Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Estadística

Título del Trabajo

Problema de Rutas de Vehículos con ventanas de tiempo

Autor:

Mario Villaizán Vallelado

Tutor/es:

Jesús Sáez Aguado

Agradecimientos

Al conjunto de profesores que a lo largo de estos años me han mostrado las bases del conocimiento.

A todos mis compañeros, en especial a los supervivientes que han resistido las muchas trabas que nos hemos encontrado.

A mis padres, que con su apoyo han logrado allanar un camino pedregoso.

A mi tutor, Jesús, que con cuya dedicación, esfuerzo y ayuda ha hecho posible este trabajo.

A todos ellos: Gracias

Índice general

Resumen: En el presente Trabajo de Fin de Grado, trataremos de obtener una solución para el problema de rutas de vehículos con ventanas de tiempo. Este problema nos lo encontramos cuando el distribuidor da la posibilidad al demandante de seleccionar el intervalo de tiempo en el que quiere recibir la mercancía. Afrontaremos su resolución mediante algorítmicos, los cuales nos proporcionan una aproximación a la solución óptima en un tiempo razonable.

Palabras Clave: Ruta de vehículos, ventanas de tiempo, heurísticas, metaheurísticas.

Abstract: In this document, we will try to obtain a solution to the problem of transporting goods with time windows. This problem is found when the dealer gives the applicant the possibility of selecting the time interval in which he wants to receive the merchandise. We will deal with their resolution from algorithmic models, which provide us with an approximation to the optimal solution in a reasonable time.

Key Word: Vehicle routing, time windows, heuristics, metaheuristics.

Introducción

En el presente proyecto se tratará el problema de rutas de vehículos con ventanas de tiempo y capacidad (VRPTW, *Vehicle Routing Problem with Time Windows*), buscando darle solución tanto de manera exacta como de manera aproximada. Para la comparación de los diferentes procedimientos de resolución, que se van a plantear a lo largo del trabajo, lo haremos de tres maneras distintas:

1. La cercanía del resultado frente a la solución exacta.
2. Los tiempos computacionales empleados.
3. La robustez que presenten dichas soluciones.

El problema VRPTW es una variante del problema CVRP (*Capacitated Vehicle Routing Problem*), que podrían ser considerados como una extensión del problema TSP (*Travelling Salesman Problem*) el cual fue formulado por primera vez en 1930 y es uno de los problemas de optimización combinatoria más estudiados. El TSP es uno de los muchos problemas considerados como *NP-hard*, los problemas que han sido categorizados de este modo es debido a que no se conoce ningún método de resolución exacta que sea de orden polinómico.

El problema TSP aparece en una gran cantidad de campos, pues lo que intenta es dar respuesta a la pregunta ¿cuál es el orden en el que se deben de visitar un conjunto de puntos, de modo que el costo total sea el menor posible y regresando al punto de partida? Dada la anterior pregunta parece que el problema únicamente tendría interés en el ámbito de la logística, en el transporte y distribución de materias primas o productos terminados, pero también nos resulta de utilidad si buscamos minimizar cuál tiene que ser el recorrido de un brazo mecánico para la producción de cierta pieza, o cuál es

el camino que tiene que realizar cierto cartero para el reparto de la correspondencia, entre otros. Una modificación de dicho problema nos la encontramos a la hora de intentar secuenciar el ADN[1]. Una recopilación más extensa del conjunto de situaciones en las que ha sido empleado, nos las encontramos en el libro publicado por William Cook en el 2012[1].

La extensión del TSP que se va a estudiar añade la complejidad de que los puntos deben ser visitados dentro de un intervalo de tiempo. En los conjuntos de datos que van a ser empleados, el requisito que se nos pide es que el inicio de la tarea se encuentre en ese intervalo, pudiéndose finalizar la tarea aún cuando el intervalo de tiempo haya sido superado. A esta complejidad hay que añadirle que los vehículos empleados tienen una capacidad finita, es decir, no se puede transportar en cada viaje más de la capacidad total del vehículo. El problema va a resolverse considerándose que la flota de la que se dispone es homogénea, todos los vehículos tienen las mismas características.

Los problemas de enrutamiento del vehículo surgen cuando un conjunto de vehículos está disponible para servir un conjunto de solicitudes de transporte. Cada solicitud especifica una o más ubicaciones que deben ser visitadas por un mismo vehículo y varias restricciones que restringen la manera de visitarlas. Los más comunes, tanto en la práctica como en la literatura, se refieren a la capacidad y el tiempo. Tendremos restricción de capacidad cuando las ubicaciones de los clientes requieren que los vehículos recojan o entreguen cargas y la cantidad que cada vehículo puede llevar al mismo tiempo es limitada. Las limitaciones de tiempo se presentan cuando el servicio solo puede realizarse en un intervalo de tiempo fijado, este intervalo se denomina ventana de tiempo.

El problema que vamos a analizar, trata de encontrar un conjunto de rutas que visiten cada solicitud una vez y sólo una vez a un costo mínimo. Cada ruta, comenzando y terminando en un lugar específico, llamado depósito, tiene que satisfacer las restricciones de capacidad y de ventana de tiempo. Las ventanas de tiempo las tendremos definidas del siguiente modo, dado un intervalo: $[e_i, l_i]$, el vehículo que dé servicio al cliente i deberá llegar antes de l_i, pudiendo llegar antes de e_i, en cuyo caso deberá esperar hasta que dé comienzo la ventana de tiempo. Una vez que el servicio ha comenzado, debe transcurrir un tiempo dado antes de que el vehículo pueda abandonar al cliente i, esto es conocido como el tiempo de servicio: s_i.

En las diferentes publicaciones dedicadas al problema que vamos a tratar,

[1]El problema de ensamblado de fragmentos de ADN (FAP, *fragment assembly problem)*

aparecen tres objetivos diferentes, todos ellos buscan que el resultado final sea el mínimo posible:

1. Minimizar el número de vehículos necesarios.
2. Minimizar el tiempo total del viaje.
3. Minimizar los tiempos parciales que componen cada ruta.

En la siguiente figura vemos la relación entre los diferentes problemas de rutas de vehículos. Paolo Toth y Daniele Vigo realizan una descripción de cada uno de ellos en su publicación de 2001[2].

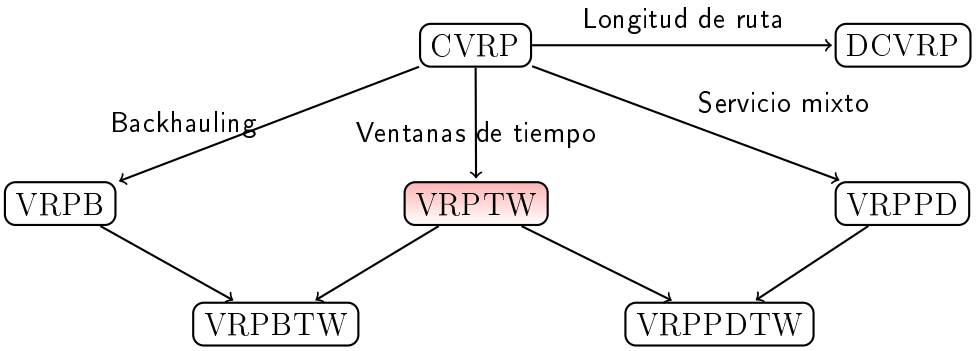


Figura 1: Los problemas básicos de la clase VRP y sus interconexiones[2]

Una descripción de los diferentes procedimientos que pueden ser aplicados para la resolución del problema de rutas de vehículos con capacidades (CVRP), nos la encontramos en el trabajo presentado por Alba Fernández Hernández en 2016[3].

En definitiva, el objetivo de este proyecto es la implementación de algunas de las resoluciones más famosas que a lo largo de los años se han propuesto con el objetivo de resolver el problema en VRPTW. Proporcionando una herramienta de aplicación de dichas técnicas de un modo intuitivo, sin tener la necesidad de conocerlas en profundidad.

Es por tanto un proyecto con el que mejorar la visión empresarial de una empresa de transporte, mejorando por tanto los beneficios que se obtienen, al minimizar el coste que se produce a la hora de realizar el reparto de mercancías, añadiendo además una mejor calidad en el servicio. Otro punto a tener en cuenta es la mejora de la responsabilidad social corporativa, puesto

que al buscar la empresa recorrer el menor número de kilómetros, se evita una emisión de gases por parte de sus vehículos a la atmósfera y consumiendo un menor número de materias primas no renovables.

El aspecto más innovador de este proyecto, lo encontramos en la implementación de una nueva heurística de construcción, no descrita hasta el momento en ningún artículo, obteniéndose mediante esta heurística una buena aproximación a la ruta óptima. El procedimiento heurístico, busca la simplificación de los algoritmos de construcción de rutas dependientes de parámetros y tratando de aunar procedimientos clásicos.

Estructura del documento

A lo largo del proyecto realizaremos un estudio progresivo en 6 capítulos, avanzando en los diferentes procedimientos que a lo largo del tiempo se han empleado para la resolución del problema al que intentamos dar solución.

En el capítulo actual se realiza una introducción general, plantearemos el interés del problema al que pretendemos dar solución y las diferentes tecnologías empleadas a lo largo de todo el estudio realizado.

En el capítulo 1 plantearemos el modelo exacto, el cual sería aplicado si quisiéramos obtener la mejor solución. Tal y como veremos, el tiempo requerido para la obtención de la mejor solución es inasumible, lo que nos obligará a realizar una aproximación a dicha solución.

En el capítulo 2 desarrollaremos un conjunto de heurísticas, las cuales buscan una aproximación determinista a la solución óptima. Estos procedimientos, comienzan desde una ruta no óptima, a la que se le aplican una serie de intercambios de clientes que reducen el costo global.

En el capítulo 3 desarrollaremos un conjunto de procedimientos que eliminen el carácter determinista de los procedimientos desarrollados en el anterior capítulo. Estos algoritmos buscan, mediante procedimientos aleatorios, aumentar el espacio de búsqueda de soluciones, logrando de este modo que realizando varias repeticiones se logre en la gran mayoría de las ocasiones, una mejor aproximación a la solución óptima.

En el capítulo 4, recopilaremos el conjunto de resultados computacionales que han sido obtenidos mediante los procedimientos implementados. El

conjunto de resultados, serán mostrados mediante tablas y gráficos, con los que facilitar la interpretación de las diferentes rutas.

En el capítulo 5 obtendremos el conjunto de conclusiones que se han ido obteniendo en el resto de capítulos, proponiendo un conjunto de futuras vías de investigación.

Herramientas empleadas

El conjunto de herramientas que han sido empleados para la obtención de los diferentes resultados podríamos categorizarlos en tres apartados:

- Obtención de una solución exacta: Xpress-Mosel, empleando XPRESS IVE v.7.8.
- Aproximación heurística y metaherística a la solución exacta: algoritmos implementados en R[4].
- Gráficos: librerías de R.

El conjunto de implementaciones ha sido realizado en un ordenador con las siguientes características: Intel(R) Core(TM) i7-4790 CPU 3.60 GHz con 16 GB de RAM.

Xpress-Mosel es un entorno para el modelado y resolución de problemas de optimización. Para este objetivo, se proporciona un lenguaje que es a la vez un modelador y un lenguaje de programación. La originalidad del idioma Mosel es que no hay separación entre una declaración de modelado (por ejemplo, la que se declara una variable de decisión o expresar una restricción) y un procedimiento que realmente resuelve el problema (por ejemplo, llamar a un comando de optimización). Gracias a esta sinergia, se puede programar un algoritmo de solución compleja mediante la combinación de modelado y la solución de declaraciones.

Cada categoría de problema viene con sus propios tipos particulares de variables y restricciones y un solo tipo de solucionador no puede ser eficiente en todos los casos. Para tener esto en cuenta, el sistema Mosel no integra un solucionador de forma predeterminada, pero ofrece una interfaz dinámica de solucionadores externos proporcionados en forma de módulos. En cada

módulo viene un solucionador con su propio conjunto de procedimientos y funciones que extiende directamente el vocabulario y las capacidades del lenguaje de Mosel. El vínculo entre el lenguaje Mosel y un módulo de resolución es alcanzado en el nivel de memoria y no requiere ninguna modificación del núcleo sistema.

Esta arquitectura abierta también puede ser utilizada como un medio para conectar Mosel a otros softwares. Por ejemplo, un módulo podría definir la funcionalidad necesaria para comunicarse con una base de datos específica.

Las tareas de modelado y resolución, por lo general, no son las únicas operaciones realizadas por una Aplicación de software. Esta es la razón de que se proporcione el medio Xpress-Mosel ya sea en bibliotecas o como un programa independiente.[2]

Por otro lado, el conjunto de heurísticas y de metaheurísticas que se han ido proponiendo a lo largo de la historia por los diferentes autores fue implementado en R, y las cuales quedarán alojadas en el repositorio público de código propio para su difusión.

R es una implementación de software libre del lenguaje S pero con soporte de alcance estático. Se trata de uno de los lenguajes más utilizados en investigación por la comunidad estadística, siendo además muy popular en el campo de la minería de datos, la investigación biomédica, la bioinformática y las matemáticas financieras. A esto contribuye la posibilidad de cargar diferentes bibliotecas o paquetes con funcionalidades de cálculo y gráficas.

R es parte del sistema GNU y se distribuye bajo la licencia GNU GPL. Está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux.

Fue desarrollado inicialmente por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en 1993. Sin embargo, si se remonta a sus bases iniciales, puede decirse que se inició en los Bell Laboratories de AT&T y ahora Alcatel-Lucent en Nueva Jersey con el lenguaje S. Este último, un sistema para el análisis de datos desarrollado por John Chambers, Rick Becker, y colaboradores diferentes desde finales de 1970. La historia desde este punto es prácticamente la del lenguaje S. Los diseñadores iniciales, Gentleman y Ihaka, combinaron las fortalezas de dos lenguajes existentes, S y Scheme. En sus propias palabras: "El lenguaje

[2]Fragmento traducido del manual alojado en: `http://home.deib.polimi.it/malucell/didattica/appunti/mosel/mosel-language_reference.pdf`

resultante es muy similar en apariencia a S, pero en el uso de fondo y la semántica es derivado desde Scheme". El resultado se llamó R "en parte al reconocimiento de la influencia de S y en parte para hacer gala de sus propios logros". Su desarrollo actual es responsabilidad del *R Development Core Team*[3].

La decisión de desarrollar el producto mediante esta tecnología, radica en la sencillez que proporcionan las librerías que contiene el lenguaje para facilitar una visión de usuario, sin que sea necesario tener un gran conocimiento intrínseco del problema a la hora de realizar una buena experiencia de usuario.

Para el desarrollo de los diferentes algoritmos se han empleado un conjunto diverso de librerías. Las librerías *data.table*[5] y *matrixStats*[6] para el manejo de los datos; las librerías *ggplot2*[7], *DT*[8] y *shiny*[9] para la visualización de los resultados; la librería *log4r*[10] para el control del flujo de la aplicación.

[3]Fragmento extraído de wikipedia: `https://es.wikipedia.org/wiki/R_(lenguaje_de_programaci%C3%B3n)`

Capítulo 1

Modelo exacto, la mejor solución

Siempre que nos enfrentamos ante un problema, nuestro objetivo debería ser el de lograr la mejor solución. Para el problema VRPTW podríamos llegar a la solución óptima resolviendo un modelo matemático de restricciones lineales. Roberto Cordone y Roberto Wolfler Calvo publicaron en 2001[11] uno de los muchos modelos matemáticos que dan respuesta exacta al problema. Definieron el modelo del siguiente modo:

Sea $G = (N, A)$ un grafo dirigido, donde $N = 1, ..., n$ es el conjunto de clientes y $A = \{(i, j) \colon i, j \in N,\ i \neq j\}$ es el conjunto de arcos posibles que los pueden interconectar. El nodo 1 representa el depósito y $N' = \{2, ..., n\}$ corresponde al conjunto de clientes a visitar. Cada arco (i, j) tiene asociado un coste de viaje $c_{ij} \geq 0$ y un tiempo de recorrido $t_{ij} \geq 0$. Cada cliente i tiene una petición de servicio q_i, la cual podría ser considerada como la demanda de dicho cliente, un tiempo de servicio s_i y una ventana de tiempo $[e_i,\ l_i]$. Todos los vehículos tienen la misma capacidad Q y el mismo coste $h \geq 0$.

El modelo presupone que $c_{ij} = t_{ij},\ \forall (i, j) \in A$ y que h es lo suficientemente alto como para garantizar que el objetivo principal es minimizar el número de vehículos. Sin pérdida de generalidad, podemos reducir las ventanas de tiempo estableciendo $e_i = max(e_1 + t_{1i}, e_i)$ y $l_i = min(l_1 - t_{i1}, l_i)$. De hecho, esto elimina porciones que no pueden ser utilizadas de manera factible, planteado por Desrochers en 1992[12].

Las variables de decisión x_{ij}, son binarias, $x_{ij} = 1$ si el arco (i, j) es usado, es decir, que desde el cliente i se vaya al cliente j, en caso contrarío $x_{ij} = 0$; p_i representa el comienzo del servicio en el nodo i; y_i es la carga del vehículo que

abandona el nodo i. De manera matemática podemos formular el problema VRPTW del siguiente modo:

$$\min \sum_{j \in N'} h x_{1j} + \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{1.1}$$

sujeto al siguiente conjunto de restricciones:

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in N' \tag{1.2}$$

$$\sum_{i \in N} x_{ij} = 1 \qquad \forall j \in N' \tag{1.3}$$

$$\text{si } x_{ij} = 1 \Longrightarrow p_i + s_i + t_{ij} \leq p_j \qquad \forall (i,j) \in A \tag{1.4}$$

$$e_i \leq p_i \leq l_i \qquad \forall i \in N' \tag{1.5}$$

$$\text{si } x_{ij} = 1 \Longrightarrow y_i + q_j \leq y_j \qquad \forall (i,j) \in A \tag{1.6}$$

$$q_i \leq y_i \leq Q \qquad \forall i \in N' \tag{1.7}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{1.8}$$

La ecuación 1.1 representa la función objetivo, en el primer sumando aparece representado el costo asociado al número de vehículos que han sido empleado para satisfacer a los diferentes clientes. En el segundo sumando de la ecuación, aparece recogida la suma de los costes de cada uno de los arcos que han sido empleados para la satisfacción del conjunto de restricciones.

En la ecuación 1.2 queda recogido que de cada cliente solo puede salir un arco, es decir, solo podemos ir a un único cliente. En la ecuación 1.3, quedaría recogido que a cada cliente solo se puede llegar desde uno solo.

En las ecuaciones 1.4 y 1.5 quedan recogidas las restricciones de cada una de las ventanas de tiempo. Para poder realizar un modelo compresible por el programa que va a ser empleado, será sustituida la ecuación 1.4, por la siguiente:

$$p_i + s_i + t_{ij} - p_j \leq M \cdot (1 - x_{ij}) \qquad \forall (i,j) \in A \tag{1.9}$$

donde M representa un valor grande.

En las ecuaciones 1.6 y 1.7 quedan recogidas las restricciones de capacidad de cada uno de los vehículos, junto con la demanda que éstos satisfacen en

cada una de las rutas. De igual modo que antes, sustituiremos la ecuación 1.6 por la siguiente:

$$y_i + d_j - y_j \leq M \cdot (1 - x_{ij}) \qquad \forall (i,j) \in A \qquad (1.10)$$

donde M representa un valor grande.

En la ecuación 1.8 queda recogido el carácter de la existencia del arco que une al cliente i, con el cliente j.

El modelo que ha sido implementado no es una propuesta única para la resolución del problema. Autores como Fisher y Jaikumar (1981)[13], Fisher (1997)[14] y Kohl y Madsen (1997)[15], han propuesto diferentes formulaciones que dan respuesta a VRPTW. El principal problema de todos estos procedimientos es el gran tiempo que requieren hasta llegar a la solución óptima, no tenemos que olvidar que el problema que estamos intentando dar solución es del tipo *NP-hard*, y no se conoce aún una solución exacta que sea de orden polinómico. Este conjunto de propuestas fueron estudiadas en el proyecto final de grado de Arturo Alonso Alonso, presentado en el 2016[16]. En el capítulo de resultados, en el apartado 4.1, veremos el tiempo que ha sido requerido hasta la obtención de las diferentes soluciones.

En los siguientes capítulos veremos cómo afrontar la resolución del problema VRPTW, teniendo en cuenta que debemos ir asumiendo que la obtención de la solución óptima es un objetivo inalcanzable. Tenemos que encontrar un procedimiento que sea capaz de obtener una solución razonablemente buena, en un tiempo "pequeño" de modo que la resolución a este problema pueda ser aplicable en situaciones reales.

Capítulo 2

Heurísticas, aproximación a la solución óptima

Dado el modelo formulado en el capítulo anterior, y siendo ahora conscientes que alcanzar la solución óptima requiere en la gran mayoría de conjuntos de datos de un tiempo demasiado grande, podemos concluir que el procedimiento planteado hasta el momento no es el adecuado. Nos encontramos ante la tesitura siguiente ¿Qué hacemos si no podemos lograr la solución óptima? La respuesta es sencilla, aproximarnos a ella lo más posible en un tiempo razonable.

Una vez que somos conscientes de que lograr la solución óptima podría ser considerado como un objetivo inalcanzable, nos plantemos generar una serie de algoritmos que, con iteraciones sucesivas, logren alcanzar una solución aproximada a la óptima. Un manera lógica de comenzar nuestra búsqueda de soluciones sería siguiendo los siguientes pasos:

1. Comenzar generando una ruta inicial, que generalmente no será una solución óptima del conjunto de datos que se nos ha presentado. Este conjunto de heurísticas son conocidas como heurísticas de construcción y con ellas únicamente estamos buscando un camino inicial factible al que poder aplicar los siguientes puntos.

2. Aplicar sucesivos intercambios, los cuales provocan una disminución en la distancia global recorrida por el conjunto de vehículos. Este conjunto de algoritmos son conocidos como heurísticas de mejora. Los cambios que se recogen en este documento los podríamos clasificar dentro de dos

categorías, según el número de rutas de vehículos a las que impliquen.

- Cambios *intra-rutas*: cambios que afectan al orden en el que son visitados los clientes de cierta ruta, produciendo la alteración del orden de visita una reducción de la distancia total recorrida. Estudiaremos las ideas propuestas por G. A. Croes[17], Shen Lin[18] y Or[19].
- Cambios *entre-rutas*: cambios que afectan a varias rutas, estos cambios pueden ser muy diversos, ya sea la relocalización de un cliente en otra ruta, el intercambio de varios clientes entre dos rutas, o el cambio de varios clientes entre algún conjunto de rutas. Estudiaremos las ideas propuestas por Alex Van Breedam[20] y por Gerard AP Kindervater y Martin WP Savelsbergh[21].

