

TESIS DOCTORAL

**DISEÑO DE UNA HEURÍSTICA DE CONSTRUCCIÓN Y NUEVOS
OPERADORES DE MEJORA PARA LA RESOLUCIÓN DE PROBLEMAS DE
ASIGNACIÓN DE RUTAS A VEHÍCULOS CON RESTRICCIÓN DE
VENTANAS DE TIEMPO**



Presentada por Roberto Carballado Morillo

dentro del Programa de Doctorado en

INGENIERÍA INFORMÁTICA Y TELECOMUNICACIÓN

Dirigida por:

José Fernando Díaz Martín

El Director

El Doctorando

Bilbao, septiembre de 2015

Diseño de una heurística de construcción y nuevos operadores de mejora para la resolución de problemas de asignación de rutas a vehículos con restricción de ventanas de tiempo

Autor: Roberto Carballedo

Director: Fernando Díaz

Texto impreso en Bilbao

Primera edición, septiembre de 2015

*A mi familia y especialmente a Soraya,
por su apoyo y comprensión en los buenos y malos momentos.*

*“La sabiduría no ejerce ninguna autoridad y
aquellos que ejercen la autoridad no son sabios.”*

Jiddu Krishnamurti

Resumen

El transporte es un sector que juega un papel muy importante en la sociedad de hoy en día. La distribución de mercancías, el movimiento de personas, o la provisión de servicios como la recogida de residuos, son actividades que impactan directamente en la calidad de vida, pero que también tienen una considerable repercusión en la economía y el medio ambiente.

En el ámbito científico, el transporte tiene una relevancia igualmente crucial puesto que en torno a esta área de aplicación han surgido un sinfín de problemas difíciles de resolver debido a su complejidad computacional. Esto provoca que en muchas ocasiones sea prácticamente imposible la obtención de soluciones factibles, siendo necesario el uso de técnicas de aproximación que permitan obtener una buena solución en un tiempo limitado. En este sentido, surgen un amplio abanico de problemas de optimización y técnicas de resolución que tienen como objetivo principal la mejora continua para la obtención de soluciones cada vez más precisas, y con costes computacionales permisibles.

La presente tesis doctoral se enmarca en el área de los problemas de generación de rutas para vehículos. Estos tipos de problemas han sido ampliamente estudiados durante los últimos 50 años, y siguen siendo foco de interés científico hoy en día, ya que aún no existe una técnica capaz de resolver de manera adecuada algunas tipologías de problemas. Es por esta razón que es posible encontrar y abordar puntos de mejora que contribuyan a la optimización de los procesos de resolución existentes.

Con todo lo anterior, el trabajo de tesis doctoral presentado en este documento focaliza sus esfuerzos en la mejora del proceso de resolución de una de las variantes más ampliamente estudiadas del problema básico de asignación de rutas a vehículos: la variante con ventanas de tiempo (VRPTW). En concreto, se abordará el desarrollo de algoritmos para dos de las fases del proceso de resolución: la construcción de la solución inicial y la mejora iterativa de soluciones. La principal novedad del enfoque propuesto se encuentra en la definición de operadores de mejora basados en la reducción del número de rutas (que es uno de los criterios utilizados para medir la calidad de una solución). Tras una extensa experimentación comparando los métodos propuestos con las técnicas más representativas, y un exhaustivo y riguroso análisis de los resultados obtenidos, puede concluirse que los algoritmos generados representan un complemento idóneo para ser integrado en cualquiera de las técnicas heurísticas o metaheurísticas que atesoran los mejores resultados para los juegos de ensayo referentes en el contexto del VRPTW.

Abstract

Transportation is an activity that plays a very important role in our modern society. The distribution of goods, transport of people, or the provision of services (such as waste collection), are activities that have a direct impact on our quality of life, economy and environment.

Transportation has also a crucial relevance in the scientific community. A large and wide variety of problems have arisen around this area. All these problems have similar characteristics in terms of computational complexity, which makes them difficult to solve. This leads, in many cases, to serious difficulties in obtaining feasible solutions. For this reason, it is necessary to use approximation techniques that allow to obtain good solutions in a limited time. As a result, a wide range of optimization problems and solving techniques have emerged. In order to obtain increasingly better solutions and have a lower computational cost, these techniques are based on continuous improvement strategies.

The present research focuses on vehicle routing problems. Although these problems have been extensively studied over the past 50 years, they still have scientific interest. This is because there is no technique capable of solving some problems in a polynomial time. Therefore, it is possible to find and address areas for improvement, which contribute to the optimization of existing resolution techniques.

Therefore, this doctoral thesis focuses on improving the solving process of one of the most widely studied variations of the basic vehicle routing problem: the one that includes time windows (VRPTW). In particular, the work focuses on developing improvements in two phases of the solving process: the construction of the initial solution and the iterative improvement of the solution. The main novelty of the proposed approach is related to the purpose of the improvement process. In that sense, the process focuses on minimizing the number of routes (which is a criterion that measures the quality of a solution). The work done includes an extensive experimentation to compare the proposed algorithms and the most representative techniques. Also, with the aim of obtaining fair and objective conclusions, the results have been subjected to a rigorous analysis. With all this, it can be concluded that the new algorithms represent a good complement for any of the heuristics and meta-heuristics that obtain the best results for the most representative VRPTW benchmark problems.

Agradecimientos

Llegado este punto, cuando el trabajo de elaboración de este documento ya está casi terminado, toca pararse y agradecer la labor de todas las personas que directa o indirectamente son responsables de que yo haya sido capaz de realizar esta tesis doctoral.

En primer lugar quiero comenzar agradeciendo desde el corazón toda la paciencia, comprensión, ayuda y motivación que me ha proporcionado Soraya tanto en la elaboración del presente trabajo, como en el día a día. Ella es el faro que me guía y da luz, ayudándome a encontrar el rumbo tanto en mi vida como en el trabajo. ¡Gracias cariño!

Continúo dando las gracias al resto de mi familia, empezando por mis padres Feli y Edelmiro, mi hermano Javi, y Carmen, Jesús, Jesús y Winchi. Todos ellos tienen parte de culpa del resultado del trabajo que he realizado. ¡Gracias familia!

Siguiendo con los agradecimientos, y pasando al ámbito profesional, me gustaría empezar por dar las gracias a Begoña. Ella es sin duda la persona de la que más he aprendido en mi vida profesional, la que me enseñó a tener gusto por la inteligencia artificial y la docencia; y con la que he compartido tantos momentos y situaciones que la relación profesional se ha convertido en un vínculo familiar. Continuando en el ámbito laboral, quiero agradecer también a Verónica, Oliver, Zubía, Iñaki Vázquez, Anselmo y Jon los ratos agradables de trabajo, comida y café que compartimos a diario. También quiero tener un agradecimiento especial para mi vecino de despacho, que a pesar de que últimamente no coincidimos demasiado, espero que podamos continuar viéndonos y pasando buenos momentos fuera de los muros de la Universidad. ¡Gracias Chefo! Para cerrar este grupo de personas a las que tengo que agradecer su dedicación y ayuda en el día a día, y en especial en el aspecto final de este documento quiero dar las gracias a otro vecino, esta vez de descansillo. ¡Gracias JosuKa!

Además de los compañeros de la Facultad, no quiero olvidarme de mis compañeros de DeustoTech, a todos les agradezco lo fácil y cómodo que es trabajar con ellos. Doy las gracias especialmente a Pablo, Itziar, Asier Moreno y Nacho que son los que más directamente comparten su día a día conmigo. Por último, me gustaría agradecerle a Eneko toda la ayuda y momentos que hemos compartido desde que comenzó siendo mi discípulo hasta convertirse en mi maestro. ¡Gracias Maldini!

También, quisiera agradecer a Fernando la motivación, y toda la ayuda que me ha prestado en todos estos años en los que he estado realizando el presente trabajo de tesis doctoral, y en especial en la recta final. ¡Gracias Fernando!

Por último, quería terminar esta sección de agradecimientos dando especialmente las gracias a Nekane, Alberto, Juanma y Unai ya que ellos me ayudan a sustentar y sobrellevar las dificultades del día a día en el trabajo, y hacer que todo sea más llevadero y humano. ¡Gracias amigos!

Índice de contenidos

1	Introducción y motivaciones.....	1
1.1	Relevancia social y científica del trabajo realizado	2
1.2	Planteamiento del problema.....	4
1.3	Hipótesis y objetivos	5
1.4	Metodología de investigación	7
1.5	Estructura del documento de tesis doctoral	9
2	Problemas de generación de rutas de vehículos.....	11
2.1	El problema del viajante.....	11
2.1.1	Definición del problema.....	12
2.1.2	Formulación matemática del problema.....	14
2.2	El problema de asignación de rutas a vehículos.....	15
2.2.1	Definición del problema.....	15
2.2.2	Formulación matemática del problema.....	17
2.3	Variantes del VRP	19
2.3.1	Variante con almacenes múltiples - MDVRP	19
2.3.2	Variante con flota de vehículos heterogénea - HVRP	19
2.3.3	Variante con rutas periódicas - PVRP	20
2.3.4	Variante con entregas fraccionadas - SDVRP	20
2.3.5	Variante con recogida - VRPB.....	20
2.3.6	Variante con recogida y entrega - VRPPDP.....	21

2.3.7 Variante con varias rutas por vehículo - VRPMT	21
2.3.8 Variante abierta - OVRP	21
2.3.9 Variante con regulaciones laborales	21
2.3.10 Variante con ventanas de tiempo - VRPTW	22
2.4 Clasificación de las variantes del VRP	23
3 Técnicas de resolución de problemas de tipo VRP	27
3.1 Heurísticas constructivas	28
3.1.1 Heurísticas de construcción para el VRPTW	31
3.2 Técnicas de búsqueda local	41
3.2.1 Vecindarios de intercambio de nodos y arcos	45
3.2.2 Heurísticas de reducción del número de rutas	54
3.2.3 Mejora de la eficiencia de la búsqueda local	56
3.3 Metaheurísticas	67
3.3.1 Metaheurísticas basadas en vecindarios	68
3.3.2 Metaheurísticas basadas en poblaciones de soluciones	70
3.3.3 Metaheurísticas híbridas	71
3.3.4 Metaheurísticas paralelas y cooperativas	72
3.3.5 Metaheurísticas más eficientes para el VRPTW	73
3.4 Características clave del diseño de metaheurísticas para el VRP	74
4 Heurística constructiva de inicialización y operadores transformativos de mejora para la resolución del problema VRPTW	77
4.1 Heurística de construcción para el VRPTW	78
4.1.1 Bases formales de la heurística de construcción para el VRPTW	78
4.1.2 Diseño de la heurística de construcción para el VRPTW	81
4.2 Operadores de mejora basados en la reducción del número de rutas	84
4.2.1 Motivación para el diseño de los operadores de mejora	85

4.2.2	Estructura general del operador reducción del número de rutas.....	88
4.2.3	Operador de reasignación de clientes alejados (RcR-opt).....	92
4.2.4	Operador de eliminación de rutas pequeñas (SrE-opt)	94
4.2.5	Operador aleatorio de eliminación de rutas (RrE-opt).....	97
4.3	Entorno de visualización y simulación de problemas de tipo VRP	97
5	Experimentación y validación de resultados	103
5.1	Comparación de técnicas de resolución de problemas de optimización combinatoria.....	104
5.1.1	Criterios generales de comparación de técnicas de resolución del VRP.....	104
5.1.2	Buenas prácticas para la implementación y comparación de técnicas de resolución de problemas de optimización combinatoria	107
5.1.3	Conjuntos de instancias de problemas de tipo VRPTW	109
5.2	Configuración de la experimentación.....	111
5.2.1	Descripción de la función objetivo	111
5.2.2	Representación de las soluciones.....	112
5.2.3	Implementación de las heurísticas de construcción	114
5.2.4	Ajuste de los parámetros de las heurísticas de construcción	115
5.2.5	Operadores de intercambio de arcos y nodos	117
5.2.6	Implementación algoritmo de búsqueda local de referencia	121
5.2.7	Entorno de ejecución y lenguaje de desarrollo	126
5.3	Resultados y pruebas estadísticas	127
5.3.1	Resultados de la experimentación en torno a la heurística de construcción para el VRPTW	127
5.3.2	Resultados de la experimentación en torno a los operadores de mejora basados en la reducción del número de rutas	132
5.4	Análisis de los resultados	142

5.4.1	Análisis de resultados de la heurística de construcción para el VRPTW	142
5.4.2	Análisis de resultados de los operadores de mejora para el VRPTW	143
6	Conclusiones y líneas de trabajo futuras	147
6.1	Conclusiones	148
6.2	Líneas de trabajo futuras	151
7	Bibliografía	155
A.	Parámetros de configuración de la experimentación	171
A.1	Parámetros de configuración de las heurísticas de construcción	171
B.	Resultados ampliados de la experimentación	175
B.1	Resultados ampliados de la heurística de construcción para el VRPTW	176
C.	Formato de entrada del entorno de visualización y simulación para problemas de tipo VRP	179

Índice de figuras

Figura 1-1: Metodología de investigación empleada.....	8
Figura 2-1: Instancia de un TSP y una posible solución del mismo.	12
Figura 2-2: Instancia de un VRP y una posible solución del mismo.	15
Figura 3-1: Espacio de soluciones de una búsqueda local.	42
Figura 3-2: Generación de una solución no válida.....	44
Figura 3-3: Operador 2-opt.	46
Figura 3-4: Operador Or-opt.	47
Figura 3-5: Operador IOPT.....	48
Figura 3-6: Operador 2-opt*.	49
Figura 3-7: Operador de inserción.....	50
Figura 3-8: Operador de intercambio.	50
Figura 3-9: Operador de cruce.....	51
Figura 3-10: Operador de cruce o CROSS-exchange.	52
Figura 3-11: Operador intercambio GENI.....	53
Figura 3-12: Operador GENICROSS.	54
Figura 3-13: Descomposición de un operador 2-opt*.	61
Figura 3-14: Clasificación de técnicas metaheurísticas.....	67
Figura 4-1: Solución a un problema de tipo VRPTW-1.	86

Figura 4-2: Solución a un problema de tipo VRPTW-2.	87
Figura 4-3: Funcionamiento del operador de reasignación de clientes alejados.	93
Figura 4-4: Funcionamiento del operador de eliminación de rutas pequeñas.	96
Figura 4-5: Vista principal del entorno de visualización y simulación.	99
Figura 4-6: Vista tabular de la información de una ruta.	100
Figura 4-7: Vista de la simulación de una ruta.	101

Índice de tablas

Tabla 2-1: Taxonomía de problemas de tipo VRP de Eksioglu y otros.....	24
Tabla 2-2: Clasificación de variantes del VRP en base a sus características.	25
Tabla 3-1: Heurísticas de construcción de rutas para el VRP.	30
Tabla 3-2: Heurísticas de dos fases.	31
Tabla 3-3: Comparativa de heurísticas de construcción para el VRPTW.	40
Tabla 5-1: Variantes del algoritmo de búsqueda paralela con vecindario variable.	125
Tabla 5-2: Resultados obtenidos por la versión básica de cada una de las heurísticas de construcción.	128
Tabla 5-3: Resultados obtenidos por cada una de las heurísticas de construcción utilizando la mejora IRCL.	129
Tabla 5-4: Resultados obtenidos por cada una de las heurísticas de construcción utilizando la mejora IIRC.	130
Tabla 5-5: Resultados obtenidos por las tres variantes de la heurística I1.	130
Tabla 5-6: Resultados obtenidos por las tres variantes de la heurística IRCL.	131
Tabla 5-7: Resultados obtenidos por las tres variantes de la heurística IMPACT.	131
Tabla 5-8: Mejores resultados obtenidos por las heurísticas de construcción.	132
Tabla 5-9: Mejor solución inicial previa a la ejecución de la búsqueda local.	133
Tabla 5-10: Resultados obtenidos por la variante PVNSInter	133
Tabla 5-11: Resultados obtenidos por la variante PVNSIntra	134

Tabla 5-12: Resultados obtenidos por la variante PVNSMin	134
Tabla 5-13: Resultados obtenidos por la variante PVNSRrE	135
Tabla 5-14: Resultados obtenidos por la variante PVNSInter-Intra.....	135
Tabla 5-15: Resultados obtenidos por la variante PVNSMin-Intra.....	136
Tabla 5-16: Resultados obtenidos por la variante PVNSRrE-Intra.....	137
Tabla 5-17: Resultados obtenidos por la variante PVNSAll	137
Tabla 5-18: Resultados obtenidos para la variante PVNSAll-RrE.....	138
Tabla 5-19: Resumen del número de vehículos obtenido por las mejores variantes..	138
Tabla 5-20: Resumen de la distancia obtenida por las mejores variantes.	139
Tabla 5-21: Ranking promedio obtenido mediante el test de Friedman para todas las variantes.	139
Tabla 5-22: Valores p ajustados y no ajustados del test de Holm para el número de vehículos.....	140
Tabla 5-23: Valores p ajustados y no ajustados del test de Holm para la distancia total recorrida.	141
Tabla 5-24: Número de iteraciones realizadas por cada una de las variantes.	146

Introducción y motivaciones

El transporte es un sector que juega un papel muy importante en la sociedad de hoy en día. La distribución de mercancías, el movimiento de personas, o la provisión de servicios como la recogida de residuos, son actividades que impactan directamente en la calidad de vida, pero que también tienen una considerable repercusión en la economía y el medio ambiente.

Por otro lado, la globalización, el crecimiento de la población mundial, los movimientos de personas de las zonas rurales a los grandes núcleos urbanos, el aumento de las emisiones de gases de efecto invernadero, y el agotamiento de las reservas de combustibles de origen fósil, hacen que sea necesario un continuo desarrollo y mejora de los servicios de transporte para que éstos sean cada vez más eficaces y eficientes.

En el ámbito científico, el transporte también tiene una relevancia crucial puesto que en torno a este área de aplicación han surgido un sinfín de problemas difíciles de resolver debido a su complejidad computacional. Esto provoca que en muchas ocasiones sea prácticamente imposible la obtención de soluciones factibles, siendo necesario el uso de técnicas de aproximación que permitan obtener una buena solución en un tiempo limitado. En este sentido, surgen un amplio abanico de problemas de optimización y técnicas de resolución que tienen como objetivo principal la mejora continua en los procesos de resolución para la obtención de soluciones cada vez mejores, y con costes computacionales más pequeños.

Problemas como la generación de rutas minimizando distancia, coste o consumo de combustible; la distribución de elementos en contenedores para maximizar la ocupación; la estibación de la carga en un barco para garantizar la estabilidad del mismo; o la planificación de los horarios de un servicio de transporte de pasajeros, son sólo algunos ejemplos de problemas complejos de resolver para los que, a día de hoy,

todavía es prácticamente imposible encontrar la solución óptima en un tiempo razonable. Esto exige, como ya se ha adelantado en el párrafo anterior, la necesidad de desarrollar técnicas de resolución sub-óptimas, también conocidas genéricamente como heurísticas o metaheurísticas, que representan un ámbito de gran actividad e interés científico.

Con esta situación en mente, la presente tesis doctoral aborda una de las problemáticas más ampliamente estudiadas en el área de la optimización combinatoria, que son los problemas de asignación de rutas a vehículos. El trabajo de investigación realizado se centra en la mejora del proceso de resolución mediante el diseño de nuevas técnicas y operadores que permitan mejorar la calidad de las soluciones obtenidas y la reducción del tiempo de ejecución.

Este primer capítulo está organizado de acuerdo a cinco secciones: la sección 1.1 ahonda en la relevancia social y científica que posee el transporte; la sección 1.2 esboza el planteamiento genérico de la problemática en la que se encuadra el trabajo de investigación realizado; en la sección 1.3 se describen las hipótesis y los objetivos definidos para la realización de la presente tesis doctoral; la sección 1.4 presenta la metodología de investigación empleada; y por último, se cierra el capítulo con la sección 1.5 que describe la estructura del presente documento.

1.1 Relevancia social y científica del trabajo realizado

El transporte, tanto de bienes como de personas, juega un papel muy importante en la economía y la sociedad de hoy en día. El texto que se incluye a continuación destaca la relevancia que posee el transporte en Europa:

“... la industria del transporte en Europa emplea a más de 10 millones de personas y representa en torno al 5% del Producto Interior Bruto (PIB), las actividades logísticas y de transporte suponen un 10-15% del coste final de los productos, y el gasto medio por hogar en bienes y servicios de transporte ronda el 14% del presupuesto anual.

Por último, el transporte también es el responsable de un alto porcentaje de las emisiones de gases de efecto invernadero y uno de los mayores consumidores de carburantes de origen fósil.”

Fuente: (The European Commission 2015)

Con estos datos, aparte de confirmar el impacto del transporte en los tres vectores del desarrollo sostenible; economía, sociedad y medio ambiente, se puede asegurar que el desarrollo de sistemas de transporte eficaces y eficientes que reduzcan los costes de los desplazamientos, mejoren su calidad, minimicen el consumo de combustibles, y

controlen las emisiones de gases de efecto invernadero, es una actividad relevante y de interés social.

Dentro de las diferentes modalidades de transporte, el transporte por carretera es sin duda el más utilizado a nivel nacional y regional. Por ese motivo, los problemas de diseño y generación de rutas de vehículos han despertado un gran interés en la comunidad científica desde hace más de 200 años, cuando fue formulado el primer problema de generación de rutas de vehículos: el problema del viajante (*Travelling Salesman Problem* - TSP); y varios años después, en 1959 (Dantzig y Ramser 1959), cuando se introdujo uno de los problemas más ampliamente estudiados en el ámbito del diseño de sistemas de transporte por carretera: el problema de asignación de rutas a vehículos (*Vehicle Routing Problem* - VRP).

Tras más de 50 años de intenso trabajo de investigación en el ámbito del problema de asignación de rutas a vehículos (Laporte 2009), todavía hoy en día sigue siendo un área de interés científico, puesto que al igual que ha evolucionado el sector del transporte, este ámbito de investigación ha ido evolucionando en paralelo para adaptarse a las exigencias del mismo. Esto ha provocado un aumento de la complejidad del problema en cuanto al número de vehículos y ubicaciones que deben contemplarse para la generación de las rutas o a las restricciones que deben cumplirse a la hora de resolverlo. Por otro lado, una razón adicional para que actualmente siga siendo un área de interés científico es la complejidad computacional del problema: no es posible obtener la solución óptima global en un tiempo factible. Esto hace que todos los métodos prácticos de resolución sean aproximados, luego susceptibles de mejora continua. Además, un atractivo adicional de este ámbito de trabajo, es el reto científico que supone la definición de nuevas técnicas que sean cada vez más simples, eficaces y eficientes.

Prueba de la repercusión de este ámbito de investigación es el hecho de que en la literatura científica pueden encontrarse múltiples referencias a problemas de generación de rutas de vehículos entre las que destacan especialmente las dos ediciones del libro de Toth y Vigo (Toth y Vigo 2002a) (Toth y Vigo 2015) y el libro de Golden, Raghavan y Wail (Golden, Raghavan y Wail 2008). Por otro lado, existe un buen nutrido conjunto de revistas científicas especializadas en transporte e investigación operativa como: *Transportation Science* (<http://pubsonline.informs.org/journal/trsc>), *Operations Research* (<http://pubsonline.informs.org/journal/opre>), *Journal of the Operational Research Society* (<http://www.palgrave-journals.com/jors/index.html>) y *European Journal of Operational Research* (<http://www.journals.elsevier.com/european-journal-of-operational-research/>) en las que se publican los últimos avances de este ámbito de investigación. Además, hay centros tecnológicos especializados en esta temática como el CIRRELT de Canadá (<https://www.cirrelt.ca/default.aspx>) que es un referente en las técnicas de resolución de este tipo de problemas, e incluso empresas

especializadas en desarrollo de software de optimización como Dassault Systèmes (<https://www.3ds.com>) que tienen productos específicos para la resolución de problemas de generación de rutas de vehículos (Dassault Systèmes 2015).

