada uno de ellos con los valores dados por el mejor criterio del nodo inicial en cada caso.

Modelo	Clarke & Wright		Secuencial		Paralelo		Christofides	
	Solución (Km)	Tiempo ejecución(s)	Solución (Km)	Tiempo ejecución(s)	Solución (Km)	Tiempo ejecución(s)	Solución (Km)	Tiempo ejecución(s)
E021-06m	444,42	0,032	469,12	0,000	488,51	0,00	510,53	0,000
E022-06m	508,22	0,063	542,57	0,000	514,34	0,00	511,24	0,000
E023-05S	576,42	0,094	746,66	0,000	725,07	0,00	665,97	0,016
E026-08m	626,78	0,094	702,31	0,000	745,36	0,01	708,24	0,015
E030-04S	534,45	0,141	577,45	0,000	752,14	0,01	605,26	0,016
E031-09h	625,22	0,187	613,44	0,015	679,09	0,01	667,50	0,016
E033-05s	843,10	0,218	912,80	0,015	952,37	0,01	971,13	0,015
E036-11h	731,47	0,358	771,18	0,015	884,35	0,01	808,83	0,016
E041-14h	899,50	0,577	954,55	0,015	1079,05	0,01	987,62	0,016
E045-04f	747,75	0,624	972,68	0,016	1253,26	0,02	836,12	0,000
E051-05e	584,64	1,235	592,18	0,015	653,34	0,02	573,58	0,016
E072-04f	258,23	4,789	339,80	0,031	375,69	0,04	268,14	0,016
E076C09r	1344,15	6,433	1567,85	0,015	1975,77	0,03	1534,71	0,062
E101-14s	1143,45	19,337	1327,87	0,019	1477,68	0,02	1275,19	0,124
E121-07c	1071,95	42,446	1117,07	0,050	1464,39	0,06	1395,10	0,062
E151-12c	1096,26	96,415	1272,54	0,063	1496,33	0,05	1260,89	0,281
E200-17c	1425,89	299,078	1612,17	0,100	1869,82	0,10	1647,74	0,780

Tabla 12. Resultados obtenidos de las cuatro heurísticas de Construcción simples con sus mejores valores.

[1] Resultados procedentes del Trabajo Fin de Grado *Problemas de rutas de vehículos: modelos, aplicaciones logísticas y métodos de resolución* de Ana Benito Quintanilla (2015).

El algoritmo de Clarke & Wright ofrece los mejores resultados en el 88,24% de los modelos con una mejora potencial media de 6,93% (Gráfico 6), pero a costa de emplear un tiempo computacional mucho mayor (Gráfico 7). Por consiguiente, se tratará de mejorar en los siguientes capítulos los resultados iniciales de las heurísticas de inserción.

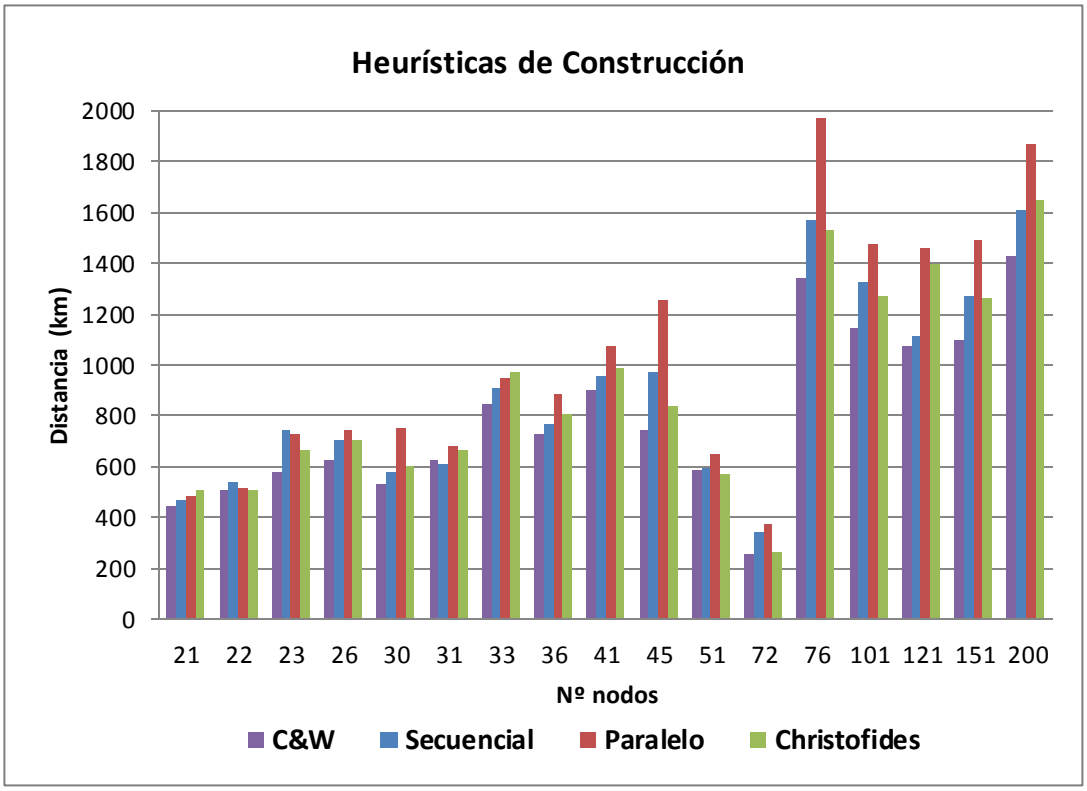

Gráfico 6. Distancia recorrida en función de la heurística de Construcción para el nodo inicial fijo.

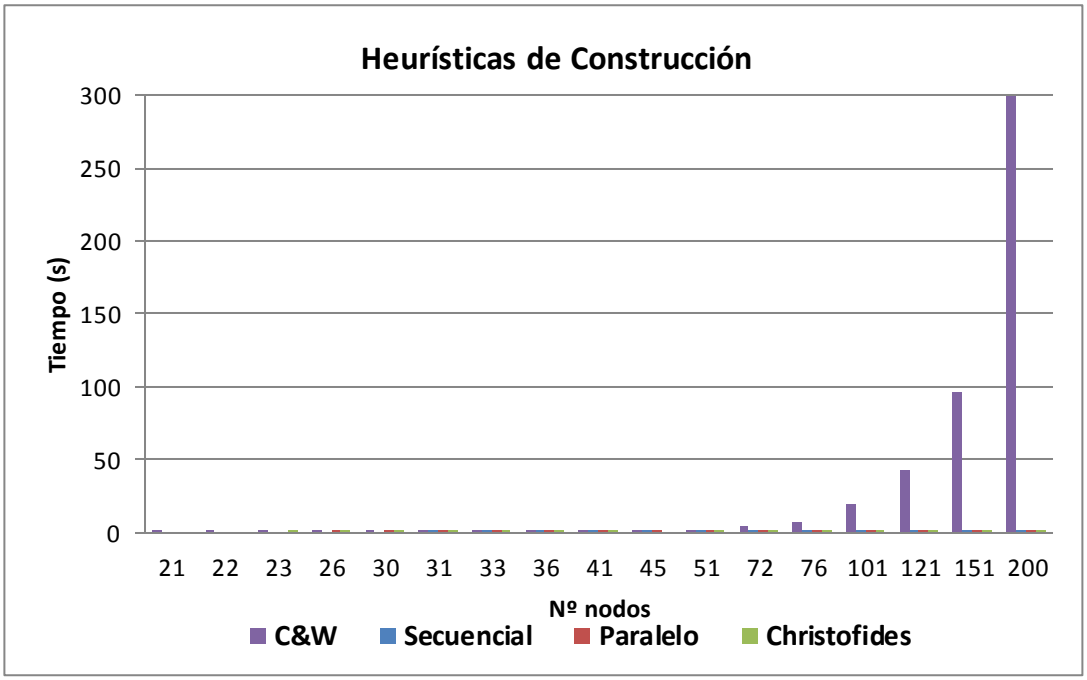

Gráfico 7. Tiempo de ejecución en función de la heurística de Construcción para nodo inicial fijo.

Descartando el algoritmo de Clarke & Wright de nuestro análisis, se pone de manifiesto que para el 52,94% de los modelos, las mejores soluciones se obtienen con el método Secuencial de Mole & Jameson y las segundas mejores se obtienen en el 47,06% con el método de Christofides, siendo el método en Paralelo el que ofrece peores resultados en comparación con los otros dos (Tabla 13).

Versión	Mejor solución
Secuencial	52,94%
Paralelo	0,00%
Christofides	47,06%

Tabla 13. Frecuencia relativa de las mejores soluciones para los diferentes métodos de Inserción.

En el Gráfico 8 se muestra una ampliación del tiempo de ejecución para las tres heurísticas de Inserción:

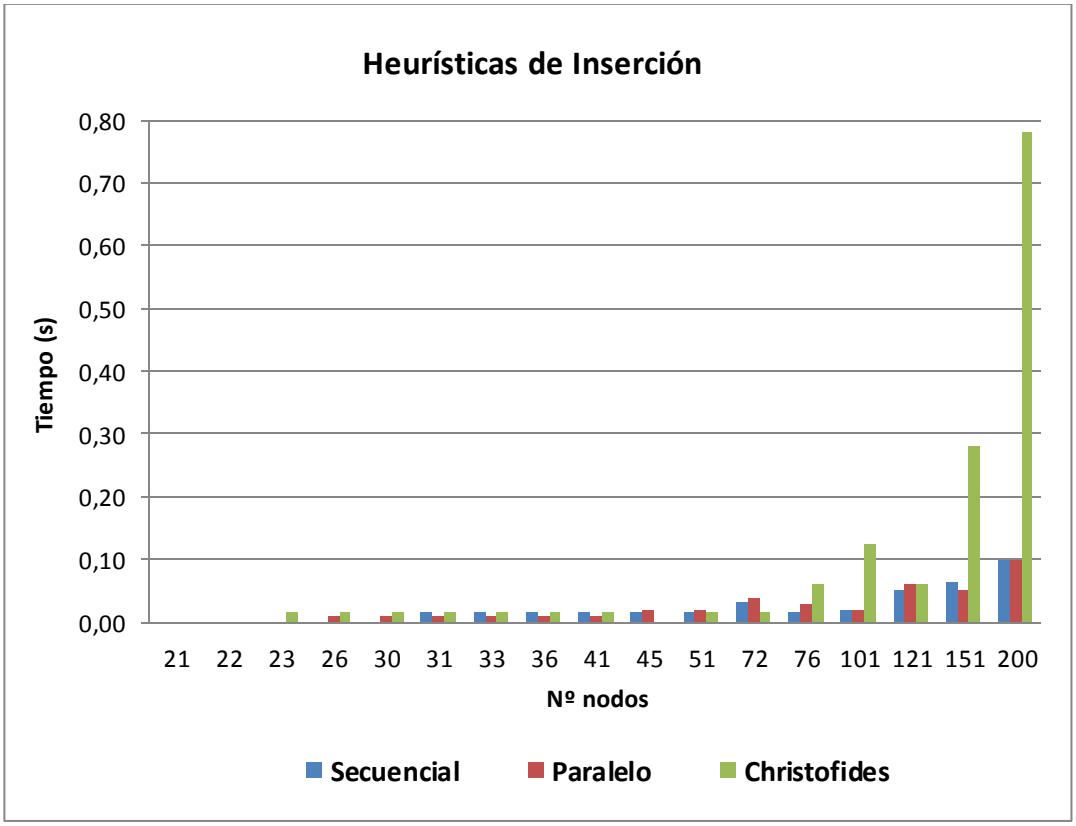

Gráfico 8. Tiempo de ejecución en función de la heurística de Inserción para nodo inicial fijo.

Como se puede evidenciar, el tiempo de cálculo es ínfimo, no llegando tan siquiera al segundo para el método que supone más tiempo cuando el número de nodos es suficientemente alto.

Por todo lo evaluado anteriormente, se concluye con que la heurística de Inserción Secuencial de Mole & Jameson es la que ofrece los mejores resultados a la vez que dedica de los menores tiempos computacionales para su objetivo.

A partir de aquí nos centraremos en la heurística de Inserción Secuencial, la cual ha sido la que ha ofrecido una mejor información de salida, con intención de seguir obteniendo resultados mejores.

- *Métodos de construcción greedy aleatorizados*

Con motivo de mejorar la heurística de Inserción Secuencial, se insta a emplear un método greedy aleatorizado. Fueron varias las variantes realizadas para introducir esta aleatoriedad en los algoritmos, pero finalmente, tras observar que algunas no ofrecían mejora alguna, se decidió estipular que los nodos iniciales de cada ruta se generaran de forma aleatoria para una mayor diversificación, y repitiendo este proceso durante N veces para quedarse finalmente con la mejor solución obtenida.

Ejecutando los modelos para observar la mejora en la solución final que supone el aumento del número de iteraciones N = {1, 100, 500}, se tienen los resultados que aparecen en la Tabla 14, así como sus gráficos correspondientes (Gráfico 9 y Gráfico 10).

Modelo	N=1		N=100		N=500	
	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)
E021-06m	456,90	0,000	444,94	0,05	438,74	0,34
E022-06m	656,41	0,000	499,02	0,05	499,02	0,19
E023-05S	659,59	0,000	631,14	0,11	631,14	0,44
E026-08m	720,57	0,000	640,98	0,09	641,09	0,47
E030-04S	650,79	0,000	568,73	0,27	568,73	0,73
E031-09h	718,68	0,015	638,64	0,13	613,43	0,58
E033-05s	917,18	0,015	887,62	0,17	869,49	0,69
E036-11h	849,02	0,015	738,67	0,11	730,00	0,47
E041-14h	973,66	0,015	905,82	0,11	905,00	0,56
E045-04f	1077,78	0,016	953,67	0,31	913,28	1,42
E051-05e	653,20	0,015	565,17	0,39	558,00	1,83
E072-04f	365,37	0,031	301,16	1,89	297,56	5,90
E076C09r	1487,87	0,015	1455,23	1,41	1435,63	7,18
E101-14s	1300,27	0,019	1301,87	2,05	1241,72	9,90
E121-07c	1365,67	0,050	1143,90	5,22	1131,71	26,28
E151-12c	1457,64	0,063	1263,29	6,10	1233,59	30,02
E200-17c	1743,47	0,100	1611,04	10,82	1568,33	53,73

Tabla 14. Resultados obtenidos de la heurística de Inserción Secuencial con el nodo inicial aleatorizado en función del número de iteraciones N={1, 100, 500}.

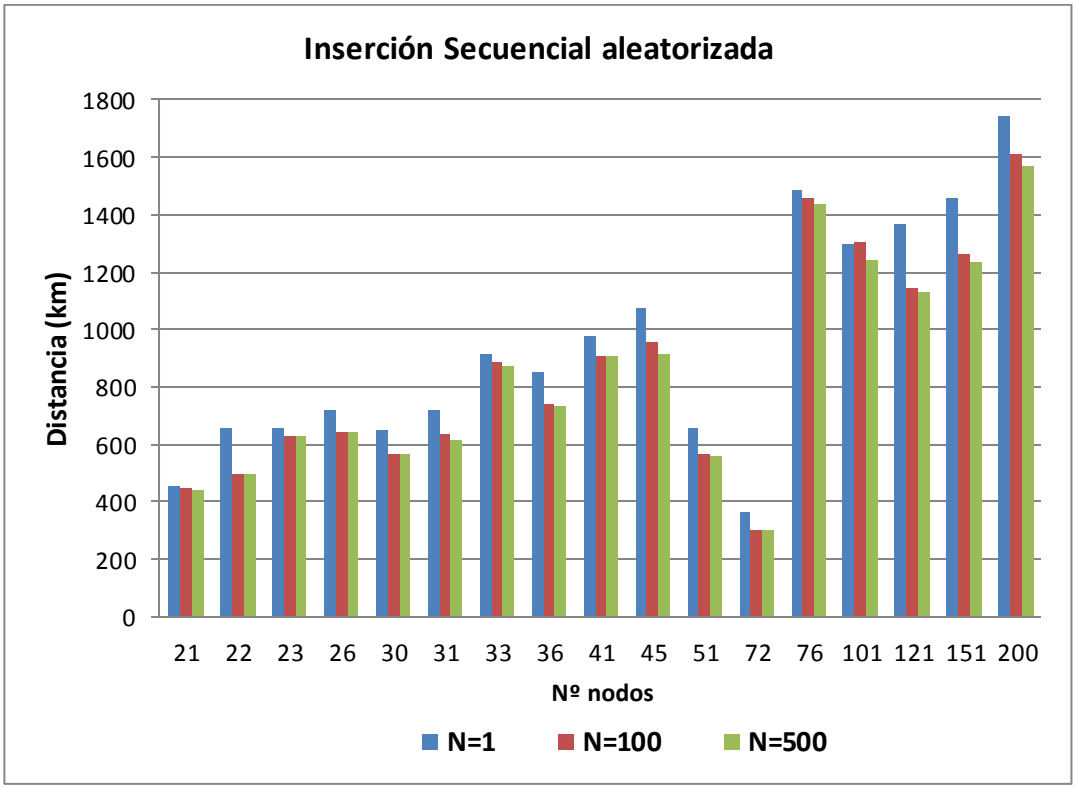

Gráfico 9. Distancia recorrida de la heurística de Inserción Secuencial en función del número de iteraciones N.

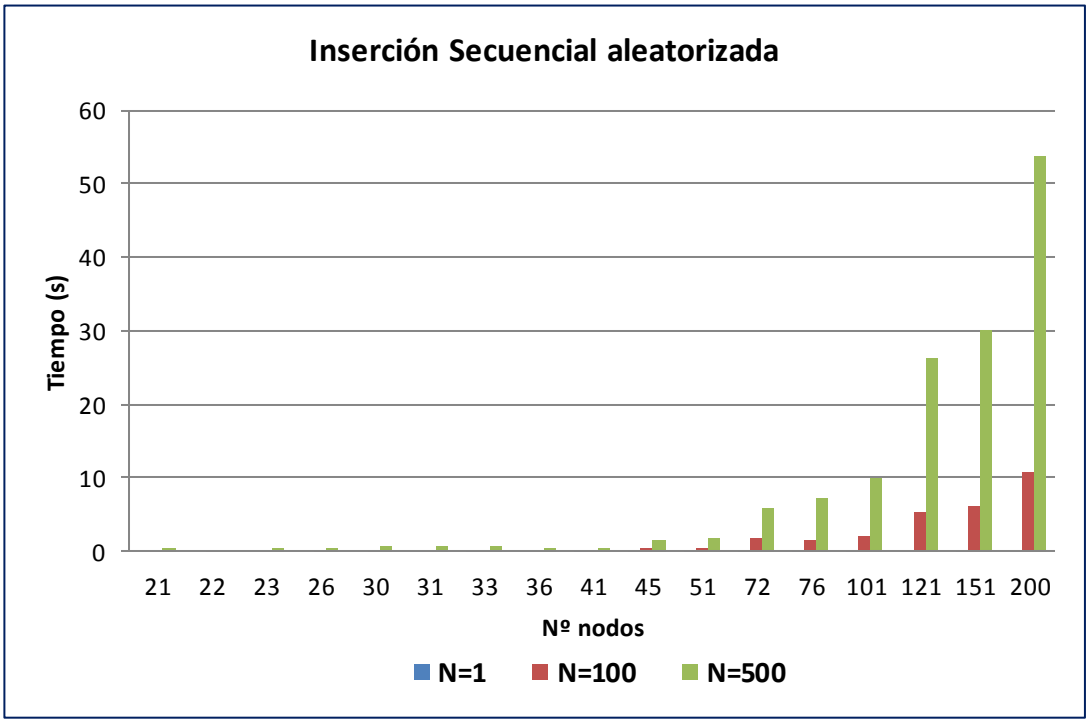

Gráfico 10. Tiempo de ejecución de la heurística de Inserción Secuencial en función del número de iteraciones N.

La mejora media que se produce con el método aleatorizado al aumentar el número de iteraciones es de un 10,04% para N=100 y de 11,52% para N=500. La mejora dada al aumentar de N=100 a N=500 se produce a costa de quintuplicar el tiempo computacional para obtener los resultados.

Lista Restringida de Candidatos (RCL)

Otra forma de introducir aleatoriedad a este algoritmo es mediante la utilización de una lista restringida de candidatos (RCL: Restricted Candidate List), en la cual se incluyen los mejores aspirantes a ser partícipes de la solución. Es de esta lista desde donde se selecciona de forma aleatoria el nodo a insertar en vez de considerar todo el conjunto.

Siguiendo las conclusiones obtenidas anteriormente de que la elección del nodo más alejado al depósito ofrecía mejores resultados, se procede ahora a evaluar la calidad de las soluciones en función de un tamaño fijo de la RCL (restricción por cardinalidad) para los K={4,8,16} nodos más alejados del depósito y con un número de iteraciones N=100 (Tabla 15).

Modelo	K=4	K=8	K=16
	Solución (Km)	Solución (Km)	Solución (Km)
E021-06m	430,89	438,74	433,54
E022-06m	519,01	499,02	500,43
E023-05S	633,74	633,74	633,74
E026-08m	654,60	626,90	667,53
E030-04S	626,31	590,92	576,48
E031-09h	612,29	611,99	628,63
E033-05s	902,09	889,08	887,62
E036-11h	713,26	716,41	731,95
E041-14h	895,41	877,99	914,73
E045-04f	959,36	955,84	947,13
E051-05e	562,63	562,63	565,02
E072-04f	338,50	339,80	334,75
E076C09r	1421,76	1407,96	1446,71
E101-14s	1204,86	1208,33	1222,46
E121-07c	1105,14	1105,16	1094,33
E151-12c	1189,06	1188,92	1192,79
E200-17c	1491,78	1496,45	1528,41

Tabla 15. Resultados obtenidos de la heurística de Inserción Secuencial aleatorizada con RCL en función del parámetro K={4,8,16} para N=100.