2.1. Heurísticas de construcción de rutas

Las heurísticas de construcción de rutas, son un conjunto de algoritmos que nos permiten la obtención de una solución factible en un tiempo de orden polinómico, siendo las soluciones que se obtienen en la mayoría de las situaciones lejanas a la solución óptima. Los algoritmos que van a ser analizados serán propuestas de creación de rutas del tipo secuencial, comúnmente conocidas como heurísticas "miopes", puesto que tras realizar la inserción de un individuo en una ruta, no se reevalúa de nuevo si dicha asignación es la que menor costo proporciona de manera global. Existen dos propuestas heurísticas diferentes:

1. Un primer conjunto de heurísticas de inserción secuenciales. En ellas se va asignando de manera secuencial clientes a una ruta hasta que ningún cliente puede ser añadido en dicha ruta, siendo en ese momento en el que si hay aún clientes sin estar asignados a una ruta, se genera una nueva ruta. Este procedimiento se repite hasta que no queda ningún cliente sin estar asignado a alguna ruta.

2. Un segundo grupo de heurísticas de inserción en paralelo. En ellas es necesario conocer el número de rutas necesarias para satisfacer al conjunto de clientes, esto se realiza mediante un procedimiento secuencial. Un vez que conocemos el número de rutas, y el primer cliente de cada una de las rutas, vamos añadiendo el resto de clientes según un criterio que pondere el costo de añadir cada cliente en cada una de las rutas.

Estudiaremos las propuestas realizadas por Solomon[22], pasando por versiones basadas en las ideas de Solomon y que fueron propuestas por G. Ioannou, G. Prastacos y M. Kritikos[23], y terminando con una propuesta que simplifica la de este último grupos de autores creada por Mario y con la que se han obtenido aproximaciones de gran calidad.

Todos los procedimientos que van a ser analizados difieren únicamente en la función de costo que se emplea para analizar la inserción de cada cliente en cada momento.

2.1.1. Algoritmo vecino más cercano *nearest neighbor*

La heurística del vecino más próximo, para el problema VRPTW aparece descrita en el artículo que Marius M. Solomon publicó en 1987[24]. Se describe como una heurística perteneciente a la clase de heurísticas de construcción secuencial. Podríamos definir los pasos que da la heurística del siguiente modo:

1. Comenzamos una primera ruta inicial desde el depósito.
2. Añadimos el cliente más próximo al último cliente rutado, entre el conjunto de clientes aún no rutados y que cumplen las restricciones de capacidad del vehículo y de ventanas de tiempo que cada cliente ha impuesto.
3. Repetimos el punto anterior mientras que existan clientes que pueden ser rutados. Pudiendo ocurrir las siguientes posibilidades:
   a) Existen clientes que cumplen las condiciones de capacidad y de ventanas de tiempo. Se realiza el punto 2.
   b) No existen clientes que cumplen las condiciones de capacidad y de ventanas de tiempo. Se finaliza la ruta añadiendo el depósito. En caso de existir clientes sin rutar se vuelve al punto 1. En otro caso, se termina.

El procedimiento, tal y como ha sido descrito, proporciona un resultado que en la mayoría de las ocasiones, se encuentra muy lejano al valor óptimo. Es necesario reseñar que el tiempo de cómputo necesario para la obtención de una primera solución, en el peor de los casos, es de orden polinómico $O(n^2)$.

Para intentar obtener una mejor solución Solomon añadió tres criterios para ponderar el impacto que ocasiona la inserción de cada uno de los clientes candidatos a ser rutados:

(I) $IS_u =$ **Impacto propio**, es decir, la relación entre el tiempo de llegada al cliente u (a_u) y el primer instante de tiempo en el que se puede comenzar el servicio (e_u).

(II) $IU_u =$ **Impacto externo**, es decir, el impacto de la inserción del cliente u en una ruta en construcción respecto al resto de clientes aún no rutados.

(III) $IR_u =$ **Impacto interno**, es decir, el impacto de la inserción del cliente u en la ruta en construcción respecto a los clientes que componen la ruta en construcción.

Estos criterios serán los que empleen años después G. Ioannou, G. Prastacos y M. Kritikos [23] y serán descritos en profundidad en el momento en el que desarrollemos su planteamiento. La diferencia entre ambos artículos radica en que Solomon emplea los tres criterios de manera separada, es decir, únicamente emplea un criterio para la inserción de los diferentes clientes en las rutas, mientras que la propuesta de años después radica en una ponderación de dichos criterios, aunando en un único valor el impacto que provoca la inserción de cada cliente en cada momento. Otra diferencia que encontramos es que Solomon plantea en todo momento una inserción secuencial, mientras que en la nueva propuesta se puede realizar la inserción de un cliente no rutado en cualquier posición de la ruta en construcción.

2.1.2. Algoritmo inserción paralela

El algoritmo de inserción en paralelo es una propuesta realizada por Jean-Yves Potvin y Jean-Marc Rousseau en 1993[25], estos autores proponen iniciar un conjunto de rutas de una vez, e ir añadiendo el conjunto de clientes que aún no han sido asignados a ninguna ruta según un nuevo criterio que será detallado más adelante.

El primer escollo que nos encontramos al emplear esta nueva metodología, es la elección del número de rutas que se van a inicializar. Si elegimos un número pequeño de rutas, podemos encontrarnos antes la tesitura de que

en ninguna de ellas podemos seguir asignando clientes que aún no han sido asignados. Por el contrario, si seleccionamos un número elevado de rutas, podemos encontrarnos ante la tesitura de tener un conjunto de rutas que admitirían nuevos clientes, cuyo número podría reducirse si se realizara una asignación más óptima. Es por tanto la elección del número a iniciar lo que debería de hacerse de un modo adecuado. Estos autores plantean que para tomar la decisión del número de rutas que van a ser necesarias, se realice mediante la inserción secuencial que planteó Solomon, y conservar en cada ruta aquel cliente que se encuentre más alejado del depósito.

Una vez que tenemos el número de rutas, y el primer cliente asignado a cada una de ellas, estos autores describen una nueva métrica con la que seleccionar del conjunto de clientes, aún no rutados, el siguiente cliente al que asignarle ruta, y el lugar en el que asignarlo. Comienzan localizando inicialmente al cliente al que rutar, siendo aquel que minimiza la suma ponderada de dos medidas.

$$c_2(u^*) = \max_u \left[c_2(u)\right] \tag{2.1}$$

$$c_2(u) = \sum_{r \neq r'} \left[\ c^*_{1r}(i_r(u),\ u,\ j_r(u)) - \min_{r'=1,\ldots,n_{r'}} c^*_{1r'}(i_{r'}(u),\ u,\ j_{r'}(u))\ \right] \tag{2.2}$$

$$c_{1r}(i_r(u),\ u,\ j_r(u)) = \alpha_1 \cdot (d_{i_r,u} + d_{u,j_r} - d_{i_r,j_r}) + \alpha_2 \cdot (b_{u,j_r} - b_{j_r}) \tag{2.3}$$

$$\alpha_1 + \alpha_2 = 1,\ \alpha_1 \geq 0,\ \alpha_2 \geq 0 \tag{2.4}$$

La selección del cliente, la ruta en la que va a ser enrutado y la posición en la que va a ser insertado, estos autores la realizan como una ponderación del incremento de la distancia que se produce en cada una de las rutas si se realizara la inserción del cliente que se está tratando, y el cambio en el momento de inicio del servicio del siguiente cliente al punto en el que se insertaría al cliente candidato.

El algoritmo, tal y como ha sido descrito, proporciona una solución en un tiempo polinómico $O(n^2)$, para la obtención de una primera solución. La utilización del procedimiento descrito, nos plantea una serie de cuestiones, hasta qué punto la generación de dos conjuntos de rutas factibles diferentes, de las cuales se descarta la primera, nos acerca a una solución más óptima antes de aplicar los algoritmos de mejora de rutas. Otro aspecto negativo que

tenemos al trabajar con este procedimiento, es la elección del valor de los dos parámetros que ponderan ambos sumando de la ecuación 2.3.

Una propuesta más reciente de construcción de rutas de manera paralela, ha sido realizada en el 2011 por King-Wah Pangs[26], la propuesta de este autor es algo más compleja que la que ha sido desarrollada. Este autor propone la ponderación de tres criterios: distancia, urgencia y espera, con los que obtiene una medida con la que seleccionar al cliente que va a ser insertado y el lugar que va a ocupar.

2.1.3. Algoritmo IMPACT

Este algoritmo fue publicado en Palgrave Macmillan Journals en el 2001, siendo sus autores G. Ioannou, G. Prastacos y M. Kritikos [23]. Este algoritmo surge al observar que en las propuestas que se habían realizado en años anteriores, no se estaba teniendo en cuenta las relaciones que se establecen entre las diferentes ventanas de tiempo entre cada uno de los clientes. Este conjunto de autores, de acuerdo con la teoría de Atkinson[27], desarrollaron un nuevo criterio de selección e inserción de clientes y fueron capaces de obtener un nuevo método que generaba nuevas soluciones. La idea básica detrás de este criterio es que la selección del cliente u que va a ser rutado, debe minimizar el impacto de los clientes frente a la ruta en construcción y el impacto en la ventana de tiempo del cliente u. Para medir ese impacto definieron tres nuevos criterios:

(a) $IS_u =$ **Impacto propio**, es decir, la relación entre el tiempo de llegada al cliente u (a_u) y el primer instante de tiempo en el que se puede comenzar el servicio (e_u).

(b) $IU_u =$ **Impacto externo**, es decir, el impacto de la inserción del cliente u en una ruta en construcción respecto al resto de clientes aún no rutados.

(c) $IR_u =$ **Impacto interno**, es decir, el impacto de la inserción del cliente u en la ruta en construcción respecto a los clientes que componen la ruta en construcción.

Podríamos por tanto, clasificar esta propuesta de construcción de rutas dentro de los algoritmos que tienen presente el “costo" de introducir un nuevo cliente en la ruta en construcción, buscando al minimizar dicho “costo”,

al mejor candidato en la mejor posición. La forma de medir el "costo", se realiza de acuerdo con la ponderación de los tres impactos. A continuación nos encargaremos de definir con más detalle cada uno de los tres impactos anteriores.

El *impacto propio* es la diferencia entre el tiempo en el que podemos llegar al cliente u y el tiempo en el que podríamos comenzar el servicio. Viene definido mediante la siguiente ecuación:

$$IS_u = a_u - e_u \tag{2.5}$$

Un valor IS_u cercano a cero proporciona un margen para añadir clientes antes o después de u. Por consiguiente, un valor cercano a cero en IS_u hace aumentar la posible inserción del cliente en la ruta en construcción.

Para ilustrar el segundo criterio, el *impacto externo*, hemos considerado que el cliente u y el cliente j, son ambos clientes candidatos para su inserción en la ruta actual. La ventana de tiempo entre estos dos consumidores queda representada en la Figura 2.1. Después de insertar el cliente u, el problema de seleccionar un nuevo cliente para insertarlo en la ruta se complica, esto es así porque se solapan las ventanas de tiempo de los dos clientes. La condición necesaria para que el vehículo pueda visitar j después de la inserción del cliente u es:

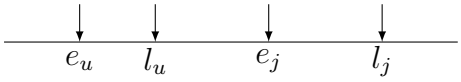


Figura 2.1: Ventanas de tiempo para los clientes u y j

$$e_u + d_{uj} \leq l_j \vee e_j + d_{ju} \leq l_u \tag{2.6}$$

En la fórmula anterior hemos considerado ambos casos que se muestran en la Figura 2.1, es decir, el cliente u precede al cliente j y viceversa. Es obvio que la inserción del cliente u que minimiza la no negativa diferencia de $[l_j - (e_u + d_{uj})]$ y $[l_u - (e_j + d_{ju})]$, es de esperar que sea una buena selección. Esto es así porque al minimizar la diferencia disminuye los componentes de la ventana de tiempo del cliente j afectado, debido a la inserción del cliente u en la ruta actual. Hay que denotar que las relaciones entre el cliente j y el cliente u, una de las dos diferencias es positiva.

La medida del impacto externo calcula la cobertura real de la ventana de tiempo del cliente no rutado resultado de la inserción del cliente seleccionado en la ruta. Examinando todos los posibles casos formulamos el siguiente criterio:

$$IU_u = \sum_{j \in J - \{u\}} \left\{ \frac{1}{|J| - 1} \right\} \times max \left\{ (l_j - e_u - d_{uj}), (l_u - e_j - d_{uj}) \right\} \tag{2.7}$$

donde J es el conjunto de clientes en cualquier etapa del procedimiento de construcción de la ruta. La ecuación anterior expresa formalmente el promedio de la ventana de tiempo no utilizada sobre todos los clientes no rutados.

Para definir el *impacto interno*, se requieren tres medidas: perturbación local y global resultante de la inserción de un cliente seleccionado en la ruta, y la accesibilidad actual de este cliente a la ruta actual. Podemos definir estas medidas de la siguiente manera: sea u el cliente seleccionado para ser insertado entre clientes i, j de la ruta actual. Esta inserción provoca un aumento de la distancia:

$$c_{1u}(i, j) = d_{iu} + d_{uj} - d_{ij} \tag{2.8}$$

Además, un retardo de tiempo relacionado con la llegada del vehículo, C_{2u}, se introduce al cliente j. La Figura 2.2 ilustra este momento, que es el margen de tiempo factible del cliente u. Se define como la diferencia de tiempo de llegada al cliente j, antes y después de la inserción de u en la ruta actual. De manera matemática quedaría definido C_{2u} como:

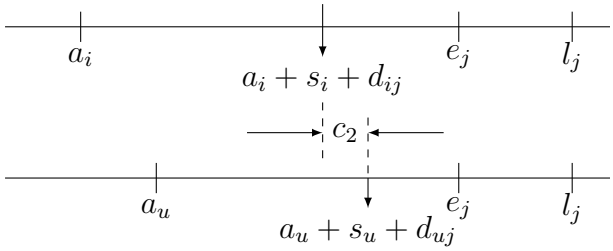


Figura 2.2: Margen en las ventanas de tiempo de dos clientes

$$c_{2u}(i, j) = [l_j - (a_i + s_i + d_{ij})] - [l_j - (a_u + s_u + d_{uj})] \tag{2.9}$$

La viabilidad marginal calcula la urgencia del cliente j rutado. Es la medida de Solomon, siendo el retraso de la llegada al cliente j debido al haber insertado al cliente u en la ruta.

Por último la inserción del cliente u entre el cliente i y j crea un espacio de tiempo, c_{3u}, entre el último momento en el que se puede iniciar el servicio l_u del cliente u y el tiempo en el que el vehículo llega al cliente u. Esta medida expresa la compatibilidad de la ventana de tiempo del cliente seleccionado, con el lugar especifico de inserción en la ruta actual. La fórmula para esta medida es:

$$c_{3u}(i,j) = l_u - (a_i + s_i + d_{iu}) \tag{2.10}$$

Es evidente que un cliente no rutado u que tiene un coste menor, debe ser insertado en la ruta.

La mejor posición para la inserción de un cliente no rutado viene definida por la suma ponderada de la combinación de criterios previamente descritos, es decir, de la distancia extra, de la viabilidad del tiempo marginal y de la compatibilidad de las ventanas de tiempo. Matemáticamente se puede definir este criterio, que hemos definido por perturbación local, LD_u, de la inserción del cliente u entre el cliente i y j como lo que sigue:

$$LD_u(i,j) = b_1 c_{1u} + b_2 c_{2u} + b_3 c_{3u} \tag{2.11}$$

Siendo $b_1 + b_2 + b_3 = 1$, y b_1, b_2 y $b_3 \geq 0$

Para la ruta actual r y la viabilidad del cliente u no rutado, se define la perturbación global IR_u como:

$$IR_u = \sum_{(i,j)\in I_r} \frac{LD_u}{|I_r|} \tag{2.12}$$

donde I_r es el conjunto de lugares viables para la inserción del cliente u dentro de la ruta r, y $|I_r|$ es el cardinal de este conjunto. Hay que denotar que los elementos de I_r son pares ordenados de clientes (i,j) que pertenecen a la ruta actual. La inversa de la media de la perturbación local define la accesibilidad, es decir:

$$ACC_u = (IR_u)^{-1} \tag{2.13}$$

La medida de la accesibilidad expresa el impacto interno del cliente seleccionado u en la ruta actual, ya que un cliente con una gran accesibilidad causa un impacto pequeño en la ruta actual. Este es el resultado de la combinación de las tres submedidas que forman la accesibilidad. Nuestro enfoque heurístico requiere que el cliente seleccionado para la inserción debe tener la mejor accesibilidad para la ruta actual.

Ahora pasamos a definir el criterio de selección de clientes que cuenta con todos los tipos de impactos descritos anteriormente. Usamos una relación

simple lineal similar a la usada en 2.12, para fusionar el impacto interno, el externo y otro tipo de impacto para el cliente no rutado u:

$$Impact(u) = b_s IS_u + b_e IE_u + b_r IR_u \tag{2.14}$$

$b_s + b_e + b_r = 1$, y b_s, b_e y $b_r \geq 0$. Denotar que b_s, b_e y b_r definen la contribución relativa de cada medida individual (interno, externo y el propio impacto) para el criterio general. Los valores pueden ser definidos tras la realización de experimentos estadísticos.

El criterio de selección para el cliente propuesto de 2.14 asegura que el cliente u, seleccionado para su inserción dentro de la ruta, puede minimizar el impacto de todos los clientes ya asociados a la ruta en construcción, en todos los clientes no rutados y en si mismo. De este modo, el criterio debe conducir al método aleatorizado a una buena solución ya que este permitirá la exploración de un espacio de soluciones y tendrá en cuenta las relaciones entre consumidores mediante las ventanas de tiempo. Además 2.14 incorpora tres parámetros adicionales relacionados con $LD_u(i, j)$, es decir, b_1, b_2 y b_3. En consecuencia $Impact(u)$ depende de 3 x 3 combinaciones de parámetros. Un buen ajuste de dichos parámetros hará que obtengamos un buen conjunto de soluciones aproximadas.

Para determinar el cliente que debe ser insertado, la heurística calcula, mediante $Impact(u)$ para cada cliente no rutado u usando las ecuaciones 2.5–2.14, y selecciona el que minimiza el valor de la medida 2.14. El cliente seleccionado u se ubica dentro de la ruta entre los cliente (i, j) en la ruta en construcción tal que LD_u de la ecuación 2.11 es mínimo. Este proceso se repite hasta que ningún cliente no rutado pueda ser insertado en la ruta en construcción. En este caso, una nueva ruta se inicia con un nuevo cliente y el bucle se inicia de nuevo hasta que todos los clientes son asignados a una ruta. Entonces el algoritmo termina dando el número de rutas, el número de vehículos activos (igual al número de rutas), los clientes que han sido asignados a cada vehículo, la secuencia en la que los clientes son visitados, el total de tiempo, distancia y coste de la solución. Hay que destacar que la viabilidad de la solución con respecto al modelo exacto es garantizada en cada etapa del algoritmo. El pseudocódigo de la heurística se presenta a continuación:

Algorithm 1 Algoritmo que pondera el impacto

procedure GENERAL
  n: Número de clientes
  K: Capacidad de los vehículos
  c_{ij}: Matriz de costos de viajar de un cliente a otro
  e_i: Tiempo más temprano de inicio de servicio en el cliente i
  l_i: Tiempo más tardío de inicio de servicio en el cliente i
  $S \leftarrow \{\emptyset\}$: Solución, conjunto de rutas
  $C \leftarrow \{2, ..., n\}$: Conjunto de clientes que aún no han sido rutados
  aux: Ruta en construcción
  while $C \neq \emptyset$ **do**
    if $aux = \emptyset$ **then**
      $aux \leftarrow maslejos(C)$ "Generalmente el más lejano de la base"
      $C \leftarrow C - \{aux\}$
    else
      $menor, i, j \leftarrow Impact(cumple(C, aux))$ Cumple restringe el conjunto total de candidatos únicamente a los que cumplen la restricción de carga
      if $menor \neq \emptyset$ **then**
        $inserta(menor, i, j)$
        $C \leftarrow C - \{menor\}$
      else
        $S \leftarrow S \cup \{aux\}$
        $aux \leftarrow \emptyset$
  return S

procedure IMPACT(C)
  $menor$: Cliente cuya inserción genera menor impacto y la posición entre que dos debe ser insertado
  for all C **do**
    $imp_u \leftarrow b_s * IS(u) + b_e * IU(u) + b_r * IR(u)$ Funciones descritas en las siguientes ecuaciones 2.5–2.14
    if $menor \geq imp_u$ **then**
      $menor \leftarrow imp_u$
  return menor

2.1.4. Algoritmo simplificado de vecino más próximo

El algoritmo que se va a presentar a continuación es una propuesta propia, y ha sido inspirada en el conjunto de propuestas que a lo largo del tiempo han surgido. La simplificación del conjunto de propuestas, podría llevarnos en alguna ocasión a quedarnos con la ruta inicial cuya solución quedara lejos de la solución óptima, lo que nos ocasionaría aplicar un gran número de intercambios mediante los procedimientos de mejora.

Las propuestas que requieren de un refinamiento de parámetros que ponderan diferentes métricas, las cuales son empleadas para la decisión de qué cliente es el que tiene que ser insertado y lugar en el que insertarlo, son propuestas que funcionan correctamente cuando la selección de parámetros se realizada adecuadamente.

En el punto de resultados, veremos que una mala selección de los parámetros nos llevará a una solución lejana a la solución óptima, es decir, el tiempo de computación empleado para llegar a dicha solución no compensa frente al costo de la ruta obtenida. Hay que apuntar, que las heurísticas que ponderan diferentes criterios adecuadamente, logran obtener una solución próxima a la solución óptima, pero que dejaría de serlo si cambiara el conjunto de datos. Dicha combinación de parámetros podría considerarse como una tarea inútil al dejar de ser viable.

Una vez que han sido descartadas el conjunto de heurísticas de construcción en las que hay que seleccionar el conjunto de pesos que van a tomar las diferentes métricas, pasamos a plantearnos qué métrica sería más adecuada emplear. Si nos remontamos a los comienzos, en los que Solomon[24] proponía una heurística de inserción secuencial del siguiente vecino más próximo, sin tener en cuenta ninguna de las otras métricas que más tarde propuso, y revisamos la propuesta que realizaba Potvin[25] en la que proponía una inserción de clientes en cualquier punto de las diferentes rutas en construcción, trataremos de aunar ambas propuestas en un método nuevo.

Una vez que tenemos ambas propuestas, podemos plantear un procedimiento similar al planteado por Solomon:

1. Comenzamos una primera ruta inicial desde el depósito.

2. Añadimos el cliente más próximo a cualquiera ya rutado, entre el conjunto de clientes aún no rutados y que cumplen las restricciones de

capacidad del vehículo y de ventanas de tiempo.

3. Repetimos el punto anterior mientras que existan clientes que pueden ser rutados. Pudiendo ocurrir las siguientes posibilidades:

   a) Existen clientes que cumplen las condiciones de capacidad y de ventanas de tiempo. Se realiza el punto 2.

   b) No existen clientes que cumplen las condiciones de capacidad y de ventanas de tiempo. Se finaliza la ruta añadiendo el depósito. En caso de existir clientes sin rutar se vuelve al punto 1. En otro caso, se termina.