1.2 Planteamiento del problema

Tal y como se ha ido esbozando anteriormente, el presente trabajo de investigación se enmarca en el área del transporte, y más concretamente en los problemas de generación de rutas para vehículos. Este tipo de problemas puede encontrarse en multitud de situaciones del mundo real, como por ejemplo, la distribución de carburantes (que fue uno de los primeros ámbitos de aplicación abordados desde el contexto de la inteligencia artificial y la investigación operativa), el reparto de paquetería, la recogida de residuos, o el transporte de pasajeros.

Recientemente (en agosto de este mismo 2015), desde el centro de investigación CIRRELT, se ha presentado un extenso estudio en el que se revisa la literatura especializada de los últimos 15 años en torno a las aplicaciones reales en el ámbito del transporte de mercancías (Coleho, Renaud y Laporte 2015). Dicho estudio, además de poner de manifiesto el interés de esta área de investigación, esboza unas conclusiones que sirven de justificación para el desarrollo de un trabajo de investigación como el presentado en esta tesis.

En primer lugar, los autores destacan que la mayor parte de los trabajos analizados se corresponden con aplicaciones del sector privado, existiendo muy poca aplicación de los resultados científicos en el sector público.

Por otro lado, justifican que el uso de técnicas de investigación operativa permite ahorrar hasta un 10% en los costes de la actividad de transporte, lo que representa un valor considerable en los ámbitos en los que la planificación de las actividades de transporte se realiza a diario.

En relación a las técnicas o algoritmos concretos, argumentan que no existe una clara superioridad de una técnica frente a otra, ya que se utilizan desde procesos simples de construcción de soluciones, pasando por técnicas de mejora basadas en búsqueda local, hasta heurísticas híbridas que combinan búsquedas de vecindario variable (estas técnicas serán revisadas con detalle en el capítulo 3).

Por último, concluyen que la mayor parte de los sistemas de resolución de problemas de transporte siguen siendo hechos “a medida”, porque todavía no existe una alternativa de software/sistema comercial que pueda adaptarse de manera adecuada al vasto y heterogéneo ecosistema de problemas existentes.

A tenor de las conclusiones puestas a la luz en este reciente estudio, y teniendo en cuenta la relevancia social y científica que se ha descrito en la sección 1.1, el trabajo de investigación desarrollado en la presente tesis doctoral aborda una problemática que es relevante y de interés actual. Esto es debido a que todavía no existe una única técnica capaz de resolver de manera adecuada algunas tipologías de problemas. Por lo tanto es posible encontrar y abordar puntos de mejora que contribuyan a la optimización de los procesos de resolución existentes.

Con todo lo anterior, el trabajo de investigación presentado en esta tesis doctoral focaliza sus esfuerzos en la mejora del proceso de resolución de una de las variantes más ampliamente estudiadas del problema de asignación de rutas a vehículos: la variante con ventanas de tiempo (que será presentada con detalle a lo largo del capítulo 2). En concreto, se abordará el desarrollo de aportes en dos ámbitos concretos del proceso de resolución: en la construcción de la solución inicial y en la mejora iterativa de soluciones con el foco puesto en la reducción del número de rutas de una determinada solución (que es uno de los criterios utilizados para medir la calidad de la misma).

A modo de reseña final, para clarificar el contexto general del trabajo realizado, los nuevos aportes diseñados se basarán en la aplicación de técnicas de inteligencia artificial para la resolución de problemas de optimización combinatoria.

1.3 Hipótesis y objetivos

Una vez presentada la problemática en torno a la que girará el trabajo a desarrollar, y con el fin de que dicho trabajo se aborde de una manera eficiente, se han definido dos hipótesis que se validarán durante el desarrollo de la presente tesis doctoral:

HIPÓTESIS 1: *Es posible definir una técnica de construcción de soluciones iniciales para problemas de asignación de rutas a vehículos con restricción fuerte de ventanas de tiempo que mejore a las principales heurísticas de construcción conocidas.*

HIPÓTESIS 2: *Es posible definir un nuevo operador de mejora para problemas de asignación de rutas a vehículos con restricción fuerte de ventanas de tiempo que facilite la reducción del número de rutas de una solución.*

Como parte de este trabajo de investigación, se plantean también dos objetivos generales, cuya consecución ratificará el cumplimiento de las dos hipótesis definidas:

- *Diseñar una heurística de construcción para problemas de asignación de rutas a vehículos con restricción fuerte de ventanas de tiempo.*
- *Diseñar un operador de mejora para problemas de asignación de rutas a vehículos con restricción fuerte de ventanas de tiempo.*

El desarrollo de la presente tesis doctoral ha sido guiado en todo momento por la consecución de los objetivos generales planteados, ya que este hecho permitirá la validación satisfactoria de las dos hipótesis definidas.

Además de los objetivos generales, se han definido un conjunto de objetivos específicos que están relacionados con la metodología llevada a cabo en el trabajo de investigación:

- *Identificar puntos de mejora en el área de los problemas de generación de rutas de vehículos.* En este sentido, será necesario revisar la literatura de referencia para adquirir el conocimiento necesario que permita detectar puntos de mejora y definir las estrategias apropiadas para abordarlos.
- *Diseñar e implementar los algoritmos que permitan la validación positiva de las hipótesis.* A partir de las hipótesis y los objetivos generales, y una vez adquirido un dominio profundo de la literatura de referencia, se debe especificar detalladamente el proceso o método que facilite la consecución de los objetivos generales y la validación satisfactoria de las hipótesis. Dichos algoritmos, que se materializarán en un conjunto de métodos o algoritmos, deben presentar un diseño sencillo y ofrecer un adecuado equilibrio entre la calidad de la solución obtenida y tiempo de ejecución necesario. Otro aspecto especialmente relevante en relación con este objetivo será el uso de pautas de diseño de software que hagan que el resultado sea fácil de modificar, extender e integrar en otras técnicas o sistemas de información.
- *Definir y configurar un entorno de experimentación para la validación de los algoritmos generados.* Con el fin de evaluar los algoritmos generados, será necesario especificar y configurar debidamente el entorno en que se llevará a cabo la experimentación. Para ello se identificarán las técnicas de referencia con las que se comparará el trabajo realizado, así como las instancias de problemas (o juegos de ensayo) utilizados durante la experimentación. Además, para controlar el impacto que pueden tener tanto los lenguajes de programación como los estilos de codificación, todas las técnicas y algoritmos utilizados en la experimentación serán implementadas íntegramente por el autor de la presente tesis doctoral.
- *Analizar los resultados obtenidos durante la fase de experimentación.* Tras llevar a cabo la experimentación, los resultados obtenidos serán analizados para extraer las conclusiones que permitan la validación de las hipótesis definidas. El análisis de los resultados se centrará fundamentalmente en la calidad de la solución y los tiempos de ejecución. Para garantizar que la comparación de los resultados sea

adecuada se utilizarán parámetros estadísticos. Además, se realizarán dos test estadísticos (el test no paramétrico de Friedman y el test post-hoc del Holm) para garantizar que las comparaciones sean rigurosas y objetivas.

Además de los objetivos específicos, el autor de la presente tesis doctoral se ha auto-impuesto un objetivo personal que también será contemplado a lo largo de la realización del trabajo de investigación:

- *Ofrecer libremente a la comunidad los algoritmos generados durante el trabajo de investigación.* Este último objetivo está directamente relacionado con la visión que tiene el autor acerca de la propiedad del conocimiento y el software que lo soporta. Por ese motivo, los algoritmos generados y la experimentación realizada serán descritos con el mayor nivel de detalle posible para que puedan ser reproducidos o utilizados directamente por cualquier persona que así lo desee. Así, tanto el presente documento, como el código fuente y los ficheros de resultados serán accesibles de forma libre y totalmente gratuita.

Por último, el autor pretende generar una librería para el modelado y resolución de problemas de asignación de rutas a vehículos que pueda ser utilizada y modificada libremente bajo licencia GNU GPL, de tal forma que pueda formar parte aplicaciones o sistemas de información mayores.

1.4 Metodología de investigación

Tal y como se ha esbozado en los apartados iniciales, la presente tesis está enmarcada en el ámbito de la resolución de problemas de asignación de rutas a vehículos, que es sin duda alguna uno de los campos de la optimización combinatoria que más intensamente han evolucionado en los últimos años. Por ese motivo, para el desarrollo del trabajo de investigación ha sido necesario el empleo de una metodología iterativa que permite contemplar de una manera continua las novedades aportadas por la actualización del conocimiento que se genera en la comunidad científica.

Con esta premisa, se ha definido un proceso metodológico iterativo en el que cada ciclo contribuye a la mejora de los algoritmos que validan las hipótesis planteadas. De esta forma, los algoritmos propuestos evolucionarán para transformarse en técnicas cada vez más prometedoras, más simples, y que obtengan unos resultados comparables con las técnicas referentes que pueden ser encontradas en la literatura.

La metodología definida se esquematiza en la **figura 1-1**. Dicho esquema ilustra el carácter iterativo de la metodología, siendo las fases principales de la misma las que se detallan a continuación:

- *Revisión del estado del arte:* el objetivo de esta fase es la recopilación y análisis de la literatura de referencia del campo de estudio para identificar los campos de mejora que den pie al planteamiento (en la primera iteración) y refinamiento (en sucesivas iteraciones) de la hipótesis de trabajo. Para lograrlo se revisarán los trabajos publicados en libros, revistas científicas, actas de congresos e informes generados por instituciones y organismos de investigación.
- *Diseño e implementación de los nuevos algoritmos:* una vez revisada la hipótesis, y teniendo en cuenta el conocimiento adquirido durante la fase de revisión del estado del arte, se diseñan e implementan los algoritmos que servirán como soporte para la validación de la hipótesis planteada.

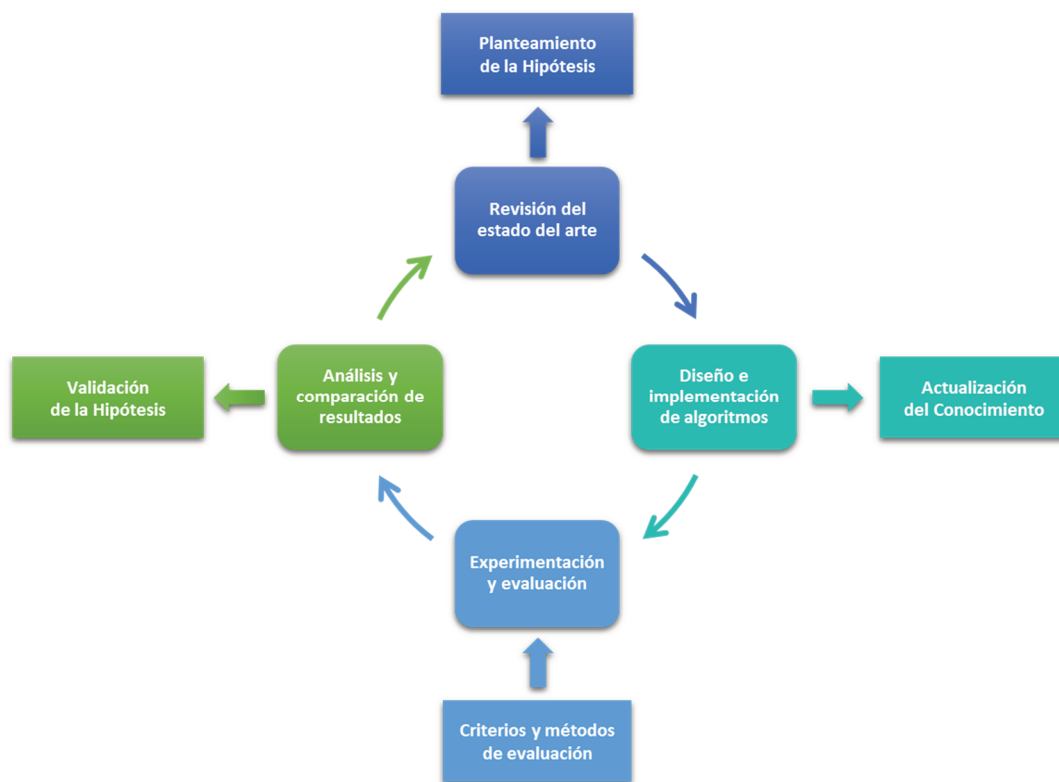


Figura 1-1: Metodología de investigación empleada.

- *Experimentación y evaluación:* esta fase es sin duda una de las más importantes del trabajo puesto que tiene por objeto la experimentación y evaluación de los algoritmos generados para confirmar si la hipótesis definida puede ser corroborada. Para la realización de este paso será necesario utilizar un conjunto de criterios y métodos de evaluación, que incluyen tanto los parámetros de configuración como los juegos de ensayo o instancias de problemas utilizados como referencia.

- *Análisis y comparación de resultados:* tras la fase de experimentación, los resultados obtenidos tienen que ser analizados y comparados con los obtenidos por las técnicas de naturaleza similar que existen en la literatura. Esta fase culmina con validación de las hipótesis planteadas, finalizando el proceso en cuanto dichas hipótesis se validen correctamente; en caso contrario, se iniciará un nuevo ciclo, hasta que se complete la validación de las hipótesis de una manera satisfactoria.

1.5 Estructura del documento de tesis doctoral

En esta sección se describe la estructura del presente documento de tesis doctoral. El documento se ha organizado en torno a seis capítulos:

- El primer capítulo, el presente, incluye: el planteamiento de la problemática, la contextualización científica y social del problema, la descripción de las hipótesis y los objetivos del trabajo realizado, y por último, la metodología de investigación utilizada.
- El segundo capítulo ahonda en la problemática en la que se enmarca el trabajo de tesis doctoral: los problemas de asignación de rutas a vehículos. En dicho capítulo se describen los tipos de problemas fundamentales así como las variantes más relevantes, junto con una nutrida colección de referencias bibliográficas que configuran un excelente punto de partida para la revisión del estado del arte del ámbito en que se enmarca el trabajo realizado.
- El tercer capítulo puede considerarse como una continuación del segundo, pero en este caso, se abordan las técnicas de resolución de problemas de asignación de rutas a vehículos. En concreto, se presenta información acerca de las principales técnicas constructivas, las técnicas de búsqueda local, las metaheurísticas, y se finaliza el capítulo con un apartado en el que se destacan los factores clave que debiera contemplar una metaheurística para ser aplicada de manera satisfactoria a problemas de asignación de rutas a vehículos.
- En el cuarto capítulo se realiza una descripción detallada de los algoritmos que sustentan la validación de las hipótesis planteadas en el presente trabajo de tesis doctoral. Junto con la descripción de los algoritmos, se explican las aportaciones y contribuciones de los mismos, así como las similitudes y diferencias respecto a otras técnicas de carácter similar existentes en la literatura.
- El quinto capítulo se centra en la experimentación llevada a cabo para evaluar los algoritmos propuestos, así como el análisis de los resultados obtenidos. Los diferentes apartados que componen este capítulo describen las técnicas, los

parámetros de configuración, los juegos de ensayo utilizados durante el proceso de experimentación y el análisis de los resultados obtenidos. Además, se incorpora un apartado relacionado con las buenas prácticas o pautas que debieran emplearse para la correcta comparación y presentación de resultados de una nueva técnica de resolución.

- Finalmente, el sexto y último capítulo cierra el documento con las conclusiones y las líneas futuras de trabajo que quedan abiertas tras la realización del presente trabajo de tesis doctoral.

2

Problemas de generación de rutas de vehículos

Este capítulo pretende ofrecer una visión global de los problemas de generación de rutas de vehículos recopilando información acerca de las diferentes variantes o tipologías de problemas, y acompañando cada una de ellas con las referencias bibliográficas más significativas. El capítulo está organizado en cuatro secciones: la sección 2.1 presenta el problema del viajante, la sección 2.2 introduce el problema básico de asignación de rutas a vehículos, la sección 2.3 describe las variantes más representativas del problema de asignación de rutas a vehículos, y para cerrar el capítulo, la sección 2.4 presenta dos enfoques para la clasificación de las variantes del problemas de asignación de rutas a vehículos.

2.1 El problema del viajante

En esta sección se define y formula el que es, sin duda, la referencia en el ámbito de problemas de generación de rutas de vehículos: el problema del viajante.

2.1.1 Definición del problema

El problema del viajante (en adelante TSP - *Travelling Salesman Problem*) es un tipo de problema¹ de optimización combinatoria tipificado como *NP-Complejo*² ampliamente estudiado desde la investigación operativa y las ciencias de la computación. Su definición es muy sencilla:

Dado un conjunto de ciudades geográficamente dispersas, y las distancias entre cada una de ellas; el problema consiste en encontrar la ruta más corta que pase por todas las ciudades una única vez.

En la **figura 2-1** se muestra una instancia de un problema de tipo TSP con 8 ciudades (es decir, $n = 8$) y una posible solución.

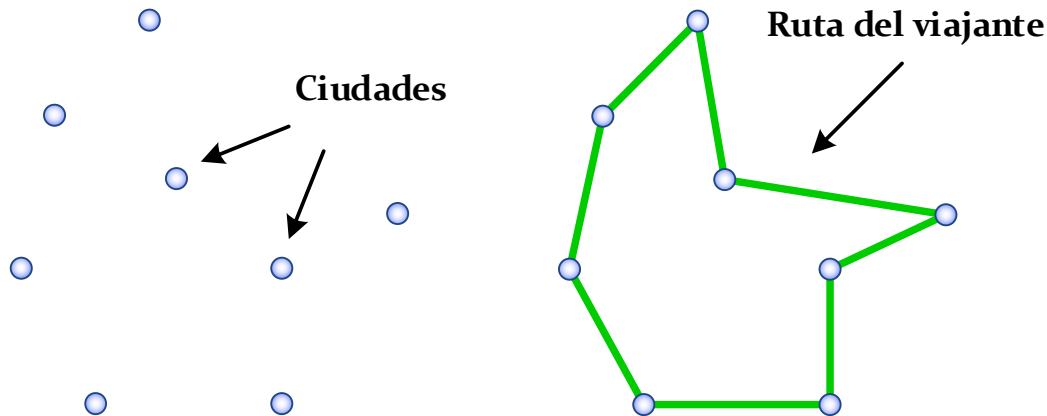


Figura 2-1: Instancia de un TSP y una posible solución del mismo.

El problema fue definido en 1800 por los matemáticos Hamilton y Kirkman, pero su primera formulación data del año 1930, y fue definida por Karl Menger. Desde entonces, el problema del TSP es uno de los problemas de optimización combinatoria más intensamente estudiados y está considerado como uno de los problemas canónicos de

¹ En el ámbito de la optimización combinatoria, un problema representa realmente a un conjunto o tipología de problemas, es decir, cuando se habla del problema del TSP, se hace referencia al conjunto de instancias de problemas de la clase TSP. Esta relación entre instancias y clases, es similar a la relación entre el concepto de suma, y el ejemplo de suma concreta: $2 + 2$.

² Los problemas de optimización combinatoria se clasifican de acuerdo con el coste computacional (recursos necesarios) del mejor algoritmo conocido que resuelve el problema, y más específicamente con el orden de complejidad temporal de: a) El mejor algoritmo conocido que verifica si una solución concreta del problema es la solución óptima global (algoritmo de verificación); b) El mejor algoritmo conocido que obtiene la solución óptima global del problema (algoritmo de resolución). De acuerdo con esto, se tienen cuatro clases no disjuntas de problemas: los problemas P, que poseen algoritmos de resolución de complejidad temporal potencial, los problemas NP, que poseen algoritmos de verificación de complejidad temporal potencial, los problemas NP-completos, que poseen algoritmos de verificación de complejidad temporal potencial, pero se conjetura que sólo pueden resolverse mediante algoritmos de complejidad temporal superior a la potencial, y finalmente los problemas NP-difíciles, que no poseen algoritmos conocidos de verificación (ni por supuesto de resolución) de complejidad temporal potencial o inferior.

optimización combinatoria, puesto que sirve como banco de pruebas para la validación de muchos métodos y algoritmos de optimización.

El objetivo básico de este problema es minimizar la distancia total recorrida por el viajante. Por lo tanto, un enfoque intuitivo y sencillo para la resolución de este problema sería analizar todas las ordenaciones posibles de ciudades sin repetición, es decir todas las permutaciones de ciudades, hasta encontrar la ruta de menor distancia recorrida. Este enfoque, conocido como *fuerza bruta*, permite obtener la solución exacta del problema pero tiene una complejidad computacional del orden de $O(n!)$, lo que hace intratable la resolución del problema de un tamaño considerable utilizando esta técnica. Por ejemplo: para una instancia del problema con 20 ciudades ($n = 20$) sería necesario analizar $2,433 \cdot 10^{18}$ permutaciones para encontrar la solución exacta, lo que se antoja prácticamente imposible tanto por tiempo como por capacidad de cómputo.

Debido a la elevada complejidad computacional del problema, salvo en el caso de instancias pequeñas (con pocas ciudades), la mayor parte de enfoques de resolución se basan en la búsqueda de soluciones sub-óptimas por medio del empleo de técnicas heurísticas, metaheurísticas, o de relajación y descomposición del problema. Este tipo de técnicas son capaces de obtener buenas aproximaciones a la solución real de una instancia concreta del problema pero sin garantías de que dicha solución sea la óptima. De hecho, la bondad de las técnicas de aproximación se mide en base a los resultados obtenidos al resolver instancias de problemas con soluciones conocidas que se utilizan como batería de prueba (o *benchmark*) durante el proceso de validación de un nuevo método o algoritmo.

El problema del TSP tiene multitud de aplicaciones en la vida real, desde las más directas en el ámbito del transporte y la logística, hasta otras no tan obvias como puede ser el diseño de circuitos integrados (donde se puede utilizar el TSP para determinar la ubicación de los componentes electrónicos y las soldaduras).

Además, existen multitud de variantes del problema clásico, que surgen al intentar resolver problemas de la vida real en los que existen restricciones o limitaciones, como por ejemplo: límites temporales o de distancia máxima recorrida, ventanas de tiempo en las que debe pasarse por una ciudad (o cliente), tiempos variables de recorrido entre dos ciudades debido a las condiciones del tráfico o topología de las carreteras, e incluso cuestiones relacionadas con las condiciones y horarios de trabajo del “viajante” que realizará la ruta. En (Bellmore y Nemhauser 1968), (Lawler, y otros 1985), (Reinelt 1994) (Applegate, y otros 2007) y (Davendra 2010) se puede encontrar información más detallada acerca del TSP, sus variantes y sus aplicaciones prácticas.

2.1.2 Formulación matemática del problema

El problema de tipo TSP puede definirse formalmente de la siguiente forma:

Sea $G = (\mathcal{V}, \mathcal{E})^3$ un grafo completo y no dirigido, donde con $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ representa al conjunto de nodos o clientes que deben ser visitados por el viajante. El conjunto de arcos o aristas $\mathcal{E} = \{(v_i, v_j) : v_i, v_j \in \mathcal{V}, i \neq j\}$ representa la posibilidad de viajar directamente de un nodo $v_i \in \mathcal{V}$ a otro nodo diferente $v_j \in \mathcal{V}$, y tiene un coste d_{ij} (normalmente la distancia) asociado al trayecto entre el cliente i y el cliente j . El objetivo del TSP es la generación (o búsqueda) de una ruta tal que, comenzando y finalizando en el mismo cliente, cumpla las siguientes restricciones:

- Todos los clientes deben ser visitados una única vez.
- El coste total (normalmente medido en distancia) del transporte debe ser minimizado.