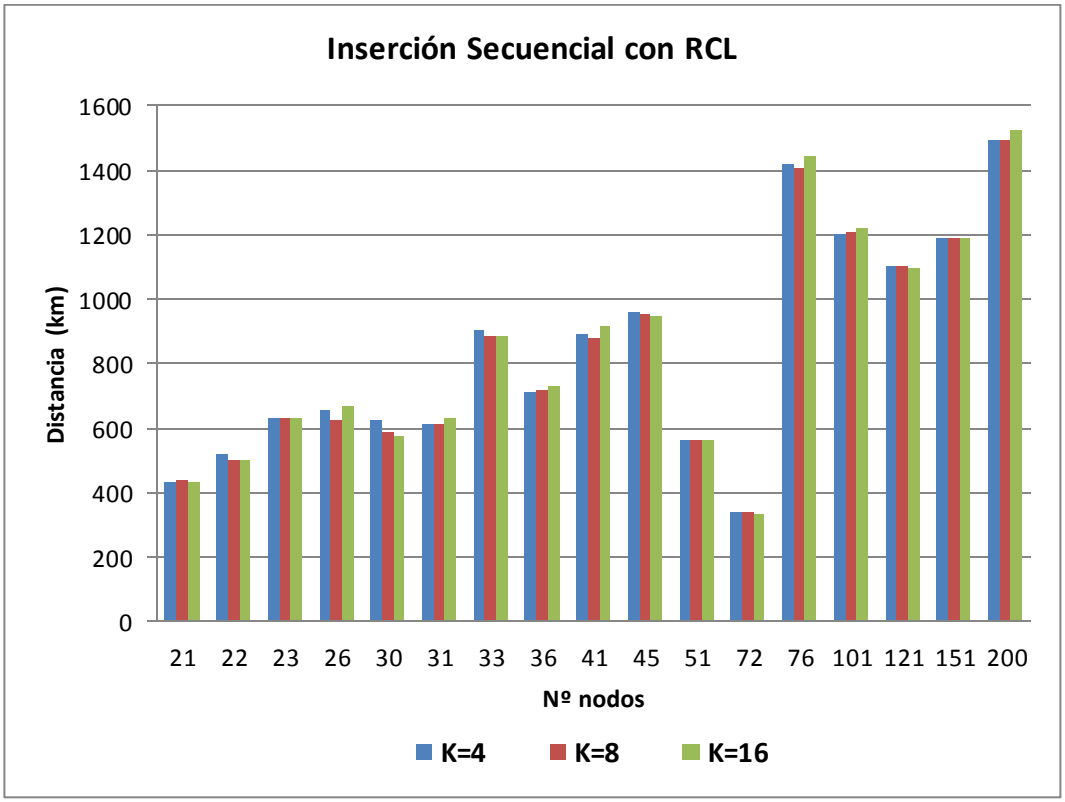

Gráfico 11. Distancia recorrida de la heurística de Inserción Secuencial aleatorizada con RCL en función del parámetro K={4, 8, 16} para N=100.

Los mejores resultados se obtienen para un K=8 con el 47,06% de las veces (Tabla 16).

RCL N=100	
K=4	35,29%
K=8	47,06%
K=16	35,29%

Tabla 16. Frecuencia relativa de los mejores resultados de la heurística de Inserción Secuencial aleatorizada con RCL en función del parámetro K.

En segundo lugar, se realiza una comparación para un valor K=8 en función del número N= {100, 500} de iteraciones (Tabla 17).

Modelo	N=100		N=500	
	Solución (Km)	Tiempo ejecución(s)	Solución (Km)	Tiempo ejecución(s)
E021-06m	438,74	0,093	430,89	0,374
E022-06m	499,02	0,078	499,02	0,390
E023-05S	633,74	0,156	633,74	0,764
E026-08m	626,90	0,156	630,95	0,562
E030-04S	590,92	0,250	580,52	1,217
E031-09h	611,99	0,156	611,99	0,811
E033-05s	889,08	0,249	889,08	1,217
E036-11h	716,41	0,234	706,89	1,029
E041-14h	877,99	0,250	878,43	1,300
E045-04f	955,84	0,500	959,36	2,481
E051-05e	562,63	0,570	562,63	3,121
E072-04f	339,80	2,028	338,50	10,078
E076C09r	1407,96	1,498	1393,42	7,500
E101-14s	1208,33	1,950	1194,45	9,838
E121-07c	1105,16	4,919	1102,23	24,520
E151-12c	1188,92	6,252	1187,09	30,799
E200-17c	1496,45	10,621	1482,43	56,931

Tabla 17. Resultados obtenidos de la heurística de Inserción Secuencial aleatorizada con RCL en función del número de iteraciones N={100,500} para K=8.

La mejora media que se produce con el método aleatorizado y restringiendo la lista de candidatos a 8 nodos al aumentar el número de iteraciones de 100 a 500 es de un 0,45%. La diferencia de la mejora obtenida resulta poco significativa para el aumento del tiempo computacional que supone (Gráfico 12 y Gráfico 13).

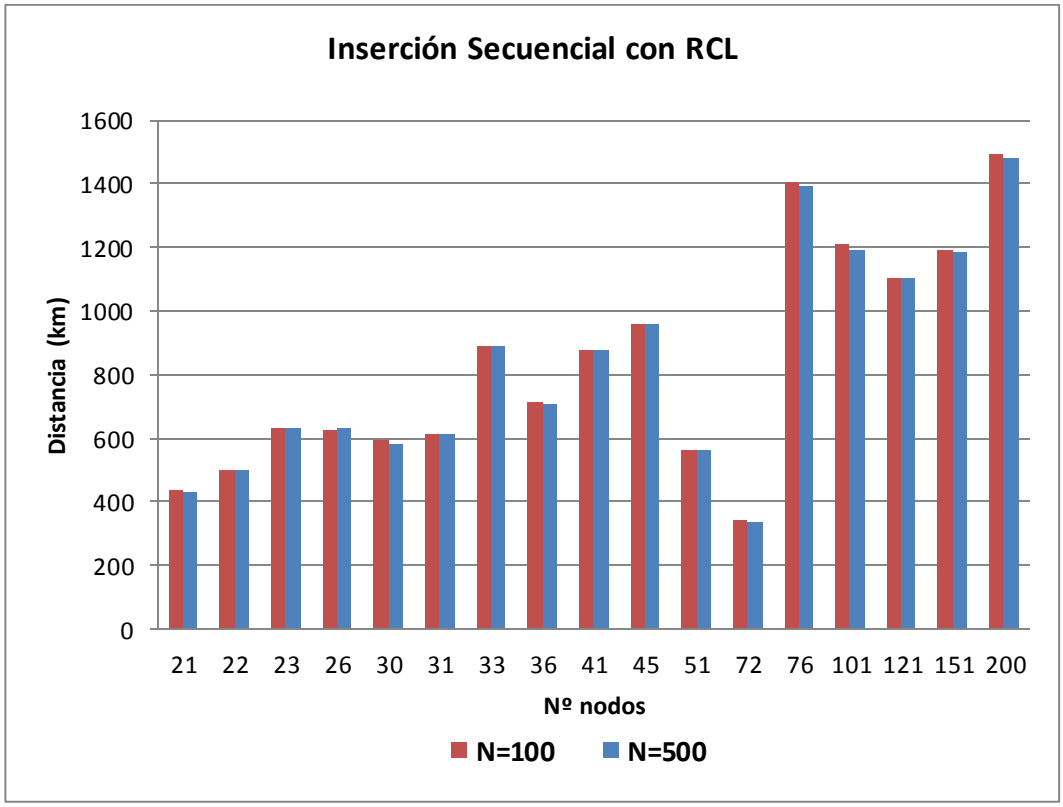

Gráfico 12. Distancia recorrida de la heurística de Inserción Secuencial aleatorizada con RCL en función del número de iteraciones N={100, 500} para K=8.

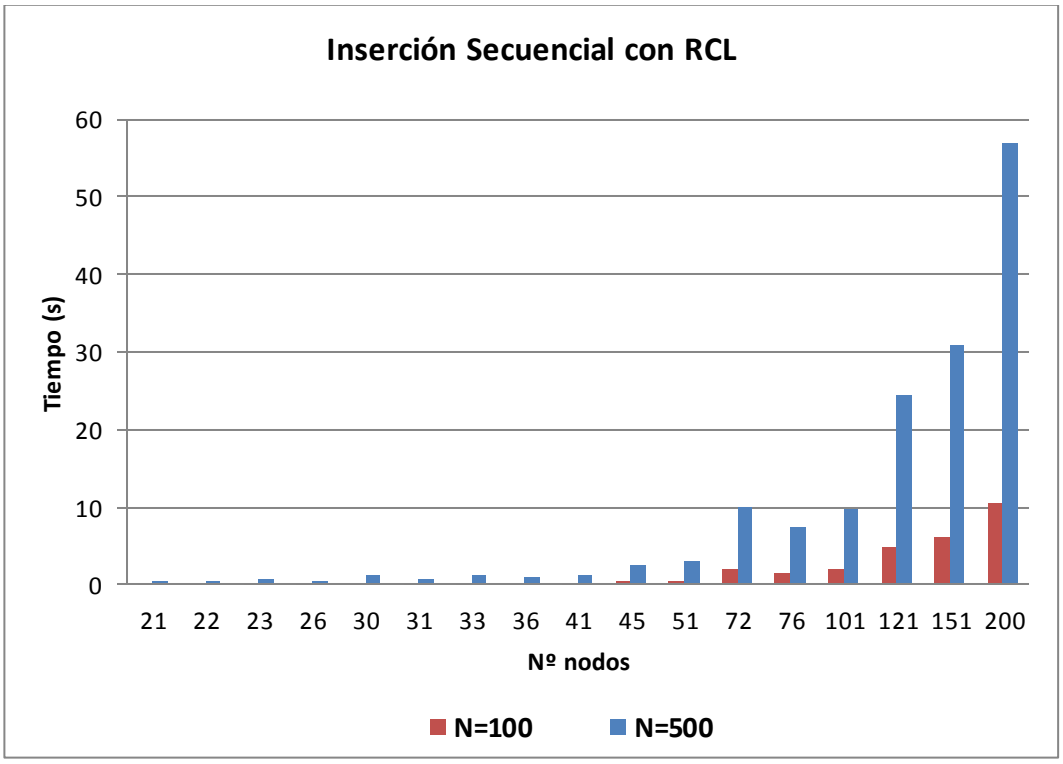

Gráfico 13. Tiempo de ejecución de la heurística de Inserción Secuencial aleatorizada con RCL en función del número de iteraciones N={100,500} para K=8.

Resultados finales

Tras analizar el efecto que supone la modificación de los parámetros que definen los diferentes procedimientos, se muestran en la Tabla 18 y Gráficos 14 y 15 un resumen con las versiones y los mejores cambios seleccionados anteriormente. Dichas versiones son las siguientes:

- Heurística de Inserción Secuencial de Mole & Jameson con nodo inicial determinista (el más distante o el más próximo al depósito).

- Heurística de Inserción Secuencial de Mole & Jameson con nodo inicial aleatorizado para N=500 iteraciones.

- Heurística de Inserción Secuencial de Mole & Jameson con nodo inicial aleatorizado con una lista restringida de los 8 candidatos más alejados del depósito y N=100 iteraciones.

Modelo	Determinista		N=500		RCL K=8, N=100	
	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)
E021-06m	469,12	0,00	438,74	0,34	438,74	0,093
E022-06m	542,57	0,00	499,02	0,19	499,02	0,078
E023-05S	746,66	0,00	631,14	0,44	633,74	0,156
E026-08m	702,31	0,00	641,09	0,47	626,90	0,156
E030-04S	577,45	0,00	568,73	0,73	590,92	0,250
E031-09h	613,44	0,02	613,43	0,58	611,99	0,156
E033-05s	912,80	0,02	869,49	0,69	889,08	0,249
E036-11h	771,18	0,02	730,00	0,47	716,41	0,234
E041-14h	954,55	0,02	905,00	0,56	877,99	0,250
E045-04f	972,68	0,02	913,28	1,42	955,84	0,500
E051-05e	592,18	0,02	558,00	1,83	562,63	0,570
E072-04f	339,80	0,03	297,56	5,90	339,80	2,028
E076C09r	1567,85	0,02	1435,63	7,18	1407,96	1,498
E101-14s	1327,87	0,02	1241,72	9,90	1208,33	1,950
E121-07c	1117,07	0,05	1131,71	26,28	1105,16	4,919
E151-12c	1272,54	0,06	1233,59	30,02	1188,92	6,252
E200-17c	1612,17	0,10	1568,33	53,73	1496,45	10,621

Tabla 18. Resultados obtenidos de las diferentes versiones de la heurística de Inserción Secuencial.

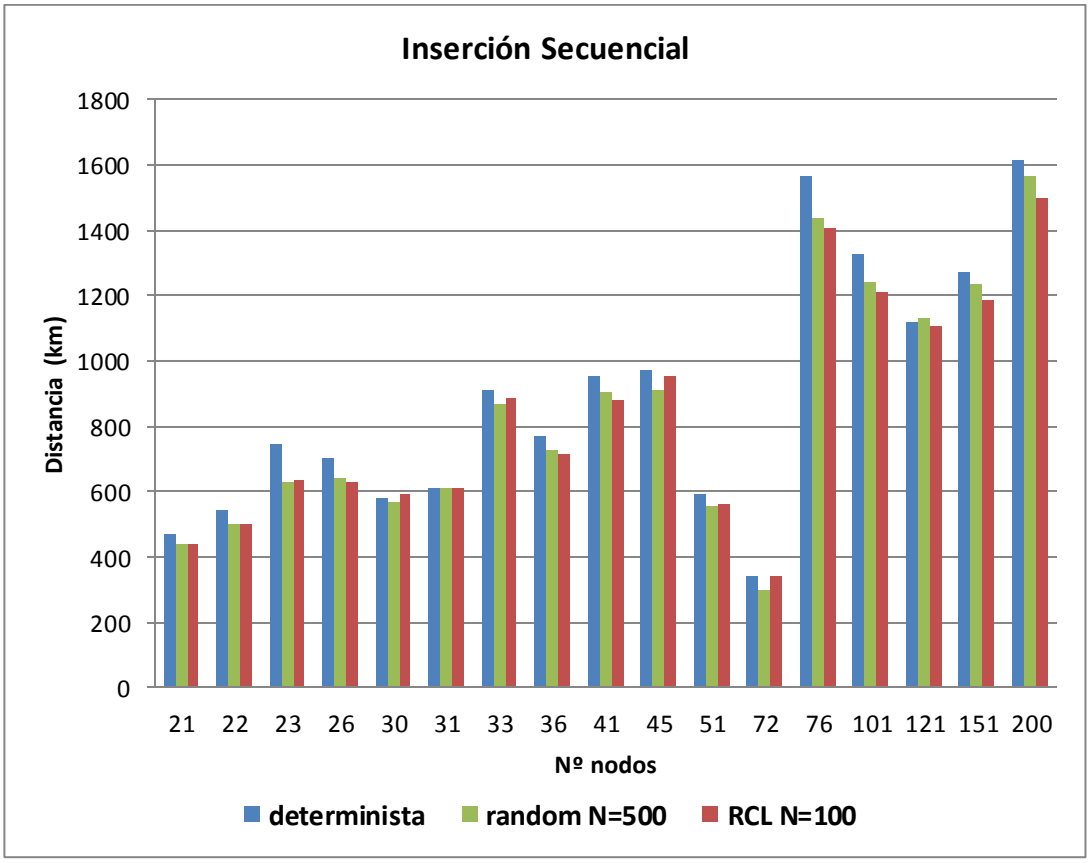

Gráfico 14. Distancia recorrida de las diferentes versiones de la heurística de Inserción Secuencial.

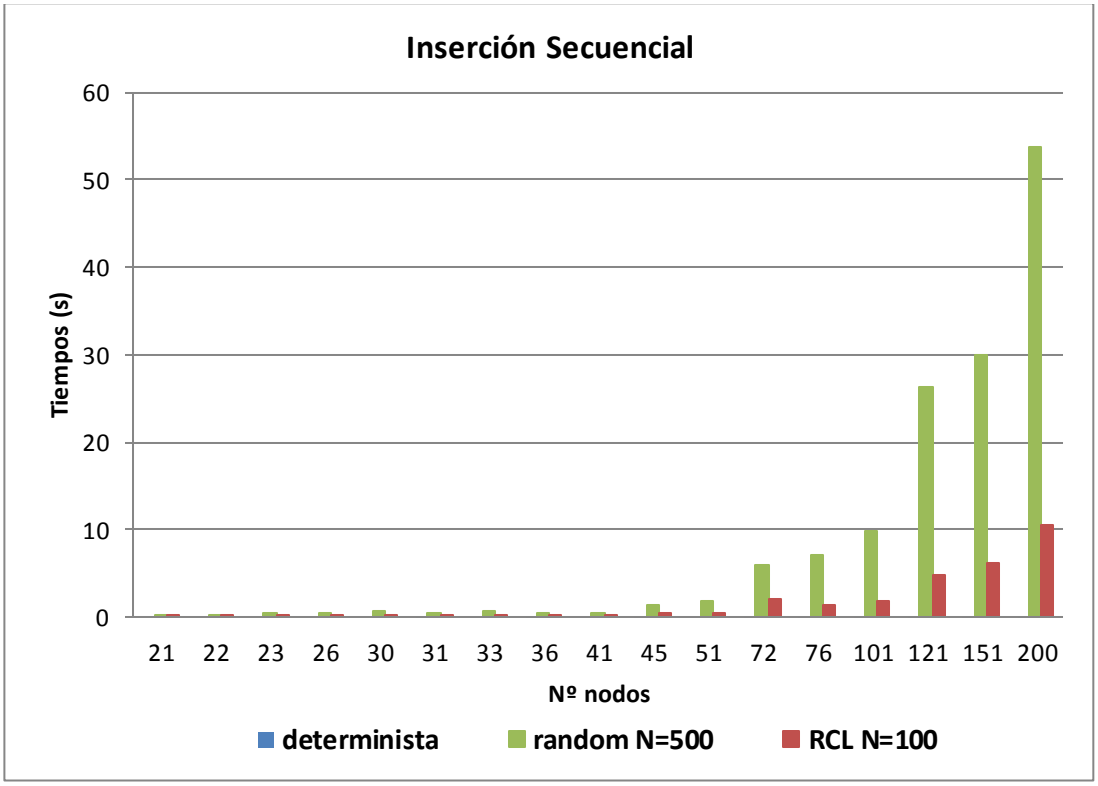

Gráfico 15. Tiempo de ejecución de las diferentes versiones de la heurística de Inserción Secuencial.

Finalmente, se puede concluir con que en el 64,71% de los modelos, el método de RCL ofrece los mejores resultados y con un tiempo muy pequeño, seguida del método aleatorizado simple. Además, con la elección del nodo inicial de forma determinista no se alcanza nunca la mejor solución y existe una mejora media del 6,05% al aplicar el método RCL y del 5,83% para el método completamente aleatorizado (Tabla 19).

Versión	Mejor solución
determinista	0,00%
Random N=500	47,06%
RCL K=8, N=100	64,71%

Tabla 19. Frecuencia relativa de las mejores soluciones para las diferentes versiones de la heurística de Inserción Secuencial.

CAPÍTULO 4. MÉTODOS DE MEJORA

4.1. Contextualización

Dentro de los métodos de mejora, los procedimientos de búsqueda local son los más utilizados. Dichos procedimientos son aquellos que tratan de mejorar de forma progresiva la solución inicial de la que se parte hasta que ya no exista ninguna solución con un mejor valor dentro de un determinado entorno.

La idea fundamental del método de búsqueda local se concentra en mejorar una solución factible dada. Esta mejora se lleva a cabo tras la generación de forma iterativa de una serie de movimientos que dan lugar a las diferentes soluciones del entorno.

Para el caso de problemas de optimización, se tiene una función a minimizar $f(x)$ sujeto a $x \in X$, donde $f(x)$ puede ser una función lineal o no lineal y X el conjunto de todas las soluciones factibles.

Conceptos:

- Entorno de x: es el conjunto de soluciones $N(x) \subseteq X$ asociadas a cada solución $x \in X$.

- Movimiento: es la operación por la cual de cada solución x puede obtenerse una solución de su entorno $x' \epsilon\, N(x)$.

Un método de búsqueda local parte de una solución inicial x_0, sobre la cual se realiza una exploración del entorno para después aplicar un movimiento y obtener la solución $x_1 \in N(x_0)$. Después de efectuar n movimientos, se obtiene una trayectoria en el espacio de soluciones:

$$x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n$$

Si la aplicación de estos movimientos se limita a aquellos que mejoren la solución actual, se obtiene un método de descenso. En este caso:

$$f(x_0) \geq f(x_1) \geq f(x_2) \geq \cdots \geq f(x_n)$$

Los pasos a realizar para desarrollar un método de búsqueda local según el método de descenso son los siguientes:

Paso 1. Seleccionar $x \in X$ para comenzar con el proceso.

Paso 2. Buscar una solución $x' \epsilon\ N(x)$ de tal forma que se mejore la solución actual $f(x') < f(x)$.

Paso 3. Si no se ha podido encontrar una solución $x' \epsilon\ N(x)$ tal que $f(x') < f(x)$, se termina el procedimiento, ya que x es un óptimo local.

En caso contrario, sustituir x por x' y volver al paso 2.

Existen dos distinciones a la hora de implementar el método de descenso. La primera es conocida por el nombre de mayor descenso, *steepest descent*, *best improvement (BI)* o *hill climbing* (en el caso de maximización), y se basa en seleccionar la mejor solución dentro de todas las soluciones posibles del entorno. La segunda denominada *first improvement (FI)* se aplica, particularmente, cuando existen muchos elementos dentro del entorno, y en este caso se selecciona la primera solución que mejore la actual.