De manera formal podríamos definir el procedimiento del siguiente modo:

$$\min_{(i,j)\in I_r} [c(i,\ u,\ j)] \quad \forall u \mid \text{ruta válida} \tag{2.15}$$

$$c(i,\ u,\ j) = d_{i,u} + d_{u,j} - d_{i,j} \tag{2.16}$$

En la ecuación 2.16, hemos definido la métrica como el incremento que tenemos en la distancia total recorrida cuando añadimos un nuevo cliente entre el cliente i y el cliente j. En la ecuación 2.15 minimizamos todos los costes y nos quedamos con el cliente y el punto de inserción que lo haga mínimo, siempre y cuando la ruta resultante sea factible.

Respecto al costo computacional, es notablemente superior al obtenido con la heurística propuesta por Solomon, dado que para cada una de las inserciones candidatas tenemos que comprobar si dicha inserción es factible. Si suponemos el coste de comprobación como una constante, podríamos tener, en el peor de los casos, un costo de inserción del orden $O(n^3)$.

En los resultados computacionales veremos que la simplificación realizada, proporciona un conjunto de rutas factibles, en un tiempo razonable. Dada la sencillez de esta heurística, la no necesidad de tener que afinar con ningún parámetro, un tiempo de cómputo asumible y los buenos resultados que proporciona con los conjuntos de prueba que han sido empleados, es una heurística candidata a emplear para realizar una primera aproximación a la ruta factible.

2.2. Heurísticas de mejoras de rutas

De la sección anterior se nos pueden plantear un conjunto de cuestiones a las que en este epígrafe intentaremos dar respuesta, ¿el orden de los clientes de todas las rutas es el que menor distancia proporciona?, ¿la asignación de cada cliente a cada ruta es la de menor distancia?

En relación con la primera cuestión, podemos considerar cada ruta como un problema individual, lo que puede ser usado para disminuir el tiempo de computo si se realiza una implementación de procesos paralelizados. Para estudiar el conjunto de permutaciones existentes del orden en el que son visitados los clientes de una ruta, estudiaremos las ideas propuestas por G. A. Croes[17], Shen Lin[18] y Or[19], y serán vistas en el apartado de mejoras intra-rutas.

La segunda cuestión afecta a los clientes que componen al menos dos rutas. Es lógico pensar que la composición inicial no será la correcta y que tendremos que realizar permutaciones de clientes entre las diferentes rutas, hasta obtener la asignación de los clientes que proporcione la menor distancia encontrada. Este conjunto de permutaciones serán estudiadas en el punto de mejoras entre-rutas, considerando las ideas propuestas por Alex Van Breedam[20] y por Gerard AP Kindervater y Martin WP Savelsbergh[21].

2.2.1. Mejoras intra-rutas

En este apartado estudiaremos diferentes métodos heurísticos que nos ayuden a ordenar la visita a cada uno de los clientes que componen una ruta, de modo que el resultado final obtenido sea el que menor costo ocasione. Las ideas que van a ser recopiladas fueron planteadas para el problema comúnmente conocido como el *"problema del viajante"* (TSP), pero añadiendo las restricciones que deben de cumplir nuestras rutas, podemos emplearlas para el problema que intentamos resolver.

Comenzaremos estudiando el intercambio denominado **2-opt**, este intercambio fue propuesto por G. A. Croes (1954)[17]. El procedimiento que Croes planteó se basaba en el problema de distancias euclídeas, si dos arcos que unen a dos clientes de una ruta se cruzan, podemos sustituir dichos arcos por otro par, de modo que la ruta quede de nuevo unida, y produciendo este intercambio de arcos una disminución en la distancia total recorrida. Una vez que

encontramos el cruce entre dos arcos que unen a dos clientes, y procedemos a su eliminación, la reconstrucción de la ruta es única y produce un cambio en el orden en el que van a ser visitados algunos de los clientes. Vemos un ejemplo en la Figura 2.3.

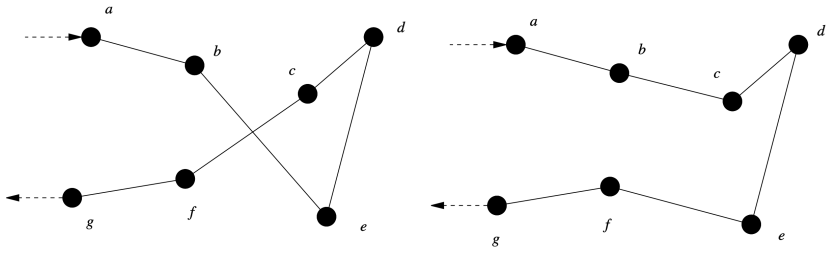


Figura 2.3: Intercambio 2-opt

En la Figura 2.3 vemos que si eliminamos los arcos que se encuentran cruzados, y añadimos los arcos necesarios para reconectar el conjunto de clientes, comprobamos que la distancia total recorrida se reduce. Tal y como poníamos de manifiesto, el orden en el que son visitados ciertos clientes se invierte.

El planteamiento queda recogido en el Algoritmo 2. El procedimiento descrito realizaría el intercambio que mejor resultado proporcione, del conjunto de nodos introducidos que pertenecen a una misma ruta. Si quisiéramos realizar todos los posibles intercambios 2-opt factibles, tendríamos que realizar el procedimiento hasta que no se produjeran cambios en el orden de los clientes que el procedimiento recibió. El siguiente procedimiento ha sido definido de manera genérica y si quisiéramos adaptarlo a nuestro problema deberíamos añadir las restricciones de las ventanas de tiempo.

Algorithm 2 Algoritmo 2-opt

```
1: procedure GENERAL
2:     n: Número de clientes de la ruta
3:     c_ij: Matriz de costos de viajar de un cliente a otro
4:     i_aux, j_aux Variables auxiliares
5:     for i = 0; i < n − 1; i++ do
6:         for j = i + 1; j < n; j++ do
7:             if c_{i,i+1} + c_{j,j+1} − c_{i,j} − c_{i+1,j+1} > 0 then
8:                 i_aux, j_aux ← i, j Si el ahorro resultante es mayor
9:     Actualizamos el orden de los clientes
```

En 1965, Shen Lin[18] planteo el intercambio **3-opt**, este intercambio realiza la eliminación de tres arcos y la posterior reconexión del conjunto de los clientes, esta reconexión a diferencia del planteamiento anterior no es única. Vemos un ejemplo de este intercambio en la Figura 2.4.

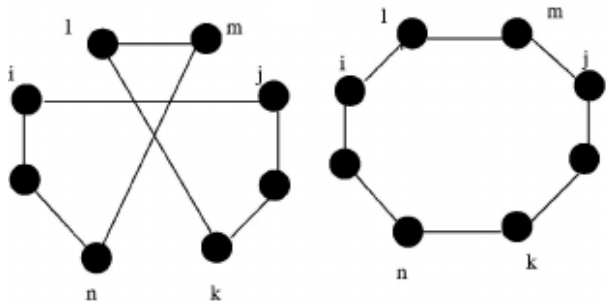


Figura 2.4: Intercambio 3-opt

Si nos fijamos con detenimiento en la Figura 2.4 veremos que tras la eliminación de tres arcos, existen ocho formas distintas de realizar la reconexión de los diferentes clientes. Teniendo que una de ellas produciría la misma reconexión inicial, tres de ellas producirían un intercambio 2-opt, y las cuatro restante un intercambio 3-opt.

Fue en 1973 cuando apareció publicada la generalización de los dos planteamientos anteriores, tomando el nombre de **k-Opt**. Apareció publicada en un artículo escrito por Shen Lin y por Brian W. Kernighan[28] donde se proponía un procedimiento para la selección de k arcos que van a ser eliminados y la posterior reconexión del conjunto de clientes, de modo que la ruta final sea de menor distancia que la ruta inicial.

El número de posibles combinaciones que tendríamos que comprobar en una ruta que contiene n clientes, y en la que se eliminan k arcos sería $\binom{n+1}{k}2^{k-1}(k-1)!$. Tendríamos que la comprobación de si un recorrido contiene algún intercambio de orden k factible sería en el peor de los casos del orden de $O(n^k)$.

De los procedimientos presentados de mejoras intra-rutas, han surgido diversas modificaciones, una de las primeras fue la sugerida por Ihan Or en 1976, y que es conocida con el nombre de **Or-opt**[19], la heurística plantea un movimiento de clientes sucesivos, conservando el orden en el que eran visitados, entre un par de cliente de la ruta. Vemos un ejemplo de dicho intercambio en la Figura 2.5, donde los clientes que se encuentran entre el

cliente i_2 y el cliente i_3 son movidos entre los clientes j_1 y el cliente j_2, para que la ruta permanezca conexa se añade un arco entre el cliente anterior a i_2 y el cliente posterior a i_3.

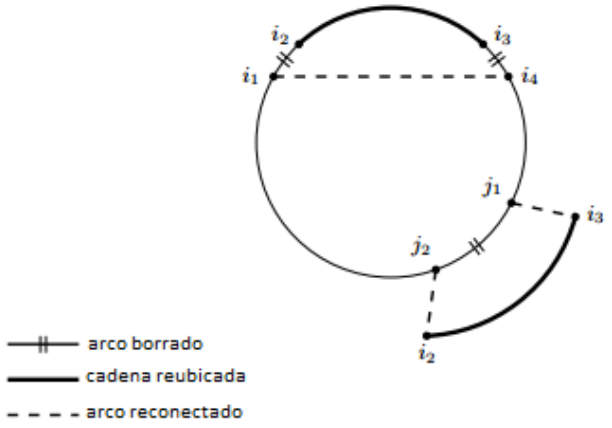


Figura 2.5: Intercambio Or-opt[29]

El modo en el que ha sido aplicada esta heurística es muy diverso Golden y Stewart en 1985[30], aplicaron la heurística después de su método de construcción y llegaron a la conclusión de que en general, el procedimiento Or-opt produce soluciones que son comparable a las obtenidas con el procedimiento 3-opt, en calidad de solución, obteniendo la solución en un tiempo más cercano al procedimiento 2-opt.

En el 2002 apareció publicado un artículo firmado por DS. Johnson, G. Gutin, LA. McGeoch, A Yeo, W. Zhang y A. Zverovitch[31]. Con él se intentaba poner de manifiesto que el procedimiento Or-opt podía dejar de ser considerado como un competidor serio a la hora de encontrar soluciones óptimas aproximadas. En el artículo aparecen una serie de recomendaciones con las que ahorrar comprobaciones innecesarias.

1. Evitar las redundancias del espacio de búsqueda: al implementar k-opt, el tiempo se puede ahorrar evitando algunos intercambios que no puede mejorar la solución actual.

2. Listas de vecinos delimitadas: considerar solamente los p vecinos más cercanos de un vértice cuando se realizan las reconexiones (esta regla

fue implementada por Zweig, 1995[32]).

3. Evitar considerar ciertos movimientos que han resultado infructuosos en el pasado.

4. Representar el conjunto de intercambios mediante estructuras de árboles, esta representación reduce el número de comprobaciones necesarias y acelera el tiempo de cálculo.

Cogiendo el testigo, Gilbert Babin, Stéphanie Deneault y Gilbert Laporte en el 2007[29] publicaron un artículo en el que realizaban la comparación entre la heurística clásica del intercambio 2-opt, la propuesta inicial del Or-opt y una versión de esta última.

Vemos que el número de maneras de implementar las heurísticas de mejora intra-ruta es muy amplia, incluso las heurísticas más básicas. Además, las comparaciones pueden verse afectadas no sólo por la implementación realizada, sino también por las instancias utilizadas para realizar las pruebas.

Una alternativa básica para evaluar la calidad de una heurística de mejora es si se debe partir de una ruta cualquiera o de una "buena" ruta producida por una heurística de construcción. Otra cuestión es si se debe implementar la primera mejorar (FI) en cada iteración o la mejor mejora (BI). Estas cuestiones fueron abordadas por Hansen y Mladenović (2005), quienes concluyeron que para el 2-opt, FI es ligeramente mejor pero mucho más rápido que BI si uno comienza en una ruta arbitraria, pero el resultado inverso se mantiene si una buena ruta es empleada como punto de partida.

2.2.2. Mejoras entre-rutas

De las heurísticas de construcción se desprende una conclusión, es posible que la distribución de los clientes que componen cada una de las rutas no sea la combinación más óptima, por tanto, necesitamos realizar intercambios de clientes entre las diferentes rutas, de modo que la distancia total recorrida por el conjunto de los vehículos sea la menor posible.

En este epígrafe describiremos la propuesta que realiza Alex Van Breedam, y su posterior adaptación realizada por Gerard AP Kindervater y Martin WP Savelsbergh.

Breedam recoge en su tesis publicada en 1994[33] cuatro posibles mejores que involucran a dos rutas. Las cuatro propuestas se hicieron para el problema de CVR, por tanto la propuesta no tiene presente que el intercambio no siempre lo podremos realizar puesto que tenemos unas condiciones iniciales del problema que no pueden ser incumplida:

- Las ventanas de tiempo en las que cada cliente quiere que el servicio comience.
- La capacidad de los vehículos que recorren cada una de las rutas.

Es en el artículo de Kindervater y Savelsbergh de 1997[21] donde se recoge una propuesta para el problema VRPTW. En este articulo se realiza una descripción de las condiciones necesarias que tienen que cumplirse para poder realizarse el intercambio. La propuesta que estos dos autores publicaron se podría considerar de idéntica naturaleza a la que propuso Breedam años antes y que fueron recogidas en el libro publicado en 1994[20]. Las propuestas de Alex Van Breedam son las siguientes:

1. String Cross (SC): Este intercambio se realiza cuando dos rutas tienen dos arcos cruzados, produciéndose al eliminar el cruce una disminución de la distancia recorrida.

   En la Figura 2.6, podemos ver el efecto que tendría si se eliminan los dos arcos que se encuentran cruzados y se introducen otros dos nuevos de modo que las dos rutas nuevas que se generan disminuyen la distancia global.

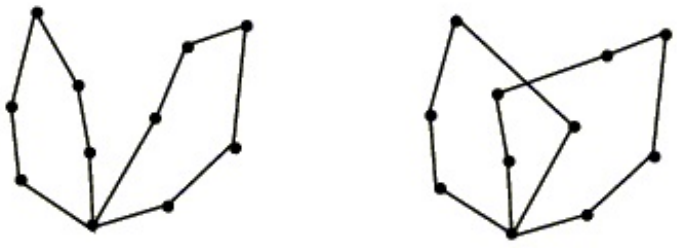


Figura 2.6: String Cross (SC)

2. String Exchange (SE): Este intercambio se plantea como el intercambio de fragmentos consecutivos de clientes entre las dos rutas.

En la Figura 2.7 el intercambio que se realiza únicamente afecta a un cliente de cada una de las rutas, pero el planteamiento es más genérico.

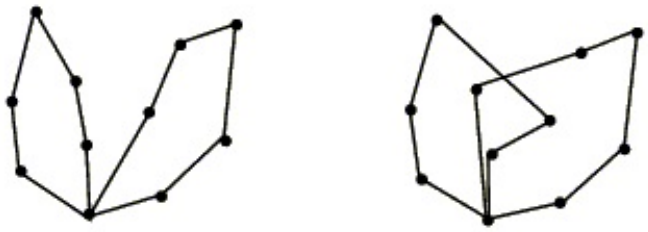


Figura 2.7: String Exchange (SE)

3. String Relocate (SR): Este intercambio plantea cambiar a un cliente de una ruta a otra, alojándole en la ruta receptora en la posición que menor distancia global produzca.

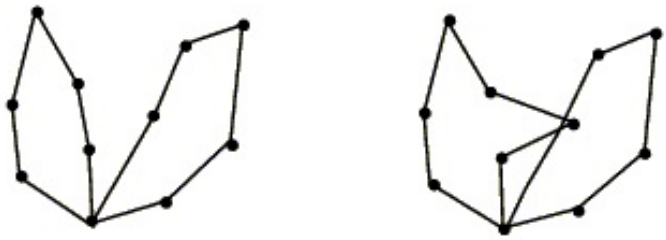


Figura 2.8: String Relocate (SR)

4. String Mix (SM): Este intercambio se plantea como el mejor intercambio encontrado entre los procedimientos de SE y SR.

Alex Van Breedam propone dos estrategia a seguir, la evaluación perezosa del conjunto de soluciones, es decir, aplicar el primer intercambio que mejore la función objetivo, o por el contrario, la evaluación del conjunto global de posibles intercambios y realizar aquel que produzca el mayor ahorro. Van Breedam define un conjunto de parámetros que pueden influir en el comportamiento del sus propuestas:

- La solución inicial, si ésta se encuentra alejada de la verdadera solución, el número de intercambios que hay que realizar hasta acercarse a ella será elevado.

- El número de clientes que permitamos intercambiar en los intercambios de tipo SE, SR y SM. Breedam plantea emplear un conjunto reducido de clientes, pero la implementación que se ha realizado no tiene presente dicha restricción, tomándose de un modo más genérico la propuesta inicial.

- La estrategia a seguir a la hora de realizar la búsqueda de posibles permutaciones, es decir, la evaluación de todo el conjunto de posibles intercambios o la evaluación hasta encontrar el primer intercambio que reduzca la distancia global. Este planteamiento fue expuesto al final del epígrafe mejoras intra-rutas.

Una vez que no se encuentran intercambios de clientes entre las diferentes rutas, se nos plantea de nuevo la pregunta de si la ordenación de los clientes dentro de cada ruta es la correcta, lo que nos haría volver a revaluar las heurísticas de mejora intra-ruta. Una vez que no se encuentren movimientos posibles dentro de cada una de las diferentes rutas, se nos volvería a plantear la duda de si la asignación de los clientes a las diferentes rutas es la correcta, lo que haría que volviéramos a revaluar las heurísticas de mejora entre-rutas. Terminaremos por tanto, en el momento que no logremos encontrar movimientos posibles, teniendo como resultado una aproximación a la solución óptima.

Capítulo 3

Metaheurísticas, ampliando la búsqueda de soluciones

Las metaheurísticas son un conjunto de algoritmos generadores de soluciones que exploran el espacio de soluciones factibles, buscando identificar buenas soluciones empleando algunas de las construcciones de rutas estándar y heurística de mejora. Una de las principales diferencias con los enfoques clásicos, nos la encontramos en que las metaheurísticas permiten un deterioro, e incluso la admisión de soluciones intermedias no viables en el transcurso del proceso de búsqueda.

A lo largo del tiempo, se han propuesto diferentes algoritmos que buscan la ampliación del espacio de búsqueda. Este conjunto de procedimientos se desarrollaron buscando eliminar el aspecto determinista que subyace a las heurísticas clásicas, las cuales obtendrían la misma solución si realizáramos una nueva ejecución.

Olli Bräysy y Michel Gendreau en 2005[34] publicaron un artículo en el que se recopila un conjunto de procedimientos metaheurísticos, como son los algoritmos genéticos y la búsqueda tabú. Años después, Mario César Vélez y José Alejandro Montoya en 2007[35] publican una descripción poco detallada de metaheurísticas clásicas, las cuales pueden ser aplicadas en el problema que estamos resolviendo. Todos estos procedimientos han sido combinados en los últimos años en procesos de hibridación.

Las metaheurísticas que vamos a desarrollar en este documento, serán dos algoritmos que plantean eliminar el determinismo de los procedimientos

clásicos en dos situaciones diferentes:

1. Eliminación del determinismo que subyace a las heurísticas de construcción de rutas. La metaheurística diseñada para eliminar el determinismo en la primera etapa es GRASP (*Greedy Randomized Adaptive Search Procedure*).

2. Eliminación del determinismo que subyace a las heurísticas de mejora de rutas, tanto en el procedimiento seleccionado, como en el intercambio a realizar. La metaheurística diseñada para eliminar el determinismo en la etapa de mejora es *Simulated annealing* (Recocido Simulado).

Aún cuando el conjunto de metaheurísticas que se han propuesto en las diferentes publicaciones ha sido variada, analizaremos dos propuestas que buscan ampliar el conjunto de soluciones analizadas. En ambos procedimientos se busca, mediante la aleatoriedad, ampliar el espacio de búsqueda, evitando que el algoritmo se quede atrapado en un óptimo local.

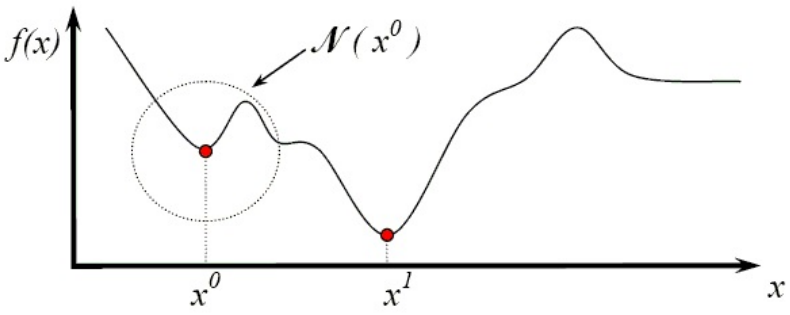


Figura 3.1: Óptimo local y óptimo global[35]

En la Figura 3.1 se ilustra un problema en el que se desea encontrar el valor de x que minimiza la función $f(x)$, donde $N(x)$ representa el vecindario de x. Si durante la búsqueda, el método de optimización llega a x^0 y el método no acepta soluciones de inferior calidad en $N(x^0)$, el método queda atrapado en el óptimo local x^0 como la mejor solución.

Al igual que nos pasaba con las heurísticas de construcción, existen numerosos procedimientos con los que guiar al algoritmo a obtener una buena solución, alguno de los procedimientos que aún no han sido mencionados son: *Threshold Accepting* (aceptación por umbrales), colonias de hormigas,

Scatter Search (búsqueda dispersa) y *Path Relinking* (Re-encadenamiento de trayectorias), entre otros muchos.

Para la realización implementación de los diferentes procedimientos metaheurísticos, se ha consultado el libro publicado por Manuel Laguna en 2003[36].

3.1. GRASP (*Greedy Randomized Adaptive Search Procedures*)

GRASP (*Greedy Randomized Adaptive Search Procedures*), es una meteheurística considerada entre las más prometedoras a la hora de la obtención de soluciones de problemas de optimización combinatoria. Aparece descrita en numerosas publicaciones, denotándose en todas ellas la facilidad de su aplicación. Thomas A. Feo, y Mauricio GC. Resende en 1995[37] realizan una descripción general, la cual permite adaptar esta metaheurística a un conjunto muy diverso de problemas. En el mismo año, George Kontoravdis y Jonathan F. Bard [38] realizan una publicación de similares características. Desde ese año, han surgido numerosas publicaciones que tratan de poner en valor este procedimiento. Es necesario reseñar la publicación de Mauricio G.C. Resende y Celso C. Ribeiro en 2016[39] donde se realiza una descripción detallada de la multitud de aplicaciones que tiene.

Algorithm 3 Algoritmo genérico GRASP

```
1: procedure GRASP
2:     input() carga de datos
3:     for criterio de parada do
4:         Creamos una solución GRASP con las heurísticas de construcción
5:         Aplicamos las heurísticas de mejora
6:         Si el costo es menor, actualizamos la solución
       return solución
```

GRASP es un proceso iterativo, compuesto de dos fases. En primer lugar se construye una solución mediante el conjunto de heurísticas de construcción. Seguidamente se realiza una fase de búsqueda local en la que se intenta reducir el valor objetivo mediante la realización de giros. En el pseudo-código 3 queda reflejado cada uno de los pasos a seguir.