En relación a la formulación matemática del TSP, en la literatura pueden encontrarse diferentes formulaciones, pero la más representativa puede ser la de (Dantzig, Fulkerson y Johnson 1954):

$$\text{Min.} \quad \sum_{(i,j) \in \mathcal{V}} d_{ij} \cdot x_{ij} \quad (1)$$

$$\text{Sujeto a} \quad \sum_{i \in \Delta^+(i)} x_{ij} = 1, \quad \forall i \in \mathcal{V} \quad (2)$$

$$\sum_{i \in \Delta^-(j)} x_{ij} \geq 1, \quad \forall j \in \mathcal{V} \quad (3)$$

$$\sum_{i \in S, j \in \Delta^+(i) \setminus S} x_{ij} \geq 1, \quad \forall S \subset \mathcal{E} \quad (4)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in \mathcal{E} \quad (5)$$

Siendo la variable x_{ij} una variable binaria que toma el valor 1 si el arco es utilizado, y 0 en caso contrario. La ecuación (1), que representa a la función objetivo, es el sumatorio del coste de todos los arcos utilizado en una solución. Normalmente este sumatorio representa la distancia total de la ruta o solución. Las restricciones (2) y (3) garantizan que todo nodo tiene que ser visitado y abandonado una sola vez, mientras que la restricción (4) garantiza que no se generan subrutas (*sub-tours*) e indica que todo subconjunto de nodos S tiene que ser abandonado al menos una vez. Esta

³ \mathcal{V} (Vertices) y \mathcal{E} (Edges) representan al conjunto de vértices y arcos/aristas del grafo respectivamente.

restricción es de suma importancia, puesto que si no estuviese presente la solución podría estar compuesta por más de una ruta.

2.2 El problema de asignación de rutas a vehículos

En esta sección se describe la variante básica del problema de asignación de rutas a vehículos, que se corresponde con la tipología de problema general sobre el que versará el trabajo de investigación realizado.

2.2.1 Definición del problema

El problema de asignación de rutas a vehículos (en adelante VRP - *Vehicle Routing Problem* o CVRP - *Capacited Vehicle Routing Problem*), al igual que el TSP, es otro tipo de problema de optimización combinatoria de la clase *NP-hard* (Lenstra y Kan 1981), que al igual que el TSP, ha sido muy estudiado en los ámbitos de la investigación operativa y la ciencia de la computación. Una definición de este problema puede ser la siguiente:

Dado un conjunto de ciudades o clientes geográficamente dispersos cada uno de ellos con una determinada demanda, el problema consiste en generar las rutas óptimas, para una flota de vehículos que salen y regresan a uno o varios almacenes, que satisfagan la demanda de todos los clientes.

La **figura 2-2** muestra la información de entrada que define una instancia de un problema de tipo VRP y una posible solución al mismo.

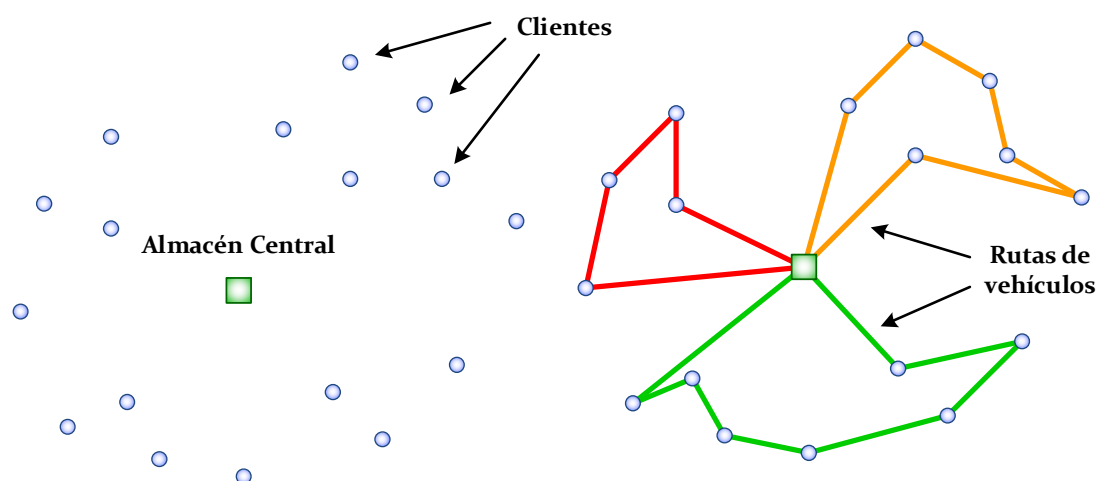


Figura 2-2: Instancia de un VRP y una posible solución del mismo.

Se puede decir que el problema de tipo VRP es una generalización del TSP si se considera que en un problema de tipo VRP hay que definir m rutas óptimas para dar servicio a todos los clientes que deben atenderse desde uno o varios almacenes, siendo cada una de las rutas una instancia de un problema de tipo TSP. Esta similitud entre los dos tipos de problemas ha hecho que muchos de los métodos de resolución de problemas de tipo TSP hayan sido adaptados a la resolución de los problemas de tipo VRP.

Por otro lado, en relación a las diferencias, mientras que el problema de tipo TSP se centra únicamente en la ordenación de los clientes para obtener la menor distancia posible de la ruta (siendo esté su único objetivo), el problema de tipo VRP debe tener en cuenta tanto la ordenación de los clientes en cada una de las rutas, como el número de rutas generadas. Por lo tanto, el problema de tipo VRP tiene una función objetivo con dos objetivos, el primero de ellos consiste en minimizar el número de rutas o vehículos, y el segundo se centra en minimizar la distancia total recorrida por todas las rutas.

Como se ha mencionado antes, y al igual que en el caso del TSP, el VRP tiene una elevada complejidad computacional, lo que dificulta el empleo de métodos exactos para la resolución de instancias del problema con varias decenas de clientes.

El problema de tipo VRP fue inicialmente definido por Dantzig y Ramser en 1959 con el nombre *Truck Dispatching Problem* (Dantzig y Ramser 1959), pero hasta varios años después no fue acuñado con el nombre *Vehicle Routing Problem* (Christofides 1976). En los últimos 50 años se ha convertido en uno de los problemas de optimización combinatoria más estudiados, puesto que intenta dar respuesta a una problemática muy extendida en el ámbito del transporte, la distribución y la logística. La importancia de las técnicas de resolución de problemas de tipo VRP en estos ámbitos viene dada porque el uso de herramientas de optimización de los procesos de transporte que mejoren la generación de rutas de vehículos puede representar un ahorro de entre el 5% y el 20% de los costes de dichos procesos (Toth y Vigo 2002a).

Desde los años 60 los problemas de tipo VRP han sido un ámbito intenso de investigación y prueba de ello son los numerosos métodos de resolución exactos, heurísticos y meta-heurísticos que pueden encontrarse en la literatura, tal y como ilustran varios artículos recopilatorios: (Baldacci, Toth y Vigo 2007), (Cordeau, y otros 2007), (Gendreau, y otros 2008), (Potvin 2009), (Laporte 2009); y libros específicos: (Golden y Assad 1988), (Toth y Vigo 2002a), (Golden, Raghavan y Wasil 2008) y (Toth y Vigo 2015).

2.2.2 Formulación matemática del problema

El problema de tipo VRP puede definirse formalmente de la siguiente forma (Vidal, y otros 2013):

Sea $G = (\mathcal{V}, \mathcal{E})$ un grafo completo y no dirigido con $|\mathcal{V}| = n + 1$ nodos o vértices. El nodo $v_0 \in \mathcal{V}$ representa al almacén donde se guardan las mercancías que deben ser distribuidas y existe una flota de m vehículos idénticos con capacidad Q . El resto de nodos $v_i \in \mathcal{V} \setminus \{v_0\}$, para $i \in \{1, \dots, n\}$, representan a los clientes que deben ser visitados, cada uno de ellos con una demanda de mercancía no negativa q_i . Los arcos/aristas $(i, j) \in \mathcal{E}$ representan la posibilidad de viajar directamente de un nodo $v_i \in \mathcal{V}$ (cliente o almacén) a otro nodo diferente $v_j \in \mathcal{V}$, y tiene un coste c_{ij} (proporcional a la distancia) asociado al transporte. El objetivo del VRP es la generación (o búsqueda) de m o menos rutas que cumplan las siguientes restricciones:

- *Todos los clientes deben ser visitados.*
- *La demanda de cada cliente tiene que ser totalmente atendida por un único vehículo.*
- *La demanda total de los clientes asociados a una ruta no debe superar la capacidad de los vehículos Q .*
- *El coste total (normalmente medido en distancia) del transporte debe ser minimizado.*

En relación a la formulación del VRP, en la literatura pueden encontrarse diferentes formulaciones pero una de las más representativas es la formulación basada en programación lineal entera propuesta por Fisher y Jaikumar (Fisher y Jaikumar 1981) puesto que engloba tanto el aspecto de asignación (de clientes a vehículos) y como de ordenación (de clientes atendidos por un vehículo) que contempla el problema de tipo VRP. Dicha formulación está basada en dos familias de variables binarias:

- y_{ik} , toma el valor 1 si el cliente i se asigna al vehículo k (y 0 en caso contrario; $y_{ok} = 1$ representa que el vehículo k realizará una ruta).
- x_{ijk} , toma el valor 1 si el vehículo k visita el nodo v_j inmediatamente después del nodo v_i ($i \neq j$).

A continuación, se muestran las restricciones que definen el problema:

$$\text{Min.} \quad \sum_{i=0}^n \sum_{j=0}^n \sum_{k=1}^m c_{ij} \cdot x_{ijk} \quad (1)$$

$$\text{Sujeto a} \quad \sum_{k=1}^m y_{ik} = 1 \quad i = 1, \dots, n \quad (2)$$

$$\sum_{k=1}^m y_{0k} \leq m \quad (3)$$

$$\sum_{i=1}^n q_i \cdot y_{ik} \leq Q \quad k = 1, \dots, m \quad (4)$$

$$\sum_{j=0}^n x_{ijk} = \sum_{j=0}^n x_{jik} = y_{ik} \quad i = 0, \dots, n; k = 1, \dots, m \quad (5)$$

$$\sum_{v_i \in S} \sum_{v_j \in S} x_{ijk} \leq |S| - 1 \quad k = 1, \dots, m; S \in V \setminus \{0\}; |S| \geq 2 \quad (6)$$

$$y_{ik} \in \{0, 1\} \quad i = 0, \dots, n; k = 1, \dots, m \quad (7)$$

$$x_{ijk} \in \{0, 1\} \quad i = 0, \dots, n; j = 0, \dots, n; k = 1, \dots, m \quad (8)$$

Las restricciones (2) – (4) presentan la estructura de un problema de tipo “carga de contenedores” *Bin Packing*⁴ con m contenedores. Estas restricciones garantizan la asignación de cada cliente a un único vehículo, el número máximo de vehículos existentes y la capacidad de los vehículos. Las restricciones (5) y (6) están relacionadas con la estructura de las rutas, garantizando la selección de un número adecuado de arcos con destino y origen a cada nodo (clientes y almacén), y eliminando sub-rutas (es decir, rutas que no empiezan y terminan en el almacén). El número de restricciones de este tipo crece exponencialmente con el número de clientes. Dada esta formulación, el TSP puede verse como un caso especial del VRP cuando $m = 1$ y $Q = +\infty$.

Además de las ecuaciones descritas, la literatura suele incluir otra adicional (9). Dicha ecuación está relacionada con el tamaño o duración máxima de la ruta. Cada cliente tiene asociado un tiempo de servicio τ_i ; y la suma de los tiempos de servicio más los tiempos de trayecto de todos los clientes asignados a un vehículo no puede exceder la duración máxima de la ruta T .

$$\sum_{i=0}^n \sum_{j=0}^n (c_{ij} + \tau_i) \cdot x_{ijk} \leq T \quad k = 1, \dots, m \quad (9)$$

⁴ El problema de *Bin Packing* es un problema de decisión que consiste en encontrar el menor número de contenedores idénticos (todos con un volumen máximo) necesarios para contener n elementos, cada uno de ellos con un volumen conocido. (Garey y Johnson 1979).

2.3 Variantes del VRP

Al igual que el problema de tipo TSP, el problema de VRP surgió como respuesta a la resolución de una problemática real que existía en el ámbito del transporte y la logística. Desde su definición inicial, y en los últimos años, han surgido multitud de variantes al problema básico para dar respuesta a problemas de la vida real.

En esta sección se describen las principales variantes del VRP, realizando un énfasis especial en la variante con ventanas de tiempo (VRPTW - *VRP with Time Windows*), que es una de las variantes más intensamente estudiadas en la literatura, y que será la base para el desarrollo de la presente tesis doctoral.

2.3.1 Variante con almacenes múltiples - MDVRP

La variante con almacenes múltiples (*Multiple Depot Vehicle Routing Problem - MDVRP*) se caracteriza por la existencia de varios almacenes desde los que puede servirse la demanda de los clientes. Cada vehículo está asignado a un único almacén debiendo iniciar y finalizar las rutas en dicho almacén (aunque existen variantes en las que se permite finalizar las rutas en un almacén distinto del original). La variante más tradicional no impone ninguna limitación en cuanto a la demanda que puede atender cada almacén, pero dicha demanda podría estar sujeta a restricciones de capacidad para el almacén.

En (Ombuki-Berman y Hanshar 2009) y (Vidal, y otros 2012) pueden encontrarse referencias recientes a esta variante.

2.3.2 Variante con flota de vehículos heterogénea - HVRP

En la variante con flota de vehículos heterogénea (*Heterogeneous fleet Vehicle Routing Problem - HVRP*) los vehículos poseen diferentes características: capacidad, duración máxima de la ruta, velocidad o coste asociado a los desplazamientos. Si el número de vehículos no está limitado, el problema se particulariza para encontrar el “mix” entre número y tipo de vehículos que minimice la función objetivo (*Fleet Size and Mix Vehicle routing problema - FSMVRP*).

Información adicional acerca de esta variante puede encontrarse en (Baldacci, Battarra y Vigo 2008).

2.3.3 Variante con rutas periódicas - PVRP

En la variante periódica del VRP (*Periodic Vehicle Routing Problem - PVRP*) las rutas se planifican de forma periódica para un horizonte temporal determinado. En el VRP tradicional el horizonte temporal suele ser un día pero en la variante periódica el horizonte se extiende a M días. En este caso, cada cliente tiene una demanda total que debe ser atendida a lo largo del horizonte temporal se acuerdo a diferentes combinaciones de aceptación para cada uno de los periodos.

Un ejemplo práctico de esta variante podría ser el siguiente: una serie de establecimientos de hostelería demandan semanalmente una cantidad de un determinado producto. Como la demanda completa no puede satisfacerse en un único día y entrega, se planifican varias entregas fraccionadas a lo largo de la semana. Cada cliente propone diferentes alternativas de fracciones de la cantidad total para satisfacer la demanda completa. El objetivo del problema de tipo PVRP consiste en diseñar las rutas periódicas que garanticen la satisfacción de la demanda completa de cada cliente respetando las restricciones de fraccionamiento junto con los objetivos básicos del VRP: minimizar el número de vehículos utilizado así como la distancia total recorrida.

Una descripción más exhaustiva de esta variante puede encontrarse en (Francis, Smilowitz y Tzur 2008) y (Gulczynski, Golden y Wasil 2011).

2.3.4 Variante con entregas fraccionadas - SDVRP

En la variante con entregas fraccionadas (*Split Deliveries Vehicle Routing Problem*), la demanda de cada cliente puede ser satisfecha por varios vehículos, por lo tanto un mismo cliente podrá formar parte de varias rutas, satisfaciendo cada una de ellas una fracción de la demanda total.

Una revisión detallada de las particularidades de esta variante puede encontrarse en: (Chen, Golden y Wasil 2007) y (Gulczynski, Golden y Wasil 2008).

2.3.5 Variante con recogida - VRPB

En la variante con entregas y recogidas o simplemente con recogidas (*Vehicle Routing Problem with Backhauls*) se diferencian dos tipos de clientes: “clientes con entrega” (*delivery or linehauls customers*) y “clientes con recogida” (*backhauls customers*). El primer tipo de cliente requiere la entrega de mercancía y el segundo la recogida o retirada de mercancía que ha de ser retornada al almacén central. Las rutas que configuran una solución a esta variante pueden contener clientes de un único tipo, o de ambos tipos. En este último caso, lo habitual es atender primero a todos los clientes con entrega y posteriormente a clientes con recogida (este esquema el que

habitualmente se utiliza en la vida real, para minimizar el tiempo de carga y descarga de la mercancía).

En (Toth y Vigo 2002b) y (Parragh, Doerner y Hartl 2008a) se pueden encontrar revisiones detalladas de la variante VRPB.

2.3.6 Variante con recogida y entrega - VRPPDP

La variante con recogida y entrega (*Vehicle Routing Problem with Pickups and Deliveries - VRPPD*) se caracteriza por el hecho de que el servicio de cada cliente tiene asociado un punto de recogida y un punto de entrega. La recogida asociada a cada servicio siempre tiene que ser atendida antes que la entrega. Esta variante surge de los problemas de la vida real denominados transporte bajo demanda o “llamada y viaje” (*dial-a-ride*) en los que un cliente solicita la recogida (de personas o mercancías) en un punto y la entrega en otro diferente.

Esta variante posee una amplia y extensa literatura entre la que destaca: (Parragh, Doerner y Hartl 2008b), (Cordeau, Laporte y Ropke 2008) y (Berbeglia, Cordeau y Laporte 2010).

2.3.7 Variante con varias rutas por vehículo - VRPMT

Esta variante contempla la posibilidad de que un mismo vehículo pase varias veces por el almacén central para dejar o coger mercancía dentro del horizonte temporal de planificación (*Vehicle Routing Problem with Multiple Trips*). De esta forma, se generan varias rutas para un mismo vehículo.

En (Taillard, Laporte y Gendreau 1995) se puede encontrar información detallada acerca de la definición de esta variante.

2.3.8 Variante abierta - OVRP

La variante abierta del VRP (*Open Vehicle Routing Problem*) se caracteriza por el hecho de no contabilizar el coste del retorno de los vehículos al almacén central, al igual que hacen las empresas de transporte (que no suelen facturar dicho coste).

Una revisión detallada de esta variante está disponible en (Li, Golden y Wasil 2007).

2.3.9 Variante con regulaciones laborales

La variante con regulaciones laborales trata de contemplar las condiciones laborales de los conductores de los vehículos. Teniendo esto presente, existen distintos tipos de

variantes que intentan contemplar las regulaciones laborales en cuanto a tiempo máximo de circulación o periodos obligatorios de descanso. Dichas restricciones normalmente están vinculadas a ventanas de tiempo asociadas a los vehículos y/o conductores.

Esta variante es una de las más recientes y menos desarrollada debido a la complejidad y diferencias que existen en cuanto a regulaciones laborales en distintas regiones. Esto hace que los trabajos realizados en torno a esta variante estén muy centrados en las regulaciones laborales específicas de Estados Unidos (Goel y Kok 2011) y la Unión Europea (Goel 2010).

2.3.10 Variante con ventanas de tiempo - VRPTW

La variante con ventanas de tiempo del VRP (*Vehicle Routing Problem with Time Windows - VRPTW*) es sin duda alguna la más estudiada en la literatura y es considerada como el prototipo de VRP “rico” (*rich*) puesto que la evaluación del cumplimiento de las ventanas de tiempo requiere la realización de tareas sofisticadas.

Esta variante asocia a cada cliente v_i un rango temporal $[e_i, l_i]$ denominado “ventana de tiempo” formado por los límites mínimo y máximo de inicio, también conocidos como el inicio más temprano (*earliest beginning*) y el inicio más tardío (*latest beginning*). Además, también se establece una ventana de tiempo u horizonte de planificación $[e_0, l_0]$ para el almacén central v_0 . Esta última ventana de tiempo especifica el rango temporal en que los vehículos han de salir del almacén central y regresar al mismo. Por último, si el vehículo llega a la ubicación del cliente v_i antes del inicio de su ventana de tiempo e_i se genera un tiempo de espera w_i (*waiting time*), cuyo sumatorio total W puede ser un criterio adicional de optimización unido al número de rutas M y la distancia total D .

La formulación matemática del VRPTW se asemeja mucho a la del VRP, e incorpora unas ecuaciones específicas relativas al cumplimiento de las ventanas temporales:

$$e_i \leq t_{ik} \leq l_i \quad v_i \in \mathcal{V}; k \in \{1, \dots, m\} \quad (10)$$

$$t_{ik} \in \mathbb{R}^+ \quad v_i \in \mathcal{V}; k \in \{1, \dots, m\} \quad (11)$$

La ecuación (10) garantiza que el servicio en el cliente comience dentro de los límites temporales establecidos por la ventana de tiempo. El valor t_{ik} representa el instante de inicio del servicio del cliente v_i en la ruta que realiza el vehículo k .

Las ventanas de tiempo en sí mismas configuran un conjunto de variantes desde la más tradicional con ventanas exigentes o absolutas (*hard time windows*); la de ventanas flexibles (VRP with *soft-time windows*), que permite el incumplimiento de los límites temporales, pero con penalizaciones en la función objetivo; la de múltiples ventanas

por cliente (VRP with *multiple time windows* - *VRPMTW*), en la que los clientes tiene un conjunto variable de ventanas de tiempo alternativas; o incluso las que contemplan ventanas de tiempo para los descansos de los conductores (VRPTW with *Lunch Breaks* - *VRPTWLB*).

En (Bräysy y Gendreau 2005a), (Bräysy y Gendreau 2005b) y (Gendreau y Tarantilis 2010) se pueden encontrar las revisiones del estado del arte más recientes en torno a la resolución de problemas del tipo VRPTW.

2.4 Clasificación de las variantes del VRP

Esta sección es una adaptación del trabajo realizado en (Vidal, y otros 2013) dónde se propone un nuevo enfoque para clasificar las variantes del VRP en función de los atributos o características que presentan. Esta clasificación recibe el nombre de VRP Multiatributo (*Multi-Attribute Vehicle Routing Problems* - *MAVRP*).

Debido a que los problemas de tipo VRP responden a problemáticas de la vida real, el número de variantes es realmente grande y consecuentemente la literatura en torno a este ámbito es vasta y compleja de analizar y clasificar. Desde los años 70 se han realizado varios intentos por definir una taxonomía de problemas de tipo VRP con el objetivo fundamental de unificar los esfuerzos y facilitar la labor de desarrollo de nuevas técnicas específicas para una variante concreta.

Así, en (Bodin 1975) puede encontrarse la primera aproximación a una taxonomía de problemas de tipo VRP. Desde entonces, motivado por la evolución de los problemas de la vida real, han surgido nuevas variantes lo que ha provocado que dicha taxonomía se haya quedado un tanto obsoleta. Más recientemente, en 2009 (Eksioglu, Vural y Reisman 2009) se presenta una taxonomía mucho más completa que integra las variantes más significativas junto con información bibliométrica relativa a número de artículos, autores referentes, temáticas y publicaciones de referencia. A modo de ilustrativo se incluyen en la **tabla 2-1** un extracto de los dos primeros niveles de dicha taxonomía. En julio de 2014 (De Jaegere, Defraeye y Van Nieuwenhuyse 2014) se actualizó el trabajo previo de Eksioglu y otros, con el análisis de la literatura generada entre 2009 y 2014.