Con el empleo de un método de descenso, con cualquiera de sus dos implementaciones, se alcanza siempre un mínimo local, una solución mejor o igual a todas las del entorno. Se debe destacar también la importancia de escoger una adecuada estructura del vecindario para una mayor eficacia del proceso. Pero la principal debilidad de este algoritmo radica en que no es capaz de escapar de este mínimo local, imposibilitando así la opción de alcanzar un posible mínimo absoluto.

En la Figura 9 podemos observar como x_n resulta ser un óptimo local, pues no existe una mejor solución en un entorno próximo a él, pero a una distancia mayor podemos evidenciar la existencia de un mínimo global x_g , el cual no se puede alcanzar mediante este método de descenso.

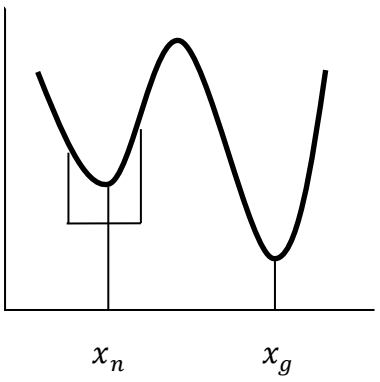

Figura 9. Visualización de óptimo local y global

Asimismo, en la Figura 10 se muestra un ejemplo de una función tal que para cada solución inicial se consideran dos soluciones vecinas, representadas por puntos a la izquierda y a la derecha de la misma. Con el método descendente la solución final que se logra es aquella situada en el valle del punto de partida en cuestión, de tal forma, que si la solución inicial es P, el proceso termina con el resultado final Q.

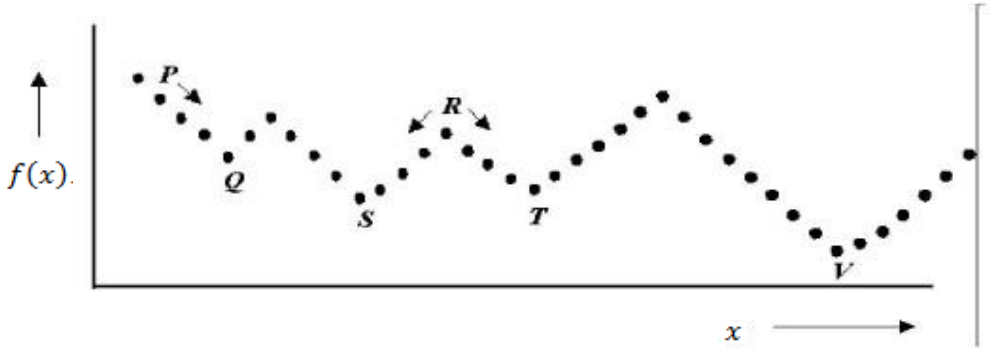

Figura 10. Visualización de óptimos locales y global

Siguiendo con el mismo ejemplo, se puede evidenciar claramente la dependencia de la solución inicial de la que se parte, así la posibilidad de que existan otras soluciones de una mayor calidad a la obtenida, bien sean otros óptimos locales (puntos S y T) o el óptimo global (punto V).

Debido a la limitación existente de no poder garantizar el óptimo global, se han diseñado otros algoritmos, dentro ya de la denominación de métodos metaheurísticos, como son la Búsqueda Tabú y el recocido Simulado, con estrategias concretas para evitar esta situación.

Implementaciones de las heurísticas de mejora:

Las heurísticas de mejora para el VRP están diseñadas para mejorar una solución inicial factible mediante un mecanismo de búsqueda. Estas heurísticas pueden clasificarse en *intra-rutas,* si solo se utiliza una única ruta para realizar la mejora, o *entre-rutas* si están involucradas varias rutas.

4.2. Mejora Intra-Rutas

Este método se centra de forma individual en cada una de las rutas, con el fin de mejorar la secuencia de una ruta al cambiar el orden de los nodos dentro de esa ruta. Son movimientos típicos del TSP.

Dentro de esta categoría podemos distinguir los métodos exactos (Tucker) y los algoritmos K-Opt (2-Opt, 3-Opt…) y Or-Opt.

4.2.1. 2-OPT

4.2.1.1. Descripción

El procedimiento de esta heurística para un problema de distancias euclídeas se basa en que si dos aristas se cruzan, puedan ser sustituidas por otras dos que no lo hagan de forma que se disminuya la distancia recorrida total, creándose así una nueva ruta como se puede observar en la Figura 11.

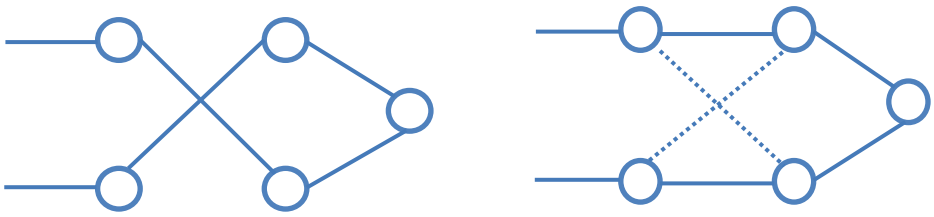

Figura 11. Intercambio 2-Opt

Un movimiento 2-Opt consiste en eliminar dos tramos de recorrido y remplazarlas por la única solución posible de unión entre estos 4 nodos. Además, al realizar este intercambio, se produce un cambio de dirección con respecto a la ruta anterior.

Gráficamente podemos observar este intercambio de los nodos *i* y *j* con sus siguientes correspondientes *s(i)* y *s(j)* como se visualiza en la Figura 12.

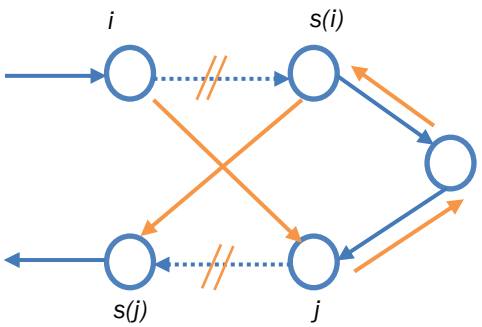

Figura 12. Intercambio 2-Opt

En la Figura 13 se muestra otra forma de representarlo:

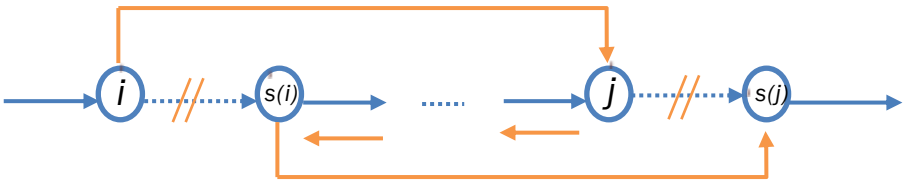

Figura 13. Intercambio 2-Opt

La mejora que se produce al realizar este cambio es la siguiente:

$$\Delta_{ij} = \underbrace{d\big(i,s(i)\big) + d\big(j,s(j)\big)}_{\text{Aristas antiguas}} - \underbrace{d(i,j) - d(s(i),s(j))}_{\text{Aristas nuevas}}$$

De tal forma que si la distancia de los recorridos antiguos es mayor que la distancia de los recorridos nuevos se procede a realizar el intercambio de los correspondientes nodos.

Si para todo *i*, *j* con $i \neq j$ es $\Delta_{ij} \leq 0$, la solución actual es un óptimo local, ya que no existe un posible intercambio que mejore la situación actual. Pero en el caso de que se encuentre algún nodo con $\Delta_{ij} > 0$, se procede a realizar el movimiento 2-Opt donde se realizan los siguientes cambios:

- $s(i) = j$

- $s(s(i)) = s(j)$

- Cambio de dirección desde el nodo *j* hasta el nodo el nodo *s(i)*.

Implementando la estrategia de "Best improvement" o mejora máxima, la solución final va a ser la resultante de realizar los anteriores cambios cuando se da el valor máximo de las mejoras calculadas para todos los nodos.

Esta heurística solo es válida para el caso simétrico STSP, puesto que se supone que la distancia entre dos puntos es independiente de la dirección del recorrido.

En el caso euclídeo, no es necesario que dos arcos se crucen para realizar un 2-intercambio como se demuestra en la Figura 14:

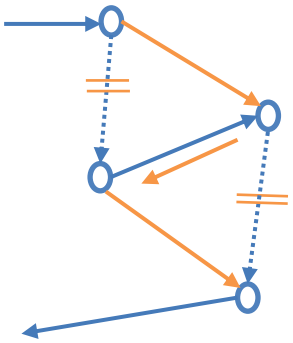

Figura 14. Intercambio 2-Opt

Examinar si existe algún movimiento 2-Opt exige un tiempo de orden $O(n^2)$ al tener que analizar todos los pares de aristas en la ruta. Se pueden aplicar mejoras para acelerar el algoritmo.

4.2.1.2. Procedimiento

El pseudocódigo de este procedimiento es el siguiente:

```
forall (toda ruta completa) do
        while (la solución de la ruta no sea el óptimo local) do
                forall (nodos i que se pueda realizar intercambio) do
                        forall (nodos j | j e i no sean nodos contiguos) do
                                - calcular la mejora que supone el intercambio
                                - seleccionar mejora_max
                        end-do
                end-do
                if (mejora_max<=0) then
                        - óptimo local. No se realiza intercambio
                else
                        - realizar intercambio de posiciones
                end-if
                - calcular la distancia con la mejora realizada
        end-do
end-do
```

4.2.2. K-OPT

4.2.2.1. Descripción

De forma general a lo comentado anteriormente, Lin, S. (1965) estableció que un k-intercambio se basa en eliminar k-arcos con k>1 y combinar las k posibles uniones de la mejor forma posible para reducir la distancia recorrida.

En una ruta en la que se visitan n clientes, existen $\binom{n+1}{k}$ formas posibles de eliminar k arcos y dada una combinación de los arcos a eliminar, hay $2^{k-1}(k-1)!$ maneras de volver a conectar la ruta, incluyendo la básica de la que se parte. Por consiguiente, el número total de k-intercambios posibles es $\binom{n+1}{k} 2^{k-1}(k-1)!$. El tiempo computacional que puede conllevar comprobar si una solución es k-óptima puede ser de $O(n^k)$ en el peor de los casos.

Los valores más comunes que puede tomar k suele ser 2 o 3. Por ejemplo, para el caso de 3-opt, al romper 3 arcos existen 8 diferentes opciones para reconectar los nodos, pero solo 4 de ellos (e,f,g,h) introducen todos los arcos nuevos. El resto de rutas, salvo la ruta inicial, son movimientos 2-Opt. Todas estas posibilidades se visualizan en la Figura 15.

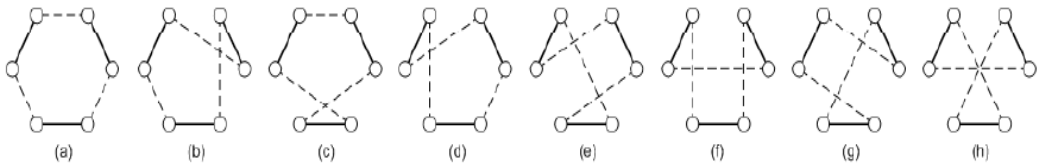

Figura 15. 3-Opt
[Fuente: Combining 2-Opt, 3-Opt and 4-Opt, with K-swap-kick perturbations for the traveling salesman problem. Andrius Blazinskas, Alfonsas Misevicius].

4.2.3. OR-OPT

Otra versión de este tipo de heurísticas situada entre la 2-Opt y 3-Opt es la Or-Opt (Or, I., 1976) Esta consiste en mover un segmento de K=3 o menos nodos conservando su orden y recolocarlos entre dos nodos vecinos en cualquier otra parte de la ruta.

Se comienza analizando las posibles opciones para un segmento de 3 clientes, luego de 2 y por último de 1, y en estos intercambios no se modifica la dirección de partida.

A continuación se muestra en la Figura 16 todas las posibles reubicaciones de los tres primeros nodos.

K=3

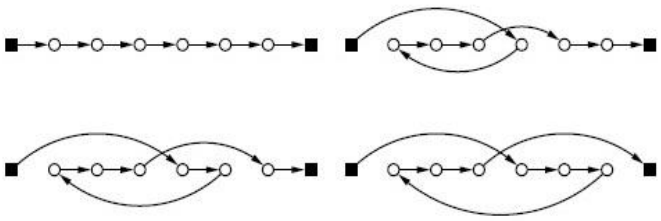

Figura 16. Or-Opt k=3
[Fuente: Heurísticas para Problemas de Ruteo de Vehículos. Alfredo Olivera].

Para el caso de reubicar 2 nodos contiguos, el esquema gráfico sería el de la Figura 17:

K=2

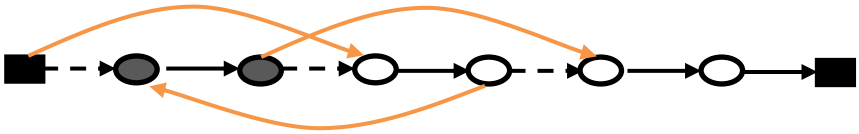

Figura 17. Or-Opt k=2

Y en la Figura 18, lo mismo para un nodo:

K=1

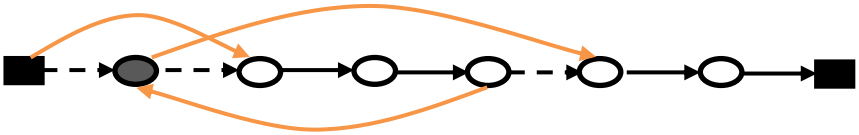

Figura 18. Or-Opt k=1

4.2.3.2. Procedimiento

El pseudocódigo es el siguiente:

```
forall (toda ruta completa) do
        k=3
        while (k>=1) do
                while (la solución de la ruta no sea el óptimo local) do
                        forall (nodos que se pueda realizar intercambio) do
                                forall (posiciones que puedan ubicarse los k nodos) do
                                        - calcular la mejora que supone el intercambio
                                        - seleccionar mejora_max
                                end-do
                        end-do

                        if (mejora_max<=0) then
                                - óptimo local. No se realiza intercambio
                                K=k-1   Pasamos a probar con otra agrupación
                                        menor de nodos
                        else
                                - realizar intercambio de posiciones
                        end-if
                            - calcular la distancia con la mejora realizada
                end-do
        end-do
end-do
```

4.2.4. Método exacto de Tucker-Miller-Zemlin

4.2.4.1. Descripción

El método de mejora que se va a aplicar en este apartado es el método exacto de Tucker-Miller-Zemlin. Dicho proceso trata de encontrar la solución exacta para cada una de las rutas que recibe de una solución inicial siguiendo el modelo de TSP explicado en el capítulo 2 y añadiéndole la restricción de Tucker-Miller-Zemlin.

En cuanto a la formulación de este modelo, se puede definir el problema como un problema de programación lineal entera donde las variables binarias x_{ij} toman el valor de 1 en el caso de que se realice el trayecto desde el nodo *i* hasta el nodo *j*.

$$Función\ objetivo: \qquad Min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$Sujeto\ a: \qquad \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 2, \dots, n \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 2, \dots, n \qquad (2)$$

$$x_{ij} \in \{0, 1\} \quad i, j = 1, \dots, n$$

Las dos restricciones de entrada y salida indican que:

(1) De cada nodo solo puede salir un arco.
(2) A cada nodo solo puede llegarle un arco.

La restricción de Tucker-Miller-Zemlin que impide la formación de subtours es la siguiente:

$$u_i - u_j + n \cdot x_{i,j} \leq n - 1 \quad \forall\, i, j \in V - \{1\} / i \neq j, \quad u_i + u_j \leq n \qquad (6)$$

$$1 \leq u_i \leq n$$

Donde la variable u_i es una variable continua adicional que representa el número de nodos visitados después de haber pasado por el cliente *i* y *n* es el número de nodos que contiene la ruta en cuestión. Se puede destacar el cumplimiento con los requisitos de conectividad:

Cuando $x_{i,j} = 0$ → $u_i \leq n$ y $u_j \geq 1$

Cuando $x_{i,j} = 1$ → $u_j \geq u_i + 1$

4.2.4.1. Procedimiento

Para este proceso, se va a crear un modelo donde se va a hallar la solución inicial mediante alguno de los métodos de inserción, y desde este se va a llamar a otro modelo, que denominaremos Subproblema, donde reciba en cada iteración la solución de una ruta con objeto de mejorarla.

En el modelo principal se necesita en primer lugar definir el subproblema y cargarlo. Una vez obtenida la solución inicial, se enviará por memoria para cada ruta el número de nodos y la matriz de distancias como datos generales y el número de datos de la ruta que corresponda y su debida secuencia. A continuación se ejecuta el Subproblema y se recibe la solución de cada ruta.

Para poder realizar este proceso en Mosel se necesita usar el módulo *mmjobs*, el cual posibilita la carga de varios problemas en memoria y su ejecución simultánea.

El pseudocódigo del problema principal es el siguiente:

```
uses "mmjobs"
tsp_tucker: Model
if compile("tsp_tucker.mos")<>0 then          ! Compile the model file
        writeln("Algún problema al compilar tsp_tucker.mos")
        exit(1)
end-if
load (tsp_tucker, "tsp_tucker.bim")
!========= mejora de la solución inicial con Tucker-Miller ============
dist_total:=0
forall (k in rutas) do
      initializations to "shmem:datos_tsp"
            n nnr indr dist
      end-initializations
      run (tsp_tucker)              ! Start model execution
      wait                          ! Wait for model termination
      dropnextevent                 ! Ignore termination event message
       initializations from "raw:"
            dist_mej as "shmem:dist_ruta"
      end-initializations
  dist_total := dist_total + dist_mej
end-do
```

En el caso del Subproblema, tras recibir los datos necesarios, se minimiza la función objetivo sujeta a las restricciones de Tucker siguiendo la formulación explicada anteriormente y se envía el resultado.

Este proceso iterativo se realiza para cada ruta definida por el método de inserción seleccionado.

El pseudocódigo del Subproblema es el siguiente:

```
uses "mmjobs"

initializations from "shmem:datos_tsp"
        n nnr indr dist
end-initializations

objetivo:=sum(i,j in nodos_ruta | i<>j) dist2(i,j)*x(i,j)

forall (i,j in nodos_ruta)x(i,j)is_binary

forall (i in nodos_ruta)res_1(i):=sum(j in nodos_ruta | i<>j)x(i,j)=1

forall (j in nodos_ruta)res_2(j):=sum(i in nodos_ruta | i<>j)x(i,j)=1

! Añadimos la restricción de Tucker-Miller

forall (i,j in nodos_ruta | i>1 and j>1 and i<>j)res_3(i,j):=u(i) - u(j) + nnr2 * x(i,j) <=
nnr2-1

minimize(objetivo)

dist_ruta:=integer(getobjval)

initializations to "raw:noindex"
        dist_ruta as "shmem:dist_ruta"
end-initializations
```

4.3. Mejora Entre-Rutas

Este procedimiento involucra varias rutas para poder realizar la mejora correspondiente.

Alex Van Breedam (1994) distingue cuatro tipos de movimientos: String Cross, String Exchange, String Relocate y String Mix.

- ***String Cross (SC)***: se intercambian dos cadenas de vértices al cruzarse dos arcos en dos diferentes rutas. Esto corresponde al intercambio de dos segmentos enteros. Gráficamente, el movimiento que se realiza es el de la Figura 19:

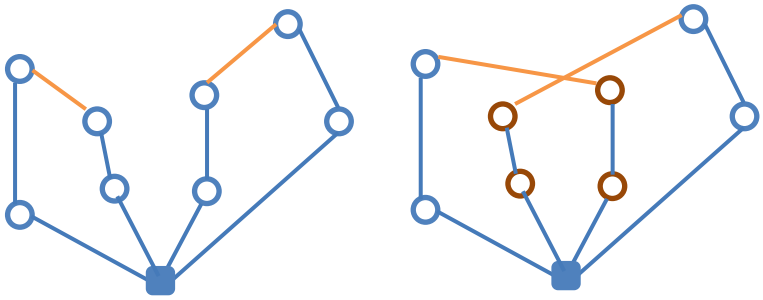

Figura 19. String Cross

- ***String Exchange (SE)***: se intercambian dos cadenas de menos de K nodos entre las dos rutas. Simbólicamente, esto puede ser representado por (x1, x2), donde $x1$ y $x2$ son enteros que definen la longitud de la cadena a intercambiar en ambas rutas. Ambos parámetros no tienen por qué coincidir, pero sí deben respetar las restricciones $1 \leq x1 \leq K$ y $1 \leq x2 \leq K$, donde K representa la máxima cantidad de nodos que se pueden intercambiar. Los valores más comunes para el número máximo de nodos son K=1 y K=2. Gráficamente, el movimiento que se realiza es el de la Figura 20:

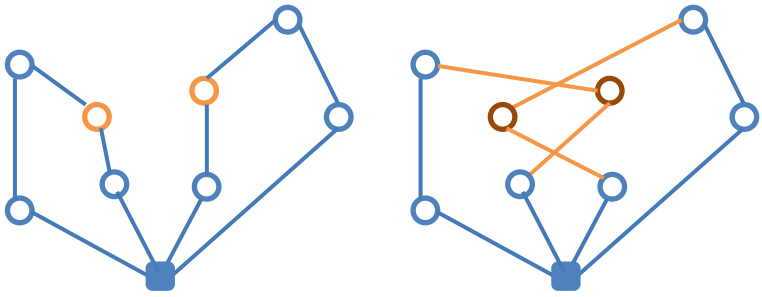

Figura 20. String Exchange

- ***String Relocate (SR)***: es el movimiento de una cadena de nodos de una ruta hacia otra ruta. Simbólicamente, esto puede ser representado por (x,0) o (0,x), donde el número de nodos está limitado por un parámetro máximo del número de nodos K, $1 \leq x \leq K$. Los valores más usados son K=1 y K=2. Con la aplicación de este tipo de movimiento se trata de reducir el número de rutas totales. Gráficamente, el movimiento que se realiza es el de la Figura 21:

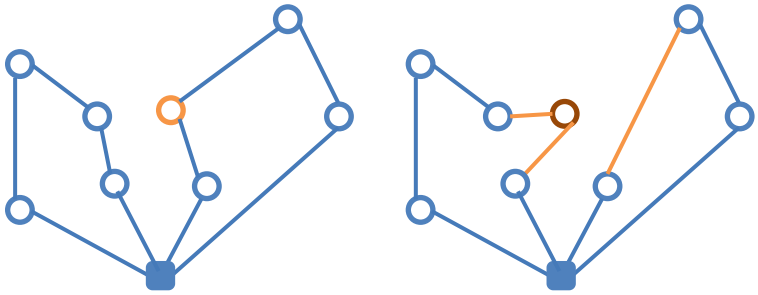

Figura 21. String Relocate

- ***String Mix (SM):*** este procedimiento es una mezcla del String Exchange y del String Relocate, el cual selecciona el mejor movimiento entre el que ofrece cada uno de estos dos métodos. El número de rutas puede ser reducido en el caso de que se seleccionen los movimientos del String Relocation.