En la primera fase, se construye una primera solución factible de manera iterativa, siguiendo la heurística de construcción deseada, insertando elementos de uno en uno. En cada iteración de construcción, la elección del siguiente elemento que se añade se determina ordenando todos los elementos de una lista de candidatos con el criterio que deseemos. Este orden mide el beneficio (miópico) de seleccionar cada elemento. La heurística es adaptativa. El orden de cada uno de los posibles clientes a ser insertados es actualizado en cada iteración de la fase de construcción, para reflejar los cambios provocados por la inserción del elemento anterior. El componente probabilístico de GRASP se caracteriza por elegir al azar uno de los mejores candidatos dentro del conjunto de clientes aún no rutados candidatos de ser insertados, pero no necesariamente el mejor candidato. La lista de los mejores candidatos se llama lista de candidatos restringidos (RCL).

Esta técnica de construcción permite diferentes rutas iniciales obtenidas en cada iteración GRASP. Una de las recomendaciones, que se proponen en las diferentes publicaciones, es la limitación del tamaño de la lista de candidatos restringidos. Si la lista únicamente permite un individuo, el resultado que obtendríamos en las diferentes iteraciones sería el mismo, dado que seleccionaríamos siempre al mejor candidato que minimice la métrica establecida. En el momento en el que se amplía el tamaño de la lista, la obtención de dos soluciones iniciales iguales se vuelve improbable. Si la lista de candidatos es pequeña, obtendremos soluciones iniciales cuyo número de giros necesarios para obtener la solución óptima será similar. Cuanto más grande sea la lista restringida, mayor variedad de soluciones iniciales obtendremos, teniendo también que realizar un mayor número de giros a posteriori.

En lo que respecta al criterio de parada, el que más se repite en la bibliografía es la realización de un número fijo de iteracciones, seleccionando la solución que menor costo obtenga. Una de las ventajas que obtenemos al emplear esta metaheurística es la obtención de soluciones parciales de un modo muy sencillo, ya que para cada iteración tendremos una solución parcial que ha quedado atrapada en un óptimo local.

Los resultados están recogidos en el epígrafe correspondiente del capítulo de resultados 4.3, donde se recoge un conjunto de tablas para los diferentes conjuntos de datos con los resultados parciales, y el resultado óptimo de la metaheurística.

3.2. Recocido Simulado (*Simulated annealing*)

Simulated Annealing es un enfoque algorítmico para la resolución de problemas de optimización combinatoria. El nombre del algoritmo deriva de una analogía entre resolver problemas de optimización y simular el recocido de sólidos. Esta analogía fue utilizada por primera vez en el artículo escrito por Metropolis en 1953[40].

El recocido simulado puede ser visto como una versión mejorada del método de mejora iterativa, en el que una solución inicial se mejora repetidamente haciendo intercambios locales hasta que tales cambios no producen una mejora en la solución. El recocido simulado aleatoriza el procedimiento de búsqueda local y en algunos casos permite cambios en la solución que la empeoran. Esto constituye un intento de reducir la probabilidad de quedar atrapado en una solución localmente subóptima, tal y como vimos en la Figura 3.1.

La naturaleza del algoritmo hace que pueda ser considerado como un proceso asintótico de optimización. En cualquier implementación práctica, sin embargo, se comporta como un algoritmo de aproximación. Los resultados que se han obtenido a lo largo del tiempo con este algoritmo, han sugerido que el método de recocido simulado es un enfoque eficiente para resolver problemas combinatorios difíciles y muy diversos.

En el conjunto de pruebas que se han realizado, se ha seguido el procedimiento planteado por Wen-Chyuan Chiang y Robert A. Russell en 1996[41]. Este procedimiento sigue los siguientes pasos:

1. Mediante los procedimientos de construcción de rutas, generar una solución inicial factible.
2. Fijar los parámetros propios del algoritmo de recocido simulado.
   - T la temperatura inicial del procedimiento que controlara el grado dispuesto a asumir una solución peor.
   - r el ratio de enfriamiento de la temperatura.
   - *Longitud* Duración de cada una de la épocas.
   - *Parada* criterio de finalización.
3. Mientras no cumplamos con el criterio de *Parada* que se haya fijado, debe llevarse a cabo lo siguiente:

a) Realizar tantas iteraciones como longitud de la época.
   1) Seleccionamos un intercambio factible de la solución actual.
   2) Cuantificamos el cambio de la función objetivo (Δ).
   3) Si el intercambio mejora la solución actual es aceptado.
   4) Si el intercambio no mejora la solución, será aceptado con una probabilidad $e^{-\frac{\Delta}{T}}$.

b) Realizar el enfriamiento del procedimiento. $T = r \cdot T$.

El "enfriamiento" progresivo que este procedimiento realiza, afecta a la hora de aceptar una solución que empeore la solución actual. En un principio, el algoritmo aceptará con gran facilidad una solución que empeore la solución actual, mientras que con el paso de las épocas, será más complicado aceptar soluciones que empeoren la actual, solo permitiéndose giros que mejoren la solución. En la bibliografía aparece representado el enfriamiento progresivo del algoritmo a lo largo de las diferentes épocas del siguiente modo:

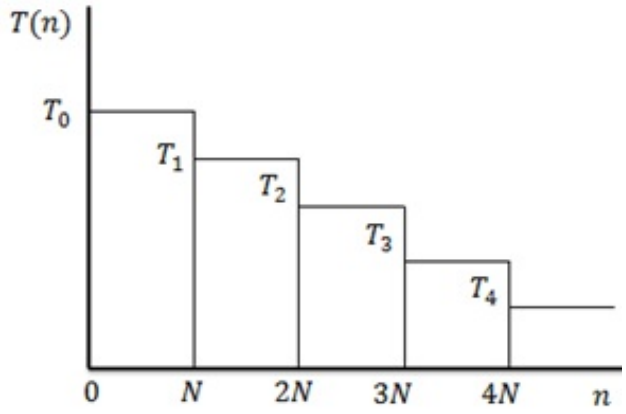


Figura 3.2: Esquema básico disminución del umbral de la temperatura[42]

A lo largo del tiempo han surgido varios enfoques en lo referente al criterio de parada que aplicar:

- Vincular el criterio a la temperatura del algoritmo, que sería lo mismo que vincular el criterio a un número fijo de épocas.
- Parar el algoritmo en el momento en el que en una época no se haya producido cambios.

El criterio de parada que se ha seleccionado para la realización de las pruebas ha sido el primero. Los resultados están recogidos en un conjunto de tablas en el epígrafe correspondiente del capítulo de resultados 4.3. El tiempo requerido para esta metaheurística está relacionado con el número de épocas que se seleccionen, es decir, relacionado con el número de giros que permitamos que empeoren la solución óptima.

Capítulo 4

Resultados computacionales

En el presente capítulo, se hará una recopilación de los resultados computacionales obtenidos mediante el conjunto de algoritmos que se han ido presentando. Buscaremos encontrar un procedimiento que proporcione un resultado aproximadamente óptimo, en un tiempo razonable.

El conjunto de datos que van a ser empleados para realizar la comparación entre los diferentes procedimientos serán los que generó Marius M. Solomon y que han sido empleados en numerosas publicaciones.

Solomon generó seis series de problemas. Su diseño destaca por varios factores que afectan al comportamiento de los algoritmos de enrutamiento. Son un conjunto de datos geográficos que pretenden abarcar la multitud de situaciones que se pudieran dar con un conjunto de datos reales, buscando influir en el número de clientes que pueden ser atendidos por un vehículo, variando el número de clientes que contienen limitaciones de tiempo, la estanqueidad y el posicionamiento de las ventanas de tiempo.

Los datos geográficos fueron generados aleatoriamente en los conjuntos de problemas R1 y R2, agrupados en conjuntos de problemas C1 y C2, y una mezcla de estructuras aleatorias y agrupadas en conjuntos de problemas por RC1 y RC2. Los conjuntos de problemas R1, C1 y RC1 tienen un horizonte de programación corto y sólo permiten unos pocos clientes por ruta (aproximadamente de 5 a 10). Por el contrario, los conjuntos R2, C2 y RC2 tienen un horizonte de programación largo que permite que muchos clientes (más de 30) sean atendidos por el mismo vehículo.

Las coordenadas del cliente son idénticas para todos los problemas dentro de un tipo (es decir, R, C y RC). Los problemas difieren con respecto a la anchura de las ventanas de tiempo. Algunos tienen ventanas de tiempo muy estrecho, mientras que otros tienen ventanas de tiempo que son apenas restrictivas. En términos de densidad de ventana de tiempo, es decir, el porcentaje de clientes con ventanas de tiempo, he creado problemas con 25, 50, 75 y 100 % de ventanas de tiempo.

Para cada conjunto de datos disponemos de la información de 100 clientes, más los datos del depósito. A lo largo del tiempo, este conjunto de datos ha sido empleado de diversas maneras, utilizando el conjunto global de clientes, o seleccionando sólo los primeros 25 o 50 clientes. En las pruebas que se han realizado, el número de clientes que ha sido el máximo, es decir, 100 clientes.

4.1. Resultados modelo exacto

En las siguientes tablas recogeremos el valor mínimo de distancia que mediante el procedimiento exacto se ha logrado alcanzar. Dado que la complejidad del problema es alta, se puso una cota de computo de cuatro horas y media, por tanto para todos los conjuntos de datos que superen un tiempo de 16.450 segundos serán problemas para los que no tendremos la garantía de que el resultado que hemos obtenido sea el mínimo.

Conjunto	Distancia mínima	Tiempo empleado	Conjunto	Distancia mínima	Tiempo empleado
C101	828.937	0.41	C201	591.557	16.594
C102	833.733	16450.6	C202	591.557	1.59
C103	967.673	16463.4	C203	588.493	16451
C104	926.344	16451.7	C204	621.529	16452
C105	828.937	1.72	C205	588.493	30.736
C106	828.937	6.206	C206	588.493	33.555
C107	828.937	2.58	C207	588.286	652.351
C108	828.937	1627.91	C208	588.324	3140.88

Cuadro 4.1: Resultados del modelo exacto conjunto C

En la tabla 4.1 observamos que para el conjunto de problemas con una estructura agrupada, en la mayoría de las ocasiones, el procedimiento exacto

ha logrado obtener la solución óptima. También observamos que el modelo ha funcionado mejor en el conjunto de datos cuyas rutas están compuestas por un mayor número de clientes.

Conjunto	Distancia mínima	Tiempo empleado	Conjunto	Distancia mínima	Tiempo empleado
R101	1388.24	1.79	R201	1099.62	4696.8
R102	1254.53	16453	R202	1085.37	16450.3
R103	1066.75	16453.2	R203	968.887	16451.5
R104	1045.99	16450.5	R204	815.895	16451.1
R105	1164.95	16451.2	R205	948.936	16456.4
R106	1127.93	16462.4	R206	915.002	16451.6
R107	1015.83	16451.3	R207	888.398	16469
R108	967.582	16452.1	R208	777.24	16453.1

Cuadro 4.2: Resultados del modelo exacto conjunto R

En la tabla 4.2 observamos que para el conjunto de problemas con una estructura aleatoria, en la mayoría de las ocasiones, el procedimiento exacto no ha logrado obtener la solución óptima.

Conjunto	Distancia mínima	Tiempo empleado	Conjunto	Distancia mínima	Tiempo empleado
RC101	1368.12	16456.3	RC201	1231.12	16451.7
RC102	1276.11	16451.1	RC202	1136.29	16451.3
RC103	1361.62	16452.4	RC203	1058.57	16460.4
RC104	1230.53	16451.9	RC204	928.222	16452.8
RC105	1284.21	16451.6	RC205	1168.03	16459.5
RC106	1252.24	16451.5	RC206	1088.49	16451.5
RC107	1279.19	16451.3	RC207	1093.73	16451.4
RC108	1189.8	16461.2	RC208	800.111	16453.6

Cuadro 4.3: Resultados del modelo exacto conjunto RC

En la tabla 4.3 observamos que para el conjunto de problemas con una estructura, mezcla entre aleatoria y agrupada, en todas las ocasiones, en ninguna de las ocasiones el procedimiento exacto ha logrado obtener la solución óptima.

Tras ver el conjunto de resultados anteriores, y siendo conscientes de que

el tiempo de cómputo que hemos destinado en los diferentes problemas, podría ser considerado ya como un tiempo inasumible en una implementación comercial, justificando la utilización del conjunto de procedimientos algoritmos que han sido descritos y descartando el modelo matemático exacto.

4.2. Resultados modelo heurístico

Una vez que somos conscientes que la solución óptima no va a poder ser alcanzada, procedemos a la comparación de un conjunto de heurísticas. Es necesario recordar el procedimiento bifásico que se sigue:

1. Generar una primera ruta inicial, solución factible del problema, y en la mayoría de las ocasiones no óptima. Esta primera ruta inicial será generada mediante tres procedimientos diferentes que han sido planteados a lo largo del tiempo:
   - *a*) El procedimiento propuesto por Marius M. Solomon, algoritmo del siguiente vecino más próximo. Es un procedimiento rápido, pero en la mayorías de las ocasiones queda muy lejano a la solución óptima. Este procedimiento ha sido descrito en el apartado 2.1.1.
   - *b*) El procedimiento paramétrico propuesto por G. Ioannou, G. Prastacos y M. Kritikos en 2001, que fue planteado como un método revolucionario para la generación de soluciones iniciales. Este procedimiento fue descrito en el apartado 2.1.3.
   - *c*) El procedimiento simplificado propio que ha sido descrito en el apartado 2.1.4, con la intención de valorar su comportamiento.
2. En una segunda fase, el procedimiento intenta, mediante giros, la reducción del coto óptimo. Esta fase se divide en dos categorías:
   - *a*) Mejora variando el orden de los clientes que componen una ruta, estos intercambios son conocidos como las heurísticas de mejora intra-rutas y fueron propuestos por diferentes autores. En este proyecto, podemos encontrar una descripción en el apartado 2.2.1.
   - *b*) Mejora intercambiando clientes entre las diferentes rutas que han sido construidas. Estos intercambios son conocidos como las heurísticas de mejora entre-rutas y fueron propuestos por Alex Van Breedam. En este proyecto, podemos encontrar una descripción en el apartado 2.2.2.

En las implementaciones que se han realizado, han sido empleadas todas las heurísticas de mejora entre-rutas y la heurística *or-opt* que contenía la mejora *2-opt*. Dado que el conjunto de datos de los que se disponían era un conjunto amplio, mostraremos los resultados con los primeros conjuntos de cada clase.

		Sin mejora		Con mejora	
Fichero	Heurística	Distancia	Tiempo	Distancia	Tiempo
C101	Vecino más cercano	878	74	829	208
	Solomon *nearest neighbor*	1871	0	829	451
C102	Vecino más cercano	1015	90	829	366
	Solomon *nearest neighbor*	1992	0	829	460
C201	Vecino más cercano	756	754	592	2191
	Solomon *nearest neighbor*	1880	0	693	1704
C202	Vecino más cercano	1023	721	592	3840
	Solomon *nearest neighbor*	1603	0	676	2119
R101	Vecino más cercano	2052	45	1681	464
	Solomon *nearest neighbor*	2623	0	1721	426
R102	Vecino más cercano	1795	61	1491	427
	Solomon *nearest neighbor*	2443	0	1508	571
R201	Vecino más cercano	1900	796	1331	2455
	Solomon *nearest neighbor*	1985	0	1276	731
R202	Vecino más cercano	1653	1481	1222	3506
	Solomon *nearest neighbor*	1813	0	1189	709
RC101	Vecino más cercano	2185	52	1767	469
	Solomon *nearest neighbor*	2711	0	1776	448
RC102	Vecino más cercano	2036	57	1519	493
	Solomon *nearest neighbor*	2647	0	1585	475
RC201	Vecino más cercano	2204	540	1477	1895
	Solomon *nearest neighbor*	2468	0	1458	807
RC202	Vecino más cercano	2186	754	1357	3178
	Solomon *nearest neighbor*	2114	0	1215	863

Cuadro 4.4: Resultados del modelo heurístico

En la tabla 4.4 se compara el modelo de Marius M. Solomon con la propuesta propia. En la tabla vemos que el tiempo requerido para obtener una primera solución mediante la propuesta de Solomon es menor a 1 segundo, con lo que obtenemos una primera solución de una manera rápida, siendo

esta solución de peor calidad a la que obtenemos mediante la heurística propia. Con la heurística propia obtenemos aproximaciones a un conjunto de solución de gran calidad, invirtiendo, inicialmente algo más de tiempo.

Podemos concluir que si queremos disponer de una solución, independientemente de la calidad que ésta tenga, deberíamos emplear la propuesta de Solomon; mientras que si nuestro objetivo es la obtención de una primera solución razonablemente óptima, deberíamos emplear la propuesta propia.

En lo referente a la solución obtenida tras aplicar las heurísticas de mejora, podemos ver que es preferible partir de una solución cercana a la óptima, es decir, partir de una solución inicial de alta calidad, dado que el tiempo que necesitaríamos para realizar la aproximación a la solución óptima menor encontrada sería menor. Vemos también que entre la propuesta de Solomon y la propuesta propia, en numerosas ocasiones, el tiempo empleado en la obtención de una solución de alta calidad no compensa el tiempo que requiere el otro procedimiento.

Procedemos a valorar los resultados obtenidos mediante el procedimiento propuesto por G. Ioannou, G. Prastacos y M. Kritikos en 2001, que era una heurística de construcción paramétrica. Tal y como recomiendan estos autores, hemos realizado la prueba con una configuración de parámetros cuya diferencia entre ellos es una décima.

						Sin mejora	
b1	b2	b3	bs	be	br	Distancia	Tiempo
0	0.7	0.3	0.1	0	0.9	853	75
0	0.8	0.2	0.1	0	0.9	853	75
0	0.9	0.1	0.1	0	0.9	853	76
0.1	0.6	0.3	0.1	0	0.9	853	76
0.8	0	0.2	0	0.9	0.1	2230	61
0.8	0.1	0.1	0	0.9	0.1	2230	61
0.8	0.1	0.1	0.1	0.8	0.1	2230	61
0.8	0.2	0	0	0.9	0.1	2230	62
0	0.1	0.9	0.9	0	0.1	4205	61
0.1	0	0.9	0.9	0	0.1	4221	57
0	0	1	0.8	0	0.2	4375	60
0	0	1	0.9	0	0.1	4586	59

Cuadro 4.5: Resultados del modelo heurístico IMPACT con C101

En la tabla 4.5, vemos que el resultado final es fuertemente dependiente de los parámetros que ponderan a las diferentes métricas. Los resultados que han sido seleccionados para mostrar corresponden a tres grupos, los cuatro primeros son los mejores resultados que habríamos obtenido si hubiéramos acertado con la configuración correcta de parámetros, los siguientes cuatro pertenecen al conjunto intermedio de resultados obtenidos con las diferentes configuraciones, el último grupo está compuesto por las cuatro peores configuraciones de parámetros existentes.

Únicamente se ha explorado el espacio total de soluciones con el conjunto de datos C101, por el gran costo computacional que requiere. Abarcando la exploración de los cuatro parámetros, produciendo variaciones en ellos de décimas, tal y como proponen sus autores, y cumpliendo el conjunto de restricciones que subyacen a cada uno de los dos grupos. Nos encontramos que tenemos que generar un conjunto de 4358 soluciones iniciales, para el conjunto que hemos analizado, y que ha requerido de un tiempo de 3 días 11 horas 16 minutos 47.4 segundos.

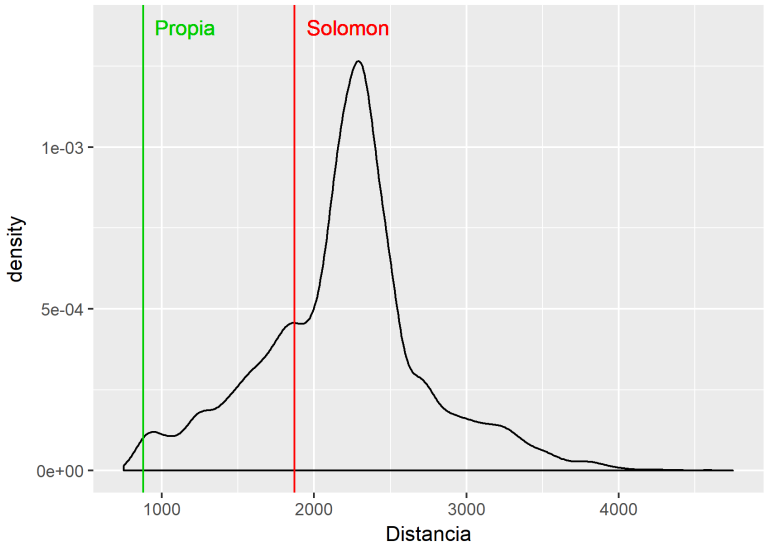


Figura 4.1: Distribución de las soluciones mediante IMPACT

En la figura 4.1 vemos como la distribución de las diferentes posibles soluciones que obtenemos con este procedimiento IMPACT, son soluciones cuyo costo es superior, en la gran mayoría de las ocasiones, a la propuesta de Solomon. Por lo tanto el costo computacional que tenemos al emplear este procedimiento no compensa con la solución que obtendríamos mediante

el procedimiento de Solomon, si la elección de parámetros se realizara de manera aleatoria. Por otro lado, vemos que la solución obtenida mediante el procedimiento propio se encuentra entre las mejores soluciones obtenidas con el procedimiento IMPACT.

En la figura 4.2 se realiza una comparación de cada uno de los tiempos computacionales requeridos por cada uno de los procedimientos. En ella se ve cómo la solución propia requiere de más tiempo que del procedimiento IMPACT, mientras que el tiempo requerido por la propuesta de Solomon podría considerarse como cero, siendo éste el motivo de que no aparezca en la imagen.

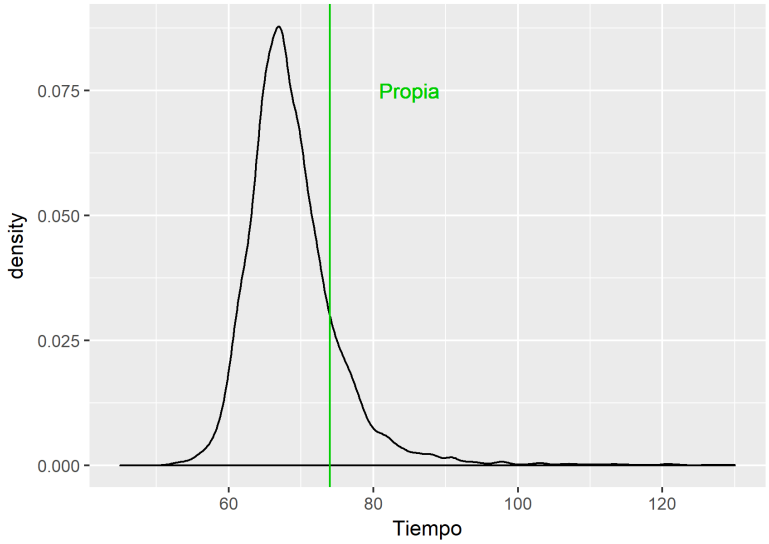


Figura 4.2: Distribución del tiempo de construcción mediante IMPACT

En el apéndice A aparecen recogidos un conjunto de gráficos que describen cada una de las rutas que se han obtenido como solución de los procedimientos mostrados en las tablas anteriores.