1. Type of study 1.1. Theory 1.2. Applied methods 1.2.1. Exact methods 1.2.2. Classical Heuristics 1.2.3. Metaheuristics 1.2.4. Simulation 1.2.5. Real-time solution methods 1.3. Implementation documented 1.4. Survey, review or meta-research	3. Problem Physical Characteristics 3.1. Transportation network design 3.2. Location of addresses (customers) 3.3. Number of points of origin 3.4. Number of points of loading/unloading facilities (depot) 3.5. Time window type 3.6. Number of vehicles 3.7. Capacity consideration 3.8. Vehicle homogeneity (Capacity) 3.9. Travel time 3.10. Objective
2. Scenario Characteristics 2.1. Number of stops on route 2.2. Load splitting constraint 2.3. Customer service demand quantity 2.4. Request times of new customers 2.5. Onsite service/waiting times 2.6. Time window structure 2.7. Time horizon 2.8. Backhauls 2.9. Node/Arc covering constraints	4. Information Characteristics 4.1. Evolution of information 4.2. Quality of information 4.3. Availability of information 4.4. Processing of information 5. Data Characteristics 5.1. Data used 5.2. No data used

Fuente: (Eksioglu, Vural y Reisman 2009)

Tabla 2-1: Taxonomía de problemas de tipo VRP de Eksioglu y otros.

De acuerdo a la revisión de la literatura realizada en (Vidal, y otros 2013), las diferentes variantes de problemas de tipo VRP pueden organizarse en torno a 3 categorías fundamentales de atributos o características:

- *Asignación de atributos o recursos (ASSIGN)*: estos atributos están relacionados con la asignación de recursos limitados que representan restricciones globales de todo el problema que deben contemplarse a la hora de asignar recursos a rutas o clientes. En esta categoría se puede encontrar la variante con múltiples almacenes, la flota de vehículos heterogénea, la variante periódica, la variante en la que un cliente es servido por varios vehículos y la variante de recolección de beneficios.
- *Elección de la secuencia (SEQ)*: estos atributos afectan a la naturaleza y estructura de las rutas. La existencia de clientes de entrega o recogida, la posibilidad de que un vehículo pueda realizar varias rutas, o la existencia de almacenes intermedios en los que los vehículos puede coger o dejar mercancía, hacen que el diseño de las rutas deba adaptarse a que un vehículo pase varias veces por el almacén en su ruta, o que un cliente sea visitado por varios vehículos; y todo ello afecta directamente a la manera en que se construyen las rutas.
- *Evaluación de la ruta (EVAL)*: estos atributos afectan a la manera en que se evalúan las rutas tanto desde el punto de vista del coste, como la idoneidad de las mismas, así como la manera en que se pondera la optimización de las mismas una vez creadas. Esta categoría se subdivide en otras dos en base a que el atributo en

cuestión haga referencia a una ruta individual o a un grupo de rutas. Ésta es sin duda la categoría con mayor número de atributos y más presentes en la literatura. En esta categoría destacan especialmente la variante abierta, la variante con ventanas de tiempo, las variantes con regulaciones laborales, la variante con restricciones especiales de carga y la variante con dependencias temporales.

En la siguiente tabla se muestran las tres categorías mencionadas junto con las principales variantes de problemas de tipo VRP asociadas a cada una de ellas.

ASSIGN	SEQ	EVAL
Múltiples almacenes (MDVRP)	Con recogida (VRPB)	Ventanas de tiempo (VRPTW)
Vehículos heterogéneos (HVRP)	Recogida y entrega (VRPPDP)	Variante abierta (OVRP)
Ruta periódicas (PVRP)	'M' rutas por vehículo (VRPMT)	Regulaciones laborales
Entregas fraccionadas (SDVRP)		

Fuente: (Vidal, y otros 2013)

Tabla 2-2: Clasificación de variantes del VRP en base a sus características.

3

Técnicas de resolución de problemas de tipo VRP

La resolución de problemas de generación de rutas de vehículos presentados en el capítulo 2 es una tarea computacionalmente compleja⁵ de abordar de una manera exacta, puesto que dichos problemas están catalogados como NP-difíciles, lo que implica la inexistencia de una técnica o algoritmo que sea capaz de resolver dichos problemas en un tiempo polinómico. Por ese motivo, las técnicas más utilizadas para la resolución de este tipo de problemas se enmarcan en el ámbito de las técnicas heurísticas y metaheurísticas que permiten la obtención de soluciones subóptimas, pudiendo en algunos casos encontrar soluciones óptimas o muy próximas a la solución óptima (para problema de un tamaño controlado).

Tal y como se comentó en el capítulo 2, el VRP es una generalización del TSP, pero su resolución es mucho más compleja. En la literatura existen técnicas aproximadas capaces de resolver problemas de tipo TSP con cientos y hasta miles de nodos, aportando soluciones que se suponen óptimas globales (sin asegurarlo) como es el caso del algoritmo Concorde (Applegate, y otros 2007). Por otro lado, el caso del VRP las mejores técnicas cuasiexactas (Fukasawa, y otros 2006) (Baldacci, Christofides y Mingozzi 2008) difícilmente resuelven de manera óptima instancias del problema de más de un centenar de nodos. Debido a que los problemas de la vida real superan el centenar de nodos, y el tiempo de respuesta necesario suele estar acotado, en la

⁵ En este caso, la complejidad hace referencia a los recursos computacionales necesarios para resolver el problema. Dichos recursos normalmente se miden en base al tiempo y la cantidad de memoria necesaria para la resolución de un problema.

práctica, sólo las técnicas subóptimas como las heurísticas y las metaheurísticas pueden ser utilizadas con garantías de éxito.

La diferencia fundamental entre una técnica heurística y una técnica metaheurística viene determinada por el tipo de información que utiliza cada una de ellas, lo que condiciona tanto el proceso de búsqueda como el tipo de problemas a los que puede ser aplicada cada una de ellas. Por un lado, las técnicas heurísticas utilizan información específica de un problema concreto, lo que limita su aplicación a dicho problema. Por otro lado, las técnicas metaheurísticas utilizan información genérica que está presente en diferentes tipos de problemas de optimización combinatoria, lo que permite su aplicación a una mayor variedad de problemas. Además, la versatilidad de las técnicas metaheurísticas, las hace especialmente atractivas para problemas complejos sujetos a múltiples restricciones u objetivos contrapuestos, como puede ser el caso de algunas de las variantes del VRP.

En este capítulo se realiza una revisión de las técnicas más utilizadas para la resolución de problemas de tipo VRP. En dicha revisión se presta una especial importancia a las heurísticas constructivas y a las búsquedas locales puesto que son la base para la definición de la aportación de la presente tesis doctoral. Además, se focaliza la revisión en las técnicas específicas para la resolución de problemas de tipo VRPTW puesto que éste es el problema elegido para el desarrollo de la tesis y la experimentación posterior. El capítulo se organiza en 4 secciones: la sección 3.1 aborda las heurísticas constructivas, la sección 3.2 recoge aspectos relacionados con las técnicas de búsqueda local y los vecindarios, la sección 3.3 se centra en las técnicas metaheurísticas más utilizadas en el contexto de los problemas de tipo VRP, y por último, para finalizar el capítulo, la sección 3.4 destaca los aspectos clave determinan el éxito de una buena técnica metaheurística para la resolución de problemas de tipo VRP.

3.1 Heurísticas constructivas

Este tipo de técnicas tuvieron mucho auge entre los años 60 y 80. El objetivo de estas heurísticas es la generación de soluciones al problema de manera “simple” tomando decisiones irreversibles durante el proceso de construcción (insertar un cliente en una ruta, combinar dos rutas en una única ruta, etc.) lo que limita en gran medida su capacidad para obtener una buena solución.

En esta sección se describen las heurísticas constructivas más relevantes aplicadas a problemas de tipo VRP; una recopilación más detallada puede encontrarse en (Laporte y Semet 2002).

Uno de los primeros ejemplos de heurística constructiva es el “método de ahorro” (*savings method/heuristic*) propuesto con Clarke y Wright (Clarke y Wright 1964). Este método comienza por una solución inicial s_0 en la que hay una ruta independiente por cada uno de los clientes. A continuación, la heurística une el primer y último cliente de dos rutas i y j (sin contemplar el almacén central), maximizando el ahorro de distancia $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, siempre que la unión sea posible. La nueva ruta generada tras cada unión reemplaza a las dos rutas combinadas. Este método ha sido revisado y mejorado en varias ocasiones (Gaskell 1967) y (Yellow 1970); en este último caso se propone una nueva función que mejora la calidad de las rutas generadas por medio de la adición de un parámetro que pondera la importancia de la distancia al almacén central ($s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}$, $\lambda \geq 0$).

Otra heurística muy conocida debido a su simplicidad, es la heurística de “barrido” (*sweep*) (Gillett y Miller 1974). Este método crea las rutas de una en una y de forma secuencial analizando los clientes de forma circular en base al ángulo polar respecto al almacén. Los clientes seleccionados de esta forma se insertan al final de la ruta actual. Si la inserción no es posible debido a que se incumple alguna restricción (por ejemplo la capacidad) se crea una nueva ruta y el proceso continúa hasta que todos los clientes están asignados a una ruta. Una vez finalizado el proceso Gillett y Miller utilizan un operador de mejora intraruta (afecta a una única ruta) λ -opt para optimizar cada ruta de forma independiente.

Otras heurísticas de construcción realizan el proceso de construcción de rutas separando las dos fases fundamentales del problema: agrupación (*cluster*) o asignación de clientes a un vehículo, y ordenación (*route*) de clientes que están asignados a un vehículo.

Por un lado estaría el enfoque *route-first cluster-second* (Newton y Thomas 1974), (Bodin y Berman 1979) y (Beasley 1983). En este caso, primero se genera una ruta gigante (*giant circuit*) en la que se incorporan todos los clientes como si se tratase de un problema de tipo TSP; posteriormente la ruta gigante se divide para definir las rutas asignadas a cada uno de los vehículos.

Por otro lado se encuentra el enfoque *cluster-first route-second* (Fisher y Jaikumar 1981) primero se agrupan los clientes en grupos (asociados a un vehículo) y después se ordenan los clientes de cada uno de los grupos como si se tratase de un TSP. Este enfoque está directamente relacionado con el enfoque visual intuitivo que utiliza una persona para resolver un problema de tipo TSP de forma manual.

Van Breedam en su trabajo de tesis doctoral (Van Breedam 1994) realiza una revisión de 11 categorías de técnicas de construcción para problemas de tipo VRP que se presenta a continuación, junto con una recopilación de las implementaciones más representativas para cada una de ellas.

- **Heurísticas de construcción de rutas/inicialización:** las heurísticas pertenecientes a esta categoría construyen las rutas de manera iterativa añadiendo clientes/nodos aún no visitados. Dentro de esta categoría se diferencian dos sub-categorías:
 - Secuenciales, que construyen las rutas una a una.
 - Paralelas, que construyen varias rutas en paralelo.

Heurísticas de construcción de rutas/inicialización	
Técnicas secuenciales	Técnicas paralelas
Vecino más cercano secuencial <ul style="list-style-type: none"> – (Tyagi 1968) – (Baker y Schaffer 1986) – (Solomon 1987) – (Balakrishnan 1993) 	Vecino más cercano paralela
Ahorro secuencial <ul style="list-style-type: none"> – (Clarke y Wright 1964) – (Gaskell 1967) – (Yellow 1970) – (Webb 1972) 	Ahorro paralela <ul style="list-style-type: none"> – (Golden 1977) – (Nelson, y otros 1988) – (Bodin 1983) – (Van Landeghem 1988)
Inserción secuencial <ul style="list-style-type: none"> – (Mole y Jameson 1976) – (Baker y Schaffer 1986) – (Solomon 1987) 	Ahorro generalizada <ul style="list-style-type: none"> – (Altinkemer y Gavish 1991) – (Desrochers y Verhoog 1989)
	Inserción paralela <ul style="list-style-type: none"> – (Potvin y Rousseau 1993)
	Inserción basada en asignación paralela <ul style="list-style-type: none"> – (Savelsbergh 1990)

Tabla 3-1: Heurísticas de construcción de rutas para el VRP.

- **Heurísticas de dos fases:** estas heurísticas construyen la solución en dos fases. Al igual que la categoría anterior, se divide en otras dos:
 - Primero agrupar-segundo ordenar (*cluster-first route-second*).
 - Primer ordenar-segundo agrupar (*route-first cluster-second*).

Otras clasificaciones de heurísticas de construcción pueden encontrarse en (Bodin, y otros 1983) y (Christofides 1985).

Heurísticas de dos fases	
<i>cluster-first route-second</i>	<i>route-first cluster-second</i>
Asignación generalizada <ul style="list-style-type: none"> – (Fisher y Jaikumar 1981) – (Nygard, y otros 1988) 	De dos fases <ul style="list-style-type: none"> – (Christofides, y otros 1979) – (Potvin y Rousseau 1993)
Barrido (<i>sweep</i>) <ul style="list-style-type: none"> – (Gillett y Miller 1974) 	

Tabla 3-2: Heurísticas de dos fases.

Las heurísticas constructivas son capaces de obtener soluciones que distan entre un 10% y un 15% de las soluciones óptimas (Vidal, y otros 2013). Por este motivo, su uso para aplicaciones del mundo real está limitado. No obstante, debido a que necesitan poco tiempo para generar una solución, son una buena alternativa para la generación de soluciones iniciales que posteriormente son “mejoradas” por otras técnicas de optimización de búsqueda local, metaheurísticas o enfoques híbridos. Este hecho hace que, en algunos contextos a estos métodos se les denomine “heurísticas de construcción” ya que representan en paso inicial para el método de resolución.

3.1.1 Heurísticas de construcción para el VRPTW

Las heurísticas constructivas descritas en el apartado anterior, fueron originalmente definidas para la versión clásica del problema de tipo VRP, pero variantes como el VRPTW requieren la redefinición de heurísticas específicas para contemplar las restricciones añadidas.

En la literatura pueden encontrarse multitud de heurísticas constructivas para problemas de tipo VRPTW; en esta sección se destacan únicamente las 3 heurísticas más representativas, que se corresponden con los trabajos desarrollados en (Solomon 1987), (Potvin y Rousseau 1993) e (Ioannou, Kritikos y Prastacos 2001). Además, se añade una heurística más reciente (Figliozzi 2010), puesto que obtiene mejores resultados que las otras 3.

En 1987 Solomon (Solomon 1987), definió 3 heurísticas secuenciales de construcción de rutas específicas para el VRPTW que han sido la referencia para la definición de muchas otras heurísticas de construcción. La primera de ellas es una extensión de la heurística de ahorro de Clarke y Wright (Clarke y Wright 1964). En este caso, para tener en cuenta la cercanía espacial y temporal de los clientes, Solomon define un límite máximo en el tiempo de espera en cada una de las rutas, que guía el proceso de construcción.

La segunda heurística de Solomon, es una variante de la heurística del vecino más cercano secuencial. El proceso comienza cada nueva ruta con el cliente más cercano al almacén central que todavía no ha sido asignado a ninguna ruta. En cada nueva iteración, se busca al cliente más cercano al último cliente añadido a la ruta, y se añade al final de la ruta. Se crea una nueva ruta cuando al insertar un nuevo cliente en la ruta actual, no se cumplen las restricciones de capacidad del vehículo o duración máxima de la ruta. El proceso termina cuando todos los clientes están asignados a una ruta. Para calcular la proximidad entre los clientes se utiliza tanto la distancia como la cercanía temporal de las ventanas de tiempo de los clientes.

La tercera heurística propuesta por Solomon, denominada I₁ (heurística de Inserción secuencial 1), es la heurística que ofrece los mejores resultados de las tres propuestas; y es una de las más utilizadas en la literatura desde su definición original en 1987. Cada ruta se inicializa con un primer cliente o semilla (*seed customer*) y el resto de clientes se van añadiendo de manera secuencial hasta que se completa la capacidad del vehículo, o se supera el horizonte de planificación. Mientras queden clientes sin asignar a una ruta, se inicializa una nueva y se repite el proceso de inserción, hasta que todos los clientes formen parte de alguna ruta. La elección del cliente inicial, que es sin duda una de las cuestiones más relevantes de las heurísticas de construcción secuenciales, se realiza en base a dos posibles criterios: el cliente que esté más alejado del almacén central, o el cliente cuya ventana temporal se inicie antes. Posteriormente, autores como Joubert y Claasen (Joubert y Claasen 2006) proponen alternativas al criterio de selección para la heurística I₁ de Solomon a partir del análisis de la compatibilidad de las ventanas de tiempo de los clientes obteniendo un cálculo que mide lo restrictiva que es una ventana de tiempo respecto a ser colocada antes o después de cualquier otra, así, utilizando este valor, se mejora la calidad de la solución obtenida para alguna de las instancias utilizadas como referencia para evaluar los procesos de construcción.

Una vez inicializada la ruta con un cliente, el método I₁ utiliza dos criterios $c_1(i, u, j)$ y $c_2(i, u, j)$ para seleccionar al cliente u e insertarlo entre los clientes consecutivos i y j , de la ruta parcial que está en proceso de construcción. Estos criterios de selección de clientes para ser asignados/insertados en una ruta han sido una fuente constante de análisis y revisión desde que fueron propuestos por Solomon. Por ese motivo, se analizan en detalle a continuación:

Dada la siguiente ruta parcial $(i_0, i_1, i_2, \dots, i_m)$, donde i_0 e i_m representan al almacén central. Para cada cliente u no asignado a ninguna ruta, primero se determina el coste de la mejor posición de inserción:

$$c_1(i(u), u, j(u)) = \min_{\rho=1, \dots, m} c_1(i_{\rho-1}, u, i_{\rho})$$

A continuación, el mejor cliente u^* , será aquel que cumpla:

$$c_2(i(u^*), u^*, j(u^*)) = \max_u \{c_2(i(u), u, j(u)) \mid u \text{ no asignado y ruta válida}\}$$

Entonces, el cliente u^* se inserta entre $i(u^*)$ y $j(u^*)$. Los cálculos y el proceso de inserción, se repiten de manera iterativa hasta que no se pueden insertar más clientes en la ruta actual. En ese momento, se finaliza la ruta actual, y se inicializa una nueva ruta, repitiendo el proceso hasta que todos los clientes están asociados a una ruta.

El criterio de inserción, $c_1(i, u, j)$ se calcula de la siguiente forma:

$$c_1(i, u, j) = \alpha_1 \cdot c_{11}(i, u, j) + \alpha_2 \cdot c_{12}(i, u, j),$$

$$\alpha_1 + \alpha_2 = 1, \alpha_1 \geq 0, \alpha_2 \geq 0,$$

siendo:

$$c_{11}(i, u, j) = d_{iu} + d_{uj} - \mu \cdot d_{ij}, \quad \mu \geq 0,$$

$$c_{12}(i, u, j) = b_{ju} - b_j$$

Los valores d_{iu} , d_{uj} y d_{ij} son las distancias entre los clientes $i - u$, $u - j$, e $i - j$ respectivamente. El parámetro μ pondera el ahorro de distancia al insertar el cliente u entre los clientes i y j . El valor b_{ju} representa el nuevo tiempo de inicio de servicio para el cliente j después de insertar al cliente u en la ruta, y b_j representa el tiempo de inicio del servicio para el cliente j antes de insertar al cliente u en la ruta.

Los parámetros α_1 y α_2 sirven para ponderar el incremento de distancia y la duración de la ruta respectivamente tras insertar al cliente u en la ruta actual. Si $\alpha_1 = 1$ y $\alpha_2 = 0$ sólo se tiene en cuenta el incremento en la distancia total de la ruta, mientras que si los parámetros toman el valor opuesto, únicamente se utiliza la duración de la ruta como criterio de elección de la mejor inserción. Para un determinado problema, lo que se suele hacer es realizar varias invocaciones de la heurística modificando los valores de α_1 y α_2 asignándoles varios valores (por ejemplo, 0.25, 0.5 y 0.75, que fueron los originalmente propuestos por Solomon en su trabajo); posteriormente, se escoge el mejor resultado.

Por último, el criterio de inserción $c_2(i, u, j)$, se calcula de la siguiente forma:

$$c_2(i, u, j) = \lambda \cdot d_{0u} - c_1(i, u, j), \quad \lambda \geq 0$$

Este segundo criterio de selección de la mejor posición de inserción para el cliente u tiene en cuenta la proximidad entre dicho cliente y el almacén central (ponderada

por el parámetro λ), y el incremento de distancia y tiempo que supone la inserción del cliente u en la ruta actual.

La heurística I_1 , es la que mejores resultados ofrece de las 3 propuestas por Solomon y ha sido la base para la creación de la solución inicial en muchos métodos de búsqueda local (incluso los más recientes), que parten de una solución inicial que se modifica de manera iterativa hasta llegar a un óptimo local.

Potvin y Rousseau en (Potvin y Rousseau 1993) proponen una versión paralela de la heurística de inserción secuencial I_1 de Solomon denominada PARIS (*parallel insertion*) en la que se construyen varias rutas de manera simultanea. La heurística propuesta se descompone en dos fases:

1. En primer lugar se determina el número de rutas que se inicializarán en paralelo. Para ello se aplica la heurística I_1 de Solomon (usando un único juego de parámetros para el proceso) y se obtiene el valor n_r que representa el número de rutas a inicializar en paralelo. Además se selecciona el cliente inicial para cada una de las rutas a partir del cliente más alejado del almacén central de cada una de las rutas obtenidas por el procedimiento secuencial de Solomon.
2. Una vez inicializadas las n_r rutas e insertados los clientes iniciales, para el resto de clientes se calcula la mejor posición de inserción en cada una de las n_r rutas utilizando un criterio de inserción parecido al propuesto por Solomon.

La mejor inserción para cada cliente u no asignado a una ruta, se calcula como:

$$c_{1r}^*(i_r(u), u, j_r(u)) = \min_{p=1, \dots, m} [c_{1r}(i_{r_{p-1}}, u, i_{r_p})], \quad r = 1, \dots, n_r;$$

$$c_{1r}(i_r, u, j_r) = \alpha_1 \cdot c_{11r}(i_r, u, j_r) + \alpha_2 \cdot c_{12r}(i_r, u, j_r),$$

$$\alpha_1 + \alpha_2 = 1, \quad \alpha_1 \leq 0, \quad \alpha_2 \leq 0$$

siendo,

$$c_{11r}(i_r, u, j_r) = d_{i_r, u} + d_{j_r, u} - d_{i_r, j_r},$$

$$d_{k,l} = \text{distancia (en unidades de tiempo) entre los clientes } k \text{ y } l$$

$$c_{12r}(i_r, u, j_r) = b_{u, j_r} + b_{j_r},$$

$$b_k = \text{instante de inicio actual para el cliente } k$$

$$b_{k,l} = \text{nuevo inicio para el cliente } l \text{ después de insertar } k \text{ en la ruta}$$

A continuación, se determina el siguiente cliente u^* a insertar de acuerdo a:

$$c_2(u^*) = \max_u [c_2(u)],$$

$$c_2(u) = \sum_{r \neq r'} [c_{1r}^*(i_r(u), u, j_r(u)) - c_{1r'}^*(i_{r'}(u), u, j_{r'}(u))]$$

$$c_{1r'}^*(i_{r'}(u), u, j_{r'}(u)) = \min_{r=1, \dots, n_r} [c_{1r}^*(i_r(u), u, j_r(u))]$$

y se inserta el cliente u^* entre $i_{r'}(u^*)$ y $j_{r'}(u^*)$. Este proceso se repite hasta que todos los clientes están asignados a una ruta o hasta que queda algún cliente que no puede ser asignado a ninguna ruta (con lo cual no se puede encontrar una solución al problema).