Procedimiento:

- Paso 1. La solución inicial es la solución actual.

- Paso 2. Seleccionar el primer movimiento dentro de la primera pareja de rutas de la solución actual, basadas en la secuencia establecida. Establecer las paradas requeridas para el tipo de movimiento en ambas rutas.

- Paso 3. Si el movimiento en cuestión no es factible o no mejora la función objetivo, ir al paso 5.

- Paso 4. Si la estrategia seleccionada es *First Improvement*, realizar el movimiento. Es entonces cuando la nueva solución se convierte en la solución actual. Ir al paso 2.

En el caso de que se seleccione la estrategia B*est Improvement* y dicha solución sea mejor que todas las soluciones anteriores, se guarda esta solución. Ir al paso 6.

- Paso 5. Si se ha completado la evaluación del ciclo entero y la estrategia seleccionada es *First Improvement*, la heurística finaliza.

En el caso de ser *Best Improvement,* si se ha completado el ciclo entero y se ha guardado alguna solución, esta entonces se convierte en la solución actual. Ir al paso 2. Si ninguna solución ha sido guardada durante el ciclo por ser mayor que el resto, la heurística termina.

- Paso 6. Seleccionar el siguiente movimiento así como el número de paradas en la ruta, o en su caso, la siguiente pareja de rutas para ser evaluadas. Si no pueden seleccionarse ningún movimiento, ir al paso 5, o si no al paso 3.

Para evaluar la calidad de los movimientos que se pueden desarrollar, Van Breedam considera las dos estrategias de mejora local (*First Improvement* si se implementa el primer movimiento que mejora la función objetivo y *Best Improvement* si se evalúan todos los posibles movimientos y se realiza el mejor de todos) y define una serie de parámetros que pueden influenciar en el comportamiento de la implementación correspondiente. Estos parámetros son los siguientes:

- La solución inicial (buena, pobre).

- La longitud de la cadena para los tipos SE, SR, SM (k=1 o 2).

- La estrategia seleccionada (FI, BI).

- La evaluación del procedimiento para una longitud de la cadena k>1 (evaluar todas las posibles longitudes de la cadena entre una pareja de rutas y aumentar k cuando en una evaluación completa no se ha identificado ningún movimiento de mejora).

La primera observación que realizó Van Breedam fue la de iniciar la búsqueda desde una solución buena mejor que desde una solución pobre, ya que ofrecía una mayor calidad en cuanto a la solución final y al tiempo de computación al tener que hacer menos evaluaciones. Además, la elección de la estrategia según sea First o Best Improvement no ofrecía diferencias significativas en las soluciones finales.

Visualización de los intercambios:

Utilizando la librería *mmive* se pueden mostrar por pantalla gráficos de funciones, diagramas, redes, etc., la cual va a servir de gran utilidad para exponer gráficamente las rutas de los modelos empleados y observar así cómo se llevan a cabo dichos intercambios.

Para todos los ejemplos mostrados a continuación se utiliza la heurística de inserción secuencial simple.

String Relocate

La Figura 22 muestra el conjunto de todas las rutas del modelo E036-11h en cuestión con sus 36 nodos y rutas correspondientes. Se puede distinguir claramente cómo hay una ruta que contiene a nodos muy alejados entre sí (nodo señalado en azul el 36 y en naranja el 24).

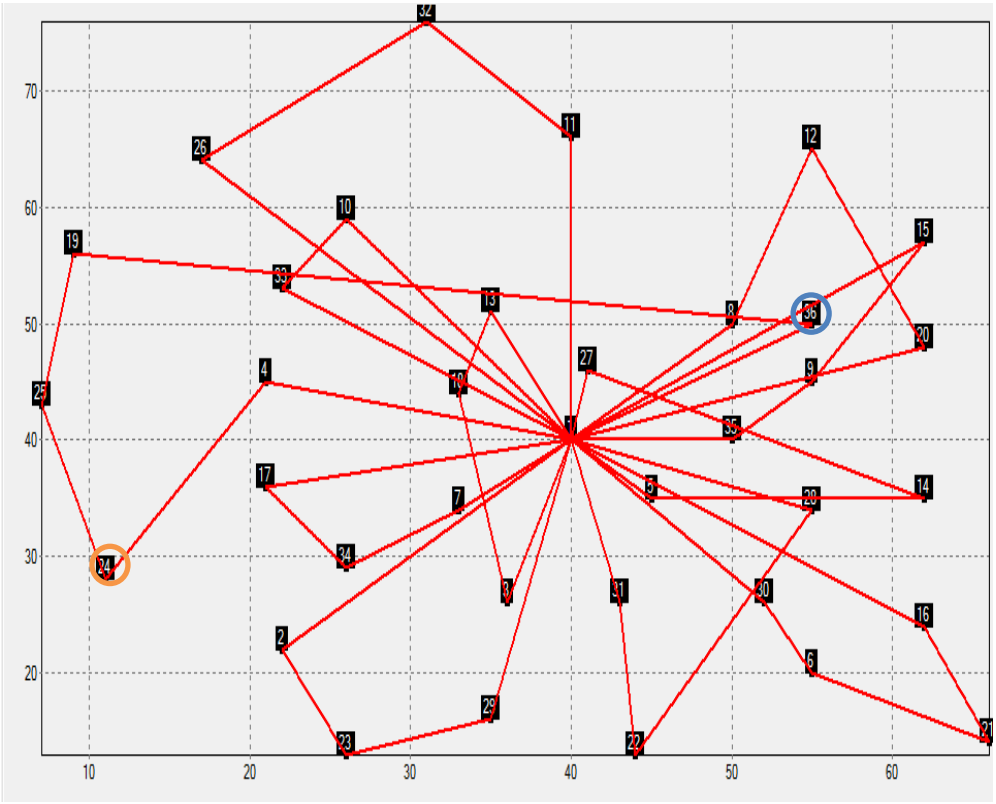

Figura 22. Situación anterior al intercambio SR

Para distinguir el cambio que se produce, se reproducen gráficamente solo las rutas relacionadas antes y después de la mejora (Figura 23 y Figura 24). Los nodos responsables de este cambio son el 24 (se inserta después del 2) y el 36 que se disponen en otras rutas más cercanas a ellos.

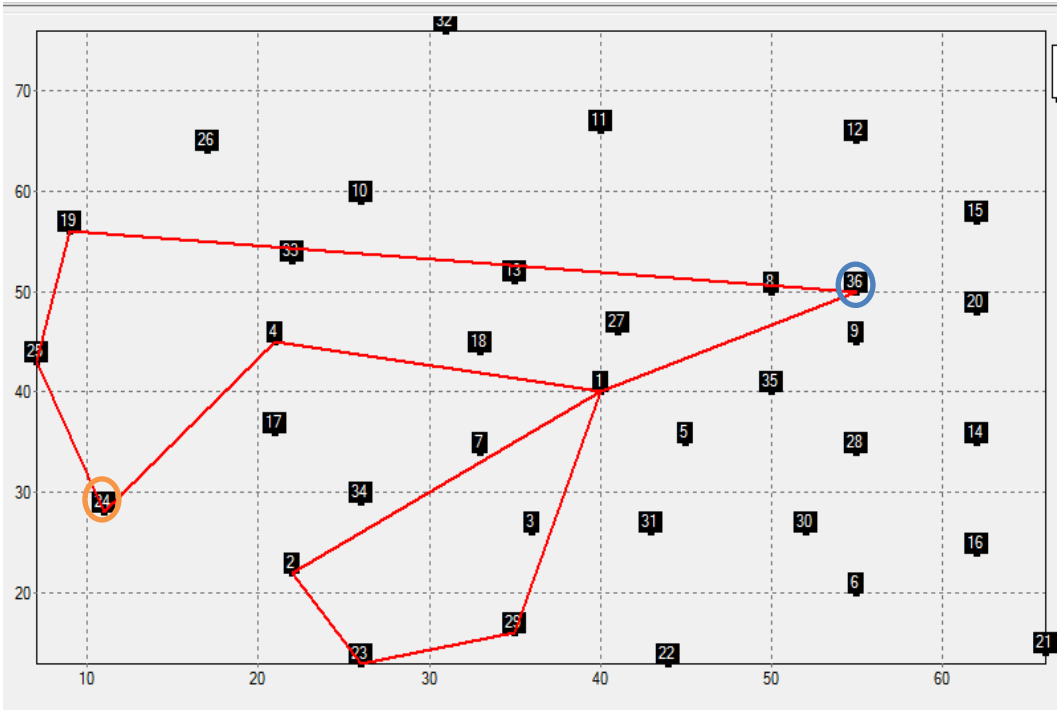

Figura 23. Situación simplificada 1 anterior al intercambio SR

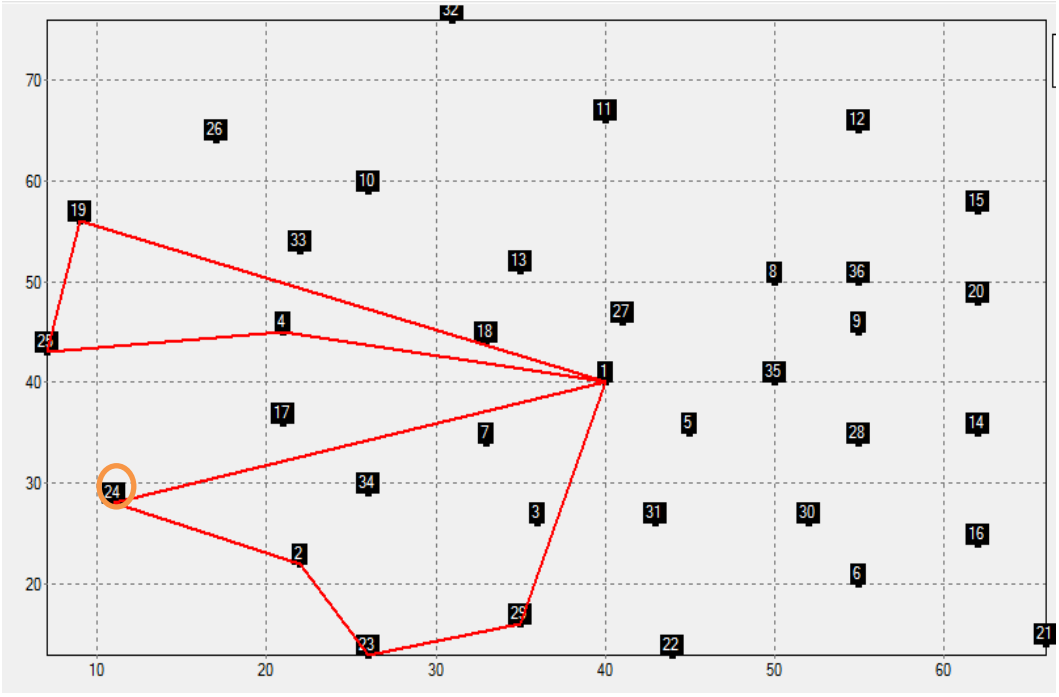

Figura 24. Situación simplificada 1 posterior al intercambio SR

El modelo E041-14h es otro ejemplo claro de la mejora que se produce con la recolocación de los nodos, por ejemplo con el nodo 24 que al ser insertado en último lugar se encuentra muy alejado de los de su ruta (Figura 25) y conviene, por tanto, intentar acoplarlo en otra más cercana (Figura 26).

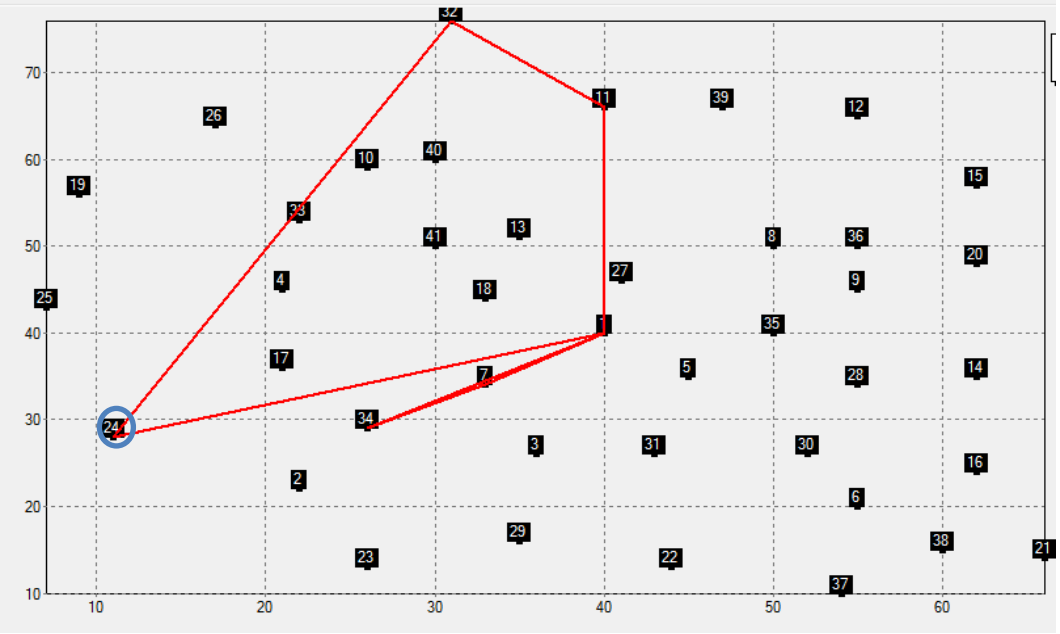

Figura 25. Situación simplificada 2 anterior al intercambio SR

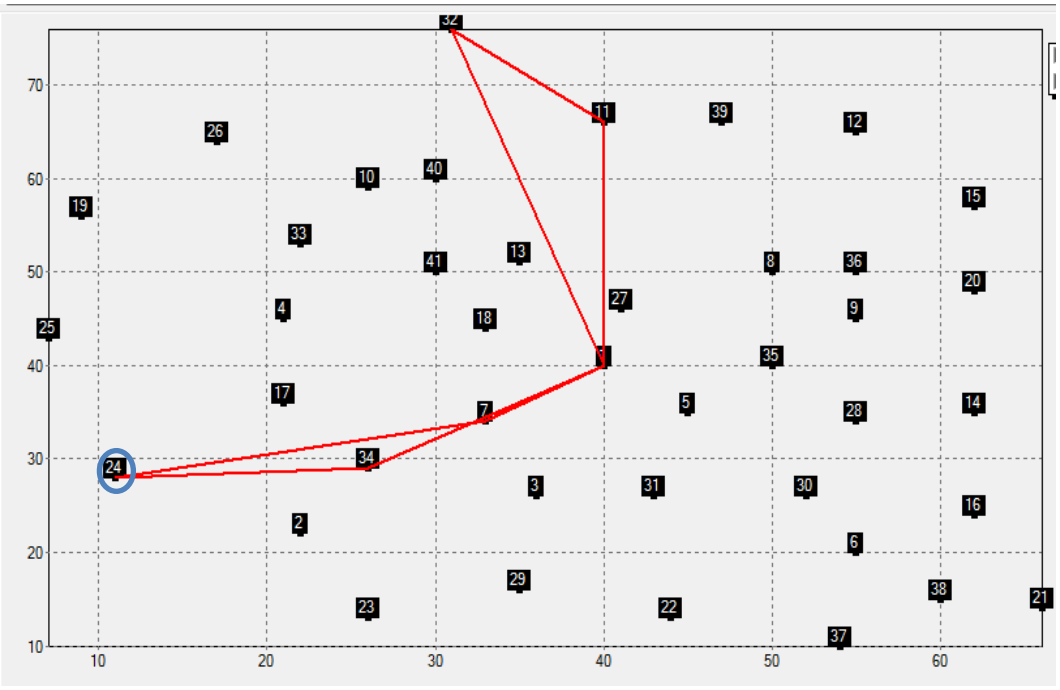

Figura 26. Situación simplificada 2 posterior al intercambio SR

String Cross

Para este método, se procede a reemplazar los dos arcos señalados en la Figura 27, intercambiándose, por tanto, el resto de nodos de las rutas correspondientes (Figura 28).

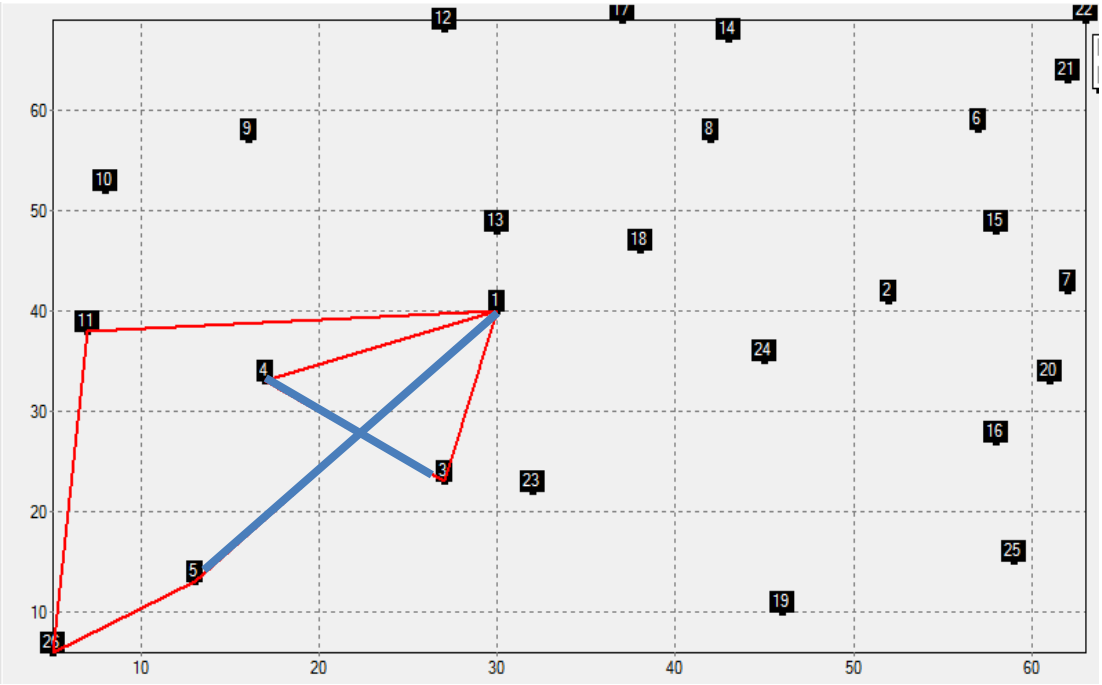

Figura 27. Situación simplificada anterior al intercambio SC

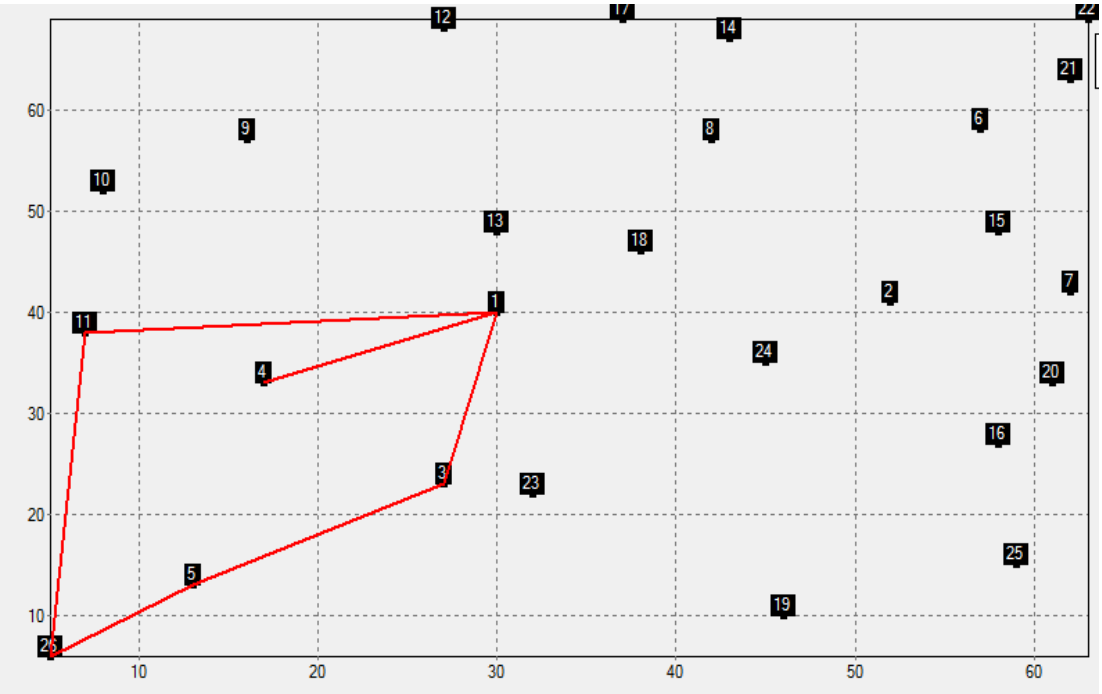

Figura 28. Situación simplificada posterior al intercambio SC

String Exchange k=1

En el modelo E101-14s se lleva a cabo un intercambio entre nodos de diferente ruta, donde el nodo 13 pasa a ser de la ruta superior y el nodo 29 de la inferior (Figura 29 y Figura 30).