4.3. Resultados modelo metaheurístico

El conjunto de modelos anteriores tienen el problema que hemos comentado en capítulos anteriores, ejecución tras ejecución, la solución que se obtendría sería la misma, explorando siempre el mismo conjunto de soluciones.

Lo que haría que fuéramos incapaces de encontrar una mejor solución aún cuando el tiempo de computación que dedicáramos fuera elevado.

Procedo a adjuntar el conjunto de soluciones que obtendríamos con la ejecución de las dos metaheurísticas que hemos descrito en el capítulo 3.

En primer lugar tenemos los resultados de la metaheurística GRASP (*Greedy Randomized Adaptive Search Procedures*), que apuesta por la aleatorización controlada del conjunto de heurísticas de generación de rutas iniciales. Esta metaheurística fue descrita en el apartado 3.1. La implementación que se ha realizado, selecciona al candidato de ser insertado en la ruta en construcción de una lista restringida de tres candidatos, este número fue fijado como consecuencia de que si la longitud de la lista es larga, el número de intercambios que tendremos que realizar mediante las heurísticas de mejora de rutas será grande, mientras que si el número es razonablemente pequeño, la ruta construida no diferirá en exceso la resultante cuando la lista únicamente esta compuesta por un único individuo.

		Sin mejora		Con mejora	
Fichero	Repetición	Distancia	Tiempo	Distancia	Tiempo
C101	1	3218	0	895	529
C101	2	2871	0	829	706
C101	3	3190	0	893	556
C201	1	2799	0	669	2227
C201	2	2593	0	665	2355
C201	3	2849	0	689	1513

Cuadro 4.6: Resultados del modelo metaheurístico GRASP C

		Sin mejora		Con mejora	
Fichero	Repetición	Distancia	Tiempo	Distancia	Tiempo
R101	1	2922	0	1719	470
R101	2	2836	0	1708	474
R101	3	3060	0	1724	466
R201	1	2530	0	1274	669
R201	2	2463	0	1249	692
R201	3	2448	0	1307	718

Cuadro 4.7: Resultados del modelo metaheurístico GRASP R

		Sin mejora		Con mejora	
Fichero	Repetición	Distancia	Tiempo	Distancia	Tiempo
RC101	1	3208	0	1727	555
RC101	2	3047	0	1855	532
RC101	3	3207	0	1744	539
RC201	1	3287	0	1424	690
RC201	2	3347	0	1432	815
RC201	3	3220	0	1429	899

Cuadro 4.8: Resultados del modelo metaheurístico GRASP RC

En las tablas anteriores vemos el comportamiento de la metaheurística GRASP, cuando el algoritmo de construcción de rutas empleado es el propuesto por Solomon. Se realizó esta selección de algoritmo de construcción de rutas por la rápida generación de rutas iniciales. La solución final que proporcionaría el procedimiento sería la repetición de menor costo, pero conservar cada una de las soluciones intermedias, brinda al usuario la posibilidad de elegir la que mejor se adecúe a sus necesidades.

En segundo lugar tenemos los resultados obtenidos mediante la metaheurística *simulated annealing*, que apuesta por la aleatorización del espacio de soluciones factibles, permitiendo en algunas ocasiones empeorar la mejor solución encontrada. Esta metaheurística fue descrita en el apartado 3.2. El conjunto que esta metaheurística necesita ha sido fijado con las siguiente cantidades:

1. $Temperatura = 500$
2. $r = 0.85$
3. $Longitud = 100$
4. *Parada* evaluar 100 épocas.

Fichero	Repetición	Distancia	Tiempo Acumulado
C101	0	1871	0
C101	3	3560	26
C101	50	1359	571
C101	101	854	1899

Fichero	Repetición	Distancia	Tiempo Acumulado
C201	0	1880	0
C201	3	3447	113
C201	50	854	3742
C201	101	689	16072

Cuadro 4.9: Resultados del modelo metaheurístico SA C

Fichero	Repetición	Distancia	Tiempo Acumulado
R101	0	2623	0
R101	3	3095	10
R101	50	2192	185
R101	101	1795	642
R201	0	1985	0
R201	3	3255	93
R201	50	1407	3125
R201	101	1249	7086

Cuadro 4.10: Resultados del modelo metaheurístico SA R

Fichero	Repetición	Distancia	Tiempo Acumulado
RC101	0	2711	0
RC101	3	3217	18
RC101	50	2010	288
RC101	101	1786	958
RC201	0	2468	0
RC201	3	4132	85
RC201	50	1666	3538
RC201	101	1342	8632

Cuadro 4.11: Resultados del modelo metaheurístico SA RC

En las tablas anteriores vemos el arranque de la metaheurística, iteraciones intermedias, y la solución final que nos proporcionaría. La heurística de construcción que ha empleado ha sido la de Solomon. En la figura 4.3 vemos la evolución de la aproximación a la solución final en cada uno de los conjuntos de datos. Se vé como la metaheurística en un principio acepta la mayoría

de permutaciones de la solución, aún cuando este intercambio aumente la solución óptima, desde un punto. El procedimiento solo acepta soluciones que disminuya el costo global.

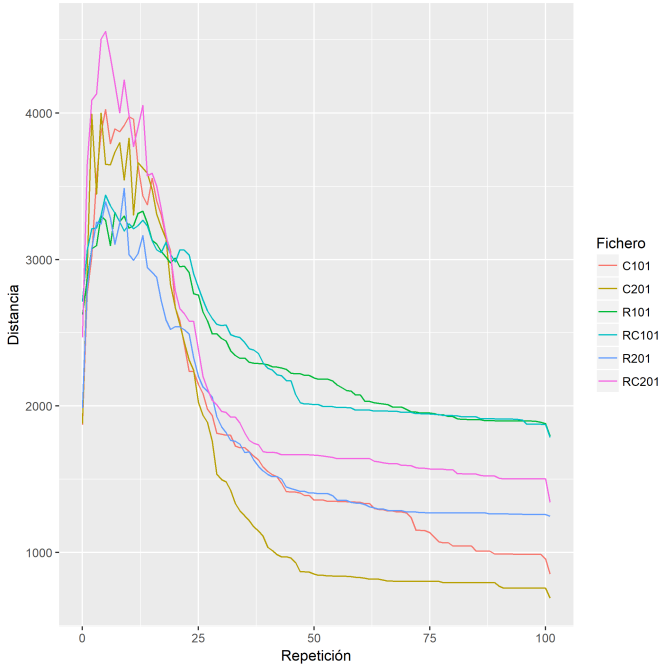


Figura 4.3: Evolución de las soluciones mediante SA

Simulated annealing padece el mismo problema que la heurística parámetrica IMPACT. Al requerir fijar un conjunto de parámetros por parte del usuario, esto puede hacer que la metaheurística tarde un mayor tiempo en realizar la exploración del espacio de soluciones, penalizando esta metaheurística frente a la exploración que realiza GRASP.

Capítulo 5

Conclusiones

A lo largo de todo el documento hemos ido analizando el problema de recogida de mercancias con ventanas de tiempo (VRPTW, *Vehicle Routing Problem with Time Windows*). El problema que se ha estudiado se encuentra dentro de la categoría de problemas NP-complejos, por tanto la obtención de la solución óptima podría llegar a considerarse como un objetivo inalcanzable, tal y como se ha visto.

La imposibilidad de alcanzar la solución óptima, nos lleva a la necesidad de plantearnos su resolución mediante un procedimiento heurístico. La utilización de heurísticas nos hace asumir desde un principio, que no podremos tener la certeza de que la solución que obtengamos sea la óptima.

Aún cuando la obtención de la solución óptima se antoja inalcanzable, tiene sentido dedicar esfuerzos en la obtención de soluciones aproximadas por la gran aplicabilidad del problema.

El problema nos lo encontramos en las decisiones que diariamente tienen que tomar los jefes de tráfico de las empresas de logística, los cuales tienen que hallar, de la mejor manera posible, el modo en el que asignar los recursos de los que disponen para maximizar los beneficios de la empresa. Pero el problema no aparece asociado únicamente al ámbito de rutas de vehículos. A lo largo de la historia ha sido aplicado en multitud de situaciones, optimizando el movimiento que tiene que realizar un brazo mecánico en un proceso de fabricación, optimizando el movimiento de telescopios (*slewing*), y más recientemente ha sido aplicado en la industria de los videojuegos.

Comenzando en los enfoques clásicos, que son puramente deterministas y que acostumbran a quedarse atrapados en mínimos locales de la función objetivo, hemos estudiado después un conjunto de metaheurísticas. Las metaheurísticas para el problema VRP, pueden encontrar excelentes soluciones a expensas de un coste de tiempo de cálculo notablemente superior. Las metaheurísticas son bastante lentas, pero también proporcionan mejores soluciones. Típicamente, los métodos clásicos producen valores de solución entre 2 % y 10 % por encima del valor óptimo, mientras que la cifra correspondiente a la mejor implementación metaheurística es a menudo inferior al 0,5 %[4]. Esta conclusión quedaba ya reflejada en el libro publicado por Paolo Toth y Daniele Vigo en el 2002[2] y lo ilustraban con la siguiente Figura 5.1.

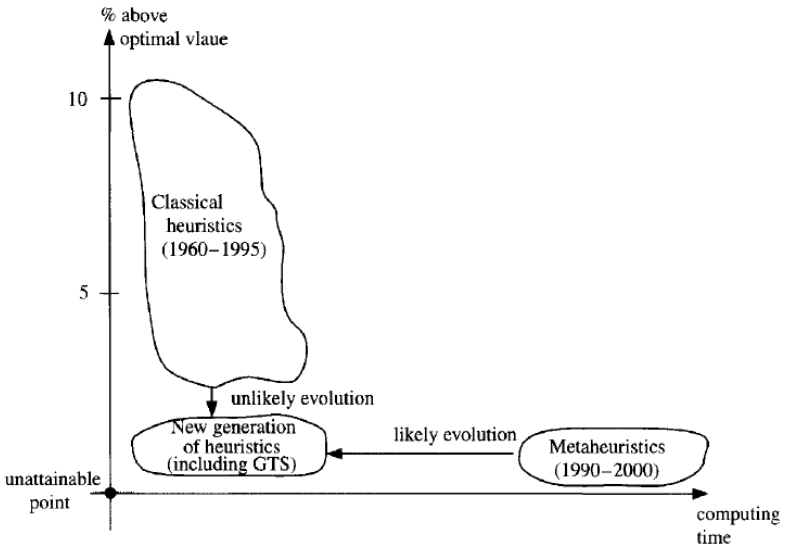


Figura 5.1: Evolución de las heurísticas para el problema VRP

Las conclusiones a las que llegan Paolo Toth y Daniele Vigo[2], son las mismas a las que llegan Olli Bräysy y Michel Gendreau[43] años después. En ambas publicaciones se pone de manifiesto que la hibridación de las diferentes metaheurísticas, podría dar buenos resultados. Este conjunto de autores es consciente, que los resultados que se han obtenido mediante procedimientos, de búsqueda tabú, recocido simulado, colonias de hormigas y procedimientos GRASP, han dado mejores resultados que los que se han obtenido mediante algorítmicos genéticos y redes neuronales, pero proponen que las futuras líneas de investigación estén centradas en su desarrollo, dado que si se com-

[4]Fragmento traducido del libro de Paolo Toth y Daniele Vigo. *The Vehicle Routing Problem*, 2002

binan adecuadamente podríamos obtener algoritmos capaces de resolver el problema de un mejor modo.

En referencia a los procedimientos de hibridación que se propuso a principios de siglo, han aparecido diversas publicaciones recientemente. Christian Prins (2009)[44] ha tratado este concepto empleándolo en la metaheurística GRASP, y fue el año pasado cuando apareció escrito por Mauricio G.C. Resende y Celso C. Ribeiro[39] un relato de GRASP especialmente encomiable por su legibilidad, que abarcan muchas facetas de esta metaheurística, como la construcción de soluciones, búsqueda, hibridaciones y extensiones.

Debemos de ser conscientes en todo momento, de que los resultados y comparaciones que a lo largo de la historia se han realizado de los diferentes algoritmos debería de ser tomado con cautela. Todos los procedimientos han sido probados en diferentes conjuntos de datos, en la gran mayoría con un número reducido de clientes, siendo esto uno de los principales motivos por los que no se haya podido realizar una ordenación de calidad de las propuestas. Hay que ser conscientes de que un enfoque de resolución para un problema en la vida real puede contener un conjunto de restricciones "sucias", es decir, poco modelables de primeras con el planteamiento presentado, este problema se lo encontró Rousseau[45].

Como futuras líneas de investigación podría proponerse, además del trabajo en métodos de hibridación, la adecuación de las diferentes heurísticas y metaheurísticas a conjuntos de datos cuya matriz de distancia no es simétrica, es decir, que la distancia entre dos clientes es diferentes dependiendo del orden en el que son visitados dichos clientes. Otra línea de desarrollo sería la adecuación del conjunto de algoritmos a problemas más complejos como son el de recogida y distribución de mercancías.

Podemos concluir remarcando que para cada conjunto de datos deberemos realizar una búsqueda de la solución óptima siguiendo todos los procedimientos heurísticos de los que dispongamos, dado que ninguno de ellos nos garantiza una solución óptima, y no sabemos cuál de todos ellos tendrá un comportamiento mejor con el nuevo conjunto de datos del que necesitamos obtener una solución. Debemos tener presente en todo momento, los buenos resultados que se han obtenido siempre con la aleatorización controlada, las estructuras de datos eficientes y el procesamiento paralelizado. Del conjunto de algoritmos que han sido propuestos en este trabajo, el que mejores resultados ha obtenido ha proporcionado la metaheurística GRASP, por lo que es recomendable su utilización.

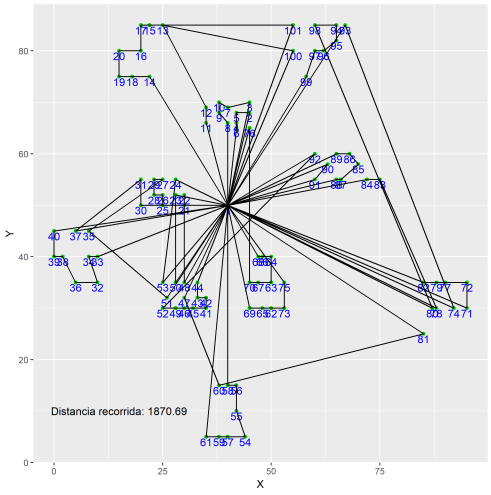
Apéndice A

Rutas modelos heurísticos

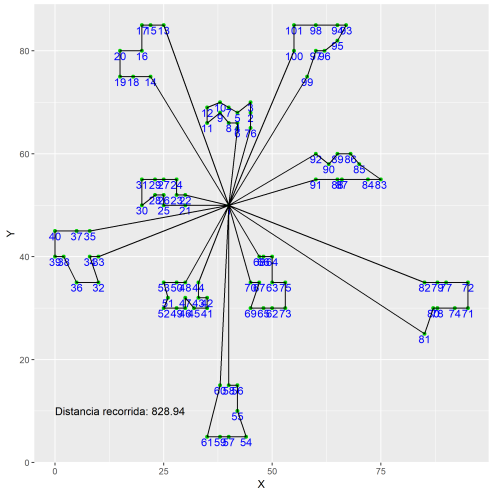
En este apéndice, se recoge un conjunto de imágenes donde se puede realizar una comparación de las soluciones de rutas que han sido generadas por cada una de las heurísticas.

En el primer grupo de imágenes se mostrará la comparación en el resultado entre la propuesta realizada por Marius M. Solomon y la propuesta propia. Estos resultados aparecían de una manera menos detallada descritos en la tabla 4.4, donde además de la solución alcanzada veíamos el tiempo computacional que había sido empleado para alcanzar dicho resultado.

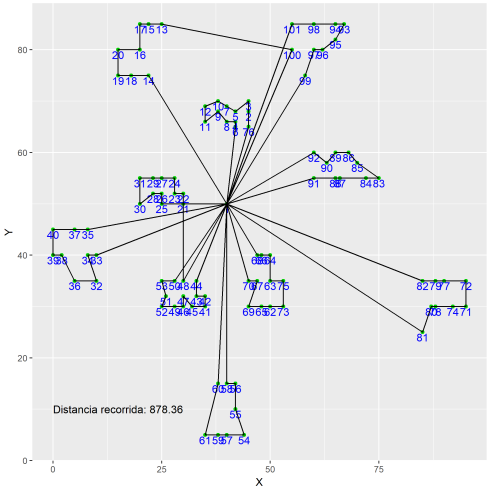
En un segundo bloque de imágenes nos encontramos los resultados tras la aplicación del algoritmo paramétrico IMPACT, este conjunto de resultados nos los encontrábamos en la tabla 4.5, donde veíamos una descripción de resultados de gran calidad, de calidad media y de baja calidad.



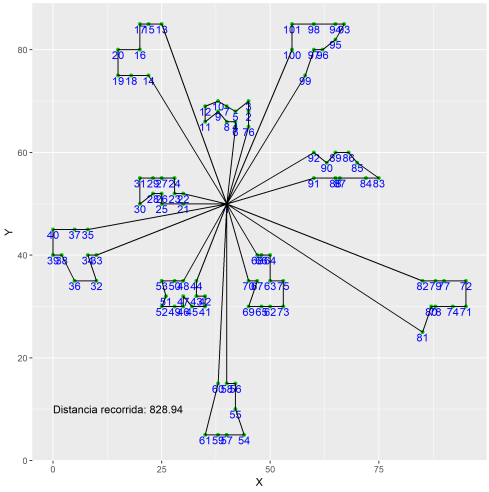
(a) Siguiente más próximo (comienzo)



(b) Siguiente más próximo (mejorados)

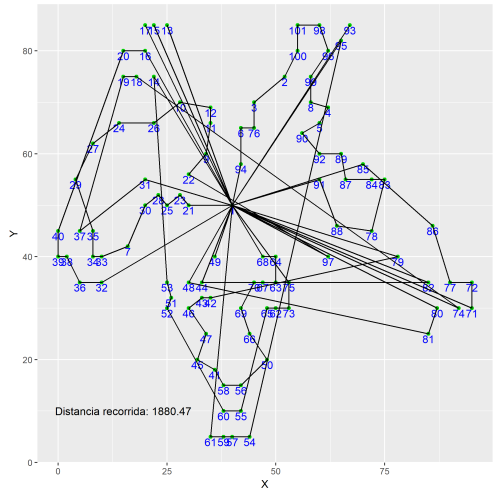


(c) Más próximo (comienzo)

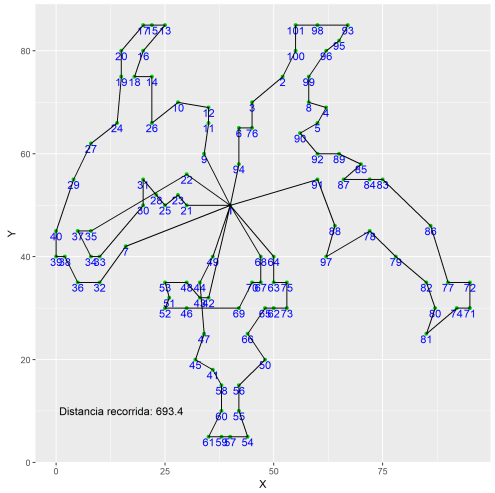


(d) Más próximo (mejorados)

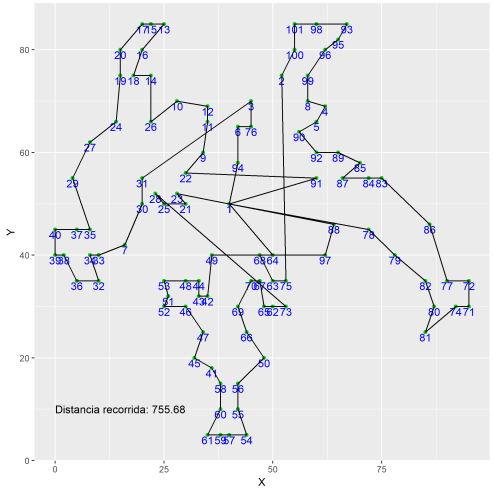
Figura A.1: Gráficos del conjunto de datos C101



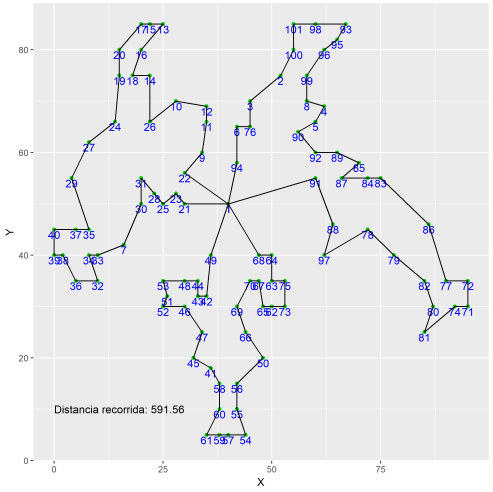
(a) Siguiente más próximo (comienzo)



(b) Siguiente más próximo (mejorados)

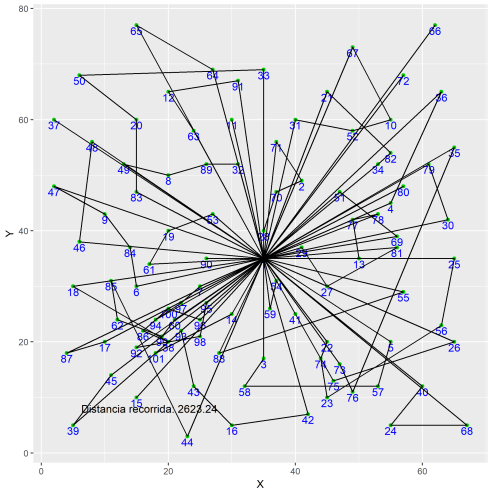


(c) Más próximo (comienzo)

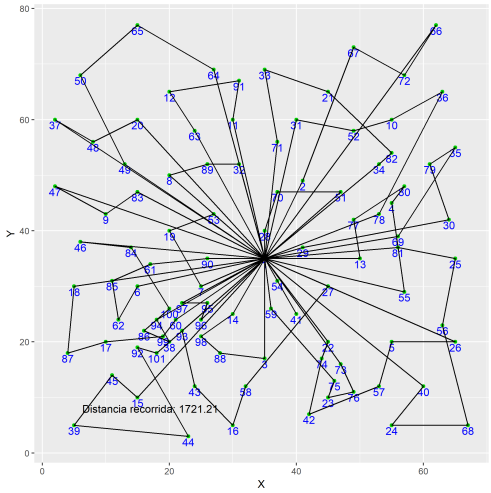


(d) Más próximo (mejorados)