La fórmula que define el coste de inserción es similar a la de Solomon, pero el criterio para elegir el siguiente cliente a asignar a una ruta $c_2(u)$ se basa en una medida de regresión sobre todas las rutas. Este valor adelanta qué se perderá posteriormente si un cliente no se inserta inmediatamente en su mejor inserción. En este contexto, un valor grande para la medida de regresión indica que existe una gran diferencia entre la mejor inserción global para un determinado cliente $c_{1r'}^*(i_{r'}(u), u, j_{r'}(u))$ y su mejor inserción en el resto de rutas. Teniendo esto presente, primero deben considerarse los clientes con un mayor valor de medida de regresión, ya que el número de buenas alternativas de inserción es menor que los clientes con un valor de medida de regresión más pequeño; lo que implica que el cliente puede ser insertado en diferentes rutas con una pérdida de coste menor.

Cuando un cliente u no puede insertarse en la ruta r , $c_{1r}^*(i_r(u), u, j_r(u))$ se establece con un valor grande y constante para contemplar también las dificultades que existen para asignar un cliente a una ruta. De esta forma, los clientes con menor número de alternativas de inserción tendrán un valor de medida de regresión mayor que el resto de clientes y por lo tanto serán elegidos antes para ser asignados a una ruta. Por otro lado, si un cliente no puede asignarse a ninguna ruta, el valor de su medida de regresión es 0; en ese caso, queda sin asignarse a ninguna ruta y se crea una nueva ruta para él.

El proceso de inserción paralela se realiza con 3 juegos de parámetros:

$$(\alpha_1, \alpha_2) = (0.5, 0.5), (0.75, 0.25), (1.0, 0.0)$$

y se elige la mejor solución obtenida para todos ellos. Además, si para un juego de parámetros el algoritmo encuentra una solución mejor que la original obtenida mediante la heurística de Solomon, el proceso se inicia de nuevo para intentar encontrar una solución con un número de rutas menor: se reduce el número de rutas inicial $n_r = n_r - 1$ y se elimina uno de los clientes iniciales s^* ; el que esté más cerca de los demás clientes:

$$s^* = \arg \left[\min_{s \in S_{seed}} \left\{ \min_{s' \in S_{seed}, s' \neq s} d_{s,s'} \right\} \right], S_{seed} = \text{clientes que inicializan las rutas}$$

El número de rutas se reduce hasta encontrar un valor $n_{r_{inf}}$ para el que no puede encontrarse ninguna solución. En ese momento se retoma el proceso actualizando $n_r = n_{r_{inf}} + 1$ y se continua con el resto de juegos de parámetros (si quedaba alguno). En algunas ocasiones (sobre todo en problemas en los que los clientes están geográficamente agrupados) la heurística necesita más rutas que las que inicialmente necesitaba la heurística secuencial de Solomon. En esos casos para cada nueva ruta, el cliente inicial se escoge como aquél que está más alejado de los clientes iniciales del resto de rutas:

$$u^* = \arg \left[\max_u \left\{ \min_{s \in S_{seed}} d_{s,u} \right\} \right], S_{seed} = \text{clientes que inicializan las rutas}$$

Los autores reconocen que el enfoque paralelo es adecuado cuando los clientes están distriuídos geográficamente de forma aleatoria mientras que ofrecen un peor resultado para situaciones en los que los clientes están agrupados.

La heurística secuencial IMPACT propuesta por Ioannou y otros en (Ioannou, Kritikos y Prastacos 2001) se basa en una combinación de las heurísticas de Solomon (Solomon 1987), y el criterio de inserción de Atkinson (Atkinson 1994) que analiza las implicaciones de la inserción de un cliente en una ruta parcialmente creada en base a 3 criterios:

- $IS_u = \text{impacto propio}$. Relación entre el instante de inicio del servicio a_u y el inicio más temprano e_u para el cliente u .

$$IS_u = a_u - e_u$$

Si este valor es próximo a 0 implica que después de insertar el nuevo cliente en la ruta existe margen para la inserción de clientes antes (ya que está disponible el intervalo de la ventana de tiempo para atrasar el inicio del servicio) y después (ya que no afecta al instante de inicio del servicio) del cliente u .

- $IU_u = \text{impacto externo}$. Implicaciones para el resto de clientes que todavía no se han asignado a ninguna ruta.

Después de asignar un cliente a una ruta, la selección de uno nuevo para ser asignado a la misma ruta se complica debido al posible solapamiento entre las ventanas de tiempo de los clientes asignados a la ruta y los que no lo están.

Para poder asignar el cliente j a la ruta en construcción después de haber seleccionado y asignado al cliente u a la ruta, debe cumplirse:

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

Estas desigualdades garantizan que el cliente j podrá asignarse a la ruta en construcción antes o después del cliente u . Por lo tanto, la asignación del cliente u que minimice la diferencia no negativa (una de las dos puede ser negativa) de $[l_j - (e_u + d_{uj})]$ y $[l_u - (e_j + d_{uj})]$ para todos los clientes aún no asignados j , será la que deje más alternativas para el resto de clientes no asignados.

El impacto externo calcula la compatibilidad de las ventanas de tiempo de los clientes no asignados y el cliente seleccionado para ser asignado.

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

J = conjunto de clientes no asignados.

- IR_u = *impacto interno*. Implicaciones para los clientes que ya están asignados a la ruta que está en proceso de construcción. El cálculo del impacto interno se realiza en base a 3 métricas: perturbación local, perturbación global y accesibilidad. La revisión de cada una de ellas se realiza a partir del análisis de la inserción del cliente u entre los clientes i y j , que ya forman parte de la ruta en construcción.

La inserción de un nuevo cliente en la ruta genera un incremento en la distancia total de la ruta:

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

Además, se incrementa el tiempo de llegada del vehículo al cliente j , que en ningún caso puede superar su límite temporal superior l_j . Este incremento se recoge en el criterio $c_{2u}(i, j)$, que se denomina compatibilidad marginal de tiempo:

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

Por último, la inserción del cliente u en la ruta define un rango temporal entre el instante de llegada a_u y el inicio más tardío del servicio l_u . Este intervalo mide la compatibilidad de la ventana de tiempo del cliente u con la inserción en una determinada posición en la ruta en construcción:

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

Con este último criterio, el cliente no asignado que tenga un menor valor, será el idóneo para ser insertado en la ruta en construcción.

La mejor posición de inserción se calcula a partir de una combinación ponderada de los 3 criterios descritos denominada “Perturbación Local” (*Local Disturbance*):

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

$$b_1 + b_2 + b_3 = 1, \quad b_1 \geq 0, \quad b_2 \geq 0, \quad b_3 \geq 0$$

El “Impacto Interno” IR_u se calcula como el sumatorio de la “Perturbación Local” LD_u para todos los posibles puntos de inserción del cliente u en la ruta en construcción:

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

$|I_r|$ = cardinalidad del conjunto de posibles inserciones del cliente u en la ruta r . Dicho conjunto está formado por pares ordenados (i, j) de clientes que ya están asignados a la ruta.

Con todo lo anterior, el criterio de selección o impacto del siguiente cliente u a asignar a la ruta r se calcula como:

$$Impact(u) = b_s \cdot IS_u + b_e \cdot IU_u + b_r \cdot IR_u,$$

$$b_s + b_e + b_r = 1, \quad b_s \geq 0, \quad b_e \geq 0, \quad b_r \geq 0$$

Los parámetros b_s , b_e y b_r definen la contribución del impacto propio, externo e interno, y sus valores se calculan a partir de un proceso estadístico de experimentación. En este sentido, los autores sugieren hacer pruebas de ajuste con incrementos de 0.1 para cada uno de los parámetros. Esta heurística mejoraba a las mejores alternativas existentes hasta la fecha de publicación de la misma (2001), incluso permitían ajustar de una manera bastante buena el número de rutas para los dos juegos de ensayo más representativos de problemas de tipo VRPTW: el de Solomon, y el de Gehring y Homberger (SINTEF 2015) (ver sección 5.1.3).

Para finalizar esta revisión de heurísticas de construcción para problemas de tipo VRPTW, se analiza la heurística secuencial constructiva IRCI propuesta por Figliozzi (Figliozzi 2010). Esta heurística utiliza dos algoritmos para la construcción de las rutas: el algoritmo auxiliar de construcción de rutas (H_r) y el algoritmo de construcción de rutas (H_c).

El algoritmo auxiliar de construcción de rutas (H_r) puede ser cualquier heurística que partiendo de un punto inicial, un conjunto de clientes y el almacén central, devuelva un conjunto de rutas que satisfacen todas las restricciones del problema. A modo de ejemplo, el autor propone el uso de una heurística del vecino más cercano generalizada (*Generalized Nearest Neighbor Heuristic - GNNH*) que utiliza una función de “coste generalizada” que contempla la ubicación geográfica, la proximidad temporal de los clientes, la capacidad restante del vehículo y el coste de añadir un nuevo vehículo si el cliente no puede ser asignado a la ruta en construcción.

Suponiendo que i representa un cliente que está asignado a la ruta y j un cliente a asignar en la ruta: d_{ij} es la distancia entre los clientes i y j ; a_j y a_i los instantes de inicio del servicio en los clientes i y j respectivamente; s_i el tiempo de servicio del cliente i ; l_j el límite superior de la ventana de tiempo del cliente j ; t_{ij} el tiempo de trayecto (medido en distancia) entre los clientes i y j ; q_i la capacidad restante del vehículo antes de asignar el cliente j ; y d_j la demanda del cliente j . El coste generalizado de insertar al cliente j después del cliente i se calcula como:

$$g(\Delta, i, j) = \delta_1 \cdot d_{ij} + \delta_2 \cdot (a_j - (a_i + s_i)) + \delta_3 \cdot (l_j - (a_i + s_i + t_{ij})) + \delta_4 \cdot (q_i - d_j)$$

- δ_1 , pondera la distancia entre los clientes i y j .
- δ_2 , contempla el intervalo que existe entre el fin del servicio del cliente i y el inicio del servicio del cliente j . Es decir, $a_j = \max\{(a_i + s_i + t_{ij}), e_j\}$
- δ_3 , pondera la “urgencia” de atender al cliente j a partir del intervalo que queda entre el instante de llegada del vehículo y el límite superior de la ventana de tiempo.
- δ_4 , tiene en cuenta la capacidad que tendrá el vehículo después de servir al cliente j .

Si un cliente j no puede ser insertado en la ruta, el coste de terminar la ruta que estaba en construcción e iniciar una nueva para atender al cliente j se calcula como:

$$g(\Delta, i, j) = \delta_0 + \delta_1 \cdot d_{0j} + \delta_2 \cdot a_j + \delta_3 \cdot (l_j - t_{0j}) + \delta_4 \cdot (q_{MAX} - d_j)$$

- δ_0 , representa el coste de añadir un nuevo vehículo.

Teniendo en cuenta lo anterior, el algoritmo auxiliar de construcción de rutas se define como $H_r(\Delta, v_i, C, v_o)$ donde $\Delta = \{\delta_0, \delta_1, \delta_2, \delta_3, \delta_4\}$ son los parámetros de la función, v_i es el punto en que se inicia la primera ruta, C es el conjunto de clientes no asignados a ninguna ruta y v_o es el almacén central donde empiezan y terminan todas las rutas, a excepción de la primera que empieza en v_i . Además, $\delta_1 + \delta_2 + \delta_3 = 1$ y $\delta_i > 0$, $i \in \{0, \dots, 4\}$.

El algoritmo de construcción de rutas (H_c) construye las rutas de manera secuencial. Dada una solución parcial y un conjunto de clientes sin asignar a ninguna ruta, este algoritmo utiliza el algoritmo auxiliar (H_r) para buscar el conjunto de rutas de menor coste que satisfagan todas las restricciones del problema. El algoritmo de construcción de rutas también utiliza una función de coste $w(v_i, C, g, W)$ tal que, dado un conjunto de clientes no asignados C , la ubicación de un cliente $v_i \notin C$ y la función de coste generalizada $g(\Delta, v_i, v_j)$, devuelve el conjunto de clientes con menor coste generalizado $g(\Delta, v_i, v_j)$ para todo $v_j \in C$. Empezando por una ruta con el almacén central, en cada

iteración el algoritmo H_c utiliza el algoritmo auxiliar H_r para seleccionar el siguiente cliente a añadir al final de la ruta (que será aquél que tenga el menor coste generalizado) hasta que no puede añadirse ningún cliente más. En ese momento, se inicia una nueva ruta y el proceso continua hasta que todos los clientes hayan sido asignados. Para su ejecución, el algoritmo va probando diferentes combinaciones del conjunto de parámetros Δ y se queda con la mejor solución de todos ellos. Según los resultados reportados por el Figliozzi, el algoritmo de construcción de rutas es muy eficiente en el cálculo del número de rutas (comparado con algoritmos meta-heurísticos) y por otro lado, domina a las heurísticas de construcción de rutas clásicas como las de Solomon, Potvin y Rousseau e Iannou y otros.

En la **tabla 3-3** se muestra una comparativa de los métodos de construcción de rutas descritos en esta sección, realizada en 2005 por Bräysy y Gendreau, utilizando como referencia el juego de ensayo de Solomon (SINTEF 2015). La primera columna de la tabla identifica el algoritmo, y el resto de columnas muestra los valores medios de número de vehículos y distancia para cada una de las clases de problemas del juego de ensayo de Solomon.

Autor	R1	R2	C1	C2	RC1	RC2
(1) I1 (Solomon 1987)	13,58 1437	3,37 1402	10,00 952	3,13 693	13,50 1597	3,88 1682
(2) PARIS (Potvin y Rousseau 1993)	13,33 1509	3,09 1387	10,67 1344	3,38 798	13,38 1724	3,63 1651
(3) IMPACT (Ioannou, Kritikos y Prastacos 2001)	12,67 1370	3,09 1310	10,00 865	3,13 662	12,50 1512	3,50 1483
(4) IRCI (Figliozzi 2010)	12,50 1262	3,09 1171	10,00 872	3,00 656	12,00 1420	3,38 1342
Tiempos medios de ejecución: (1) 0,6 min.; (2) 19,6 min.; (3) 4,0 min.; (4) 10,9 min.						

Fuente: (Bräysy y Gendreau 2005a) y (Figliozzi 2010)

Tabla 3-3: Comparativa de heurísticas de construcción para el VRPTW.

Analizando los resultados de la **tabla 3-3** se puede apreciar que la técnica de Figliozzi domina a las otras 3 en cuanto a calidad de la solución (número de rutas y luego distancia) debido a que obtiene los mejores resultados (salvo en el caso de la clase de problemas C1). Por otro lado, en relación a los tiempos de ejecución, la heurística de Solomon es claramente la más rápida de las cuatro analizadas, por ese motivo ha sido utilizadas como referencia en la generación de soluciones iniciales para multitud de técnicas heurísticas de búsqueda local y metaheurísticas.

Tal y como se argumentará más adelante en la sección 5.1.1 Criterios generales de comparación de técnicas de resolución, esta comparativa no es del todo exhaustiva y

justa ya que únicamente contempla los mejores valores de la función objetivo obtenidos por cada una de las técnicas y por otro lado no se tienen en cuenta criterios como la flexibilidad, robustez o sencillez de implementación de cada una de las técnicas. No obstante, se mantiene a modo de ilustración del tipo de prácticas no adecuadas a la hora de presentar resultados que se pueden encontrar en la bibliografía de autores que pueden ser considerados como referentes en el ámbito de la resolución de problemas de tipo VRP y sus variantes, como son Bräysy y Gendreau.

La mayor parte de las heurísticas de construcción para el problema de tipo VRPTW son el paso previo al uso de una técnica heurística o metaheurística cuyo objetivo fundamental es la optimización de la solución inicialmente obtenida mediante la heurística de construcción. No obstante, autores como Russell en (Russell 1995) proponen el diseño de una heurística de construcción híbrida, que combina el proceso de optimización de una búsqueda local durante la fase de construcción de las rutas, lo que permite mejorar los resultados obtenidos por el algoritmo de construcción utilizado de forma independiente.

En concreto, la heurística híbrida propuesta es una heurística de construcción paralela basada en la heurística de Potvin y Rousseau (Potvin y Rousseau 1993) que integra un proceso de optimización basado en un operador interruta variante del operador k-opt propuesto por Lin y Kernighan para el problema de tipo TSP (Lin y Kernighan 1973). La optimización se realiza sobre el conjunto de rutas parcialmente construidas cada vez que un determinado número de clientes ha sido asignado a una ruta.

3.2 Técnicas de búsqueda local

La búsqueda local (*LS - Local Search*) es quizá la técnica metaheurística básica más utilizada en la resolución de problemas de optimización combinatoria, tanto de forma independiente, como combinada con heurísticas y con otras metaheurísticas.

Esta técnica de resolución se basa en el concepto de la mejora iterativa de la solución de un problema a partir de la exploración de las "soluciones vecinas" o "vecindario" (*neighborhood*) de la solución actual, que está compuesto por las soluciones que se pueden generar a partir de una modificación en la solución actual. Como técnica de resolución aproximada, la búsqueda local no tiene garantías de encontrar la solución óptima a un problema (conocida como óptimo global), por ese motivo, una vez finalizado el proceso de búsqueda se dice que la búsqueda local devuelve un óptimo local.

La noción básica del funcionamiento de un algoritmo de búsqueda local, se puede ver haciendo una analogía entre el espacio de soluciones (todas las posibles soluciones) de

un problema y la orografía de un paisaje montañoso (Russell y Norving 2013). En un momento determinado, partiendo de una solución existente, lo que se pretende es avanzar por la orografía montañosa hacia la cumbre más alta o un valle más profundo, dependiendo de cómo esté definida la función objetivo: función a maximizar o minimizar. La **figura 3-1** muestra esta analogía en un plano bidimensional en el que la línea representa el espacio de soluciones del problema y la cota representa el valor de la función objetivo para una determinada solución.

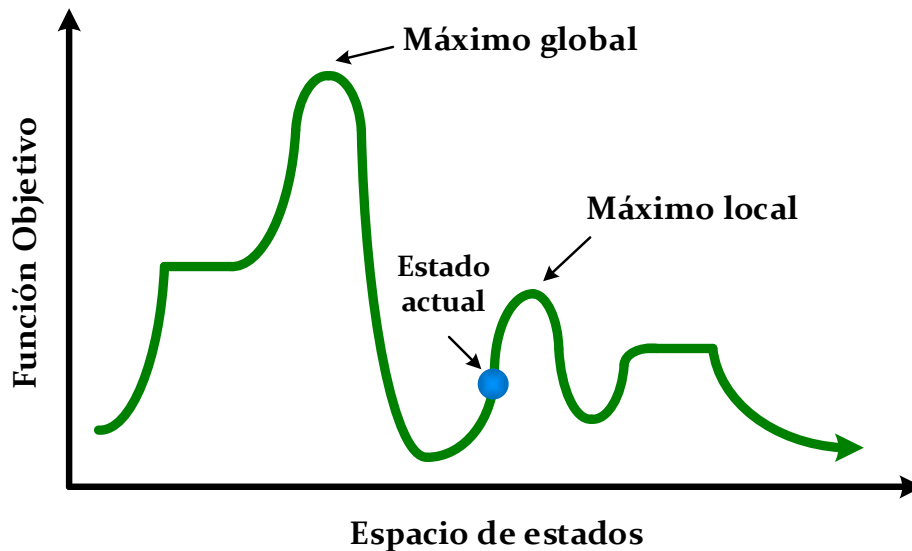


Figura 3-1: Espacio de soluciones de una búsqueda local.

En un momento dado, partiendo de un punto (que representa a la solución actual) y de forma iterativa, se van analizando los puntos limítrofes a la solución actual (que representan a los vecinos de la solución actual) y se avanza hacia otro punto en movimientos ascendentes o descendentes (dependiendo de si se trata de un problema de maximización o minimización de la función objetivo) hasta que se llega una cota en la que no se puede seguir avanzado porque todos los vecinos de la solución actual tienen un valor de la función objetivo peor. En ese instante, el proceso terminaría y se devuelve como solución la solución actual. La realidad de una búsqueda local, sería ese mismo espacio, pero en un plano multidimensional, donde en cada momento se pueden realizar múltiples movimientos, que permiten avanzar hacia cualquiera de las soluciones vecinas de la solución actual. Como se puede observar de manera intuitiva, dependiendo de cuál sea la solución inicial, la naturaleza de la orografía y la función objetivo del problema, existirán más o menos garantías de encontrar la solución óptima. Por lo general, la búsqueda local suele quedarse atrapada en un óptimo local lo que hace que su aplicación directa a determinados problemas se vea limitada. Este hecho, ha motivado el desarrollo de técnicas metaheurísticas alternativas más complejas. Estas últimas, utilizando la misma noción del avance a través de un espacio

montañoso definen estrategias para “escapar” de los óptimos locales y aumentar las posibilidades de encontrar un óptimo global.

De una manera más formal, la búsqueda local podría definirse de la siguiente manera (Aarts y Lenstra 2003):

Una instancia (X, c) de un problema de optimización combinatoria se define como $\min_{x \in X} c(x)$, donde X representa al conjunto de posibles soluciones al problema y c es la función de coste o función objetivo.

El núcleo de una técnica de búsqueda local es la definición de un vecindario \mathcal{N} , que es un mapeo de la función $\mathcal{N}: X \rightarrow 2^X$. Cada elemento $x' \in \mathcal{N}(x)$ se denomina vecino de x . Los vecinos x' con coste $c(x') < c(x)$ son “vecinos con mejora” (improving neighbors).

La búsqueda local comienza con una solución inicial $x^0 \in X$. En cada iteración t , se reemplaza la solución actual x^t hacia un vecino con mejora $x^{t+1} \in \mathcal{N}(x^t)$. El proceso termina con un óptimo local, es decir, con una solución x^t que no tiene ningún vecino con mejora en su vecindario $\mathcal{N}(x^t)$.

A continuación, se muestra el pseudocódigo de una búsqueda local genérica:

Algoritmo 3-1: Pseudocódigo de una búsqueda local genérica

```

1. Entrada: Una solución inicial  $x^0 \in X$ .
2.  $t = 0$ 
3. REPEAT
4.     BUSCAR un “vecino con mejora”  $x'$  en el vecindario  $\mathcal{N}(x^t)$  de  $x^t$ .
5.     IF existe un vecino con mejora  $x' \in \mathcal{N}(x^t)$  THEN
6.          $t = t + 1$  y  $x^t = x'$ 
7.     UNTIL no se encuentren más mejoras.
8. Salida: Un óptimo local  $x^t$ .