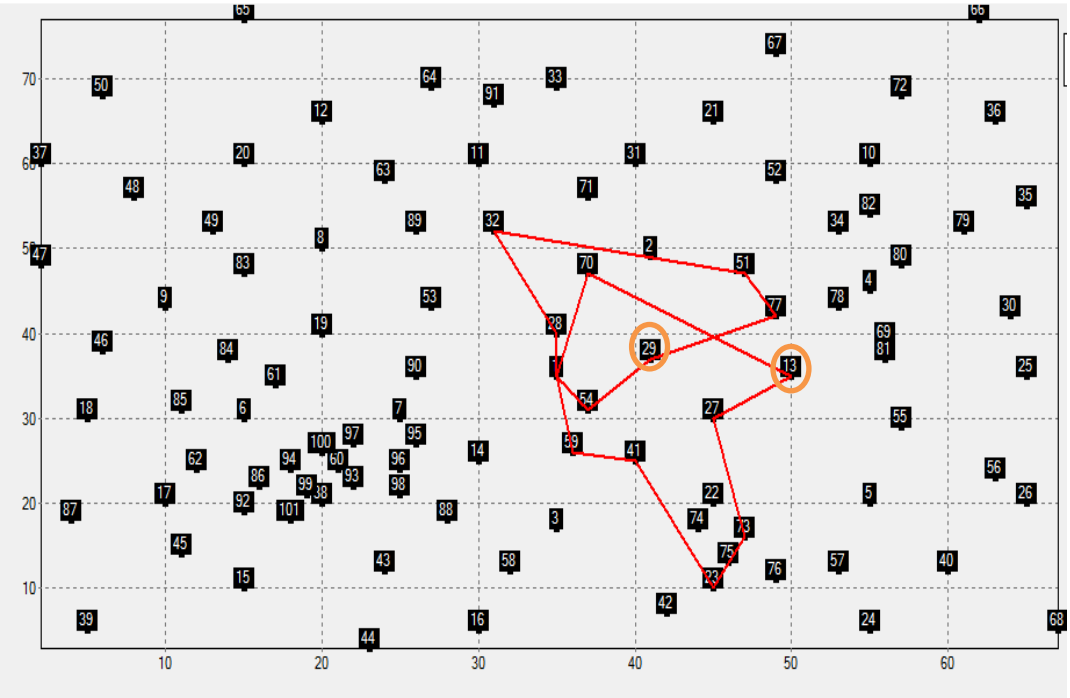

Figura 29. Situación simplificada anterior al intercambio SE k=1

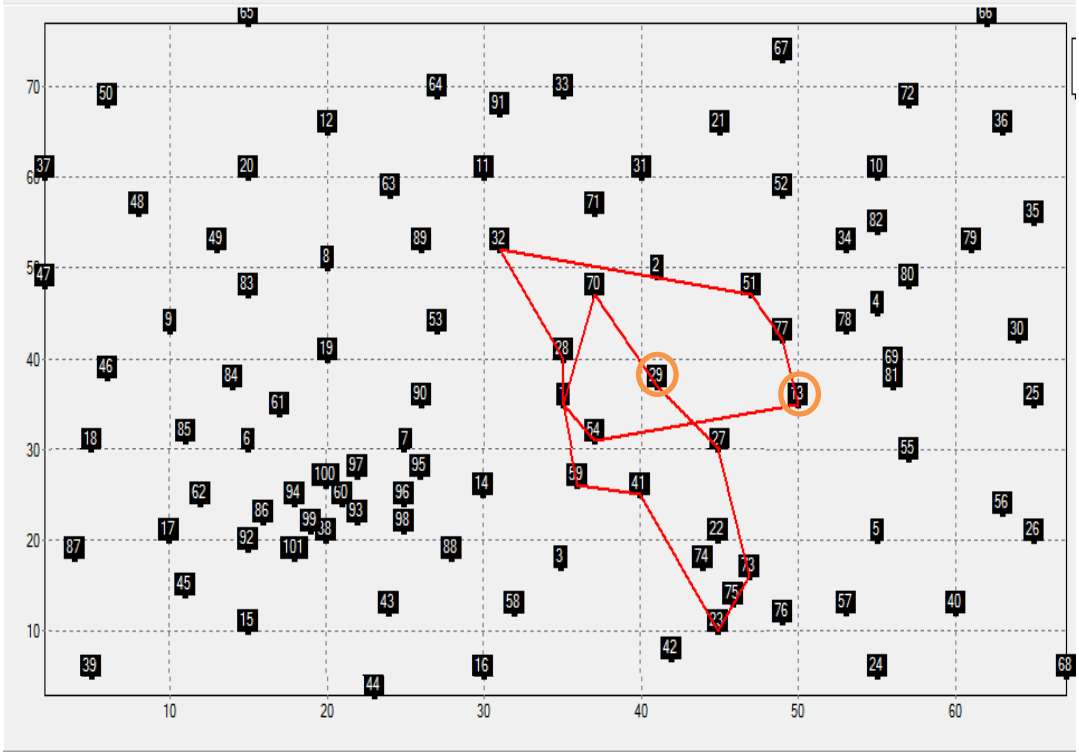

Figura 30. Situación simplificada posterior al intercambio SE k=1

String Exchange k=2

En el modelo E041-14h se produce un intercambio 2 a 2 de nodos entre diferentes rutas (Figura 31 y Figura 32).

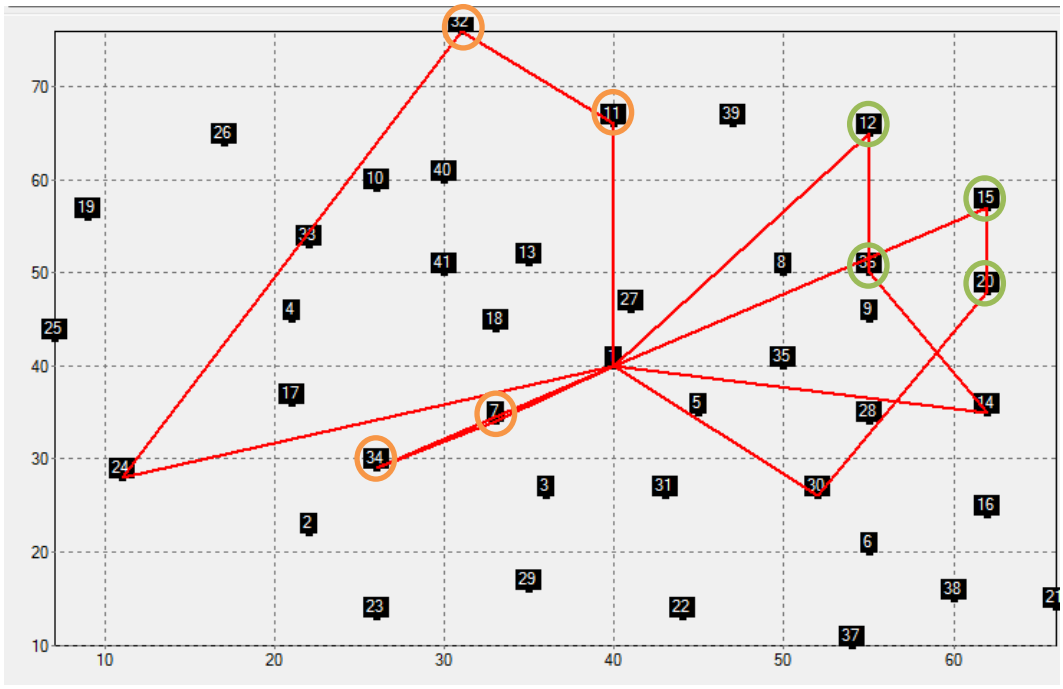

Figura 31. Situación simplificada anterior al intercambio SE k=2

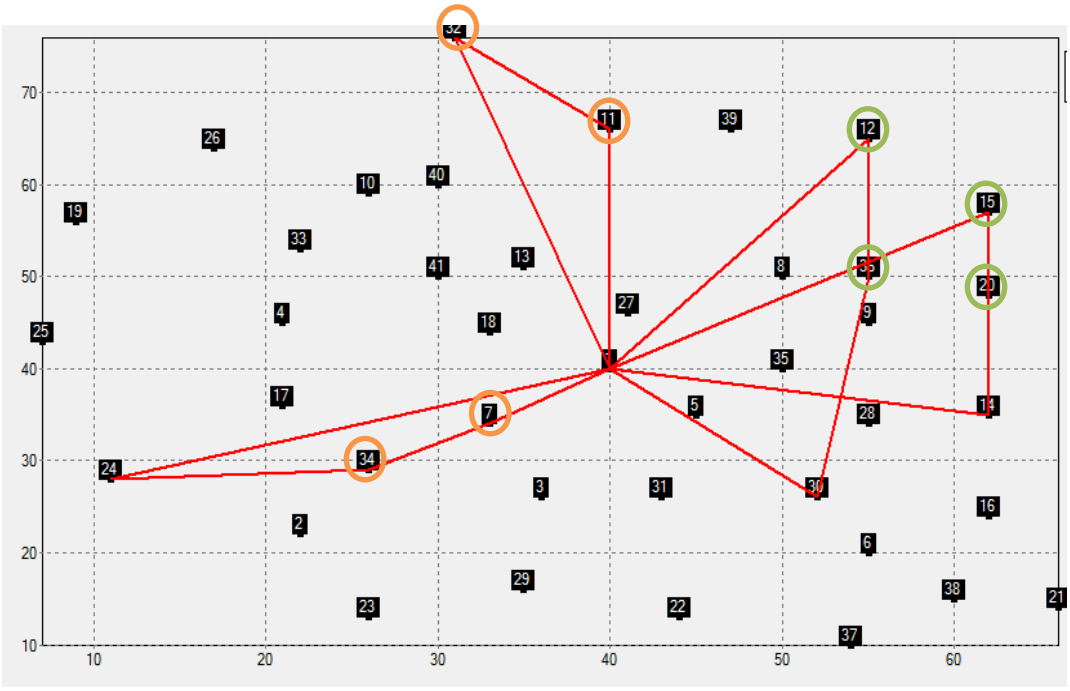

Figura 32. Situación simplificada posterior al intercambio SE k=2

4.4. Experimentación computacional

Mejora intra-rutas

En primer lugar comenzamos analizando los resultados obtenidos de los métodos de mejora intra-rutas partiendo de la heurística de inserción secuencial simple como solución inicial (Tabla 20, Gráfico 16 y Gráfico 17).

Modelo	Sin mejora		2-Opt		Or-Opt		Tucker	
	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)
E021-06m	469,12	0,00	469,12	0,002	469,12	0,002	469,12	0,069
E022-06m	542,57	0,00	542,57	0,002	542,57	0,002	542,57	0,107
E023-05S	746,66	0,00	667,95	0,012	678,24	0,012	667,23	0,974
E026-08m	702,31	0,00	702,31	0,002	701,08	0,002	701,08	0,102
E030-04S	577,45	0,00	561,41	0,012	560,59	0,012	560,24	1,865
E031-09h	613,44	0,02	613,44	0,012	613,44	0,012	613,44	0,114
E033-05s	912,80	0,02	887,21	0,017	882,04	0,017	882,04	0,811
E036-11h	771,18	0,02	767,45	0,004	767,45	0,004	767,45	0,154
E041-14h	954,55	0,02	954,55	0,004	954,55	0,004	954,55	0,135
E045-04f	972,68	0,02	972,68	0,009	951,65	0,009	950,16	2,829
E051-05e	592,18	0,02	592,18	0,014	587,05	0,014	587,05	0,563
E072-04f	339,80	0,03	324,37	0,038	332,27	0,038	324,67	39,612
E076C09r	1567,85	0,02	1521,12	0,053	1506,95	0,053	1498,22	1,811
E101-14s	1327,87	0,02	1322,82	0,028	1312,42	0,028	1312,42	0,624
E121-07c	1117,07	0,05	1107,23	0,069	1096,19	0,069	1091,57	81,380
E151-12c	1272,54	0,06	1252,97	0,085	1238,26	0,085	1236,43	6,100
E200-17c	1612,17	0,10	1588,64	0,139	1569,12	0,139	1561,50	70,379

Tabla 20. Resultados obtenidos de la heurística de Inserción Secuencial de Mole & Jameson simple con las diferentes mejoras intra-rutas.

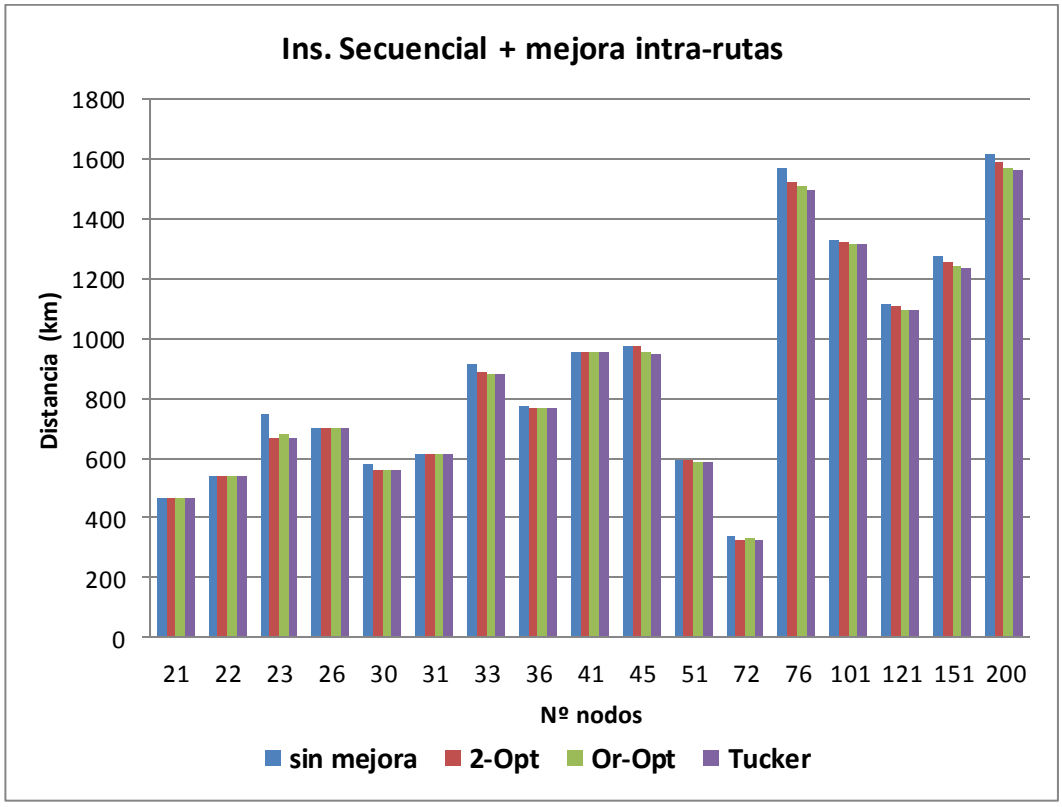

Gráfico 16. Distancia recorrida de la heurística de inserción secuencial simple mejorada en función de los tres métodos intra-rutas

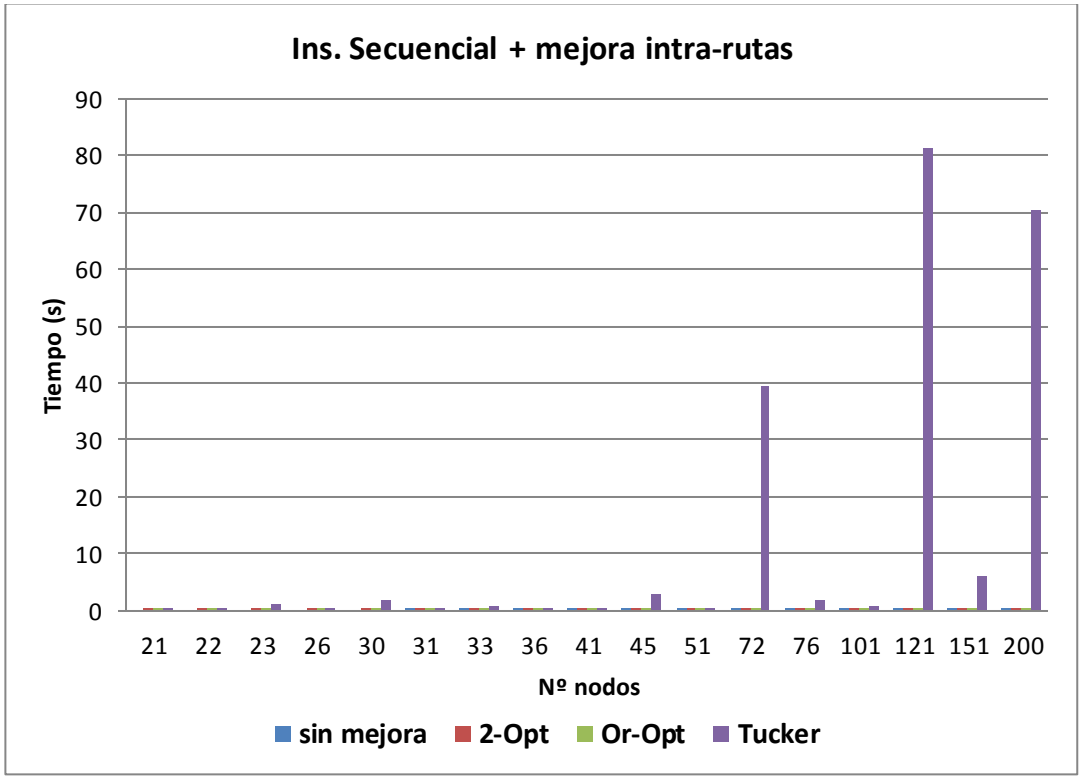

Gráfico 17. Tiempo de ejecución de la heurística de inserción secuencial simple mejorada en función de los tres métodos intra-rutas

Se produce mejora con alguno de los métodos en el 76,47% de los casos, de los cuales 23,53% son con el intercambio 2-Opt y 58,82% con el intercambio Or-Opt.

Por otra parte, con el intercambio 2-Opt se alcanza la solución exacta de Tucker-Miller-Zemlin en el 5,88% de los modelos y con el intercambio Or-Opt en el 29,41% (Tabla 21).

Método	Solución exacta	Mayor mejora
2-Opt	5,88%	23,53%
Or-Opt	29,41%	58,82%

Tabla 21. Frecuencia relativa de la solución exacta y de la mayor mejora para los intercambios 2-Opt y Or-Opt.

También, la capacidad media de mejora de cada uno de los intercambios con respecto a la solución inicial es la señalada en la Tabla 22:

Método	Variación de mejora
2-Opt	1,67%
Or-Opt	1,98%
Tucker	2,30%

Tabla 22. Frecuencia relativa de la mejora media producida para cada método.

En referencia al tiempo computacional, es notoria la gran diferencia existente con el método exacto de Tucker.

Mejora entre-rutas

Por otra parte, se analizan los métodos de mejora entre-rutas String Relocate (SR), String Cross (SC) y String Exchange (SE) para k=1 y k=2 nodos intercambiables (Tabla 23 y Gráfico 18).

La implementación utilizada para todos los modelos es la de Best Improvement, realizándose en este caso el intercambio que tiene una mayor mejora por cada ciclo.

Modelo	Sin mejora	SR	SC	SE k=1	SE k=2
	Solución (Km)	Solución (Km)	Solución (Km)	Solución (Km)	Solución (Km)
E021-06m	469,12	456,98	469,12	469,12	469,12
E022-06m	542,57	540,75	542,57	542,57	542,57
E023-05S	746,66	740,48	668,98	667,95	746,66
E026-08m	702,31	673,45	656,17	702,31	702,31
E030-04S	577,45	557,45	577,45	561,41	577,45
E031-09h	613,44	612,29	612,29	613,44	613,44
E033-05s	912,80	878,87	890,44	887,21	912,80
E036-11h	771,18	767,96	771,18	767,45	771,18
E041-14h	954,55	936,27	935,46	954,55	914,81
E045-04f	972,68	940,41	972,68	972,68	972,68
E051-05e	592,18	591,15	592,18	592,18	592,18
E072-04f	339,80	304,86	327,33	324,37	339,80
E076C09r	1567,85	1529,24	1555,89	1521,12	1567,85
E101-14s	1327,87	1261,32	1277,33	1320,80	1326,36
E121-07c	1117,07	1117,07	1116,76	1107,23	1117,07
E151-12c	1272,54	1248,11	1252,82	1252,97	1272,54
E200-17c	1612,17	1571,29	1609,30	1588,64	1612,17

Tabla 23. Resultados obtenidos de la heurística de Inserción Secuencial de Mole & Jameson simple con las diferentes mejoras entre-rutas.
[SR= String Relocate; SC= String Cross; SE= String Exchange]

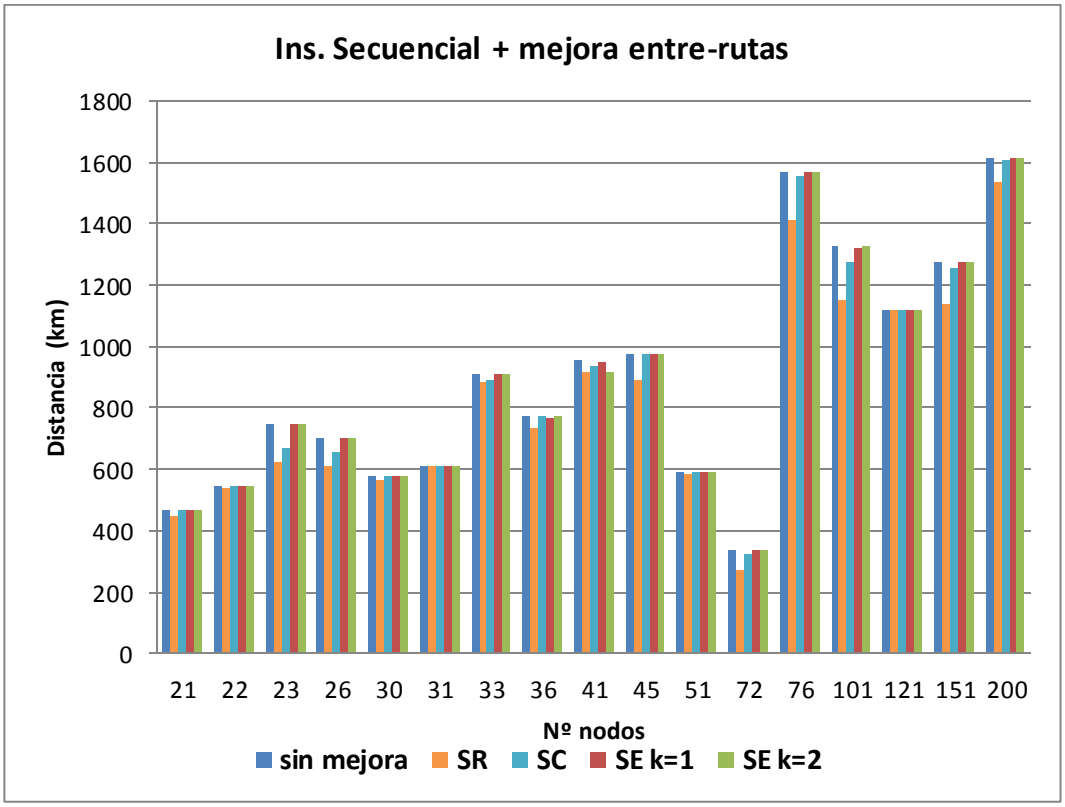

Gráfico 18. Distancia recorrida de la heurística de inserción secuencial simple mejorada en función de los cuatro métodos entre-rutas.
[SR= String Relocate; SC= String Cross; SE= String Exchange]

A continuación, en la Tabla 24 se muestra para cada uno de los métodos de mejora, la frecuencia relativa con la que consigue obtener el mayor ahorro, así como la variación positiva media que se produce.