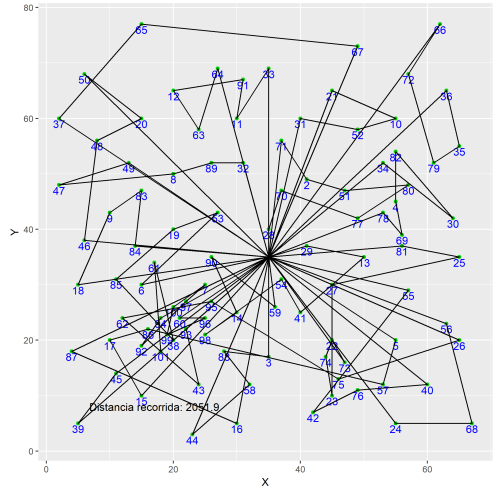
Figura A.2: Gráficos del conjunto de datos C201



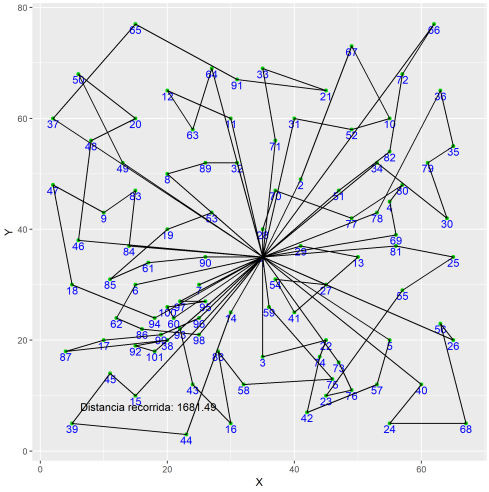
(a) Siguiente más próximo (comienzo)



(b) Siguiente más próximo (mejorados)

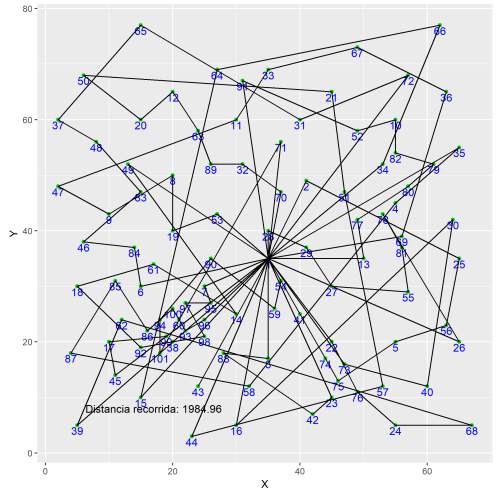


(c) Más próximo (comienzo)

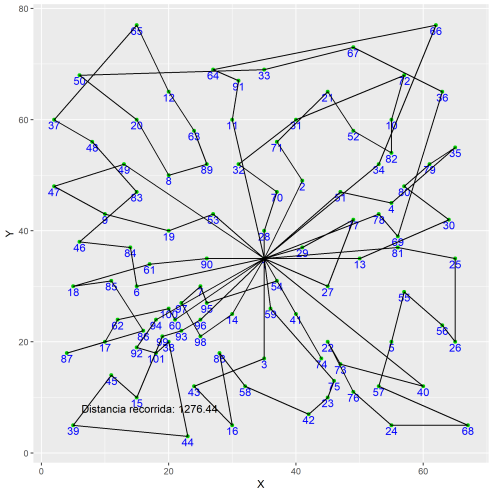


(d) Más próximo (mejorados)

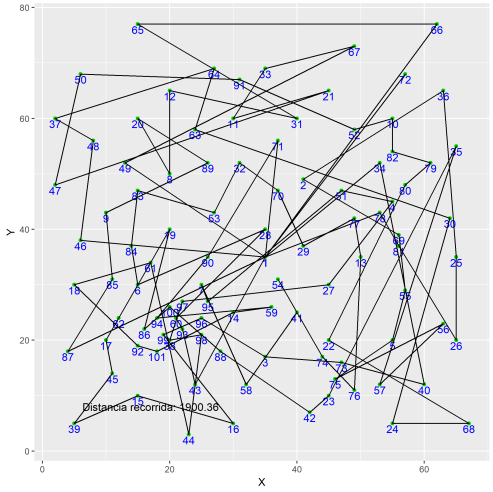
Figura A.3: Gráficos del conjunto de datos R101



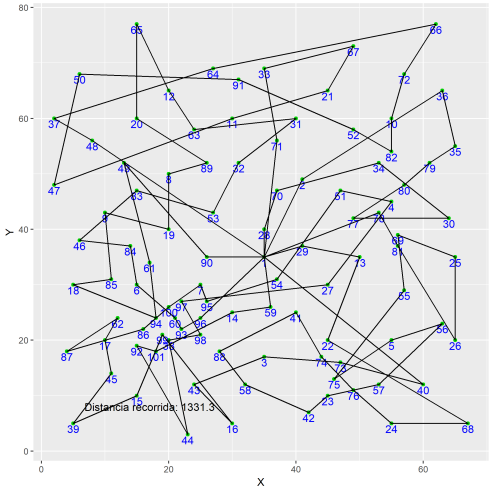
(a) Siguiente más próximo (comienzo)



(b) Siguiente más próximo (mejorados)

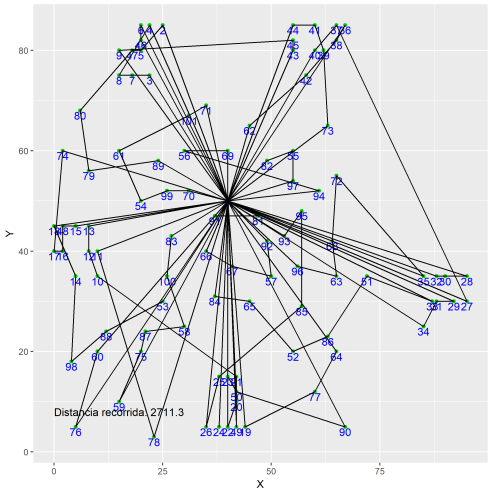


(c) Más próximo (comienzo)

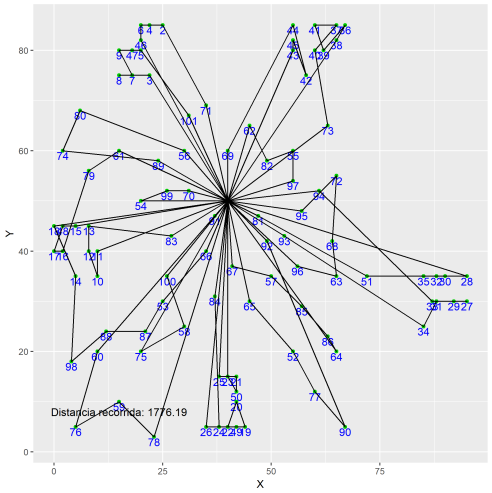


(d) Más próximo (mejorados)

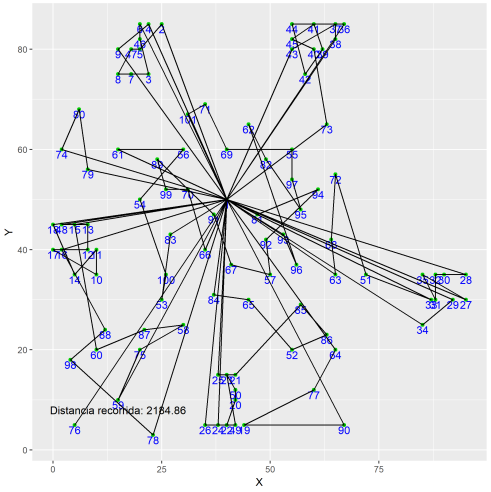
Figura A.4: Gráficos del conjunto de datos R201



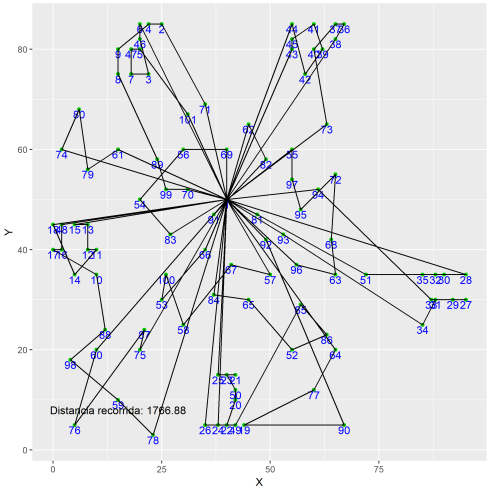
(a) Siguiente más próximo (comienzo)



(b) Siguiente más próximo (mejorados)

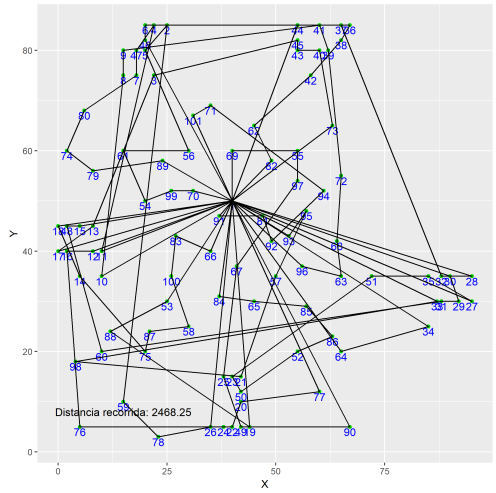


(c) Más próximo (comienzo)

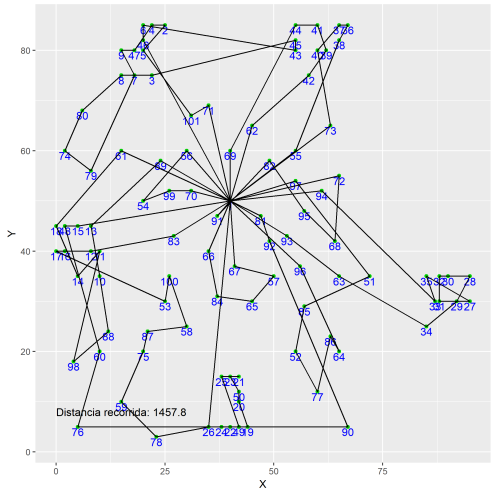


(d) Más próximo (mejorados)

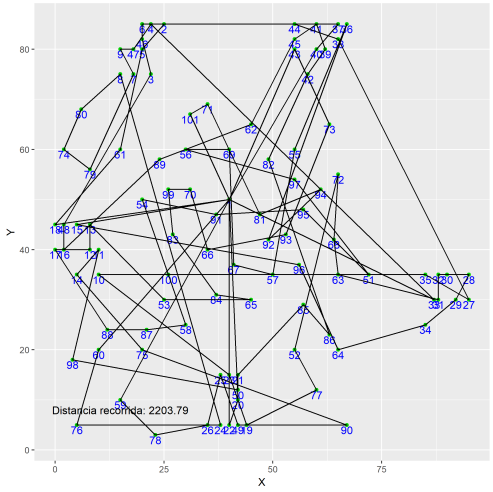
Figura A.5: Gráficos del conjunto de datos RC101



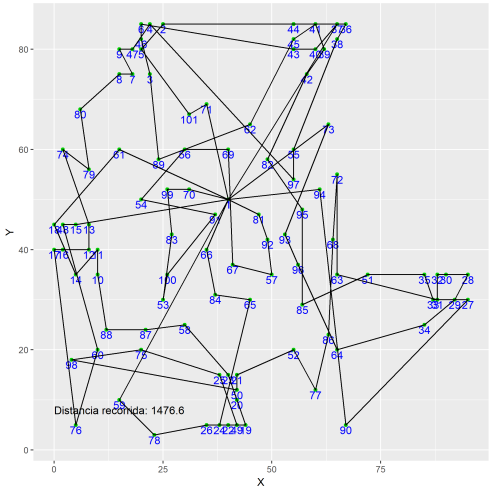
(a) Siguiente más próximo (comienzo)



(b) Siguiente más próximo (mejorados)

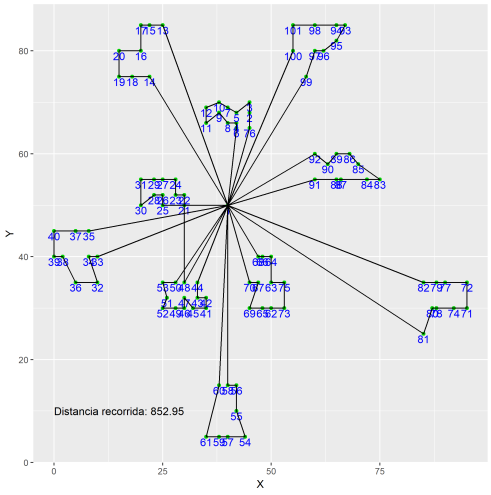


(c) Más próximo (comienzo)

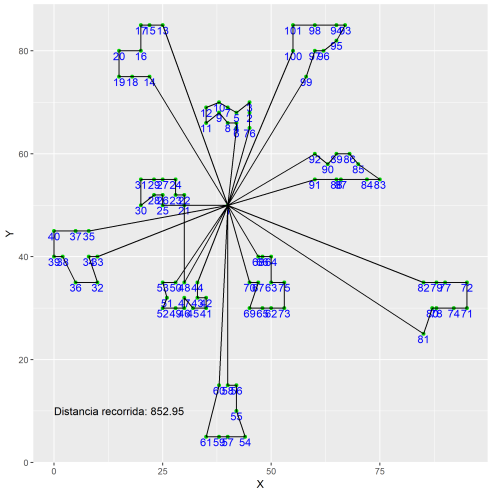


(d) Más próximo (mejorados)

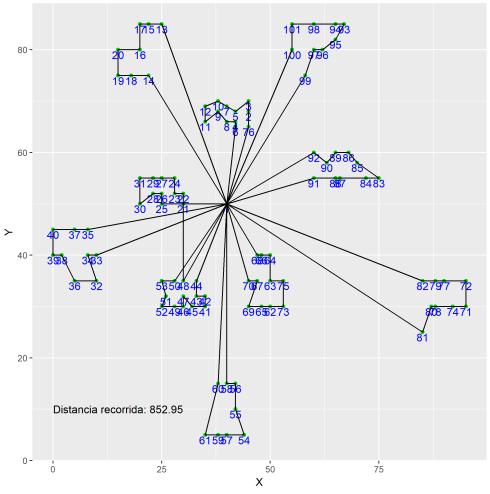
Figura A.6: Gráficos del conjunto de datos RC201



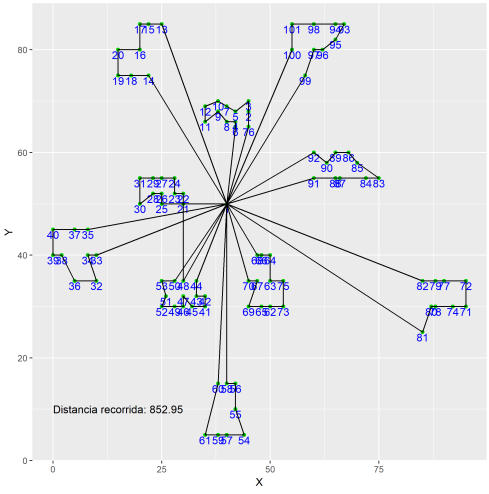
(a) IMPACT 0-07-03-01-0-09



(b) IMPACT 0-08-02-01-0-09

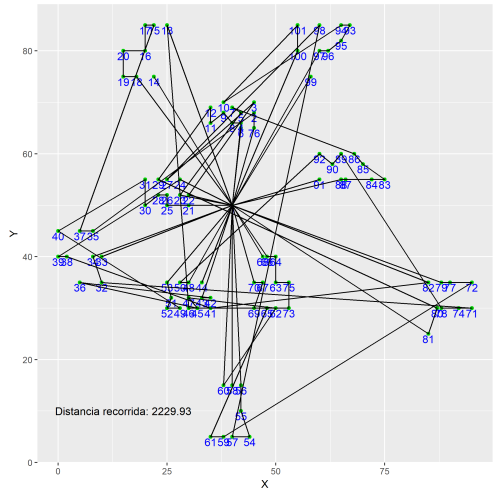


(c) IMPACT 0-09-01-01-0-09

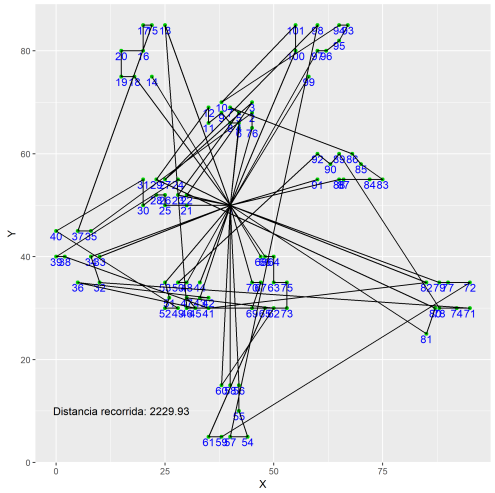


(d) IMPACT 01-06-03-01-0-09

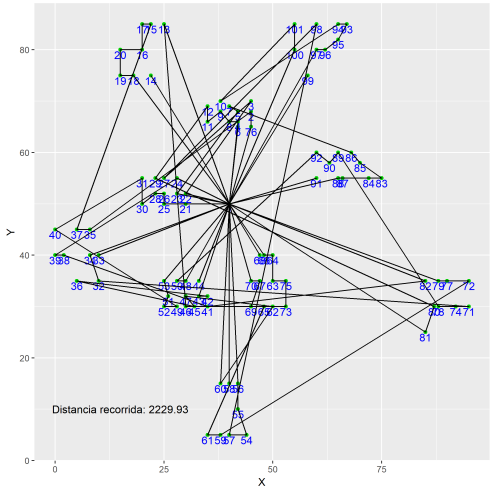
Figura A.7: Gráficos del conjunto de datos C101 IMPACT Mejores



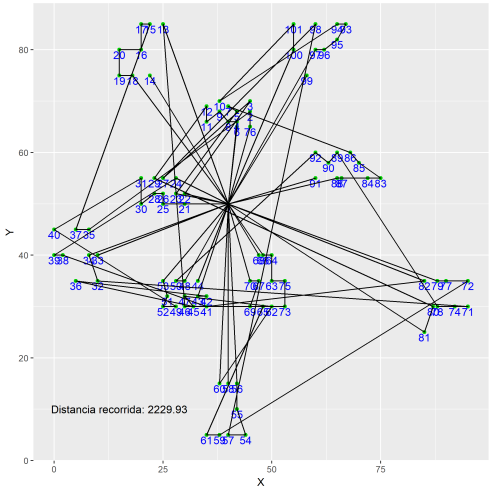
(a) IMPACT 08-0-02-0-09-01



(b) IMPACT 08-01-01-0-09-01

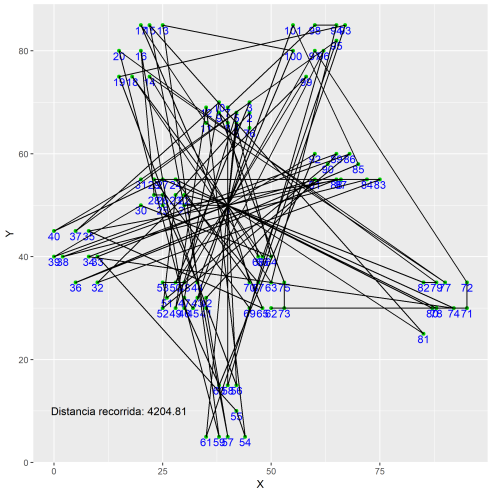


(c) IMPACT 08-01-01-01-08-01

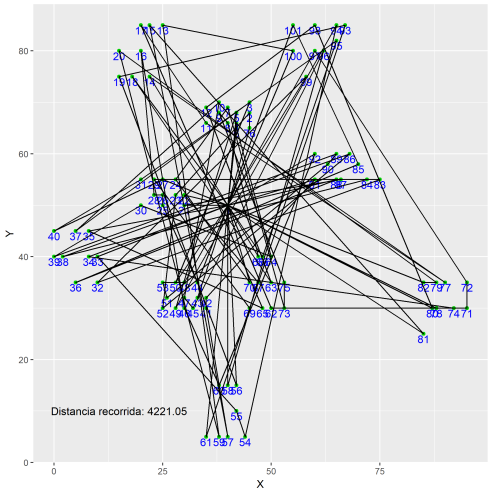


(d) IMPACT 08-02-0-0-09-01

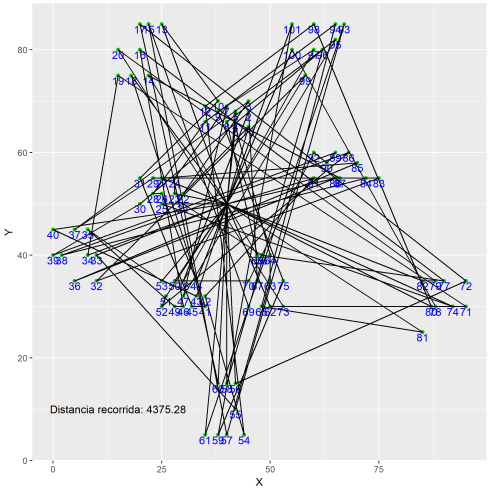
Figura A.8: Gráficos del conjunto de datos C101 IMPACT Intermedias



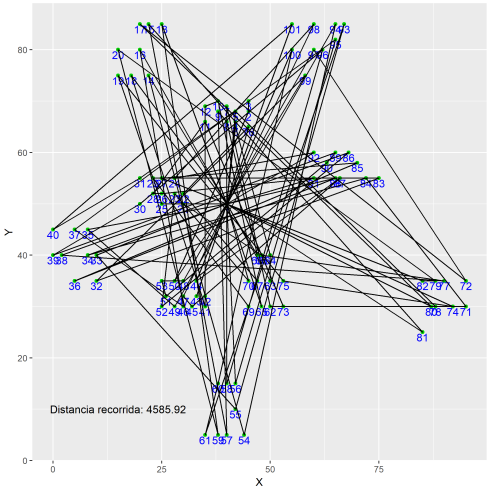
(a) IMPACT 0-01-09-09-0-01



(b) IMPACT 01-0-09-09-0-01



(c) IMPACT 0-0-1-08-0-02



(d) IMPACT 0-0-1-09-0-01

Figura A.9: Gráficos del conjunto de datos C101 IMPACT Intermedias

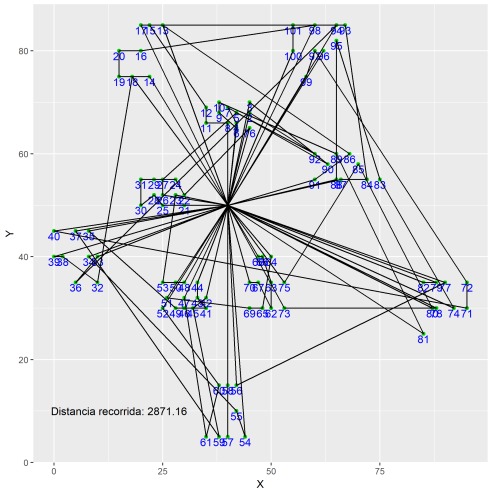
Apéndice B

Rutas modelos metaheurísticos

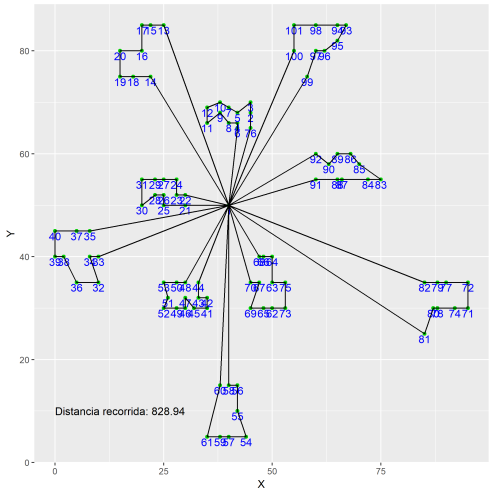
En este apéndice, se recoge un conjunto de imágenes donde se puede realizar una comparación de las soluciones de rutas que han sido generadas por cada una de las metaheurísticas.