```

El esquema básico de un algoritmo de búsqueda local está sujeto a multitud de variantes que responden a las siguientes cuestiones:

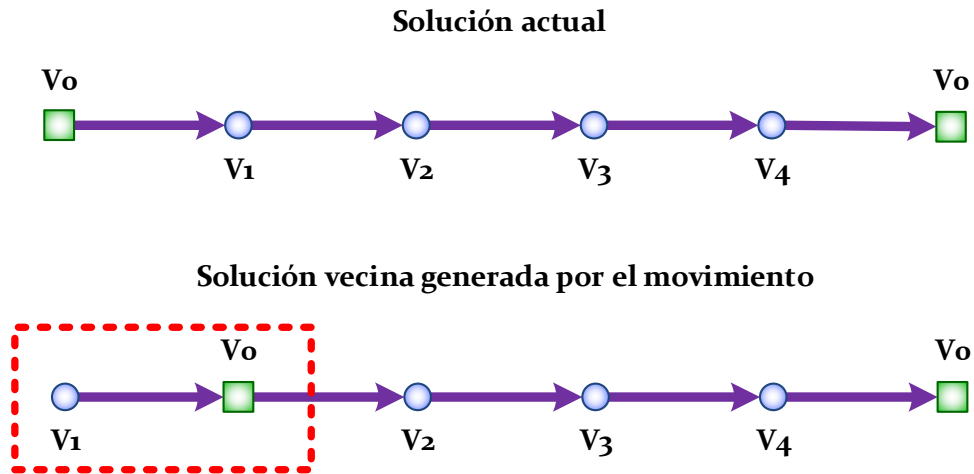
- ¿Cómo se genera la solución inicial? (paso 1)
- ¿Cuál es el mecanismo de generación de vecinos? (paso 4)
- ¿Cuál es la estrategia de selección de “vecinos con mejora”? (paso 4)
- ¿Cuál es el criterio de parada del proceso? (paso 8)

Las alternativas de respuesta para cada una de las cuestiones son variadas, y la elección de un método/técnica concreta para cada una de ellas afecta directamente tanto a la eficacia (entendida en este caso como la capacidad que ofrece el algoritmo para

encontrar una solución de calidad) como a la eficiencia (medida de acuerdo a la complejidad temporal y espacial del proceso) del algoritmo.

Por ejemplo, si el proceso de búsqueda es enumerativo (los vecinos $x' \in \mathcal{N}(x^t)$ y sus costes $c(x')$, se generan y evalúan uno a uno) las estrategias de selección de vecinos con mejora más representativas son las conocidas como: “el primer vecino con mejora” (*first-improvement*) o “el mejor vecino con mejora” (*best-improvement*). Además, existe una variante conocida como “el mejor entre d vecinos con mejora” (*d-best improvement*) que finaliza el proceso de selección cuando encuentra d vecinos con mejora, devolviendo el mejor de todos ellos.

Otro punto clave es la generación de vecinos. Normalmente, los vecindarios y las soluciones vecinas no se construyen a partir de la función $\mathcal{N}: X \rightarrow 2^X$ ni como un subconjunto de todas las posibles soluciones $\mathcal{N}(x) \subset X$; los vecindarios se definen de manera implícita mediante un conjunto de movimientos M . Un movimiento $m \in M$ aplicado a una solución válida del problema x , la transforma en un elemento $m(x)$ cuya estructura es similar a una solución válida del problema, pero puede incumplir alguna restricción del problema, lo que generaría una solución no válida (*infeasible solution*).



Se muestra una ruta de un problema de tipo VRP al que se le aplica un movimiento de generación de soluciones vecinas que consiste en el intercambio de dos nodos cualesquiera de una ruta. Si los nodos intercambiados fuesen los que ocupan la posición v_0 (el almacén central) y v_1 (el primer cliente de la ruta), se generaría una ruta no válida puesto que todas las rutas deben comenzar siempre en el almacén central. En este caso el movimiento genera una solución no válida.

Figura 3-2: Generación de una solución no válida.

Continuando con el paso 4 del **algoritmo 3-1** las tareas fundamentales que debe realizar un algoritmo de búsqueda local en ese punto son:

- Generación de las soluciones vecinas de la solución actual.

- Calcular la ganancia de coste para cada uno de los vecinos.
- Validar si cada nuevo vecino es una solución válida o no.

La manera y orden en que se realizan cada una de estas tareas así como las comprobaciones específicas afectan directamente a la eficiencia de los algoritmos de búsqueda local. En (Irnich 2008a) se puede encontrar una propuesta metodológica para el modelado de metaheurísticas basadas en búsqueda local aplicadas a diferentes variantes de problemas de tipo VRP. En dicho trabajo, se analizan los factores que afectan a la eficiencia de las búsquedas locales y se compara el enfoque tradicional de búsqueda local aplicada a problemas de tipo VRP, también denominado búsqueda lexicográfica con la búsqueda secuencial (Irnich, Funke y Grünert 2006).

3.2.1 Vecindarios de intercambio de nodos y arcos

Tal y como se ha comentado en el apartado anterior, la búsqueda local se basa en la modificación iterativa de la solución actual mediante el uso de movimientos que generan soluciones vecinas. En esta sección se describen los movimientos/operadores más utilizados para la resolución de problemas de tipo VRP. Al conjunto de vecinos obtenidos a partir de la ejecución de un determinado operador sobre una solución se le conoce como vecindario; de ahí el nombre de ésta sección, y el hecho de que en algunos ámbitos, la búsqueda local también se conozca como búsqueda de vecindarios.

Cada uno de los operadores/movimientos aquí presentados hace referencia a un tipo de operador puesto que un mismo operador se puede aplicar n veces sobre una solución actual x generando el vecindario $\mathcal{N}(x)$ compuesto por el conjunto de estados vecinos $x' \in \mathcal{N}(x)$, que podrán ser soluciones válidas o no válidas (tal y como se ilustra en la **figura 3-2**).

En el contexto de los problemas de tipo VRP, los operadores más utilizados implican el intercambio de nodos o secuencias de nodos (lo que implica una eliminación y adición de arcos - *edge-exchange*). Dichos operadores provienen directamente del ámbito del TSP y la terminología de Lin (Lin 1965) donde un vecindario λ -opt contiene al conjunto de soluciones (válidas y no válidas) que se pueden generar a partir de la eliminación y posterior inserción de λ arcos en una ruta. El tamaño de este tipo de vecindarios es $|\mathcal{N}^{\lambda-opt}| = O(n^\lambda)$. Por último, al adaptar estos operadores al VRP, las modificaciones en los arcos pueden afectar a una ruta individual, o a un conjunto de rutas. Cada una de éstas familias de operadores se conocen con los nombres de operadores intraruta (afectan a una única ruta) e interruta (afecta a más de una ruta; normalmente dos) respectivamente.

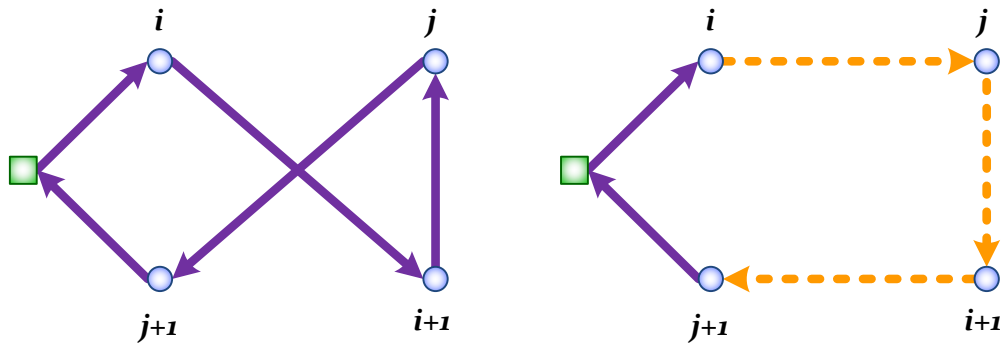
El proceso de aplicación/ejecución de un operador de intercambio de arcos implica los siguientes pasos:

- Eliminación de una serie de arcos (al menos 2) de la/s solución/es actual/es.
- Reordenación de los fragmentos resultantes después de eliminar los arcos. Opcionalmente, algunos operadores contemplan la modificación del orden de los nodos en alguno de los fragmentos obtenidos tras eliminar los arcos.
- Generación de los arcos necesarios para unir los fragmentos y crear una nueva solución.

Una vez ejecutado el operador (aunque también es posible hacerlo durante el proceso de ejecución del mismo) el algoritmo de búsqueda local debe validar que la nueva x' solución sea válida y su coste $c(x') < c(x)$ sea mejor que el de la solución actual x , de lo contrario la nueva solución obtenida será descartada. Cuanto antes se detecte que una solución no es válida o que su coste es peor que el de la solución actual, el algoritmo será más eficiente.

3.2.1.1 Operador intraruta 2-opt

El operador 2-opt, genera una nueva ruta eliminando 2 arcos de la ruta original y sustituyéndolos por otros, invirtiendo el orden de los nodos del fragmento intermedio resultante después de eliminar los arcos. El proceso de eliminación de las dos aristas genera tres fragmento que deben ser unidos de nuevo por medio de otros dos arcos para configurar de nuevo una ruta completa.



En primer lugar se eliminan los arcos $(i, i+1)$ y $(j, j+1)$. Posteriormente se crean los arcos (i, j) e $(i+1, j+1)$ para unir de nuevo los tres fragmentos generados después de eliminar los arcos iniciales. Además, se invierte el sentido del fragmento intermedio; el que incluía a los nodos $i+1$ y j .

Figura 3-3: Operador 2-opt⁶.

⁶ En esta figura y las siguientes, en cada ruta se representa el almacén central mediante un cuadrado y los clientes/nodos mediante círculos. En los operadores de interruta, el almacén central se duplica para clarificar la representación, pero realmente es el mismo. Además, la parte izquierda de la figura muestra la/s ruta/s en su estado original; y la parte derecha la/s ruta/s después de aplicar una instancia del operador

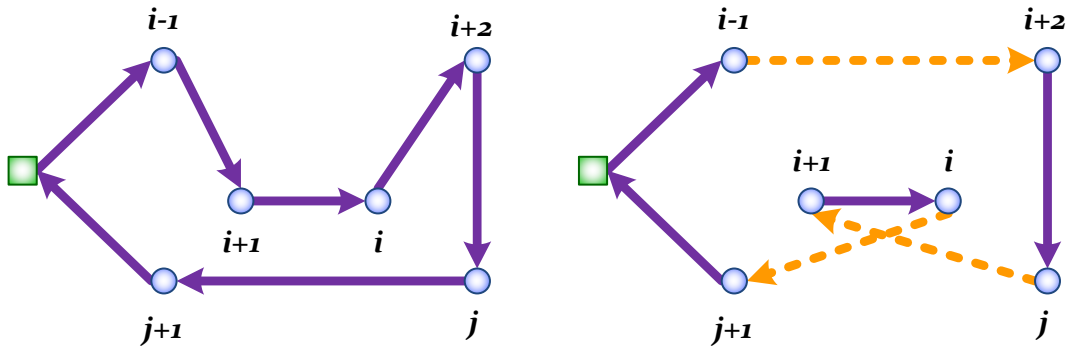
Este operador suele utilizarse de manera iterativa hasta que la ruta no puede ser mejorada. En ese caso, se dice que la ruta es “2-óptima”. Es decir, no existe ninguna combinación de eliminación e inserción de dos aristas que reduzca el coste actual de la ruta. El operador 2-opt fue originariamente definido por (Croes 1958), pero su generalización al concepto λ -opt, así como el operador 3-opt se atribuyen a Lin (Lin 1965).

Las adaptaciones de estos operadores (2-opt y 3-opt) para el VRP han sido tradicionalmente muy utilizadas debido a su sencillez de implementación y el tamaño del vecindario que generan, $|\mathcal{N}^{2-opt}| = O(n^2)$ y $|\mathcal{N}^{3-opt}| = O(n^3)$ respectivamente, lo hace que en la práctica sean operadores idóneos para optimizar las rutas ya generadas.

3.2.1.2 Operador intraruta Or-opt

El operador Or-opt, diseñado por Or (Or 1976) también surge originalmente para la resolución del TSP. Este operador genera un subconjunto del vecindario del operador 3-opt lo que reduce su tamaño a $|\mathcal{N}^{3-opt}| = O(n^2)$.

Si bien el operador 3-opt implica la eliminación y posterior generación de 3 arcos cualesquiera, el operador Or-opt extrae un fragmento formado por 3 o menos nodos consecutivos de la ruta original y reinserta el fragmento extraído entre dos nodos consecutivos sin modificar el sentido de los nodos del fragmento extraído.



En primer lugar se extrae de la ruta original el fragmento formado por los nodos i e $i + 1$. Dicha extracción se realiza a partir de la eliminación de las aristas que preceden y siguen a dicho fragmento. Posteriormente se inserta dicho fragmento entre los nodos j y $j + 1$, sin invertir sentido del fragmento extraído. Para realizar la inserción del fragmento eliminado en su ubicación definitiva, se elimina una tercera arista: la que unía inicialmente a los nodos j y $j + 1$.

Figura 3-4: Operador Or-opt.

concreto. Por último, para una mayor claridad, los nuevos arcos generados se representan con líneas discontinuas.

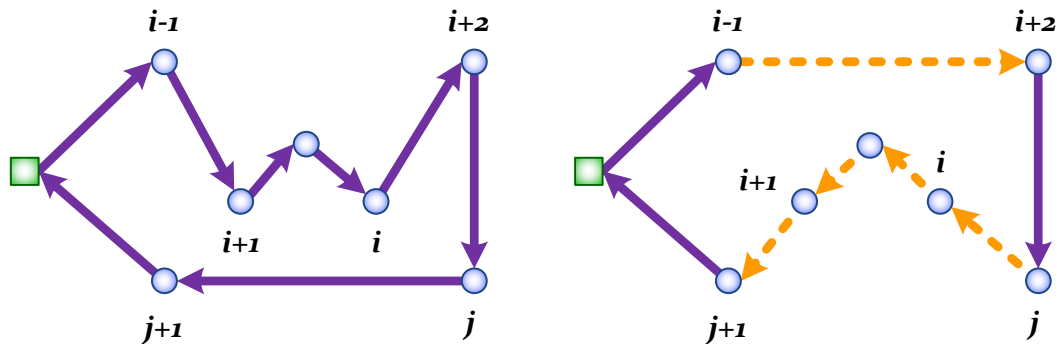
Babin y otros en (Babin, Deneault y Laporte 2007) proponen una serie de mejoras a los operadores 2-opt y Or-opt aplicados al TSP Simétrico (la distancia entre cada par de nodos es la misma en un sentido y otro, es decir, el grafo que define el problema es no dirigido).

3.2.1.3 Operador intraruta IOPT

Bräysy en (Bräysy 2003) propone una generalización del operador *Or-opt* que se denomina IOPT (la “I” hace referencia a inversión):

- El fragmento que se extrae de la ruta original puede tener cualquier longitud (número de clientes/nodos) y no está limitado a 2 o 3 nodos.
- Existe la posibilidad de invertir el orden de los nodos del fragmento que se extrae de la ruta original. Es decir, al reinsertar el fragmento extraído de nuevo en la ruta, se elige la mejor alternativa: la que conserva la orientación de los nodos, o la inversa.

Esta generalización del operador Or-opt es más exhaustiva que la versión original puesto que contempla un mayor número de alternativas. Este hecho aumenta la probabilidad de encontrar una mejor solución, pero también la complejidad temporal del operador.



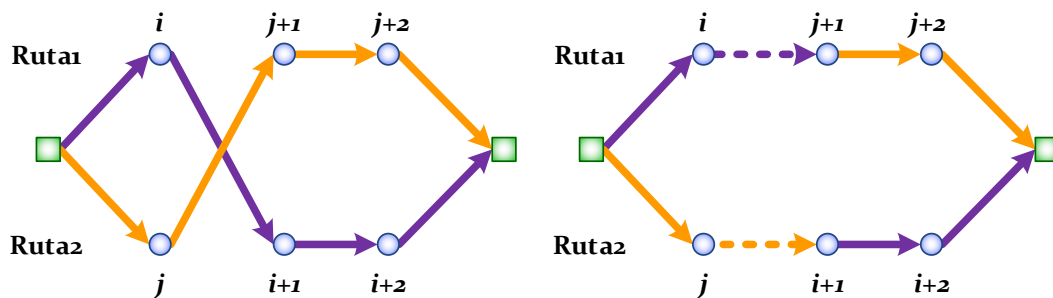
En primer lugar se extrae de la ruta original el fragmento que se encuentra entre los nodos i e $i + 1$. Posteriormente se inserta dicho fragmento entre los nodos j y $j + 1$ invirtiendo el sentido del fragmento extraído. La inversión del fragmento extraído se produce porque representa una mejora frente a la alternativa que supone no realizar dicha inversión.

Figura 3-5: Operador IOPT.

3.2.1.4 Operador interruta 2-opt*

Potvin y Rousseau (Potvin y Rousseau 1995) compararon diferentes operadores intraruta de intercambio de aristas aplicados a problemas de tipo VRPTW (2-Opt, 3-Opt y Or-Opt) y definieron un nuevo operador interruta denominado 2-opt*.

La idea básica de este operador consiste en eliminar un arco de cada una de las rutas e intercambiar el fragmento final de cada una de las rutas con la otra, manteniendo la orientación original de los nodos pertenecientes a los fragmentos intercambiados.



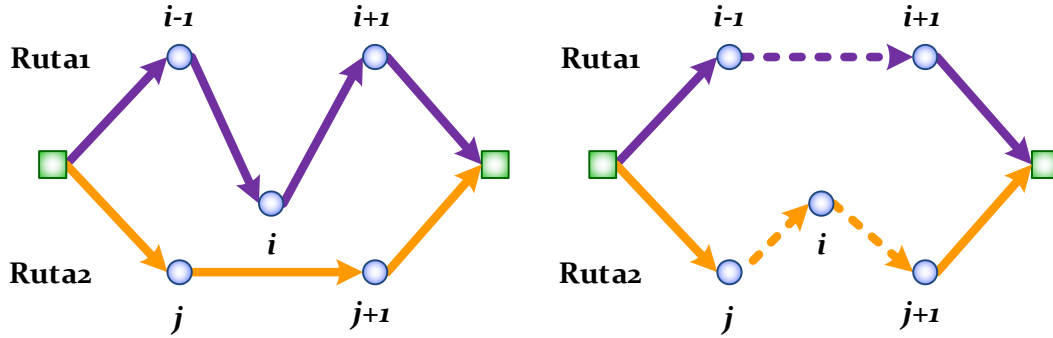
En primer lugar se extrae de cada una de las rutas un fragmento que incluye la parte final de cada una de las rutas. Posteriormente, cada uno de los fragmentos se inserta como parte final de la otra ruta, manteniendo el sentido que tenían los fragmentos originales.

Figura 3-6: Operador 2-opt*.

3.2.1.5 Operador interruta de recolocación

Tres de los operadores de intercambio de arcos interruta que más se referencian en la literatura fueron definidos por Savelsbergh (Savelsbergh 1992): recolocación (*reallocation*), intercambio (*exchange*) y cruce (*cross*).

El operador de recolocación (*reallocation/insertion*) mueve un nodo (también se puede generalizar para una secuencia de nodos) de una ruta a otra lo que supone la eliminación de 3 arcos (dos en la primera ruta y uno en la segunda), y la generación de 3 nuevos arcos.

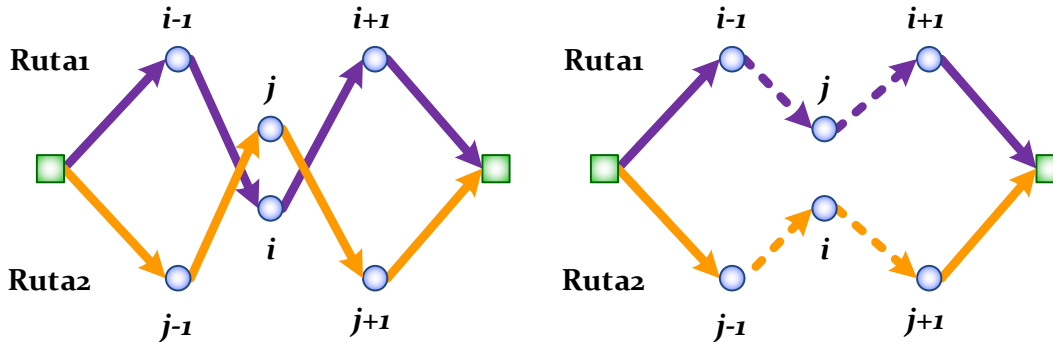


En primer lugar, se extrae el nodo i de la Ruta 1. Posteriormente el nodo i se inserta entre los nodos j y $j + 1$ de la Ruta 2 (lo que implica la generación de dos arcos nuevos) y se reconstruye la Ruta 1.

Figura 3-7: Operador de inserción.

3.2.1.6 Operador interruta de intercambio

El operador de intercambio (*exchange operator*) realiza un proceso de intercambio simultáneo de dos nodos (uno de cada ruta). Este operador es parecido al operador de recolocación, pero en este caso se intercambia un nodo de cada una de las rutas a la otra ruta. La generalización de este operador para intercambiar más de un nodo, se conoce con el nombre de “intercambio de cadena” (*String Exchange*).

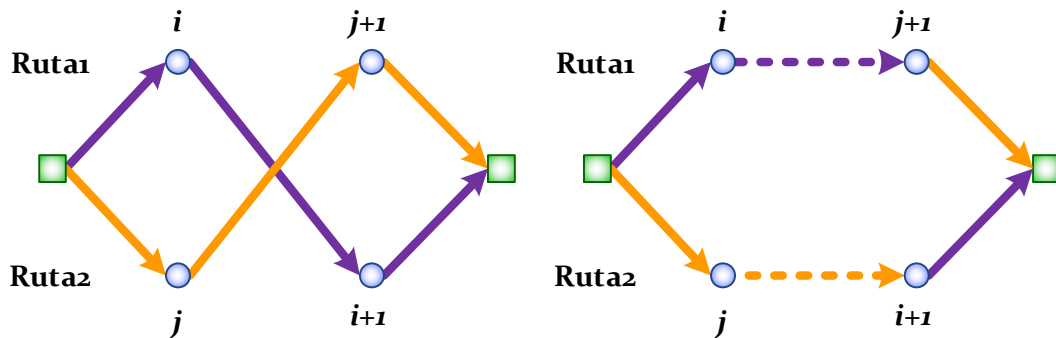


En primer lugar se extrae el nodo i de la Ruta 1 y el nodo j de la Ruta 2. Para ello se eliminan cuatro aristas $(i - 1, i)$, $(i, i + 1)$, $(j - 1, j)$ y $(j, j + 1)$. Posteriormente cada nodo se inserta en la otra ruta, generando cuatro aristas nuevas $(i - 1, j)$, $(j, i + 1)$, $(j - 1, i)$ y $(i, j + 1)$.

Figura 3-8: Operador de intercambio.

3.2.1.7 Operador interruta de cruce (CROSS-change)

El operador de cruce (*CROSS-change*) propuesto por Savelsbergh se centra en el intercambio de arcos entre dos rutas que se cruzan en algún punto. Este operador elimina los arcos que se cruzan intercambiando la parte final de cada una de las rutas. Además, también ofrece la posibilidad de combinar las dos rutas en una sola, siempre que se cumplan las restricciones del problema.

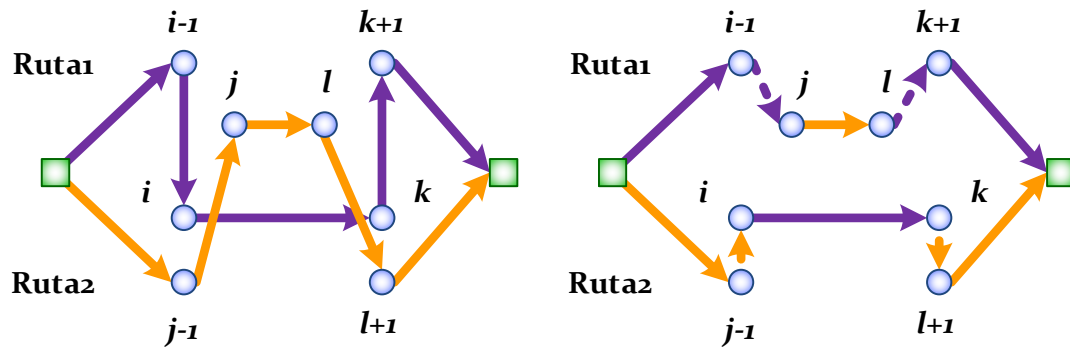


En primer lugar, se eliminan los dos arcos que se cruzan $(i, i + 1)$ y $(j, j + 1)$. Posteriormente, se crean los nuevos arcos que intercambia la parte final de ambas rutas.