Método	Mejor solución	Variación de mejora
SR	88,24%	6,93%
SC	5,88%	1,86%
SE k=1	0,00%	0,09%
SE k=2	5,88%	0,25%

Tabla 24. Frecuencia relativa de la mejor solución y la mejora media producida para cada método.

Se evidencia claramente la fuerte capacidad que presenta el método de String Relocate con respecto a los otros tres, donde la mejor solución se ofrece con este método para el 88,24% de los casos, así como la capacidad de mejora del algoritmo es de un 6,93% hacia la solución inicial de la que parte.

Como ventaja, este método presenta la posibilidad de reducir el número de rutas totales, pero por contra, viene limitado por la capacidad de utilización de los vehículos, puesto que si no existe capacidad libre en al menos una ruta no puede recolarse un nodo procedente de otra ruta.

Resultados finales

En el siguiente análisis, se compara el mejor proceso de intercambio de los métodos de mejora dentro de la misma ruta, intercambio Or-Opt junto con el método exacto de Tucker, y el intercambio String Relocate como el mejor de los métodos de intercambio entre las rutas (Tabla 25 y Gráfico 19).

Modelo	Sin mejora		Or-Opt		Tucker		SR	
	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)
E021-06m	469,12	0,00	469,12	0,002	469,12	0,069	456,98	0,002
E022-06m	542,57	0,00	542,57	0,002	542,57	0,107	540,75	0,010
E023-05S	746,66	0,00	678,24	0,012	667,23	0,974	740,48	0,010
E026-08m	702,31	0,00	701,08	0,002	701,08	0,102	673,45	0,000
E030-04S	577,45	0,00	560,59	0,012	560,24	1,865	557,45	0,010
E031-09h	613,44	0,02	613,44	0,012	613,44	0,114	612,29	0,000
E033-05s	912,80	0,02	882,04	0,017	882,04	0,811	878,87	0,010
E036-11h	771,18	0,02	767,45	0,004	767,45	0,154	767,96	0,010
E041-14h	954,55	0,02	954,55	0,004	954,55	0,135	936,27	0,010
E045-04f	972,68	0,02	951,65	0,009	950,16	2,829	940,41	0,010
E051-05e	592,18	0,02	587,05	0,014	587,05	0,563	591,15	0,010
E072-04f	339,80	0,03	332,27	0,038	324,67	39,612	304,86	0,120
E076C09r	1567,85	0,02	1506,95	0,053	1498,22	1,811	1529,24	0,092
E101-14s	1327,87	0,02	1312,42	0,028	1312,42	0,624	1261,32	0,124
E121-07c	1117,07	0,05	1096,19	0,069	1091,57	81,380	1117,07	0,060
E151-12c	1272,54	0,06	1238,26	0,085	1236,43	6,100	1248,11	0,267
E200-17c	1612,17	0,10	1569,12	0,139	1561,50	70,379	1571,29	0,170

Tabla 25. Resultados obtenidos de la heurística de Inserción Secuencial de Mole & Jameson simple con los mejores intercambios.

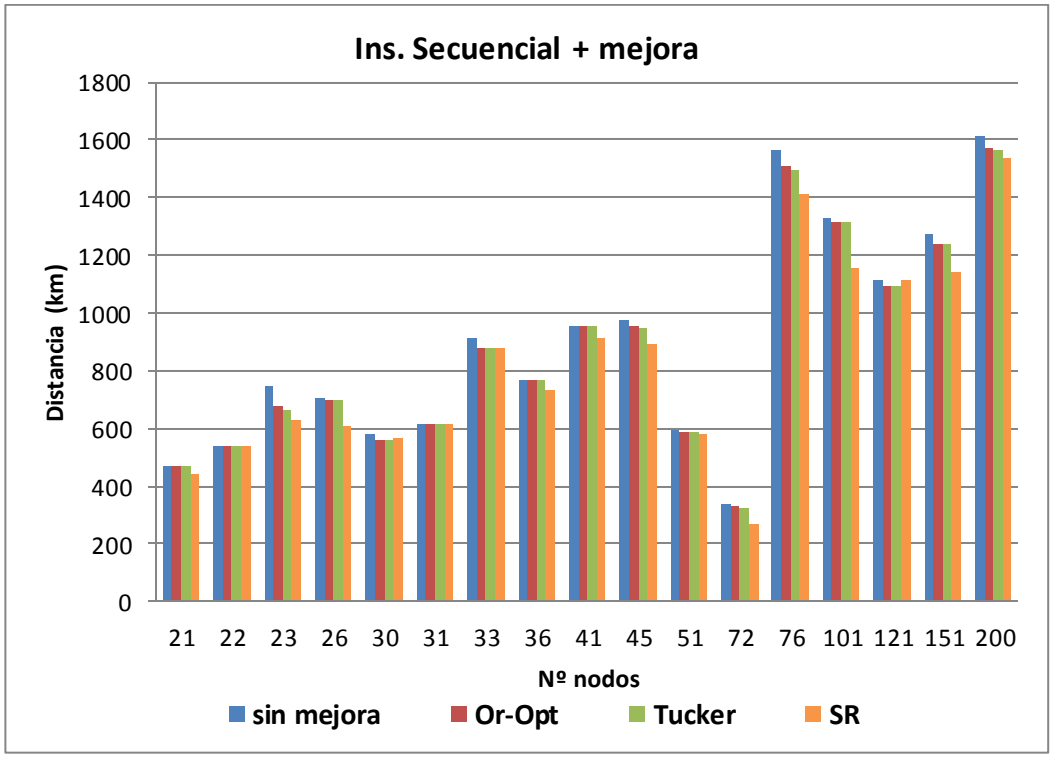

Gráfico 19. Distancia recorrida de la heurística de inserción secuencial simple mejorada en función de los métodos de mejora. [SR= String Relocate]

Suprimiendo el tiempo de ejecución del método de Tucker, se obtienen los tiempos computacionales del Gráfico 20.

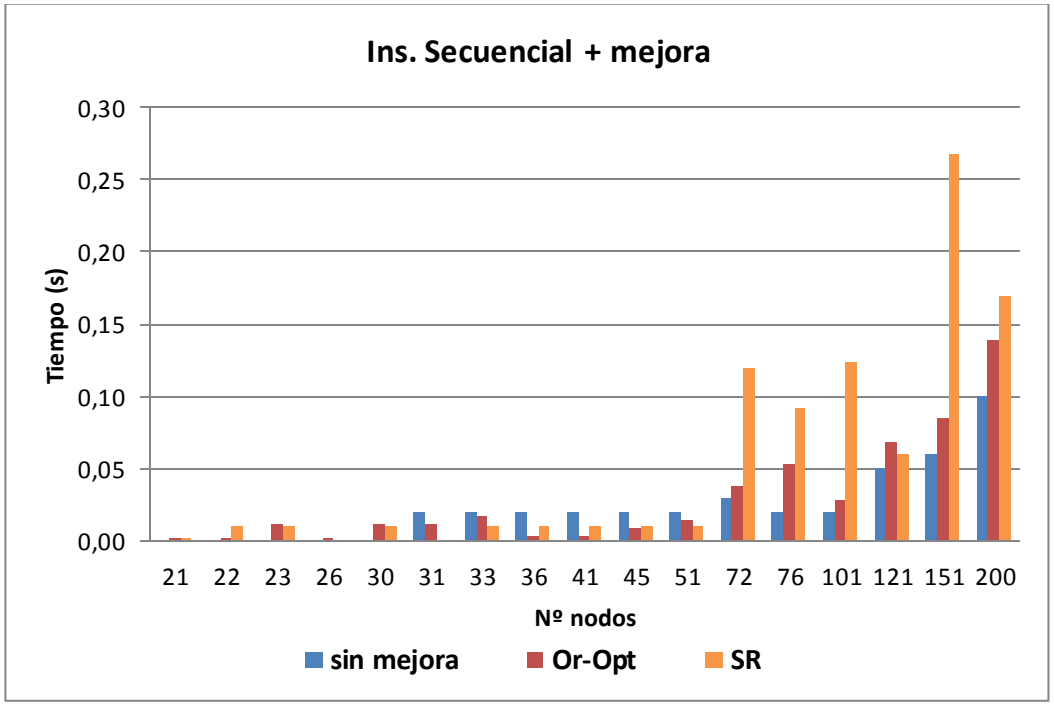

Gráfico 20. Tiempo de ejecución de la heurística de inserción secuencial simple mejorada en función de los métodos de mejora. [SR= String Relocate]

Se puede afirmar por tanto, que el método de String Relocate es el que ofrece soluciones de mayor calidad en el 58,82% de los modelos, a la vez que se consigue reducir la distancia inicial en un 2,54% (Tabla 26).

Método	Mejor solución	Variación de mejora
Or-Opt	11,76%	1,98%
Tucker	41,18%	2,30%
SR	58,82%	2,54%

Tabla 26. Frecuencia relativa de la mejor solución y la mejora media producida para cada método.

Una buena propuesta a realizar después de haber empleado el método de String Relocate, sería aplicar un método de mejora interna de las propias rutas para reordenar aquellas en las que se ha insertado un nodo.

CAPÍTULO 5. METAHEURÍSTICAS

5.1. Contextualización

Las metaheurísticas son el desarrollo más reciente dentro de los métodos aproximados para resolver problemas complejos de optimización combinatoria. Se han desarrollado y extendido rápidamente desde comienzos de los años 80 para dar respuesta a una gran variedad de problemas dentro de las áreas de negocio, comercio, ingeniería, industria y muchas otras más. Dichas técnicas surgen por la necesidad de resolver aquellos problemas donde los métodos exactos y los heurísticos fallan en términos de eficiencia y eficacia.

La definición de metaheurística enunciada por Osman y Kelly en "*Meta-Heuristics: Theory and Applications*" (1996) es la siguiente:

"Una metaheurística es un procedimiento iterativo con una estructura y una reglas generales de funcionamiento que lo caracterizan, que guía un método heurístico subordinado combinando inteligentemente diversos conceptos para explorar los espacios de búsqueda mediante estrategias aprendidas para conseguir soluciones quasi- óptimas de manera eficiente".

Además, estas nuevas técnicas permiten empeorar la solución e incluso obtener soluciones intermedias no factibles durante el proceso de búsqueda con el propósito de no quedarse atrapado en un óptimo local y tratar de encontrar óptimos locales de mejor calidad o el óptimo global. Esta ventaja en la calidad de la solución final con respecto a las heurísticas clásicas se produce gracias a una mejor exploración del espacio de soluciones, pero con el inconveniente de incurrir en tiempos de ejecución mayores.

Algunas de las metaheurísticas más importantes que se aplican a los VRP son las siguientes:

- Recocido Simulado (Simulated Annealing)

El Recocido Simulado es un método iterativo de búsqueda por entornos que hace uso de conceptos derivados de la Termodinámica.

Dicho proceso se detallará en profundidad más adelante en el apartado 5.3.

- GRASP

El método GRASP está constituido por una primera fase de construcción y por otra de búsqueda local. En la primera se establece una solución, que será mejorada en la segunda mediante algún procedimiento de intercambios.

Dicho proceso se detallará en profundidad más adelante en el apartado 5.3.

- Búsqueda Tabú (Tabú Search)

La búsqueda Tabú es un tipo de búsqueda por entornos que, según Glover (1994), tiene como fin guiar un algoritmo heurístico de búsqueda local para explorar el espacio de soluciones más allá del óptimo local. Además, aceptan moverse a soluciones peores que la actual para poder así escapar de los óptimos locales y continuar estratégicamente la búsqueda de soluciones de una mayor calidad.

En tanto y cuanto el proceso sigue operando, el método, para evitar volver a un óptimo local ya visitado, clasifica los últimos movimientos realizados como "movimientos tabú", los cuales no eran posibles repetir durante un cierto horizonte temporal. Por lo tanto, la Búsqueda Tabú mantiene una memoria de una serie de eventos relevantes del pasado para que, posteriormente, pueda alterar el entorno de la solución actual e influir en el siguiente espacio de búsqueda.

- Algoritmos Genéticos

Los Algoritmos Genéticos se basan en conservar un conjunto de soluciones lo suficientemente heterogéneo como para cubrir una gran parte del espacio de soluciones.

Goldberg (1989) los definió como "técnicas de búsqueda basadas en la mecánica de la selección natural y la genética", por lo que básicamente, simulan el proceso evolutivo de las especies.

La analogía la estableció por primera vez Holland (1975) al asociar cada individuo de una población con una posible solución y codificarla como un vector binario. En cada una de las iteraciones, se aplican unos operadores evolutivos (selección, cruzamiento y mutación) que combinan y modifican a los individuos de la población creándose una nueva.

- Colonia de Hormigas (Ant Systems)

La estrategia empleada por las colonias de hormigas a la hora de buscar alimentos fue la fuente de inspiración de Colorni et al. (1992) para desarrollar los denominados Sistemas de Hormigas.

El símil proviene de la forma en la que las hormigas buscan comida. Tan pronto como una de ellas encuentra alimento, esta evalúa la fuente y de camino a casa deja un rastro de feromona, cuya cantidad depende de la calidad del alimento y la distancia a este. Si las hormigas no detectan dicha sustancia se mueven de forma aleatoria, pero si la perciben es muy probable que se desplacen siguiendo la señal, lo cual a su vez aumenta la cantidad de la feromona. La expulsión de esta sustancia sirve para poder guiar a otras hormigas hacia una buena fuente de alimento.

- Redes Neuronales (Neural Networks)

El trabajo de Hopfield and Tank (1985) fue fundamental para el desarrollo de estos modelos inspirados en la forma en la que funciona el sistema nervioso biológico.

Las Redes Neuronales son sistemas compuestos por unidades interconectadas entre ellas que colaboran entre sí para producir un estímulo de salida. En particular, estos modelos pueden aprender de la experiencia, y por tanto, generar salidas razonables a partir de ajustes incrementales durante el proceso.

Según Taillard et al. (1998), la Búsqueda Tabú, Algoritmos Genéticos y Colonias de Hormigas son métodos que, mientras el proceso continua, guardan información de las soluciones encontradas y la usan para obtener soluciones mejores.

5.2. GRASP

5.2.1. Descripción

La metaheurística GRASP fue desarrollada originalmente por Feo y Resende (1989) y se presenta como un método multi-arranque de los más exitosos diseñado para resolver problemas difíciles de optimización combinatoria. Para cada iteración, el método opera en dos fases:

- Fase constructiva. Se genera una solución inicial, que no tiene por qué ser un óptimo local.
- Fase de búsqueda local. A partir de la solución inicial proporcionada, se trata de examinar el vecindario de esta para poder encontrar un óptimo local. Pueden aplicarse los métodos de k-opt, Tucker o entre-rutas entre otros.

Este proceso se repite un número predeterminado de veces, de tal forma que se vaya guardando la mejor solución encontrada hasta que se cumple con la condición de parada y el algoritmo termina.

GRASP es el acrónimo de Greedy Randomized Adaptive Search Procedures, en español Procedimientos de Búsqueda Ávidos, Aleatorizados y Adaptativos, ya que cada una de los adjetivos del nombre caracteriza esta metaheurística de la siguiente forma:

- Greedy. Como ya se comentó en el capítulo 3, los métodos greedy son estrategias voraces que en cada iteración seleccionan el mejor valor para un criterio en cuestión. También se les conoce como algoritmos miopes, puesto que para cada iteración solo tiene en cuenta la situación presente y no lo que podría suceder en sucesos posteriores. Se utilizan para una gran variedad de problemas, proporcionando en ocasiones especiales soluciones óptimas, en otras soluciones buenas y en la mayoría soluciones mediocres.

- Randomized. Un algoritmo está aleatorizado cuando su respuesta viene determinada tanto por los valores de entrada como por los que se generan internamente en el proceso. Estas estrategias aleatorizadas son útiles cuando existen diferentes posibilidades a la hora de caracterizar el problema, pero no se puede determinar de forma exacta cuál es la mejor de todas ellas. Esta aleatoriedad puede introducirse en el algoritmo de diversas formas como ya se comentó en el capítulo 3.

- Adaptive. Este término proviene de su uso en los campos de ingeniería de sistemas de control, donde se requieren modelos capaces de adaptarse a las situaciones en continua evolución.

Este método presenta una de las características fundamentales de los problemas de optimización como lo es el principio de diversificación e intensificación. El primero de ellos se realiza al introducir de forma aleatoria los nodos de inicio en las rutas de tal forma que no se comience siempre con los mismos nodos; y el segundo al realizarse la mejora de la solución inicial.

A pesar de su corta historia, se ha podido demostrar la potencialidad y progresión de futuro de este método en comparación con otras metaheurísticas de mayor bagaje histórico. Además, ha sido aplicado a una gran variedad de campos, como por ejemplo, secuenciación, programación y planificación, grafos, cobertura de conjuntos, problemas de localización, problemas de transporte, asignación cuadrática, lógica, etc.

5.2.2. Procedimiento

El pseudocódigo del método GRASP, donde en la primera fase de construcción se determinan las soluciones de élite o mejores soluciones y en la segunda fase se aplica el procedimiento de búsqueda local sobre dichas soluciones de élite es el siguiente:

```
N ← número máximo de iteraciones
SE ←número de soluciones de élite

forall (iter in 1..N ) do
        - Fase constructiva: creación de soluciones
        - Selección de las SE mejores soluciones
end-do
forall (e in 1..SE) do
        - Fase de búsqueda local: para cada una de las SE soluciones, aplicar
        método de mejora local
        - Selección de la mejor solución.
end-do
```

5.3. Simulated Annealing

5.3.1. Descripción

El algoritmo de Simulated Annealing (SA), en español Recocido Simulado, surgió en la década de los años 80 como una nueva estrategia heurística utilizada para la resolución de problemas complejos combinatorios al establecerse una analogía con el proceso de un campo científico totalmente diferente como lo es la Termodinámica.

El Recocido Simulado es un método de búsqueda por entornos, donde debido a la aleatorización en la selección de la siguiente solución, se reduce la probabilidad de quedar atrapado en un óptimo local, llegando a aceptar soluciones peores. Además, se ha llegado a demostrar que este método es capaz de hallar la solución óptima tras un número infinito de veces en el peor de los casos.

Según Arts (1989) recocer es "un proceso término para obtener estados de baja energía en un sólido mediante un baño térmico". Este proceso físico primero reblandece el sólido calentándolo a una temperatura elevada, y luego se va enfriando paulatinamente mientras las partículas se van colocando por sí mismas en el "estado fundamental" del sólido. El sólido puede alcanzar el equilibrio térmico si el enfriamiento se produce lo suficientemente lento, pero en el caso de que se produzca un enfriamiento rápido, el sólido puede llegar a estados meta-estables en lugar de al estado fundamental donde las partículas forman retículas perfectas y se encuentran en el nivel energético más bajo posible.

Tiempo después y de forma independiente, Kirkpatrick et al (1983) y Cerny (1985), introdujeron los conceptos del recocido simulado dentro del ámbito de la optimización combinatoria. Por ejemplo, estableciendo analogías con los estados del sistema y las soluciones del problema, el estado fundamental con la solución óptima global o los estados meta-estables con los posibles óptimos locales.

El concepto físico que se tiene sobre la temperatura, no se le correspondería un significado real en el campo de la optimización, sino que habría que tomarlo como un parámetro a ajustar. De esta forma, se trata de encontrar una similitud en cuanto al proceso que sufren las moléculas al tratar de buscar el equilibrio energético a una determinada temperatura cuando estas partículas se disponen en los niveles energéticos siguiendo una distribución de Boltzmann. Dichos posibles movimientos se llevan a cabo si la energía disminuye, o en su caso, se pueden aceptar los movimientos con una probabilidad proporcional al factor de Boltzmann. Este mismo proceso es el

que se realiza para la optimización al establecer una temperatura inicial y aceptar solo los movimientos que mejoren la solución o admitir el paso a una solución peor con una cierta probabilidad (cada vez una probabilidad menor conforme se va acercando a la solución óptima). Es por tanto que, al disminuir la temperatura, se disminuye la probabilidad de aceptar soluciones peores a la solución actual.