En el primer grupo de imágenes se mostrarán los resultados obtenidos mediante el procedimiento GRASP (*Greedy Randomized Adaptive Search Procedures*). Estos resultados aparecían, de una manera menos detallada, descritos en los cuadros 4.6 a 4.8, donde además de la solución alcanzada veíamos el tiempo computacional que había sido empleado para conseguir dicho resultado.

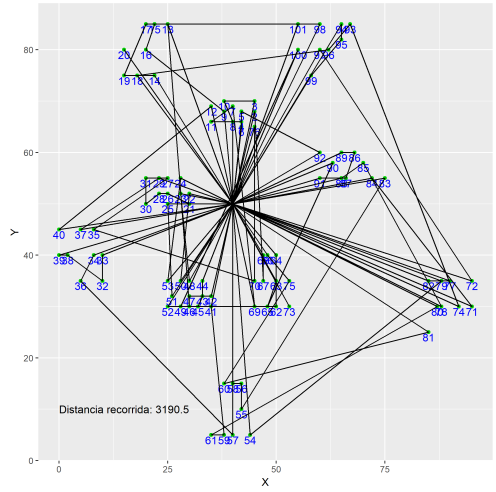
En un segundo bloque de imágenes nos encontramos una recopilación de los resultados obtenidos mediante la metaheurística *simulated annealing*, que aparecían recogidos en el apartado 4.3 y cuadros 4.10 y 4.11.



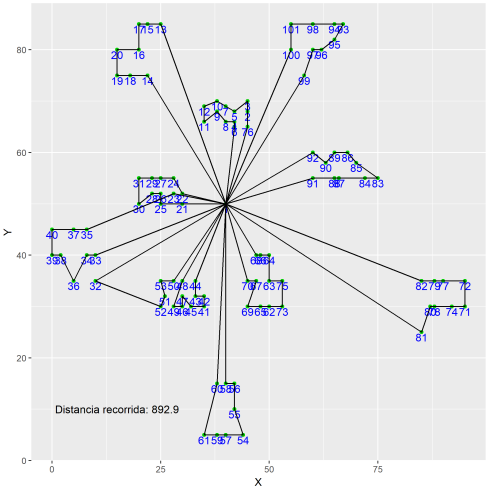
(a) Solomon + GRASP(3) inicio



(b) Solomon + GRASP(3) mejorado

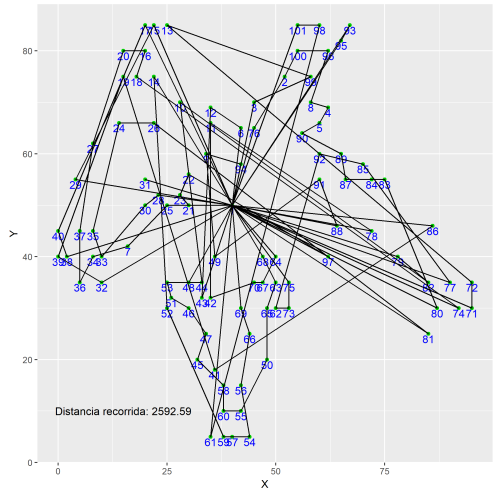


(c) Solomon + GRASP(3) inicio

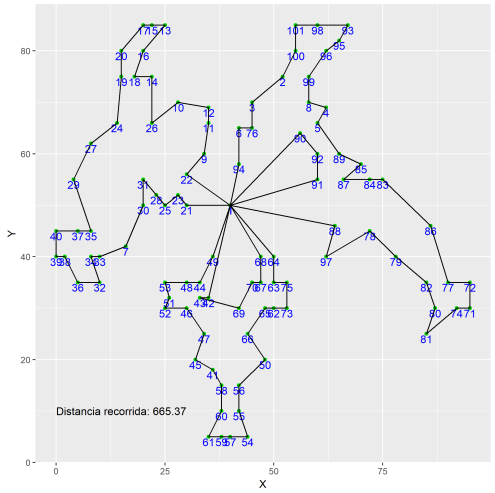


(d) Solomon + GRASP(3) mejorado

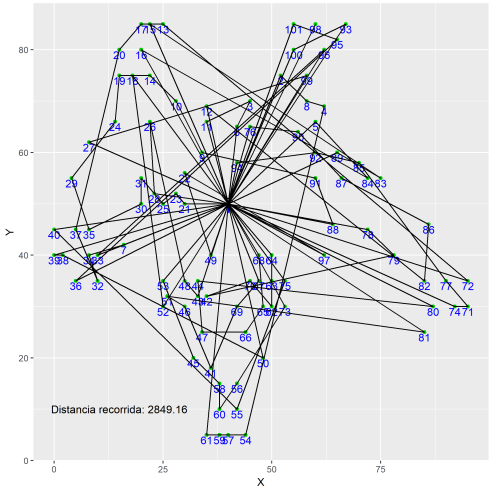
Figura B.1: Gráficos del conjunto de datos C101 con GRASP



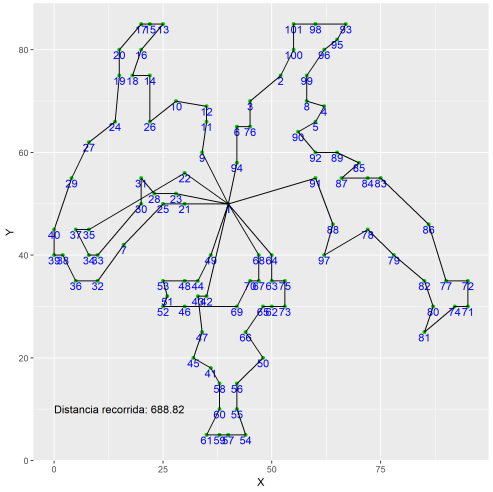
(a) Solomon + GRASP(3) inicio



(b) Solomon + GRASP(3) mejorado

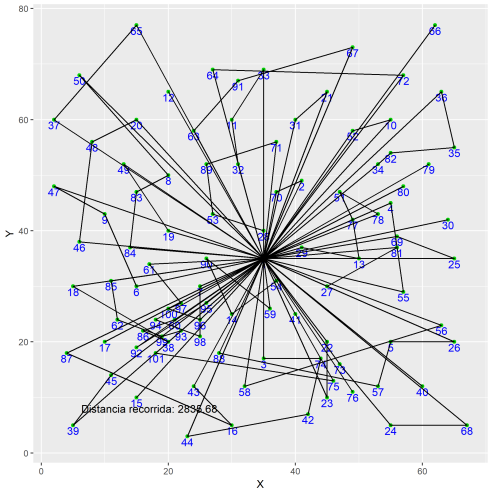


(c) Solomon + GRASP(3) inicio

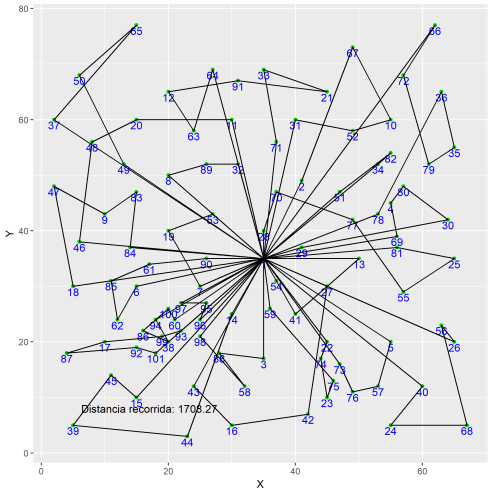


(d) Solomon + GRASP(3) mejorado

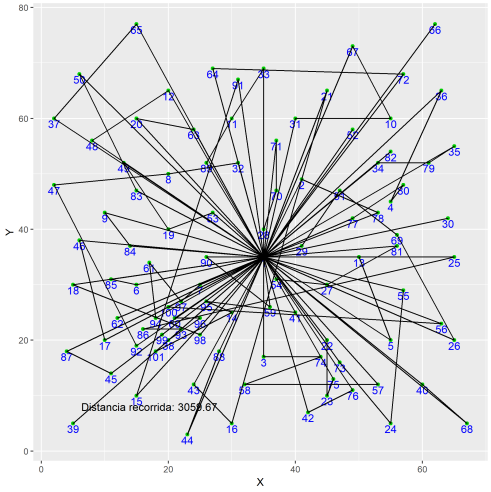
Figura B.2: Gráficos del conjunto de datos C201 con GRASP



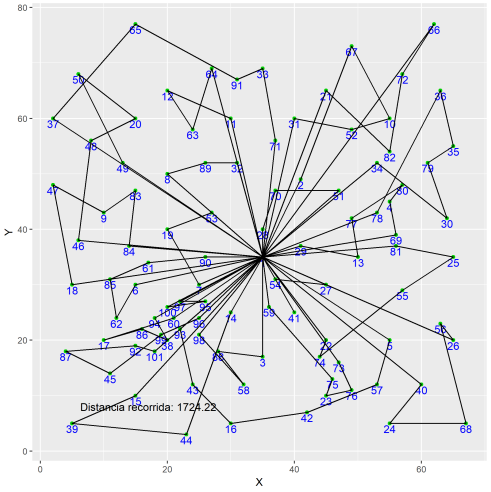
(a) Solomon + GRASP(3) inicio



(b) Solomon + GRASP(3) mejorado

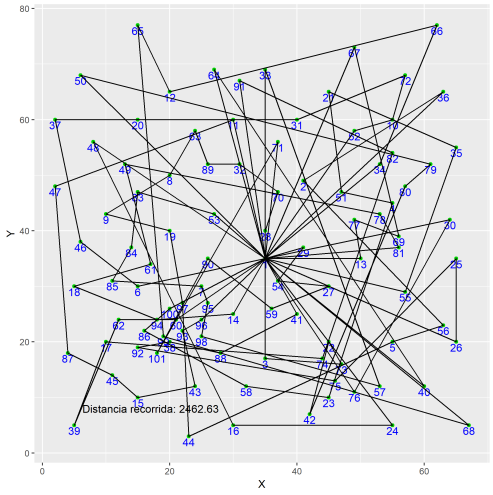


(c) Solomon + GRASP(3) inicio

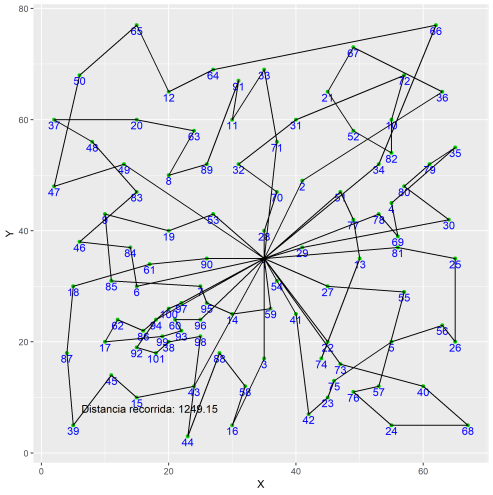


(d) Solomon + GRASP(3) mejorado

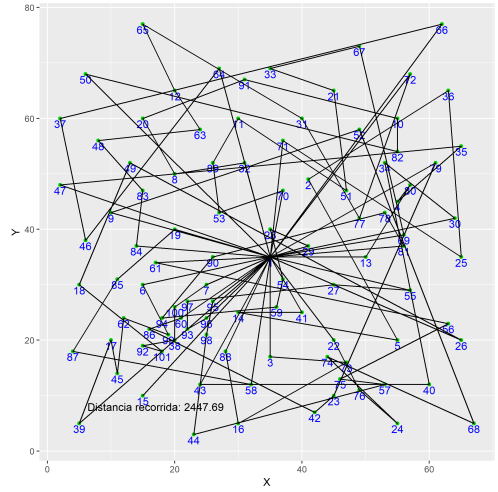
Figura B.3: Gráficos del conjunto de datos R101 con GRASP



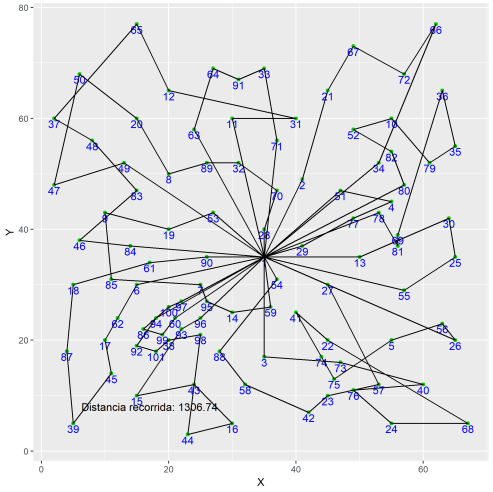
(a) Solomon + GRASP(3) inicio



(b) Solomon + GRASP(3) mejorado

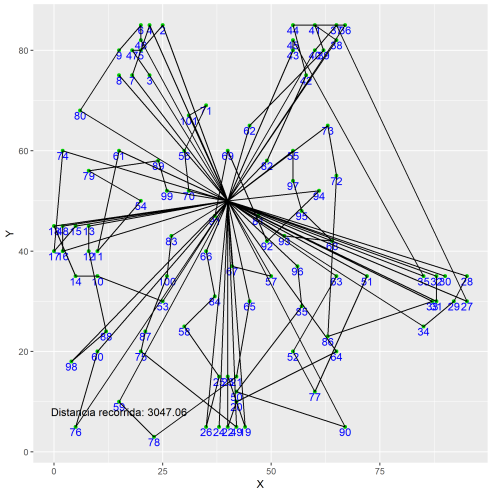


(c) Solomon + GRASP(3) inicio

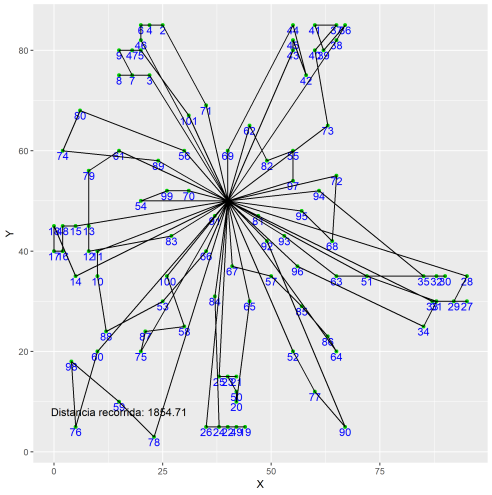


(d) Solomon + GRASP(3) mejorado

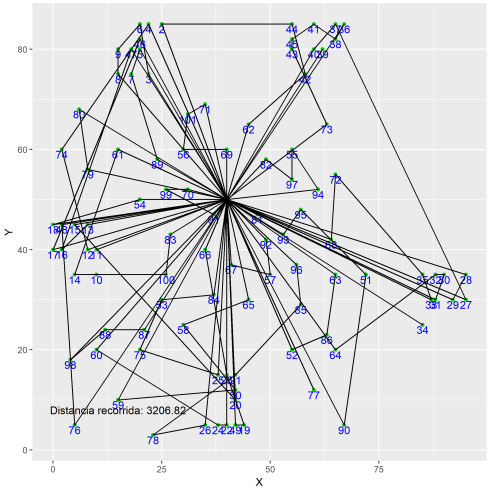
Figura B.4: Gráficos del conjunto de datos R201 con GRASP



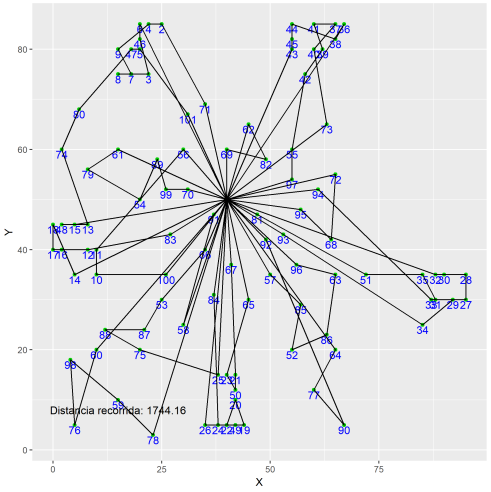
(a) Solomon + GRASP(3) inicio



(b) Solomon + GRASP(3) mejorado

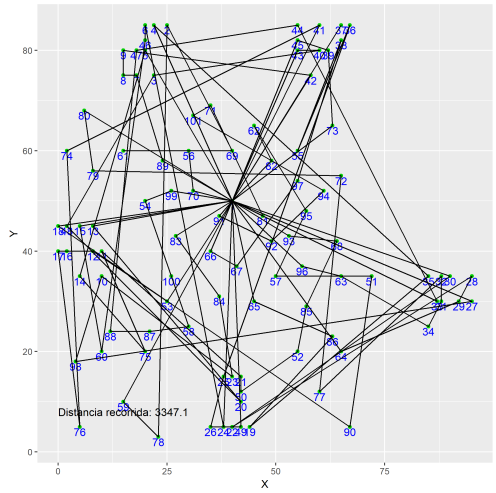


(c) Solomon + GRASP(3) inicio

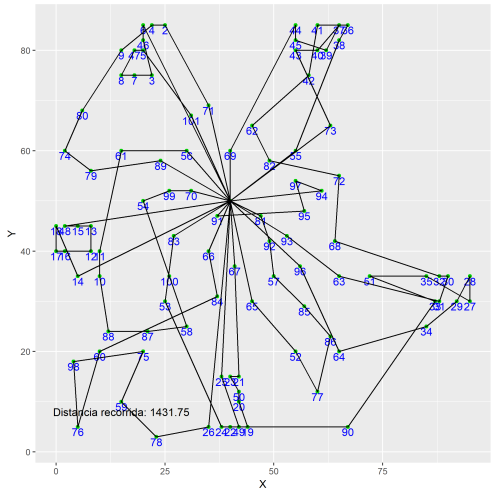


(d) Solomon + GRASP(3) mejorado

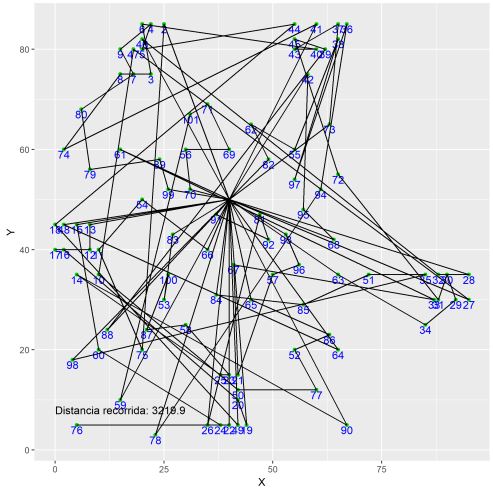
Figura B.5: Gráficos del conjunto de datos RC101 con GRASP



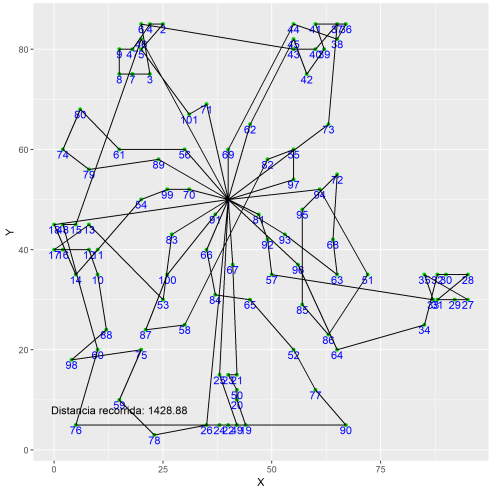
(a) Solomon + GRASP(3) inicio



(b) Solomon + GRASP(3) mejorado

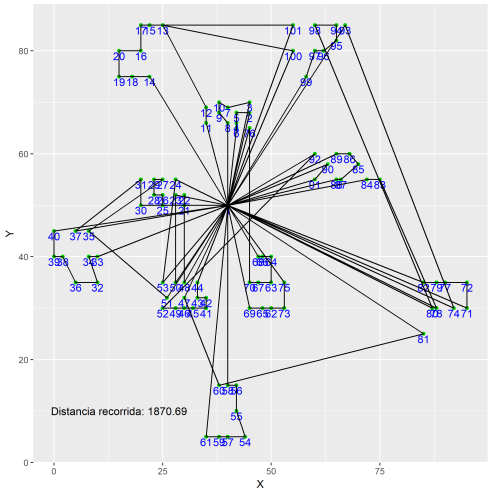


(c) Solomon + GRASP(3) inicio

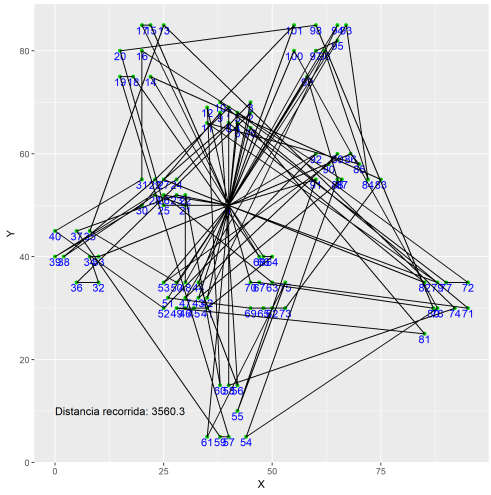


(d) Solomon + GRASP(3) mejorado

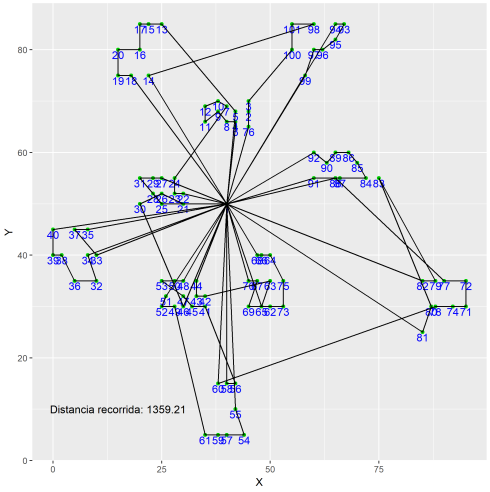
Figura B.6: Gráficos del conjunto de datos RC201 con GRASP