Figura 3-9: Operador de cruce.

3.2.1.8 Operador interruta de cruce con intercambio CROSS-exchange

El operador de cruce con intercambio (*CROSS-exchange*) (Taillard, y otros 1997) afecta a dos rutas e implica la extracción de un fragmento (de un número cualquiera de nodos) de cada una de las rutas y el intercambio de dichos fragmentos. Los fragmentos intercambiados mantienen el sentido que tenían en la ruta original. Este intercambio de fragmentos implica la eliminación de 4 arcos y la generación de otros 4 para unir los fragmentos intercambiados en la ruta destino.



En primer lugar se extrae el segmento de la Ruta 1 que está entre los nodos i y k , y el segmento de la Ruta 2 que está entre los nodos j y l . Posteriormente, los segmentos se intercambian de ruta y se generan los arcos necesarios para integrar cada segmento en la ruta destino.

Figura 3-10: Operador de cruce o CROSS-exchange.

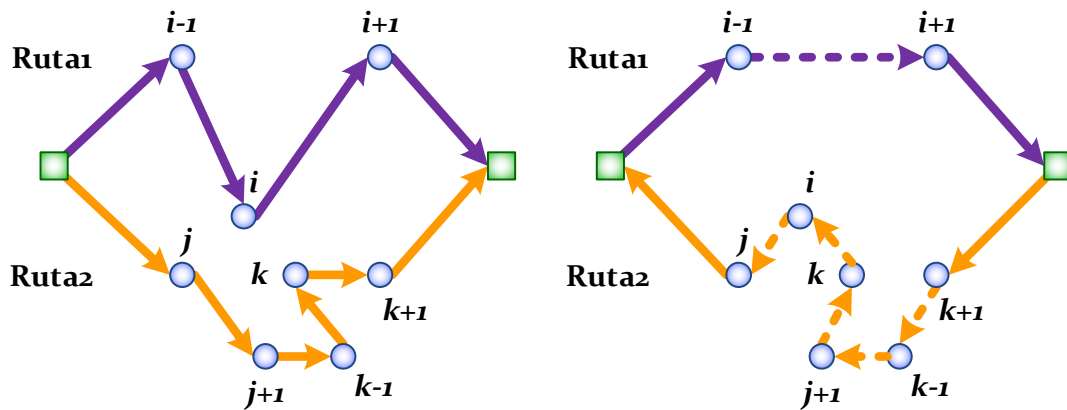
3.2.1.9 Operador interruta I-CROSS

Este operador fue propuesto por Bräysy (Bräysy 2003) como una modificación del operador CROSS de Taillard y otros. La modificación propuesta es muy parecida a la que propuso el mismo autor para el operador Or-opt (de ahí que ambos operadores comiencen por la letra “I”). En este caso, se contempla la inversión del sentido de los segmentos que se extraen de cada una de las rutas originales; eligiendo el mejor de los sentidos (directo o invertido) a la hora de insertar cada segmento extraído en la ruta destino.

3.2.1.10 Operador interruta intercambio GENI

El operador intercambio-GENI (*GENI-exchange*) (Gendreau, Hertz y Laporte 1992) es una extensión del operador de recolocación (Savelsbergh, The vehicle routing problem with time windows: Minimizing route duration 1992). Su esencia, al igual que la del operador de recolocación, consiste en eliminar un nodo de una ruta e insertarlo entre dos nodos de otra ruta.

En el proceso de inserción en la nueva ruta, el nodo se puede colocar entre los dos nodos más próximos a él, aunque éstos no sean consecutivos. Por ese motivo, a diferencia del operador de recolocación, el operador de intercambio-GENI puede modificar el sentido de alguna de las aristas de la ruta destino.



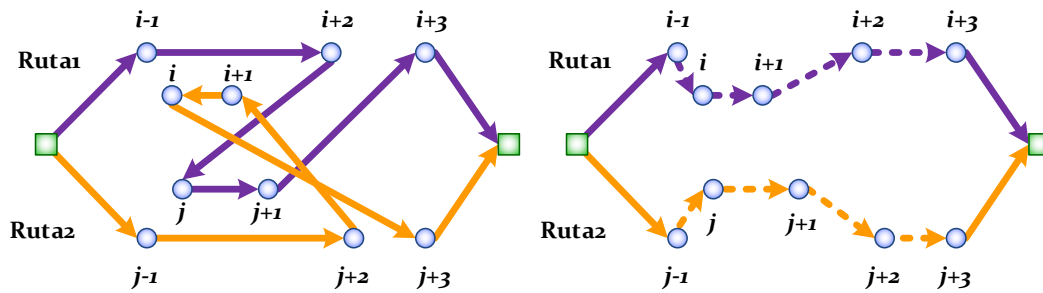
En primer lugar se extrae el nodo i de la Ruta 1 para ser insertado en la Ruta 2. Posteriormente, se inserta el nodo i entre los nodos j y k (que son los más próximos a él). Este intercambio obliga a reorientar todos los arcos de la Ruta 2.

Figura 3-11: Operador intercambio GENI.

3.2.1.11 Operador interruta GENICROSS

Bräysy y otros (Bräysy, y otros 2003) proponen una modificación del operador interruta CROSS-exchange de Taillard y otros (Taillard, y otros 1997) cuyo objetivo es el intercambio simultáneo de dos segmentos de clientes entre dos rutas, pero con las siguientes modificaciones:

- El punto de inserción en la ruta destino puede ser cualquiera y no el punto en que se extraía el segmento a intercambiar de la ruta destino.
- Se permite la inversión de los segmentos intercambiados o alguno de los arcos existentes en las rutas originales.
- Estas modificaciones permiten la generación de un operador más robusto ya que el tamaño del vecindario considerado es mucho mayor, y por otro lado, no existe un aumento significativo en el tiempo de ejecución puesto que el nuevo operador incorpora técnicas de aceleración del proceso de intercambio.



En primer lugar se extraen los segmentos $(i + 1, i)$ de la Ruta 1 y $(j, j + 1)$ de la Ruta 2. Posteriormente los segmentos se intercambian de ruta y se crean los arcos necesarios para configurar de nuevo rutas válidas. Además, se invierten algunos de los arcos, tanto los que pertenecen a los segmentos intercambiados como alguno de los arcos existentes en la Ruta 2.

Figura 3-12: Operador GENICROSS.

3.2.2 Heurísticas de reducción del número de rutas

En la sección anterior se han descrito los principales operadores de mejora de soluciones aplicados exitosamente a problemas de tipo VRP. Estos operadores, denominados en ciertos contextos “operadores de mejora”, se basan en la realización de “pequeñas” modificaciones en una única ruta, o en pares de rutas, mediante el intercambio de nodos (clientes) o segmentos (secuencias de clientes).

Estos operadores son muy eficientes y útiles para focalizar o intensificar el proceso de búsqueda en regiones del espacio de soluciones prometedoras, siendo su objetivo fundamental la reducción del coste de la solución medido en distancia o tiempo, pero tienen serias dificultades para reducir el número de rutas, salvo casos excepcionales en los que al intercambiar algún nodo, las rutas involucradas se queden vacías. Por ese motivo, han surgido varias técnicas heurísticas cuyo objetivo fundamental es la reducción del número de rutas. Estos procesos realizan modificaciones más sofisticadas que los operadores de intercambio de aristas y nodos puesto que afectan a un mayor número de rutas y por lo tanto provocan mayores alteraciones en la solución actual, lo que aumenta la capacidad de exploración (o diversificación) del proceso de búsqueda permitiendo explorar el espacio de soluciones de una manera más exhaustiva.

Una de las primeras referencias a este tipo de técnicas aplicada exitosamente al TSP, es la conocida como “cadenas de expulsión” (*ejection chains*) propuesta por Glover en (Glover 1996). Esta técnica se basa en encadenar secuencias de movimientos de nodos de su ubicación original a otra ubicación para minimizar el coste total de la solución. Durante este proceso de movimiento de nodos puede ser necesario expulsar (*eject*) algún nodo de su ubicación original “para hacer hueco”.

Esta noción, ha sido utilizada con éxito en el contexto del VRP en multitud de ocasiones, entre las que destacan (Rego y Roucairol 1996), (Rego 2001) y (Lim y Zhang 2007). Todos estos procesos comienzan extrayendo un nodo (o secuencia de nodos) de su ruta original. Los nodos extraídos se almacenan temporalmente en un “grupo de extracción” (*Ejection Pool*) del tal forma que el proceso trabaja de forma iterativa para intentar reinsertar los nodos extraídos en alguna de las rutas existentes. Durante el proceso de reinserción, puede ser necesario extraer nodos (que se añaden al “grupo de extracción”) que previamente formaba parte de una ruta. El proceso se detiene cuando el “grupo de extracción” está vacío y todos los nodos están asignados a una ruta. Dependiendo del número de nodos que sean extraídos de una ruta, este proceso, tiene una mayor probabilidad de reducir el número de rutas de la solución actual.

Uno de los mejores ejemplo de uso de esta técnica de extracción de nodos para la reducción del número de rutas, el método de Nagata y Bräysy (Nagata y Bräysy 2009a). Estos autores proponen una heurística específica para el VRPTW cuyo objetivo es la eliminación sucesiva de rutas haciendo uso del concepto de “grupo de expulsión”. Esta heurística es una pieza clave de una metaheurística (Nagata, Bräysy y Dullaert 2010) que obtiene unos resultados excelentes para las instancias de problemas de referencia en el ámbito del VRPTW, tal y como se puede comprobar en la revisión del estado del arte realizada por Gendreau y Tarantillis en 2010 (Gendreau y Tarantilis 2010) o más recientemente por Vidal y otros en (Vidal, y otros 2013) y (Vidal, y otros 2014).

La heurística de reducción del número de rutas de Nagata y Bräysy funciona de acuerdo al siguiente proceso:

- En primer lugar se selecciona aleatoriamente una ruta y se elimina de la solución actual.
- Los clientes que configuraban la ruta eliminada se agregan al “grupo de expulsión” (*Ejection Pool*). A cada cliente que se encuentra en el grupo de expulsión se le asocia un valor numérico que contabiliza los intentos fallidos de reinserción en la solución, este valor mide la dificultad de reinserción que posee un determinado cliente.
- A continuación, se realiza un proceso iterativo para reinsertar cada uno de los clientes del “grupo de expulsión” en alguna de las rutas restantes. Este proceso se detiene cuando todos los clientes han sido reinsertados satisfactoriamente en la solución actual, o cuando expira el tiempo o un número máximo de iteraciones.
 - Los clientes que forman parte del “grupo de expulsión” se procesan de acuerdo a un esquema LIFO (*Last In First Out*) de tal forma que los clientes extraídos más recientemente serán los primeros en reinsertarse, y los clientes más difíciles de reinsertar permanecerán más tiempo en el “grupo de expulsión”.

- El proceso de reinserción del siguiente cliente del “grupo de expulsión” comienza con el análisis de las posibles soluciones parciales que se obtienen a partir de la inserción de cliente a reinsertar en cualquier posición de las rutas existentes, seleccionando una de ellas de forma aleatoria.
- Si el cliente no puede ser reinsertado en ninguna de las rutas existentes, se genera temporalmente una solución no válida. El cliente se inserta en la ruta que implique una menor penalización (en base a una función que pondera tanto el exceso de capacidad como el incumplimiento de las restricciones temporales). Posteriormente se realiza un proceso de búsqueda local adicional para restaurar la validez de la solución parcial, que en caso de no ser satisfactorio finaliza el proceso de dicho cliente.
- Si el cliente no puede reinsertarse en ningún caso (directamente o tras intentar recomponer la solución no válida), se incrementa el número de intentos de reinserción y se analizan todas las posibles extracciones de clientes que permitan la inserción del cliente actual. En este punto, el cliente que pretendía reinsertarse inicialmente se mantiene en el “grupo de expulsión”, añadiendo además, los clientes extraídos que facilitaban su reinserción.

Después de extraer los nuevos clientes, se realiza un proceso de búsqueda local sobre las rutas resultantes (reordenándolas con la intención de facilitar las opciones de reinserción de nuevos clientes en la siguiente iteración). En la siguiente iteración, dichos se procesa el siguiente cliente del “grupo de expulsión”, que será el que se extrajo de su ruta en último lugar (siguiendo el comportamiento LIFO del “grupo de expulsión”).

Pese a las bondades de esta heurística, el proceso puede llegar a ser muy costoso en tiempo, Por ese motivo, el método ha sido revisado y readaptado posteriormente en (Blocho y Czech 2011) mejorando su rendimiento a partir de la limitación en el número de iteraciones, tamaño del “grupo de expulsión” y las alternativas de reinserción analizadas para cada cliente a reinsertar.

Tal y como se comentará en el capítulo 4 al describir el aporte realizado por la presente tesis doctoral, la filosofía de este método tiene similitudes con el foco del trabajo realizado en la presente tesis doctoral, aunque el alcance y los objetivos no llega a ser del todo comparables.

3.2.3 Mejora de la eficiencia de la búsqueda local

Tal y como se ha comentado en el apartado anterior, la búsqueda local es la base de muchas de las técnicas metaheurísticas y por lo tanto, la eficiencia (media en tiempo

de ejecución y cantidad de memoria necesaria) de la búsqueda local afecta directamente a la eficiencia de cualquier otra técnica que la integre en su proceso. Esta eficiencia está íntimamente ligada a la naturaleza de los vecindarios (tamaño) y la manera de analizar las soluciones vecinas. En esta sección se describen diferentes técnicas cuyo objetivo es el análisis eficiente de las soluciones vecinas para reducir el tiempo que tarda la búsqueda local en realizar su trabajo.

3.2.3.1 Búsqueda local lexicográfica y búsqueda local secuencial

En esta sección se revisa el concepto de búsqueda secuencial como alternativa a la búsqueda lexicográfica (o tradicional). Estas dos alternativas se enmarcan dentro del proceso de búsqueda local del mejor vecino, en la que se revisa completamente la vecindad de forma determinista hasta obtener la mejor solución vecina viable, para luego sustituir la solución actual por la solución vecina obtenida siempre que ésta suponga una mejora.

La diferencia esencial entre las dos alternativas consiste en lo siguiente: la búsqueda lexicográfica genera una solución vecina, evalúa el cumplimiento de las restricciones del problema y la ganancia (o mejora de la función objetivo) respecto a la solución actual, y actualiza la solución actual si se produce una mejora y la solución es válida. Por otro lado, la búsqueda secuencial divide las tareas de generación de la solución vecina en tareas elementales y va calculando la ganancia de cada una de las tareas elementales deteniendo el proceso en cuanto detecta que las tareas elementales no implicarán una mejora en la solución vecina. De esta forma, la búsqueda secuencial es capaz de descartar soluciones vecinas antes de generarlas por completo e incluso saber que la solución es válida. Así, el proceso global del vecindario se realiza de una manera más eficiente, lo que redundará en una mejora en el tiempo de ejecución de la búsqueda local.

La búsqueda secuencial fue descubierta en los años 70 en paralelo por Christofides y Elion (Christofides y Eilon 1972) y Lin y Kernighan (Lin y Kernighan 1973) como parte de algoritmos para la resolución de problemas de tipo TSP, de hecho, la heurística de Lin y Kernighan sigue siendo una de las mejores heurísticas para la resolución de problema de tipo TSP, pero hasta el año 2006 esta técnica no ha sido adaptada a problemas de tipo VRP. A continuación, se detallan las cuestiones más relevantes de la adaptación de la búsqueda secuencial a los problemas de tipo VRP realizada por Irnich y otros en (Irnich, Funke y Grünert 2006).

Tal y como se mostró en el **algoritmo 3-1** la búsqueda local define un orden de proceso de los movimientos u operadores de generación de soluciones vecinas y en cada iteración genera una solución vecina a partir de la solución actual y un operador. Seguidamente se evalúa su idoneidad (cumplimiento de las restricciones del problema) y la ganancia, y en caso de que se cumplan las dos comprobaciones, se actualiza la

solución actual. Por lo tanto, hasta que se genera por completo la solución vecina, el algoritmo de búsqueda no puede determinar si dicha solución es idónea y aporta ganancia.

El primer cambio que introduce la búsqueda secuencial es la división del movimiento u operador en “movimientos parciales” que realizan pequeñas modificaciones sobre la solución actual. Así, una descomposición $m = p_\ell \circ \dots \circ p_2 \circ p_1$ de un movimiento m en $\ell \geq 2$ movimientos parciales p_1, p_2, \dots, p_ℓ significa que una solución válida $x \in Z$ primero se transforma en $p_1(x)$, luego $p_1(x)$ se transforma en $p_2(p_1(x))$ y así sucesivamente. En el caso concreto de los problemas de tipo VRP existen dos tipos de movimientos básicos que se combinan para la definición de movimientos y operadores más complejos: añadir arcos (p_{ij}^{add} añade el arco $\{i, j\}$) y eliminar arcos (p_{kl}^{del} elimina el arco $\{k, l\}$).

Junto con la descomposición de los movimientos, la mejora en el coste (o ganancia) de los movimientos parciales es otro aspecto relevante en el funcionamiento de la búsqueda secuencial. La ganancia de un movimiento m aplicado a una solución x , se define como $g(m, x) = c(x) - c(m(x))$. Para que la búsqueda secuencial sea eficiente, la descomposición de un movimiento m en movimientos parciales $m = p_\ell \circ \dots \circ p_2 \circ p_1$ debiera tener “independencia de coste”. Es decir, la ganancia de cada movimiento parcial debiera depender únicamente de la solución actual y no de otros movimientos parciales. De esta forma, los movimientos parciales podrían aplicarse y evaluarse en cualquier orden y tendrían “independencia de orden”. De lo contrario, durante el proceso de generación de la solución vecina habría que restringir el orden de los movimientos parciales. La división de un movimiento en movimientos parciales puede tener alguna de las siguientes propiedades:

- $m = p_\ell \circ \dots \circ p_2 \circ p_1$ posee “independencia de coste” (*cost-independent*) si:
 - $g(m, x) = \sum_{i=1}^{\ell} g(p_i, x)$
- $m = p_\ell \circ \dots \circ p_2 \circ p_1$ posee “independencia de orden” (*order-independent*) si:
 - $m(x) = p_{\pi(\ell)} \circ \dots \circ p_{\pi(2)} \circ p_{\pi(1)}(x)$ se cumple para todas las soluciones $x \in Z$ y todas las permutaciones $\pi = \{1, 2, \dots, \ell\}$.
- $m = p_\ell \circ \dots \circ p_2 \circ p_1$ posee “independencia circular” (*cyclic-independent*) si:
 - El efecto del movimiento m es independiente de cualquier permutación cíclica de los movimientos parciales.

Las divisiones de movimientos en movimientos parciales con “independencia circular”, son esenciales para el desarrollo de búsquedas locales eficientes.

Para analizar las diferencias entre la búsqueda lexicográfica y la búsqueda secuencial, se considera el siguiente problema de búsqueda general:

Un vecindario $\mathcal{N}(x)$ de tamaño $\mathcal{O}(n^k)$ se define implícitamente por los movimientos m_{i_1, i_2, \dots, i_k} siendo $\{i_1, i_2, \dots, i_k\}$ un subconjunto de $\{1, \dots, n\}$ de cardinalidad k .

La forma natural de generar los k diferentes elementos $i_1 < i_2 < \dots < i_k$ es utilizar k bucles anidados. El primer bucle contempla los elementos $i_1 \in \{1, \dots, n\}$, el segundo bucle los elementos $i_2 \in \{i_1 + 1, \dots, n\}$, y en general el bucle “ p -ésimo” (p -th) contempla los elementos $i_p \in \{i_{p-1} + 1, \dots, n\}$. Éste es el enfoque que utiliza la búsqueda lexicográfica.

Algoritmo 3-2: Pseudocódigo de una búsqueda lexicográfica general

```

1. Entrada: Una solución inicial  $x \in X$  y ganancia mínima  $G_{\min} \in \mathbb{R}$ 
2.  $G^* = G_{\min}$ 
3. LOOP  $i_1 \in \{1, 2, \dots, n\}$ 
4.   LOOP  $i_2 \in \{i_1 + 1, i_1 + 2, \dots, n\}$ 
5.     ...
6.   LOOP  $i_k \in \{i_{k-1} + 1, i_{k-1} + 2, \dots, n\}$ 
7.     IF ( $m_{i_1, i_2, \dots, i_k}(x) \in X$  AND  $G^* < g(m_{i_1, i_2, \dots, i_k}, x)$ )
8.        $G^* = g(m_{i_1, i_2, \dots, i_k}, x)$ 
9.        $(i_1^*, i_2^*, \dots, i_k^*) = (i_1, i_2, \dots, i_k)$ 
10. Salida: IF ( $G^* > G_{\min}$ ) THEN RETURN  $(i_1^*, i_2^*, \dots, i_k^*)$ 

```

El **algoritmo 3-2** encuentra el mejor vecino $x' \in \mathcal{N}(x)$ en función del valor definido como ganancia mínima G_{\min} ; o indica que no puede encontrarse un vecino con mejora. El vecino con mejora x' quedaría definido por el movimiento $x' = m_{i_1^*, i_2^*, \dots, i_k^*}(x)$. En el paso 7, el algoritmo comprueba que el vecino es una solución válida y que su ganancia (o mejora) es superior al mínimo establecido inicialmente. Este paso es clave para garantizar la eficiencia del algoritmo de búsqueda puesto que la comprobación de idoneidad del vecino y el cálculo del coste son las tareas más costosas (en tiempo especialmente) del algoritmo.

La búsqueda secuencial explora los movimientos parciales de un vecindario que posea “independencia de coste” e “independencia circular” analizando la ganancia de cada movimiento parcial para detener el proceso de generación de la solución vecina en cuanto se detecta que alguno de los movimientos parciales no aportará ganancia respecto a la solución actual. De esta forma, la eficiencia viene por el hecho de que no es estrictamente necesario escanear el vecindario completo (salvo en las situaciones en la que sólo el último movimiento parcial sea el decisivo para determinar la existencia de ganancia). La exigencia de que la descomposición de los operadores, y por consiguiente los vecindarios tengan “independencia de coste” e “independencia circular” es debido al siguiente teorema de Lin y Kernighan (Lin y Kernighan 1973):

Teorema de Lin y Kernighan: Si una secuencia de números $(g_i)_{i=1}^k$ tiene una suma positiva $\sum_{i=1}^k g_i > 0$, entonces existe una permutación circular π de dichos números tal que, cualquier suma parcial es positiva, es decir, $\sum_{i=1}^{\ell} g_{\pi(i)} > 0$ para todo $1 \leq \ell \leq k$.

El **algoritmo 3-3** muestra el pseudocódigo de una búsqueda secuencial que realiza la misma búsqueda que se mostraba en el **algoritmo 3-2**; posteriormente se explicarán las implicaciones del teorema de Lin y Kernighan en dicho algoritmo.