Por consiguiente, la estrategia a utilizar será la de establecer un valor alto para la temperatura, de tal forma que al principio se permitan los cambios a soluciones peores cuando aún se está lejos del óptimo global, y al ir reduciendo la temperatura, la posibilidad de cambios hacia soluciones peores irá disminuyendo también.

El buen funcionamiento de la metaheurística va a depender en gran medida del acierto a la hora de diseñar la estructura del entorno, el patrón de enfriamiento y las estructuras de datos.

El algoritmo de Simulated Annealing no busca la mejor solución en el entorno $N(x_n)$ de la actual solución x_n, sino que selecciona aleatoriamente un candidato (solución x) de entre los que componen el entorno de la solución actual. Si la solución candidata es mejor que la actual $(F(x) \leq F(x_n))$, entonces esta es aceptada como solución actual; en caso contrario, la solución x puede pasar a ser la solución actual con una probabilidad $p(n)$, o bien $x_n$continua siendo la solución actual con probablidad $1 - p(n)$. Esta probabilidad $p(n)$ decrece según crezca la diferencia entre la solución candidata y actual $(F(x) - F(x_n))$.

Para hacer operativo el algoritmo es necesario tomar una serie de decisiones:

-Probabilidad de aceptación *p(n).* Siguiendo una analogía con la termodinámica, la distribución de Boltzmann suele ser la escogida:

$$p(n) = \exp(-\frac{F(x) - F(x_n)}{T(n)})$$

- Temperatura inicial. Se pueden utilizar diferentes fórmulas para determinar el valor de la temperatura inicial en función del tipo de problema, aunque a efectos prácticos, basta considerar una temperatura lo suficientemente alta como para admitir las diferentes soluciones del entorno al principio del proceso.

- Velocidad de enfriamiento o decrecimiento de la temperatura. Comenzando con una temperatura inicial T_o , esta se mantiene constante durante L pasos consecutivos con objeto de conseguir llegar a su estado estacionario para esa temperatura. Tras las L iteraciones, la temperatura disminuye siguiendo una distribución dada (geométrica, lineal, logarítmica…).

Para el modelo geométrico, la temperatura se actualiza en función de un multiplicador fijo $\alpha \in [0,1]$, aunque los valores típicos que suele tomar están entre 0,8 y 0,99:

$$T_{i+1} = \alpha\, T_i$$

Esto implica la configuración de los valores para los tres parámetros $T_o, \alpha\ y\ L$ definidos como la temperatura inicial, el ratio de enfriamiento y la longitud de *plateau*.

En la Figura 33 se muestra el esquema de la velocidad de enfriamiento geométrico.

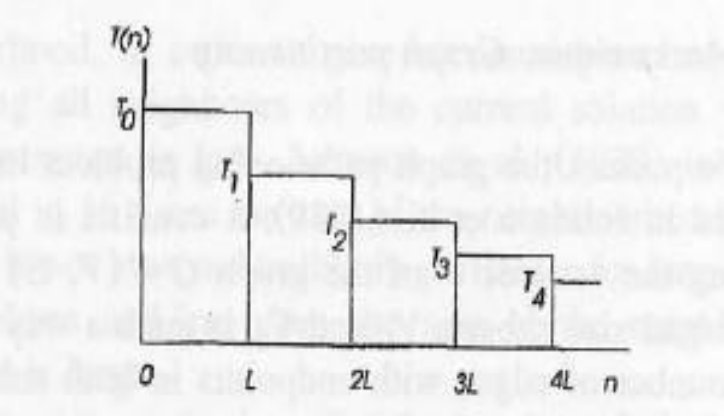

Figura 33. Esquema básico de velocidad de enfriamiento.
[Fuente: M. Pirlot / European Journal of Opeartional Research 92]

- Criterio de parada. Como criterios para terminar la implementación del algoritmo se pueden configurar otros parámetros que determinen un número máximo de iteraciones o un valor para la temperatura final.

5.3.2. Procedimiento

El algoritmo básico podría representarse de la siguiente forma:

T ←*To*
*F** ← $F(x_1)$
*x** ←x_1
while (no se cumpla condición de parada) do
- *generar aleatoriamente un* x *en el entorno* $N(x_n)$ *de* x_n
if ($F(x) < F*$) *then F** ← $F(x)$ *y x** ← *x* *end-if*
- *generar un p aleatorio entre [0,1]*

if ($F(x) \leq F(x_n)$ *or* $p \leq p(n)$) *then* x_{n+1} ← *x* *end-if*
- *se disminuye la temperatura*

end-do

5.4. Experimentación computacional

GRASP

A la hora de implementar la metaheurística GRASP, se han seleccionado los siguientes criterios:

- Como primera fase se utiliza la heurística de inserción secuencial aleatorizada con RCL para un parámetro fijo de K=8 y un número de iteraciones N=200.

- Soluciones de élite: se determina un valor de 15 para hallar las mejores soluciones con el método de inserción secuencial y ser estas sobre las que se aplique la siguiente fase.

- En la fase de mejora, se desarrollan sobre las 15 mejores soluciones el algoritmo de String Relocate para colocar nodos en otras rutas y después el intercambio Or-Opt como mejora dentro de las mismas.

Con la finalidad de analizar su eficiencia, se realiza una comparativa con el método aleatorizado con RCL, que sería la aplicación simplemente de la primera fase del método GRASP (Tabla 27, Gráfico 21 y Gráfico 22).

Modelo	RCL N=200 K=8		GRASP	
	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)
E021-06m	430,89	0,172	430,89	0,202
E022-06m	499,02	0,172	495,85	0,234
E023-05S	580,52	0,530	569,75	0,440
E031-09h	614,14	0,344	611,16	0,398
E033-05s	889,08	0,381	844,57	0,733
E036-11h	714,25	0,437	715,52	0.457
E041-14h	892,14	0,546	859,73	0,766
E045-04f	955,84	1,014	752,52	1,308
E051-05e	562,63	1,326	554,58	1,629
E072-04f	339,80	4,056	251,27	5,362
E076C09r	1406,44	2,995	1265,54	4,087
E101-14s	1202,60	3,947	1151,54	4,838
E121-07c	1103,25	9,860	1088,63	10,626
E151-12c	1195,34	12,338	1129,55	14,373
E200-17c	1476,40	21,064	1400,49	24,338

Tabla 27. Resultados obtenidos del método GRASP en comparación con la heurística de inserción secuencial aleatorizada con RCL.

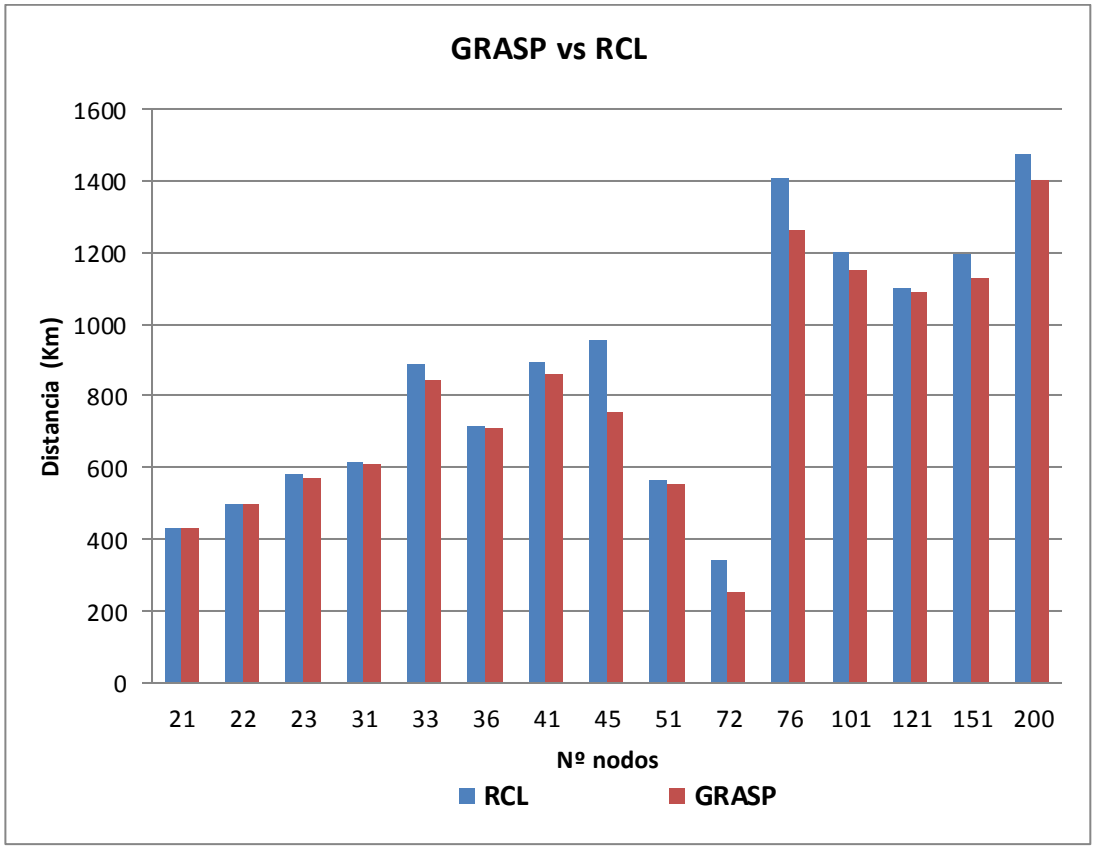

Gráfico 21. Distancia recorrida de la metaheurística GRASP en comparación con la heurística de inserción secuencial aleatorizada con RCL.

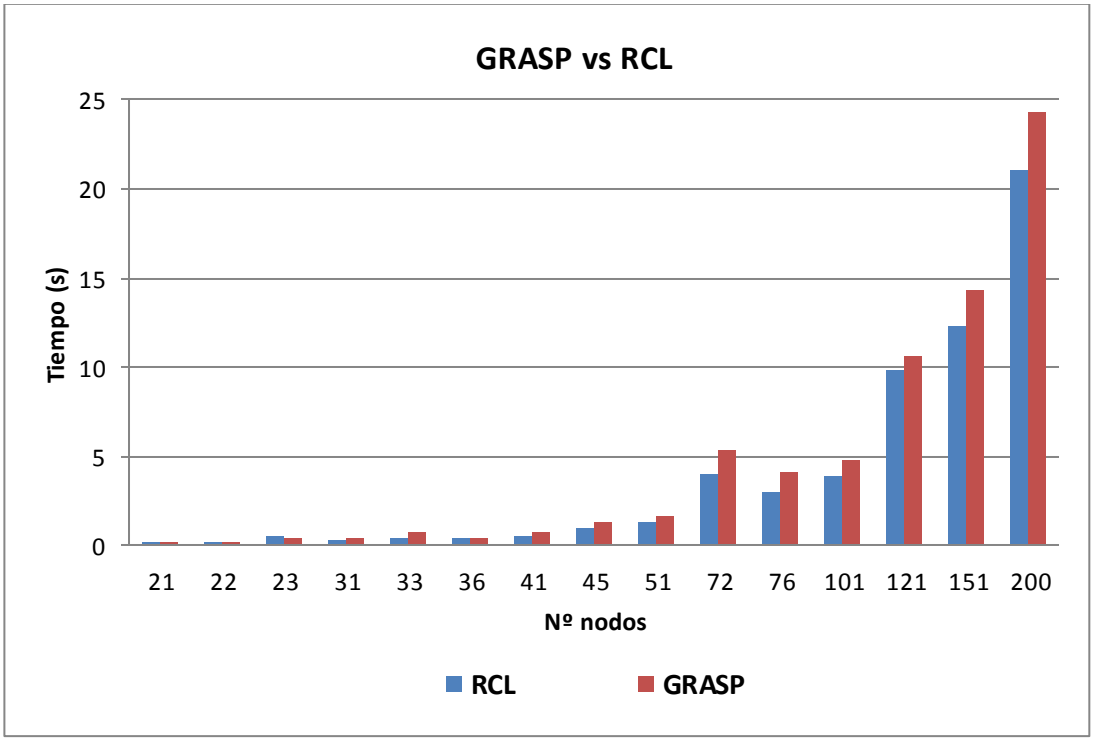

Gráfico 22. Tiempo de ejecución de la metaheurística GRASP en comparación con la heurística de inserción secuencial aleatorizada con RCL.

Para este supuesto en el que se aplica el método de String Relocate y el intercambio Or-Opt, la mejora media del método GRASP es de un 5,85% con respecto al procedimiento de la primera fase exclusivamente. Por otra parte, ambos métodos se ejecutan en un tiempo computacional similar, por lo que se evidencia aún con más motivo la eficiencia del método GRASP.

Además, este método podría incorporar la posibilidad de realizar más procesos de intercambio en el mismo procedimiento y así aumentar la calidad de las soluciones.

Simulated Annealing

Como ya se comentó anteriormente, los resultados del Simulated Annealing dependen notablemente de la estructura de entornos que se diseñe, en tanto y cuanto se acierte con los parámetros adecuados para lograr un resultado eficiente.

Se ha conseguido cuadrar los parámetros y obtener una mejora con este proceso en el 60% de los modelos utilizados (Tabla 29), para los cuales se han establecido los valores que aparecen en la Tabla 28 en función del número de nodos n.

	n ≤ 23	23< n <76	n ≥151
T inicial	100000	10000	10000
α	0,95	0,91	0,9
L	10000	100000	100000
Iter max	10000000		
Tolerancia	0, 0000000000001		

Tabla 28. Valor de los parámetros en función del número de nodos del problema. [T inicial: Temperatura inicial; α: tasa de decrecimiento de la temperatura; L: número de iteraciones fijas antes de cambiar de temperatura; Iter max: número de iteraciones máximas; Tolerancia: mínima temperatura]

Modelo	RCL N=200 K=8		Simulated Annealing (SA)	
	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)
E023-05S	633,74	1,185	568,56	200,22
E026-08m	630,01	0,858	607,65	264,53
E033-05s	889,08	0,381	846,54	183,03
E041-14h	892,14	0,546	863,59	336,03
E045-04f	955,84	1,014	757,82	213,55
E072-04f	339,80	4,056	299,32	287,39
E076C09r	1406,44	2,995	1340,65	262,05
E101-14s	1202,60	3,947	1154,47	334,65
E151-12c	1195,34	12,338	1098,67	414,09

Tabla 29. Resultados obtenidos del método Simulated Annealing en comparación con la heurística de inserción secuencial aleatorizada con RCL.

La calidad de las soluciones aumenta un 7,91% con el método de Simulated Annealing, pero supone un incremento en el tiempo de cálculo enorme, además de tener el acierto a la hora de establecer una correcta estructura del procedimiento.

La comparativa gráfica en cuanto a la solución final y al tiempo de cálculo entre los métodos mencionados se puede observar en el Gráfico 23 y Gráfico 24.

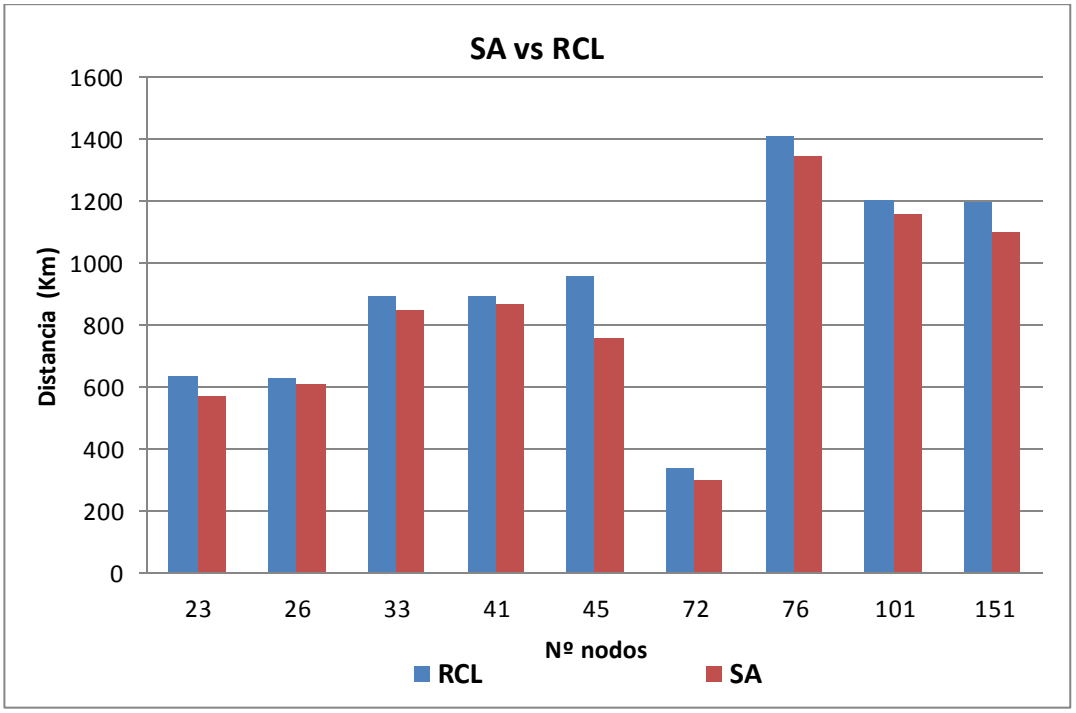

Gráfico 23. Distancia recorrida de la metaheurística SA en comparación con la heurística de inserción secuencial aleatorizada con RCL.

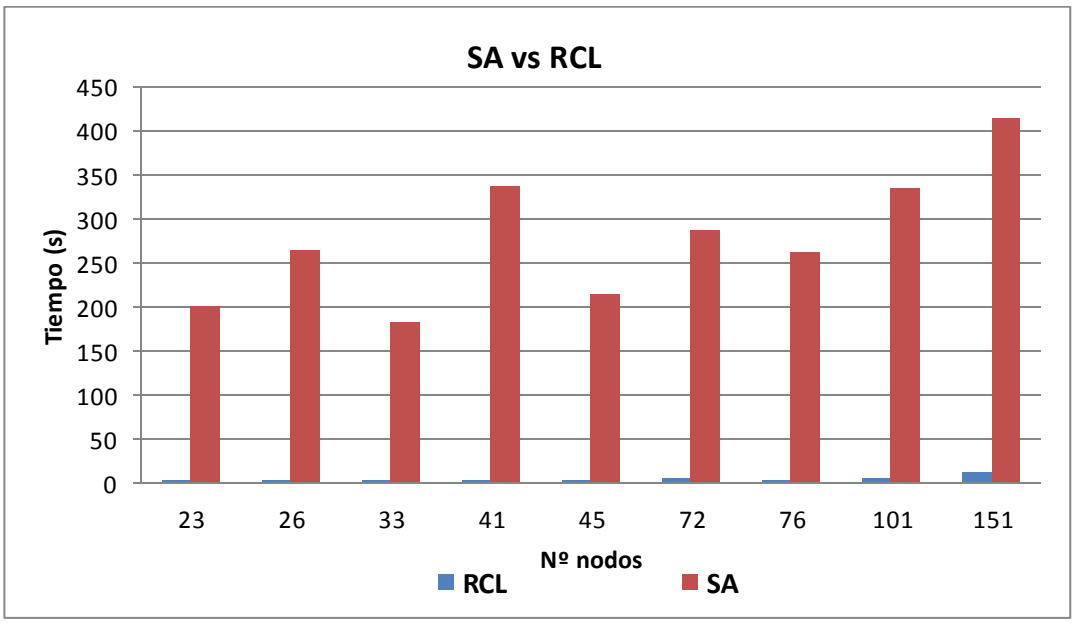

Gráfico 24. Tiempo de ejecución de la metaheurística SA en comparación con la heurística de inserción secuencial aleatorizada con RCL.

Resultados finales: metaheurísticas

Comparando ahora las dos metaheurísticas desarrolladas, se obtienen los resultados que se muestran en la Tabla 30, así como la equiparación gráficamente (Gráfico 25 y Gráfico 26).

Modelo	GRASP		Simulated Annealing (SA)	
	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)
E023-05S	569,75	0,440	568,56	200,22
E026-08m	611,16	0,512	607,65	264,53
E033-05s	844,57	0,733	846,54	183,03
E041-14h	859,73	0,766	863,59	336,03
E045-04f	752,52	1,308	757,82	213,55
E072-04f	251,27	5,362	299,32	287,39
E076C09r	1265,54	4,087	1340,65	262,05
E101-14s	1151,54	4,838	1154,47	334,65
E151-12c	1129,55	14,373	1098,67	414,09

Tabla 30. Comparativa de los resultados obtenidos con el método Simulated Annealing y con el método GRASP.

La mejor solución se halla con el método GRASP para el 66,17% de los modelos que se han conseguido mejorar la solución inicial con el método SA (Gráfico 25), con una diferencia de tiempo a favor del primero muy importante (Gráfico 26).

Destacándose además, la complejidad en cuanto a la determinación de los valores para los parámetros para el método SA.

Por todo lo comentado anteriormente, se puede afirmar que el método GRASP es el que ofrece mejores resultados en unos tiempos de ejecución muy asequibles, además de tener una línea de mejora notoria en cuanto a la posibilidad de incorporar más algoritmos de mejora de la solución inicial y del aumento de iteraciones para una mayor diversificación.