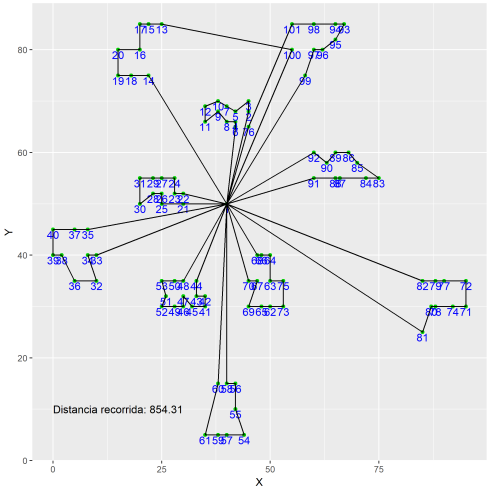
(a) Solomon + SA inicio



(b) Solomon + SA 3

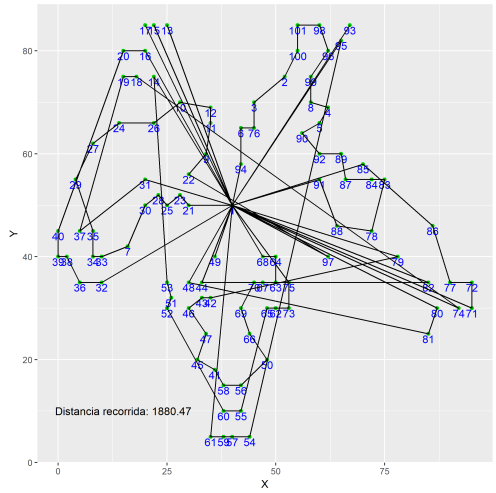


(c) Solomon + SA 50

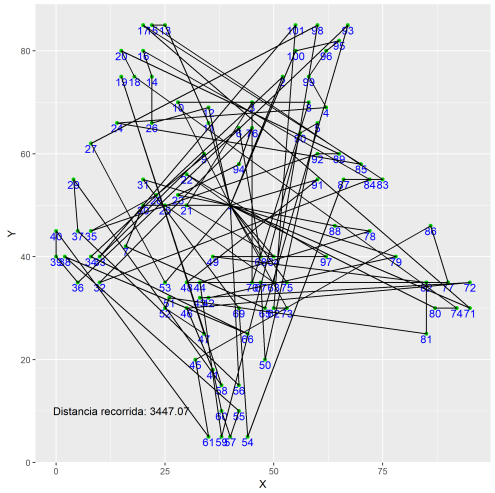


(d) Solomon + SA final

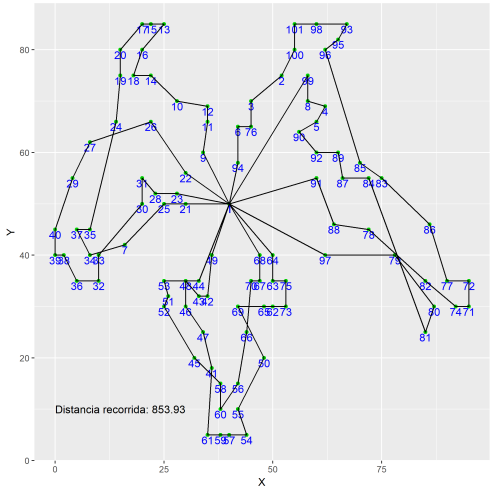
Figura B.7: Gráficos del conjunto de datos C101 con SA



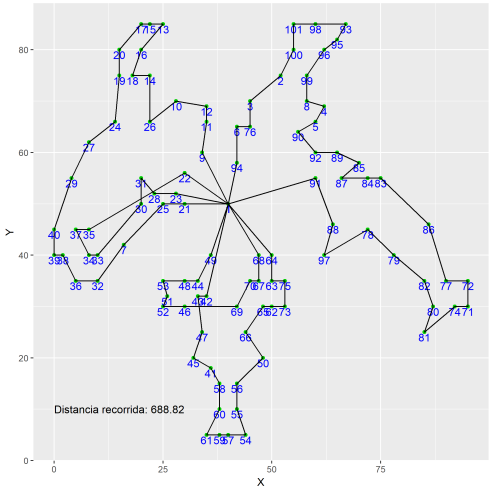
(a) Solomon + SA inicio



(b) Solomon + SA 3

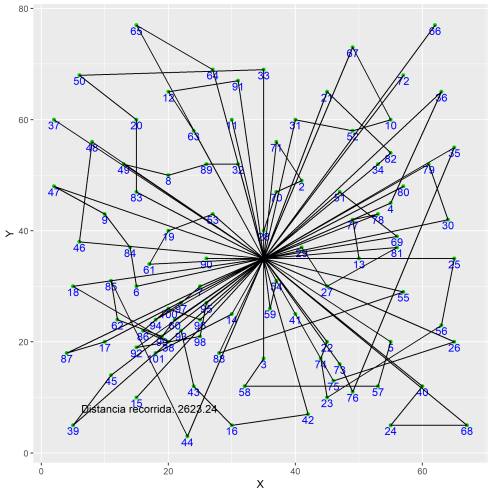


(c) Solomon + SA 50

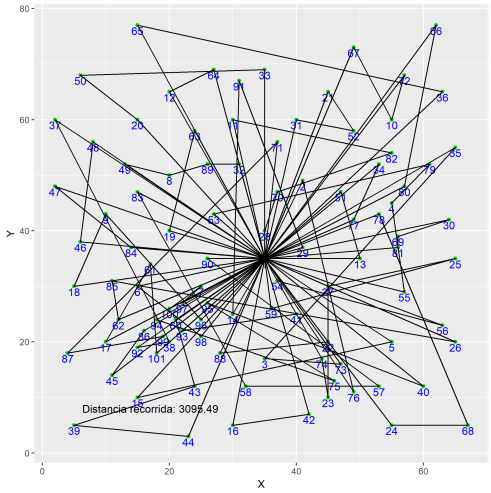


(d) Solomon + SA final

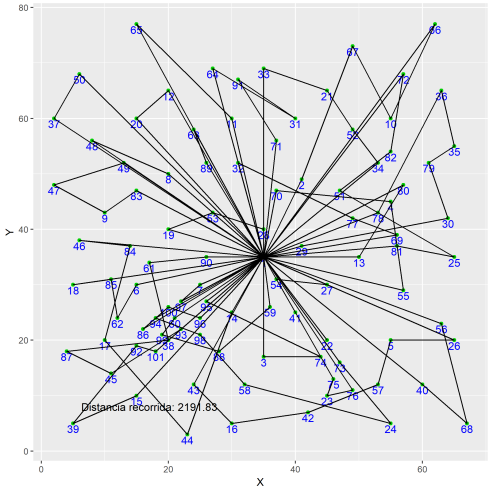
Figura B.8: Gráficos del conjunto de datos C201 con SA



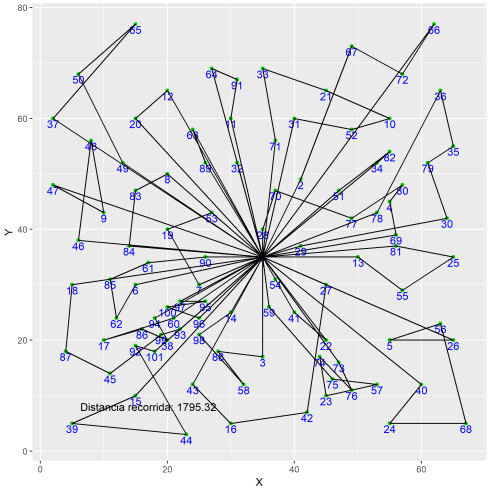
(a) Solomon + SA inicio



(b) Solomon + SA 3

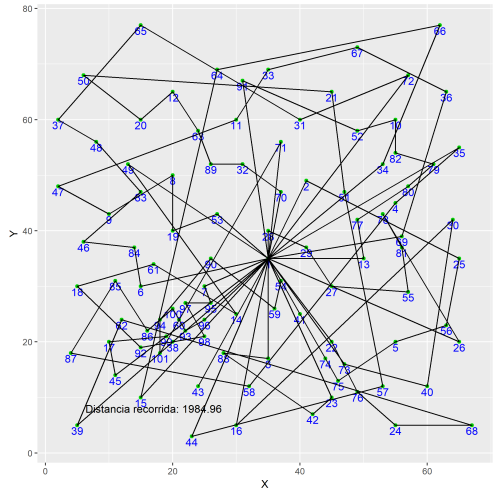


(c) Solomon + SA 50

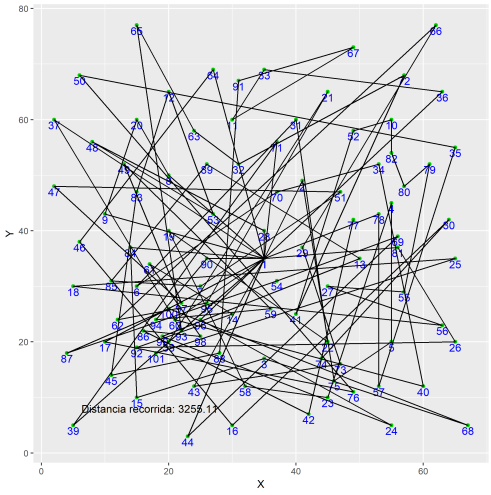


(d) Solomon + SA final

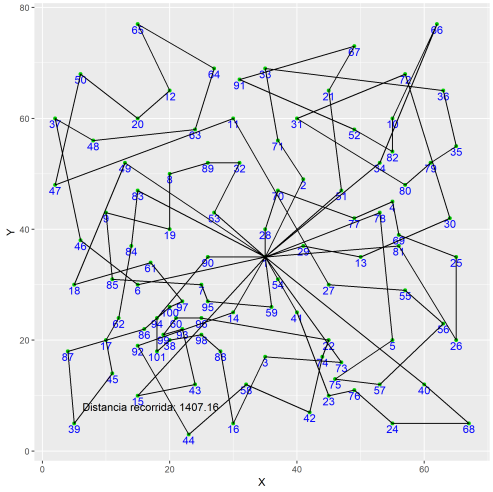
Figura B.9: Gráficos del conjunto de datos R101 con SA



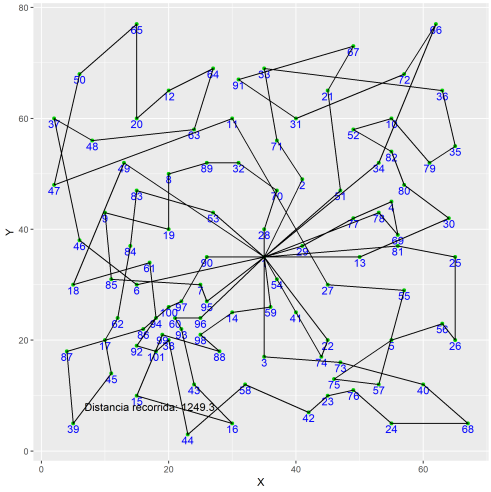
(a) Solomon + SA inicio



(b) Solomon + SA 3

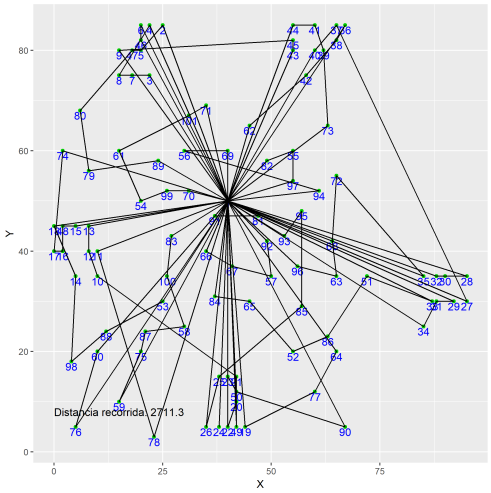


(c) Solomon + SA 50

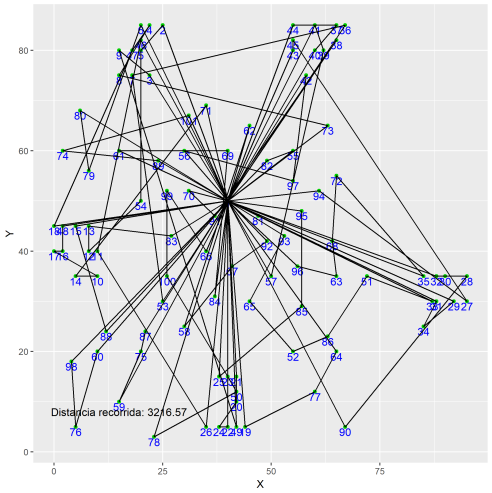


(d) Solomon + SA final

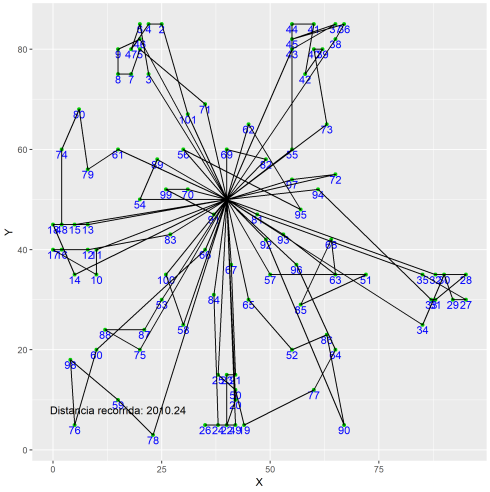
Figura B.10: Gráficos del conjunto de datos R201 con SA



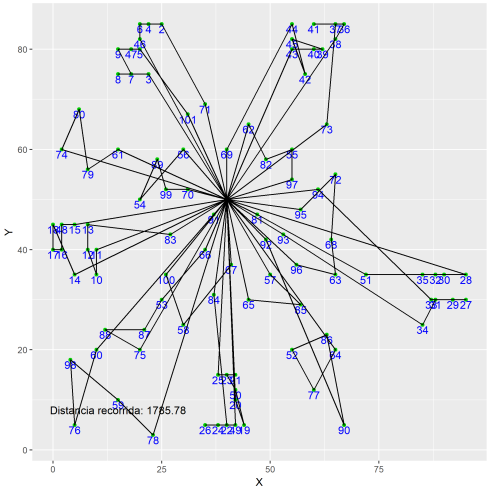
(a) Solomon + SA inicio



(b) Solomon + SA 3

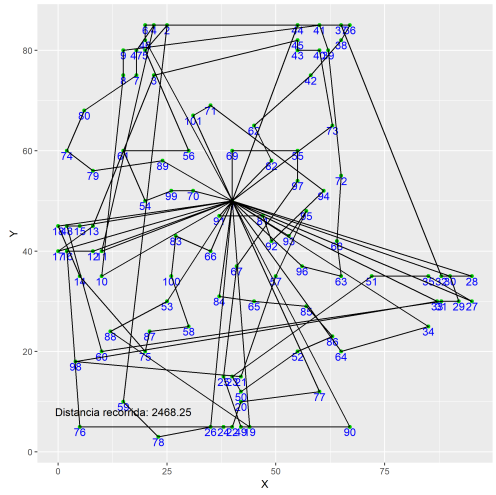


(c) Solomon + SA 50

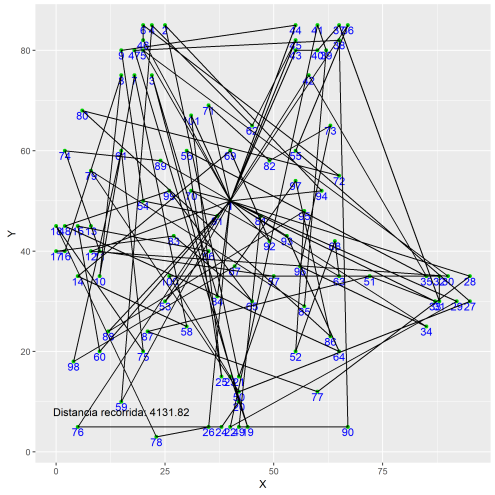


(d) Solomon + SA final

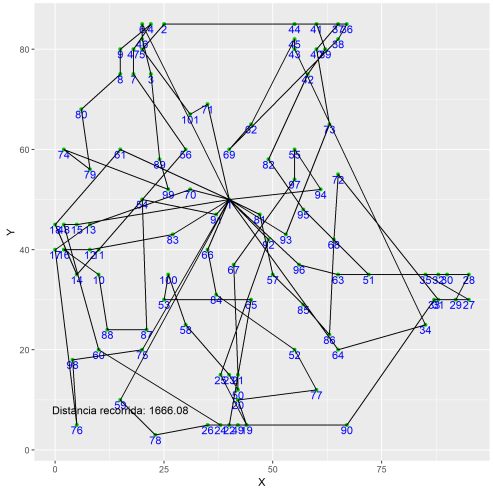
Figura B.11: Gráficos del conjunto de datos RC101 con SA



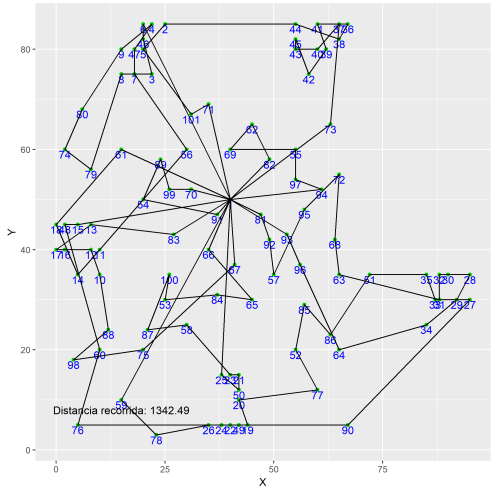
(a) Solomon + SA inicio



(b) Solomon + SA 3



(c) Solomon + SA 50



(d) Solomon + SA final

Figura B.12: Gráficos del conjunto de datos RC201 con SA

Bibliografía

[1] William Cook. *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press, 2012.

[2] Paolo Toth, Daniele Vigo. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[3] Alba Fernández Hernández. *Algoritmos heurísticos y metaheurísticos basados en búsqueda local aplicados a Problemas de Rutas de Vehículos*. Trabajos Fin de Grado. Universidad de Valladolid, 2016.

[4] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[5] Matt Dowle and Arun Srinivasan. *data.table: Extension of 'data.frame'*, 2017. R package version 1.10.4.

[6] Henrik Bengtsson. *matrixStats: Functions that Apply to Rows and Columns of Matrices (and to Vectors)*, 2017. R package version 0.52.2.

[7] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009.

[8] Yihui Xie. *DT: A Wrapper of the JavaScript Library 'DataTables'*, 2016. R package version 0.2.

[9] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2017. R package version 1.0.1.

[10] John Myles White. *log4r: A simple logging system for R, based on log4j.*, 2014. R package version 0.2.

[11] Roberto Cordone and Roberto Wolfler Calvo. A heuristic for the vehicle routing problem with time windows. *Journal of Heuristics*, 7(2):107–129, 2001.

[12] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.

[13] Marshall L Fisher and Ramchandran Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

[14] Marshall L Fisher, Kurt O Jörnsten, and Oli BG Madsen. Vehicle routing with time windows: Two optimization algorithms. *Operations research*, 45(3):488–492, 1997.

[15] Niklas Kohl and Oli BG Madsen. An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations research*, 45(3):395–406, 1997.

[16] Arturo Alonso Alonso. *Comparativa de modelos y solvers para la optimización de variantes de problemas de rutas de vehículos*. Trabajos Fin de Grado. Universidad de Valladolid, 2016.

[17] Georges A Croes. A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812, 1958.

[18] Shen Lin. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269, 1965.

[19] Ilhan Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Northwestern University Evanston, IL, 1976.

[20] Alex Van Breedam. *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints*. RUCA, 1994.

[21] Gerard AP Kindervater and Martin WP Savelsbergh. Vehicle routing: handling edge exchanges. *Local search in combinatorial optimization*, pages 337–360, 1997.

[22] Marius M Solomon and Jacques Desrosiers. Survey paper—time window constrained routing and scheduling problems. *Transportation science*, 22(1):1–13, 1988.

[23] G. Ioannou, G. Prastacos, M. Kritikos. A greedy look-ahead heuristic for the vehicle routing problem with time windows. *The Journal of the Operational Research Society*, 52(5):523–537, 2001.

[24] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[25] Jean-Yves Potvin and Jean-Marc Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993.

[26] King-Wah Pang. An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints. *Expert Systems with Applications*, 38(9):11939–11946, 2011.

[27] J. B. Atkinson. A greedy look-ahead heuristic for combinatorial Optimization: An Application to Vehicle Scheduling with Time Windows. *Journal of the Operational Research Society*, 45(6):673–684, 1994.

[28] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

[29] Gilbert Babin, Stéphanie Deneault, and Gilbert Laporte. Improvements to the or-opt heuristic for the symmetric travelling salesman problem. *Journal of the Operational Research Society*, 58(3):402–407, 2007.

[30] Bruce L Golden, William R Stewart, et al. Empirical analysis of heuristics. *The traveling salesman problem*, 4:207–249, 1985.

[31] DS Johnson, G Gutin, LA McGeoch, A Yeo, W Zhang, and A Zverovitch. The traveling salesman problem and its variations. *Chapter: Experimental Analysis of Heuristics for the ATSP*, pages 445–487, 2002.

[32] Geoffrey Zweig. An effective tour construction and improvement procedure for the traveling salesman problem. *Operations Research*, 43(6):1049–1057, 1995.

[33] Alex Van Breedam. *An Analysis of the Behaviour of Heuristics for the selection of Problems with Vehicle-Related, Customer-Related and Time-Related Constraints*. PhD thesis, University of Antwerp, Belgium, 1994.

[34] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119–139, 2005.

[35] Mario César Vélez and José Alejandro Montoya. Metaheurísticos: una alternativa para la solución de problemas combinatorios en administración de operaciones. *Revista Eia*, 8:99–115, 2007.

[36] Manuel Laguna and Rafael Martí. *Scatter Search: Methodology and Implementations in C.* Operations Research/Computer Science Interfaces Series 24. Springer US, 1 edition, 2003.

[37] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[38] George Kontoravdis and Jonathan F Bard. A grasp for the vehicle routing problem with time windows. *ORSA journal on Computing*, 7(1):10–23, 1995.

[39] Mauricio G.C. Resende and Celso C. Ribeiro. *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures.* Springer-Verlag New York, 1 edition, 2016.

[40] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[41] Wen-Chyuan Chiang and Robert A Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27, 1996.

[42] Marc Pirlot. General local search heuristics in combinatorial optimization: a tutorial. *Belgian Journal of Operations Research, Statistics and Computer Science*, 32(1-2):7–69, 1992.

[43] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104–118, 2005.

[44] Christian Prins. A grasp× evolutionary local search hybrid for the vehicle routing problem. In *Bio-inspired algorithms for the vehicle routing problem*, pages 35–53. Springer, 2009.

[45] Louis-Martin Rousseau, Michel Gendreau, and Gilles Pesant. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of heuristics*, 8(1):43–58, 2002.

Índice de figuras

Índice de cuadros