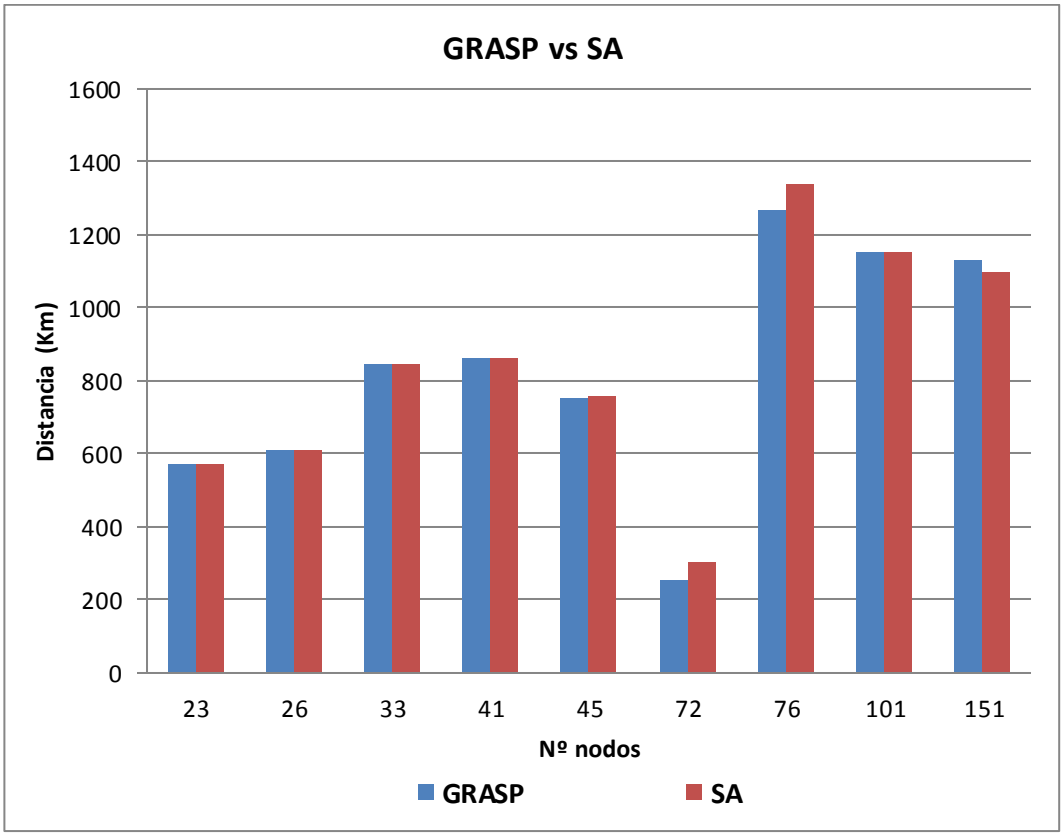

Gráfico 25. Comparativa de las distancias recorridas entre las metaheurísticas GRASP y SA

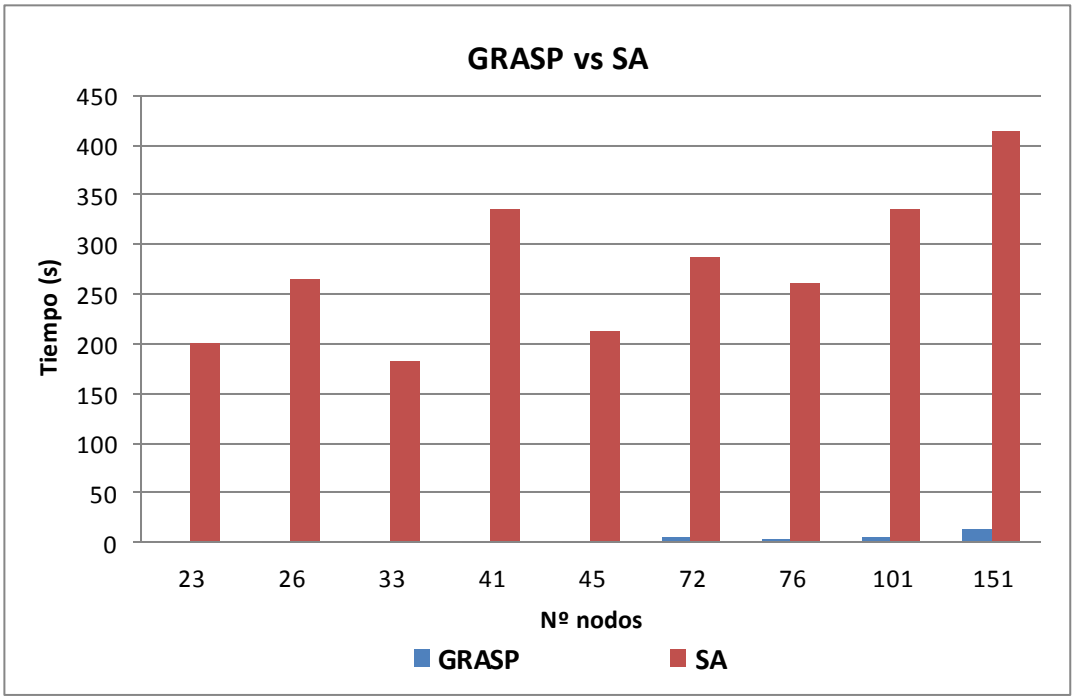

Gráfico 26. Comparativa de los tiempos de ejecución entre las metaheurísticas GRASP y SA

Resultados finales: Método exacto

Una vez se ha llegado a la conclusión de cuál es el mejor método aproximado, se pretende ahora comparar el método GRASP con el método exacto de Tucker-Miller-Zemlin.

Los datos han sido facilitados por Arturo Alonso Alonso de su Trabajo Fin de Grado *comparativa de modelos y solvers para la optimización de variantes de problemas de rutas de vehículos* (2016) y están calculados con el modelo de 2 índices basado en Tucker-Miller-Zemlin, con un tiempo de ejecución máximo de 200 segundos.

Modelo	GRASP		Tucker	
	Solución (Km)	Tiempo ejecución (s)	Solución (Km)	Tiempo ejecución (s)
E021-06m	430,89	0,202	430,89	200,60
E023-05s	569,75	0,440	568,56	200,10
E026-08m	620,25	0,411	609,76	199,15
E033-05s	844,57	0,733	1010,65	199,90
E041-14h	859,73	0,766	884,75	199,20
E045-04f	752,52	1,308	791,05	199,38
E051-05e	554,58	1,629	676,98	200,25
E072-04f	251,27	5,362	430,01	200,05
E076C09r	1265,54	4,087	1915,94	199,79
E101-14s	1151,54	4,838	1355,80	199,88
E121-07c	1088,63	10,626	1865,63	199,49
E151-12c	1129,55	14,373	1667,85	200,11
E200-17c	1400,49	24,338	2163,47	200,25

Tabla 31. Comparativa de los resultados obtenidos con los métodos GRASP y Tucker.

La mejor solución se obtiene en el 84,62% de los casos con el método GRASP y con una diferencia positiva del 18,47%. Además se refleja que para modelos pequeños ambos métodos ofrecen resultados similares, pero que a medida que va aumentando el número de nodos la diferencia del coste es mayor.

Señalar que la calidad de las soluciones para el método exacto está restringida por la limitación del tiempo de cálculo a 200 segundos, por lo que se evidencia el alto consumo de tiempo computacional.

La comparativa gráfica en cuanto a la solución final y al tiempo de cálculo entre los métodos mencionados se puede observar en el Gráfico 23 y Gráfico 24.

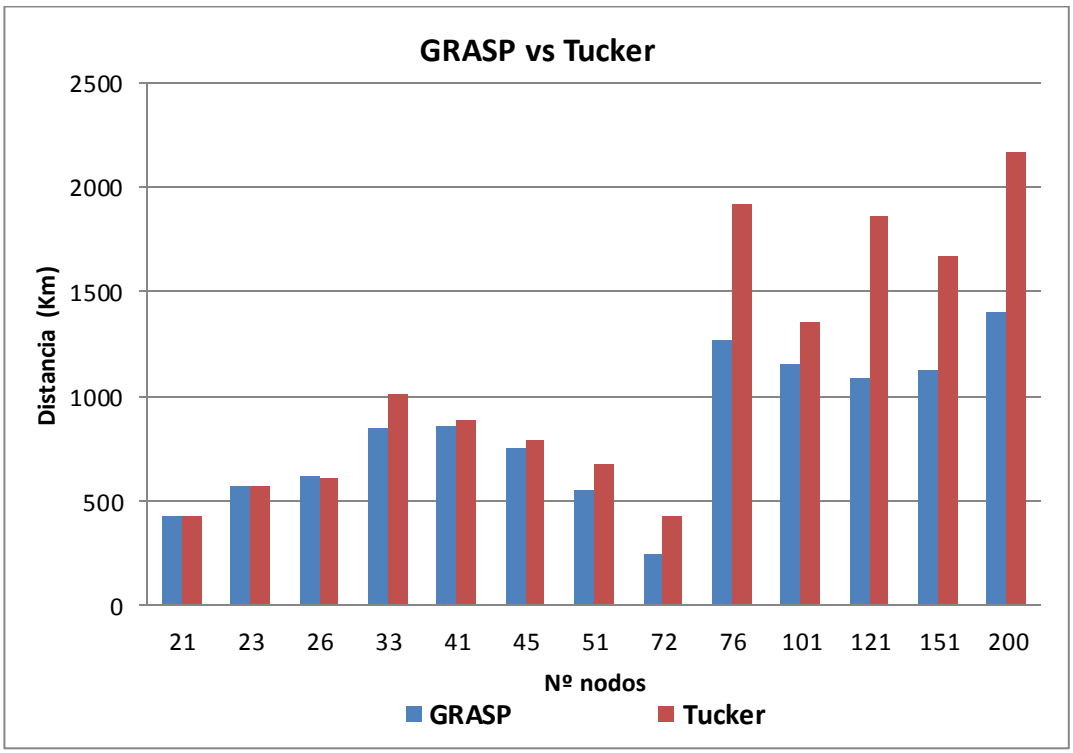

Gráfico 27. Comparativa de las distancias recorridas entre los métodos GRASP y Tucker.

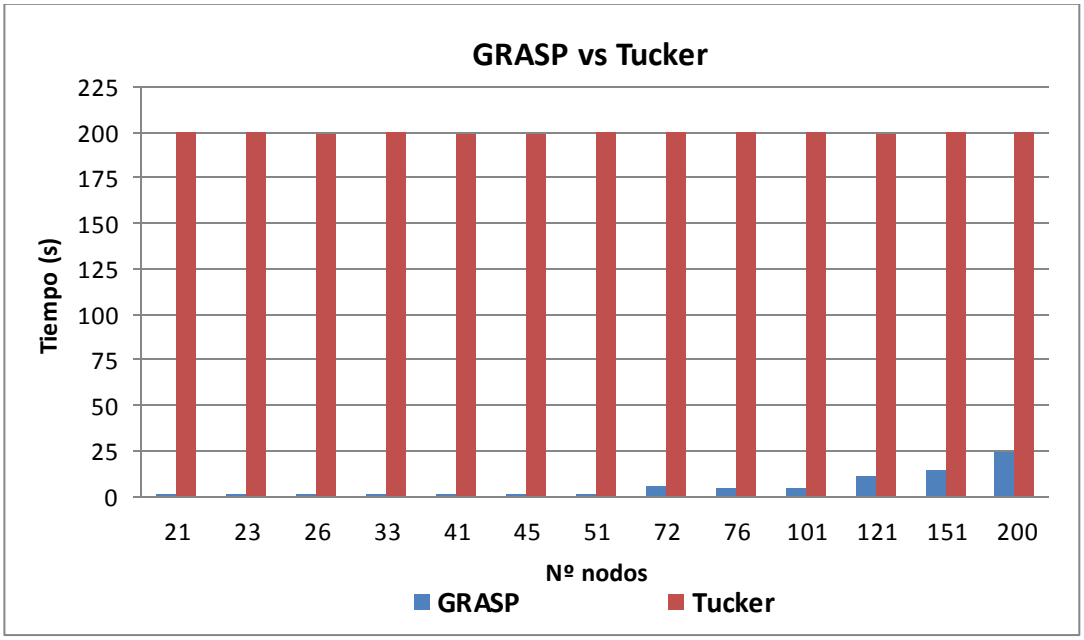

Gráfico 28. Comparativa de los tiempos de ejecución entre los métodos GRASP y Tucker.

CAPÍTULO 6. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

6.1. Conclusiones

El Problema de Rutas de Vehículos es uno de los problemas de optimización más importantes debido al gran número de aplicaciones reales a las que da cabida y las posibilidades que supone.

Para la resolución de este tipo de problemas, se pueden emplear métodos exactos y métodos aproximados. Los métodos exactos proporcionan soluciones óptimas con tiempos de ejecución elevados, mientras que los aproximados ofrecen soluciones buenas (aunque no óptimas) en tiempos inferiores.

A lo largo del presente documento se han ido evaluando diferentes algoritmos, tanto heurísticas y metaheurísticas como la comparación final con un método exacto, para concluir finalmente con el método GRASP como el que mejores resultados ha conseguido en términos de calidad y de tiempo de cálculo. Este método multi-start está constituido por un método greedy aleatorizado de la heurística de Inserción Secuencial, al ser la mejor evaluada, y por los diferentes intercambios que se pueden realizar como método de mejora.

Desde que se publicó la primera heurística para el VRP, muchas han sido las técnicas propuestas para su resolución. Los sucesivos análisis llevados a cabo en este Trabajo Fin de Grado ponen de manifiesto la alta capacidad de estos métodos heurísticos y metaheurísticos para dar respuesta a algunas necesidades instantáneas del mundo contemporáneo.

Todo ello se encuadra en un escenario donde las expectativas de los clientes han crecido notablemente, así como los productos del mercado. Además, la globalización de los mercados ha estimulado una aceleración en el comercio. Es por ello que las empresas se someten, cada vez más, a retos más dinámicos. Por consiguiente, el transporte, que ya es una función vital, tendrá aún una posición más estratégica para la industria del futuro.

6.2. Futuras líneas de trabajo

Según establecen Toth y Vigo (2002), las heurísticas clásicas cosechan valores de las soluciones finales entre el 2% y el 10% por encima del óptimo (o la mejor solución encontrada), mientras que las mejores metaheurísticas están normalmente entorno al 0,5%. En la Figura se muestra los diferentes métodos aproximados en relación con el tiempo de procesamiento y la diferencia con respecto al óptimo, así como su tendencia evolutiva.

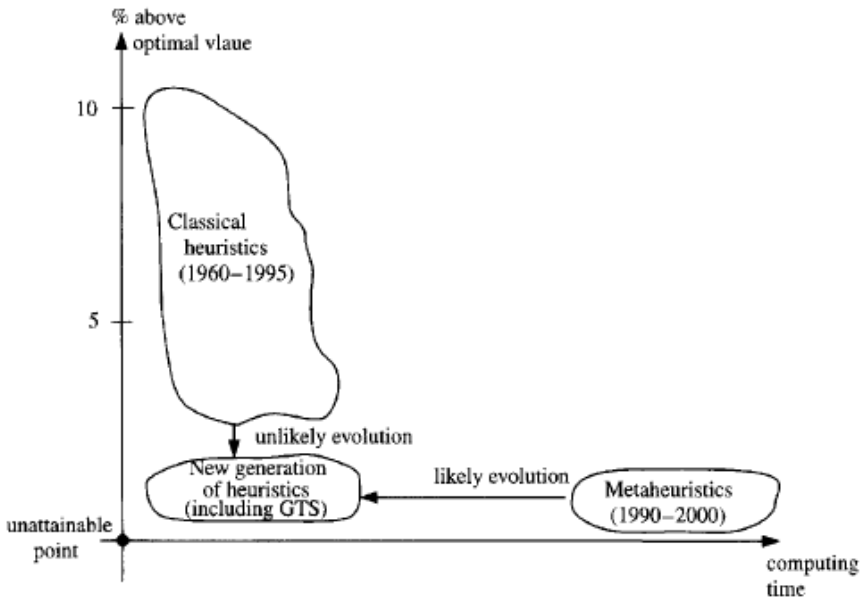

Figura 34. Evolución de las heurísticas para el VRP
[Fuente: Toth y Vigo (2002)]

La necesidad de disponer de herramientas que permitan ofrecer soluciones (aunque no sean óptimas) a problemas reales, permite augurar una continuidad en la tendencia observada.

Esta continuidad puede venir por un progreso en las mejores metaheurísticas disponibles, más que por esfuerzo en enfoques clásicos. El algoritmo *Granular Tabu Search* (GTS) propuesto por Toth y Vigo (2002) es un importante paso siguiendo esta dirección. La principal idea detrás del GTS radica en la observación de que los arcos más largos del gráfico tienen pocas probabilidades de pertenecer a la solución óptima. Consigue evitar la mayoría de las ejecuciones innecesarias llevadas a cabo por los previos algoritmos de Búsqueda Tabú.

Asimismo, otra metaheurística que se ha aplicado con éxito es la denominada Búsqueda Dispersa (en inglés *Scatter Search*), propuesta por Glover (1977). Esta se presenta como un algoritmo evolutivo o basado en poblaciones que, aunque muestra similitudes con los Algoritmos Genéticos, difiere de estos en los principios fundamentales como por ejemplo, el uso de estrategias sistemáticas en lugar de aleatorias.

Por otra parte, en este documento se ha tratado exclusivamente el Problema de Rutas de Vehículos Capacitado, por lo que otras posibles líneas de futuro serían la aplicación de algoritmos heurísticos y metaheurísticos a las diferentes variantes del VRP (con ventanas de tiempo, retornos, recogidas y entregas, multi-depósito, etc.), y también para el VRP asimétrico y el VRP con flota heterogénea.

BIBLIOGRAFÍA

[1] A GRASP Algorithm Based on New Randomized Heuristic for Vehicle Routing Problem. Abdesslem Layeb (2013).

[2] A guide to vehicle routing heuristics. Potvin, J. et al. Journal of the Operational Research Society. (2002).

[3] A heuristic algorithm for the Asymmetric Capacitated Vehicle Routing Problem. Daniele Vigo. European Journal of Operational Research 89 (1996).

[4] A Library of Local Search Heuristics for the Vehicle Routing Problem. Chris Groër (2010).

[5] A Path-Exchange-Type Local Search Algorithm for Vehicle Routing and its efficient search strategy. Narihiro Park et al. (2002). The Operations Research Society of Japan.

[6] A Review of Bio-Inspired Algorithms for Vehicle Routing. Jean-Yves Potvin (2008).

[7] Adaptation in Natural and artificial systems. Holland, J. The University of Michigan Press (1975)

[8] Adaptive memory programming: A unified view of metaheuristics. Éric D. Taillard et al. (2000).

[9] Algoritmo Simulated Annealing (Recocido simulado). J. Sáez Aguado.

[10] Algoritmos Heurísticos en Optimización Combinatoria. Rafael Martí. Departament d'Estadística i Investigació Operativa.

[11] An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a selection of problems with Vehicle-related, Customer-related, and Time-related Constraints Contents. Alex Van Breedam (1994).

[12] Aplicación de la metodología GRASP al problema de Rutificación de Vehículos. Paolo Priore et al/Dirección y Organización 48 (2012).

[13] Búsqueda Dispersa. Rafael Martí (2003). Artículo (Departamento de Estadística e Investigación Operativa, Universidad de Valencia).

[14] Combining 2-Opt, 3-Opt and 4-Opt, with K-swap-kick perturbations for the traveling salesman problem. Andrius Blazinskas, Alfonsas Misevicius.

[15] Comparativa de modelos y solvers para la optimización de variantes de problemas de rutas de vehículos Arturo Alonso Alonso. Trabajo Fin de Grado (2016)

[16] Diseño de metaheurísticos para problemas de rutas can flota heterogénea: GRASP. JOAQUÍN A. PACHECO (1999).

[17] Efficient Insertion Heuristics for Vehicle Routing and Schedulin Problems. A. Cambell and M. Savelsbergh (2004).

[18] Genetic Algorithms in Search, Optimization, and Machine Learning. Goldberg, D. Addison-Wesley Publishing Company, Inc., Reading, MA (1989)

[19] GRASP: Procedimientos de búsqueda miopes aleatorizados y adaptativos. Mauricio G. C. Resende y José Luis González Velarde (2003).

[20] Heurística 2-Opt para el problema TSP. J. Sáez Aguado.

[21] Heurísticas para Problemas de Ruteo de Vehículos. Alfredo Olivera (2004)

[22] Improvement heuristics for the Vehicle Routing Problem based on Simulated Annealing. Alex Van Breedam (1994).

[23] k-Interchange procedures for local search in a precedence-constrained routing problem. Harilaos N. PSARAFTIS (1982).

[24] Local Search. M. Pirlot / European Journal of Operational Research 92 (1996).

[25] Local Search Algorithms for Integrated Logistics. Sara Ceschia (2012). Ph.D. Thesis.

[26] Metaheuristic Search Concepts. A tutorial with Applications to Production and Logistics. Günther Zäpfel et al. (2010).

[26] Metastrategy simulated annealing and tabu search algoritms for the vehicle routing problem. Hassan Osman (1993).

[27] Multiple models and parallel solving with Mosel. Y. Colombani and S. Heipcke (2014).

[28] Multiple Phase Neighborhood Search-GRASP for the Capacitated Vehicle Routing Problem. Yannis Marinakis (2012).

[29] Problemas de rutas de vehículos: modelos, aplicaciones logísticas y métodos de resolución. Trabajo Fin de Grado. Ana Benito Quintanilla (2015).

[30] Problemas de rutas de vehículos. J. Sáez Aguado.

[31] Procedimientos de búsqueda local. J. Sáez Aguado.

[32] Procedimientos Metaheurísticos en Optimización Combinatoria. Rafael Martí.

[33] Simulated Annealing (SA). M. Pirlot / European Journal of Operational Research 92 (1996).

[34] Study of Greedy Search with Multiple Improvement Heuristics for Vehicle Routing Problems. Patrick Prosser and Paul Shaw. Department of Computer Science Research Report (1996).

[35] The Vehicle Routing Problem. Paolo Toth y Daniele Vigo (2002). *Universita degli Studi di Bologn.*

[36] Transportation Logistics. Part VI: VRP - improvement heuristics. R.F. Hartl, S.N. Parragh.

[37] Vehicle Routing Optimization Using Multiple Local Search Improvements. Juraj Fosin et al. (2014).

[38] Xpress-Mosel. Reference manual (2014).

Librería de datos:

[39] http://or.dei.unibo.it/library/vrplib-vehicle-routing-problem-library