Algoritmo 3-3: Pseudocódigo de una búsqueda secuencial general

```

1. Entrada: Una solución inicial  $x \in X$  y ganancia mínima  $G_{min} \in \mathbb{R}$ 
2.  $G^* = G_{min}$ 
3. LOOP  $i_1 \in \{1, 2, \dots, n\}$ 
4.   CALCULAR  $B_1$  (en base a  $i_1$ )
5.   LOOP  $i_2 \in NL(i_1)$  mientras  $g(p_{i_1, i_2}, x) > B_1$ 
6.     CALCULAR  $B_2$  (en base a  $g(p_{i_1, i_2}, x)$  e  $i_2$ )
7.     LOOP  $i_3 \in NL(i_2)$  mientras  $g(p_{i_1, i_2}, x) + g(p_{i_2, i_3}, x) > B_2$ 
8.       ...
9.       CALCULAR  $B_{k-1}$  (en base a  $\sum_{l=1}^{k-2} g(p_{i_l, i_{l+1}}, x)$  e  $i_{k-1}$ )
10.      LOOP  $i_k \in NL(i_{k-1})$  mientras  $\sum_{l=1}^{k-1} g(p_{i_l, i_{l+1}}, x) > B_{k-1}$ 
11.        IF  $(m_{i_1, i_2, \dots, i_k}(x) \in X \text{ AND } G^* < g(m_{i_1, i_2, \dots, i_k}, x))$ 
12.           $G^* = g(m_{i_1, i_2, \dots, i_k}, x)$ 
13.           $(i_1^*, i_2^*, \dots, i_k^*) = (i_1, i_2, \dots, i_k)$ 
14. Salida: IF  $(G^* > G_{min})$  THEN RETURN  $(i_1^*, i_2^*, \dots, i_k^*)$ 

```

En este caso, antes de comenzar un nuevo bucle, se calcula el límite de ganancia B_1, B_2, \dots, B_{k-1} que debe aportar el siguiente movimiento parcial, para podar la búsqueda o descartar la solución vecina (que está en proceso de generación) en cuanto se detecte que la ganancia parcial es inferior al límite. El cálculo del límite de ganancia se hace a partir del teorema de Lin y Kernighan, pero extendido por medio de la siguiente proposición o corolario:

Corolario de Irnich y otros: Dado un número G^* . Si una secuencia de números $(g_i)_{i=1}^k$ cumple que $\sum_{i=1}^k g_i > G^*$, entonces existe una permutación circular π de dichos números tal que, cualquier suma parcial cumple, $\sum_{i=1}^{\ell} g_{\pi(i)} > \ell \cdot G^* / k$ para todo $1 \leq \ell \leq k$.

Además, otro cambio fundamental es el hecho de limitar la búsqueda a un conjunto de todos los posibles valores asociados a cada i_p . Dicho conjunto $NL(i_{p-1})$ se conoce con el nombre de “lista de vecinos” (*neighbor list*) y contiene todos los valores en un determinado orden para garantizar que se analizan primero las alternativas con menor ganancia. Esta lista puede estar limitada en tamaño $NL^k(i_{p-1})$ y contener únicamente los “mejores valores” en ese caso recibe el nombre de “lista de candidatos” (*candidate*

list) y representa un elemento clave para optimizar el proceso de búsqueda y reducir el tamaño del vecindario. Tanto la “lista de vecinos” como la “lista de candidatos” se generan como un paso previo (o pre-proceso) antes de iniciar la búsqueda.

Una vez presentadas las dos alternativas de búsqueda por separado (búsqueda lexicográfica y búsqueda secuencial), se muestra un ejemplo de la aplicación de las mismas a un operador concreto de mejora de las soluciones de un problema de tipo VRP: el operador *2-opt** (ver o), que elimina un arco de cada una de las rutas y los sustituye por dos nuevos arcos sin alterar la ordenación de los clientes. En la **figura 3-13** se ilustra el efecto de un operador *2-opt** sobre dos rutas. El operador queda definido por las posiciones i (perteneciente a la Ruta 1) y j (perteneciente a la Ruta 2), que representan los extremos de los dos arcos a eliminar $(i, i + 1)$ e $(j, j + 1)$; siendo los nodos afectados t_1, t_2, t_3 y t_4 .

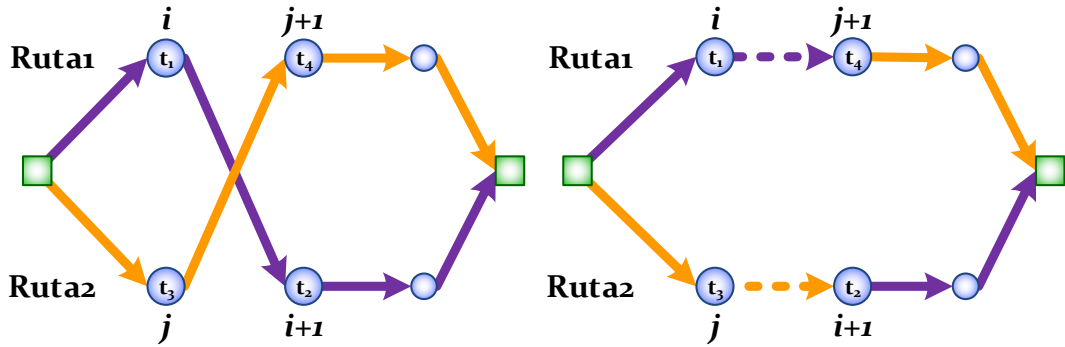


Figura 3-13: Descomposición de un operador *2-opt**.

El operador *2-opt** es un operador simétrico que puede descomponerse en dos movimientos parciales: $m_{i,j}^{2-opt*} = p_{i,j} \circ p_{j,i} = p_{j,i} \circ p_{i,j}$. El movimiento parcial $p_{i,j}$ elimina el arco $(i, i + 1)$ y añade el arco $(j, i + 1)$, y $p_{j,i}$ elimina el arco $(j, j + 1)$ y añade el arco $(i, j + 1)$. La ganancia de cada movimiento parcial es: $g(p_{i,j}, x) = c_{nodo[i],nodo[i+1]} - c_{nodo[j],nodo[i+1]}$ y $g(p_{j,i}, x) = c_{nodo[j],nodo[j+1]} - c_{nodo[i],nodo[j+1]}$ respectivamente (donde: *nodo*[i], representa al nodo que está en la posición i , que en este caso: t_1). Esta descomposición del movimiento *2-opt** posee “independencia de orden” e “independencia de coste”.

Dada una posición i y que el movimiento parcial $p_{i,j}$ tiene una ganancia positiva $g(p_{i,j}, x) > 0$, la búsqueda de la posición j puede realizarse utilizando la lista de vecinos $NL(i + 1)$. El nodo t_3 que está en la posición j tiene que ser un vecino del nodo t_2 que cumpla $c_{t_3,t_2} < c_{t_1,t_2}$. Por lo tanto, la posición i define un límite $B_1 = c_{t_1,t_2}$, y t_3 tiene que buscarse entre los vecinos de t_2 con coste $c_{t_3,t_2} < B_1$. Los siguientes

algoritmos muestran la implementación del operador 2-opt* mediante una búsqueda lexicográfica y una búsqueda secuencial.

Cabe resaltar que los dos algoritmos mostrados obtienen la misma solución final, pero el proceso de búsqueda en el segundo de los bucles es diferente: en la búsqueda lexicográfica (**algoritmo 3-4**) se itera por todos los posibles valores de la posición j , generando el coste y analizando la idoneidad de cada una de las n_j alternativas. Por el contrario, la búsqueda secuencial (**algoritmo 3-5**) itera sólo por los nodos vecinos del nodo que está en la posición $i + 1$ que reportarán una ganancia superior a B_1 , lo que implica una reducción del número de iteraciones (o como mucho se realizará el mismo número de iteraciones), y por consiguiente una reducción en el tiempo necesario para encontrar la solución.

Algoritmo 3-4: Búsqueda lexicográfica del 2-opt***Algoritmo 3-5:** Búsqueda secuencial 2-opt

1. Entrada: $x \in X$ y $G_{\min} \in \mathbb{R}$	1. Entrada: $x \in X$ y $G_{\min} \in \mathbb{R}$
2. $G^* = G_{\min}$	2. $G^* = G_{\min}$
3. LOOP $i \in \{1, 2, \dots, n_i\}$	3. LOOP $i \in \{1, 2, \dots, n_i\}$
4. $t_1 = \text{nodo}[i]$	4. $t_1 = \text{nodo}[i]$
5. $t_2 = \text{nodo}[i + 1]$	5. $t_2 = \text{nodo}[i + 1]$
6.	6. $B_1 = c_{t_1, t_2} - G^*/2$
7. LOOP $j \in \{1, 2, \dots, n_j\}$	7. LOOP $t_3 \in NL(t_2)$ mientras $c_{t_3, t_2} < B_1$
8. $t_3 = \text{nodo}[j]$	8. $j = \text{posición}[t_3]$
9. $t_4 = \text{nodo}[j + 1]$	9. $t_4 = \text{nodo}[j + 1]$
10. $G = c_{t_1, t_2} - c_{t_3, t_2} + c_{t_3, t_4} - c_{t_1, t_4}$	10. $G = c_{t_1, t_2} - c_{t_3, t_2} + c_{t_3, t_4} - c_{t_1, t_4}$
11. IF ($G > G^*$ AND $m_{i,j}^{2-opt^*}(x) \in X$)	11. IF ($G > G^*$ AND $m_{i,j}^{2-opt^*}(x) \in X$)
12. $G^* = G$	12. $G^* = G$
13. $(i^*, j^*) = (i, j)$	13. $(i^*, j^*) = (i, j)$
14. Salida: IF ($G^* > G_{\min}$) THEN	14. Salida: IF ($G^* > G_{\min}$) THEN
15. RETURN (i^*, j^*)	15. RETURN (i^*, j^*)

En el paso 11 de los dos algoritmos debe comprobarse que la solución vecina $m_{i,j}^{2-opt^*}(x)$ cumple las restricciones del problema, y es sin duda el más complejo del algoritmo puesto que dependiendo del tipo de problema concreto habrá que validar más o menos restricciones. El cálculo del límite B_1 , que se obtiene a partir del Corolario de Irnich y otros con $k = 1$ y $\ell = 2$. Otra cuestión a destacar es que en los dos algoritmos ilustrados para el operador 2-opt* se utiliza como criterio de finalización “el mejor vecino con mejora”. Por último, en el algoritmo de búsqueda secuencial se ha utilizado la lista completa de vecinos para buscar la posición j , pero en este caso, una reducción del número de vecinos puede reducir considerablemente el tiempo de ejecución con una diferencia mínima en la calidad de la misma, tal y como se puede comprobar en las pruebas realizadas en (Irnich, Funke y Grünert 2006).

3.2.3.2 Lista de vecinos y lista de candidatos

Tal y como se ha comentado con anterioridad, uno de los aspectos clave en la eficiencia de los algoritmos de búsqueda local reside en la manera en que se explora el vecindario generado. Por ese motivo, los mecanismos y procedimiento que permitan acelerar dicho proceso, o moverse por las áreas más prometedoras del vecindario resultan especialmente interesantes para problemas complejos debido al tamaño de los mismos, o al tipo de restricciones que contemplan.

Dentro de estos mecanismos, uno de los que ofrecen unos buenos resultados, sobre todo en relación a la reducción del tiempo de proceso necesario para analizar un vecindario de intercambio de nodos o arcos, es el conocido como lista de vecinos (*neighbors list*) y lista de candidatos (*candidate list*).

La esencia de estos dos mecanismos se sustenta en el hecho de que al eliminar un arco de una ruta, se rompe un enlace entre dos nodos (que lleva asociado una distancia o coste) y dicho enlace es reemplazado por otro, con un coste menor (para garantizar la existencia de ganancia). Pues bien, la lista de vecinos, almacena para cada uno de los nodos del problema todos los nodos que pueden conectarse con un determinado nodo ordenados de menor a mayor coste. De esta forma, a la hora de analizar las alternativas de unión de un nodo con otro para generar un nuevo arco, se evalúan las alternativas en orden creciente de coste. Por otro lado, la lista de candidatos, es un subconjunto de la lista de vecinos que contiene únicamente los “mejores vecinos”.

Estas listas se generan como un proceso previo a la exploración del vecindario, con lo cual, el tiempo de generación no se incorpora al proceso de búsqueda ya que se puede reutilizar durante todo el proceso puesto que las ubicaciones de los nodos no se alteran en ningún momento. En la práctica, se utiliza una lista de candidatos con un número fijo de nodos entre los que siempre se encuentra el almacén central. Un ejemplo del uso de la lista de vecinos puede encontrarse en la implementación de la búsqueda secuencial propuesta por Irnich (Irnich 2008a).

En el caso particular de los problemas de tipo VRPTW, las listas de vecinos y candidatos deben contemplar la compatibilidad de las ventanas de tiempo entre los diferentes nodos, para garantizar que cuando se intenta unir un nodo con otro, las ventanas de tiempo y el tiempo necesario para que el vehículo circule de un nodo al siguiente hagan viable un nuevo arco. Por ese motivo, estas listas se duplican y se transforman en las listas de vecinos o candidatos “previas” y “posteriores”. La primera almacena los nodos cuya ventana de tiempo permite que sean colocados en una ruta antes de un determinado nodo; mientras que la segunda almacena los nodos cuya ventana de tiempo permite que sean colocados en la ruta después de un determinado nodo. Estas listas se generan en base a los límites de las ventanas de tiempo, los tiempos de servicio en cada nodo, y el tiempo necesario para viajar de un nodo a otro. Estas listas se crean

antes de comenzar el proceso de búsqueda y posteriormente se utilizan cuando se quiere enlazar un nodo con otro, para minimizar la ocurrencia de situaciones en las que existe una ganancia en distancia al unir dos nodos, pero sus ventanas temporales son incompatibles.

3.2.3.3 Actualización de las rutas y evaluación de la validez de las soluciones

Al igual que las listas de vecinos y candidatos, que en el caso del problema de tipo VRPTW ayudan a evitar la generación de soluciones vecinas que no sean correctas por el incumplimiento de las restricciones impuestas por las ventanas de tiempo, diferentes autores han focalizado su trabajo en la mejora de la eficiencia en el proceso de evaluación de las restricciones del problema, y la actualización de los atributos o valores que se utilizan como referencia para evaluar o ponderar las soluciones respecto a la función objetivo.

En relación a la actualización de los valores que se utilizan como referencia para evaluar o ponderar las soluciones respecto a la función objetivo, la práctica más habitual consiste en almacenar una serie de variables globales (Kindervater y Savelsbergh 1997) asociadas a cada una de las rutas que recogen la distancia total recorrida (D), la duración total de la ruta (T), y la demanda asociada a los clientes pertenecientes a una ruta (Q). Estas variables globales se actualizan cada vez que se realiza un cambio en la ruta (en un tiempo constante e independiente del número de nodos $O(1)$) de tal forma que al finalizar la generación de la solución vecina, no sea necesario ningún proceso iterativo que recorra todos los clientes (que supondría un tiempo de proceso proporcional al número de nodos de la ruta $O(n)$). Esta misma noción, pero aplicada a fragmentos de rutas, es adaptada por Irnich en (Irnich 2008b) para asociar recursos (tiempo, distancia, demanda) a segmentos parciales de una ruta, con objeto de optimizar el proceso de actualización y validación de las restricciones del problema (haciendo que su complejidad temporal sea constante) cuando un cambio en una ruta implica la modificación de varios nodos de forma simultánea (como podría ser el caso de un operador interruta 2-opt*, CROSS o I-CROOS).

Las variantes más complejas del VRP como el VRPTW, requieren el uso de una variable global adicional (W) que almacena el tiempo de espera que se genera en la ruta (y que habitualmente se utiliza como parte de la función objetivo), así como valores dinámicos específicos para cada uno de los nodos que almacenan el instante real de inicio temprano o simplemente inicio (*early begin* - e_i o *begin* - b_i) y el instante real de inicio tardío (*late begin* - l_i) del servicio, que se diferencia del inicio más temprano (*earliest begin* - E_i) y el inicio más tardío (*latest begin* - L_i) que son constantes y definen el tamaño de la ventana de tiempo para cada cliente antes de ser asociado a una ruta. Estos valores dinámicos asociados a cada nodo sirven para garantizar que el vehículo no llega a un nodo después de su límite de inicio más tardío, lo que provocaría que la ruta no sea válida. El mantenimiento de estos valores debe realizarse cada vez que se

modifique una ruta, y será evaluado antes de realizar las modificaciones, para evitar la generación de soluciones vecinas que incumplan las restricciones temporales impuestas por las ventanas de tiempo.

En relación a la actualización de los valores de inicio temprano y tardío en cada nodo, en (Solomon 1987) y (Solomon, Baker y Schaffer 1988) se realiza un estudio relacionado con la actualización eficiente de esta información para el problema del TSP con ventanas de tiempo. En ese estudio se definen 2 reglas de actualización denominadas “empuje hacia delante” (*Push Forward* - PF) y “empuje hacia atrás” (*Push Backward* - PB), que utilizadas de una forma adecuada permiten controlar el tiempo necesario para propagar los cambios que implica la modificación de alguno de los arcos de una ruta siendo la complejidad computacional temporal necesaria para dicho proceso en el peor de los casos igual a $O(n)$. A continuación, se analizan con más de detalle estas dos reglas aplicadas al VRPTW, tal y como se recogen en (Campbell y Savelsbergh 2004). Cada vez que se inserta un nuevo nodo, o conjunto de nodos en un punto concreto de la ruta i , dicha inserción obliga a actualizar los valores e_k y l_k en dos direcciones: hacia delante, actualizando e_k para cada nodo que se encuentre entre la posición de inserción y el final de la ruta; y hacia atrás, actualizando l_k para cada nodos que se encuentra entre la posición de inserción y el primer nodo de la ruta:

- El “empuje hacia delante” se realiza desde $k = i$ hasta n y utiliza la siguiente regla de actualización: $e_k = \max(e_k, e_{k-1} + t_{k-1,k} + s_{k-1})$
- El “empuje hacia atrás” se realiza desde $k = i - 1$ hasta 0 y utiliza la siguiente regla de actualización: $l_k = \min(l_k, l_{k+1} - t_{k,k+1} - s_k)$

En ambos casos, el proceso se detiene en cuanto alguno de los valores de e_k o l_k no se modifica respecto al valor que tenía anteriormente; con lo cual, en la práctica, el tiempo necesario para realizar la actualización completa podría ser inferior a $O(n)$, siempre que no sea necesario procesar todos los nodos de la ruta.

Desde el punto de vista de la evaluación de la idoneidad de una solución vecina, la técnica de búsqueda local debe garantizar el cumplimiento de las restricciones del problema antes de aceptar como válida cualquier solución vecina con mejora y realizar el cambio de la ruta anterior. En este sentido, para el caso del VRP clásico, las únicas restricciones que deben contemplarse son la capacidad, y en algunos casos la duración total de la ruta. Estas comprobaciones requieren un proceso muy simple con una complejidad temporal constante $O(1)$ e independiente del número de nodos de una ruta gracias al uso de las variables globales descritas en esta misma sección. Pero en el caso de problemas más complejos como el VRPTW, la evaluación de la idoneidad de una ruta tras una modificación requiere de un proceso más complejo puesto que los cambios en una ruta afectan tanto a los valores globales asociados a la ruta, como a los

valores individuales asociados a cada nodo. Por lo tanto, además de contemplar la evaluación de las variables globales, es necesario evaluar el impacto que tiene la modificación en el cumplimiento de las ventanas de tiempo. En este sentido, normalmente se utiliza un proceso basado en la regla de actualización de “empuje hacia delante” que analiza si el cambio en la ruta provoca que el inicio del servicio en algún nodo sea superior al inicio tardío (L_i); en cuyo caso, la ruta no sería válida. Este procedimiento comienza con la definición del “empuje hacia delante” en la posición i en la que se realiza la modificación que será igual a la diferencia entre el instante de inicio del servicio en el nodo antes y después de la modificación, utilizando la siguiente fórmula:

- $PF = e_i^{new} - e_i^{old}$
 - Si $PF \leq 0$, la ruta es factible, ya que la modificación no afectará a los nodos entre i y el final de la ruta.
 - Si $PF > 0$, se modifica el inicio en el nodo que se encuentra en la posición i , con lo cual habrá que analizar el impacto de dicho “empuje” en los nodos que se encuentran entre la posición i y el final de la ruta para garantizar en todo momento que $e^{new} \leq L$. El proceso se realiza de forma iterativa hasta que $PF = 0$, se incumpla alguna ventana de tiempo, o se llega al final de la ruta.

En cada iteración, el nuevo valor del empuje hacia delante, se calcula a partir del valor actual de PF y el tiempo de espera (w_k) en el nodo actual mediante la expresión:

- $PF = \max(PF - w_k, 0)$
 - De esta forma, a partir de la posición $i + 1$ y mientras $PF > 0$, se validará que $e_k + PF \leq L_k$.

De esta manera, la validación de la idoneidad de una modificación en una ruta, sólo en el peor de los casos (cuando dicha modificación afecte a todos los nodos de la ruta) requerirá una complejidad temporal proporcional al número de nodos $O(n)$.

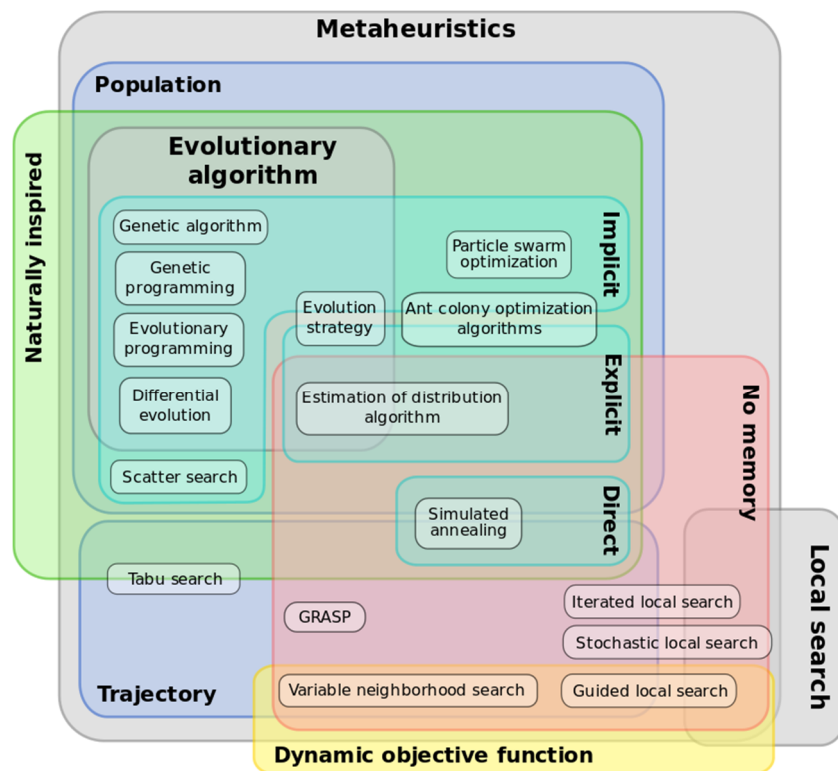
Las cuestiones tratadas en esta sección, pese a ser triviales en el caso de instancia de problemas muy pequeñas, representan una mejora considerable en los tiempos de ejecución de las búsquedas locales y los operadores de modificación de rutas, que son uno de los elementos clave a la hora de analizar la complejidad temporal de las técnicas metaheurísticas que se analizarán en la siguiente sección.

3.3 Metaheurísticas

El término metaheurística, acuñado por Glover en 1986 (Glover 1986) hace referencia a una familia de métodos generales de búsqueda de alto nivel que utilizan diferentes estrategias para explorar el espacio de búsqueda de manera eficaz y eficiente intentando evitar quedarse atrapadas en óptimos locales. Una definición de metaheurística, podría ser la siguiente (Osman y Laporte 1996):

“Una metaheurística se define formalmente como un proceso iterativo de generación que guía y subordina heurísticas mediante la combinación inteligente de diferentes conceptos para explorar y explotar el espacio de búsqueda, mientras se utilizan estrategias de aprendizaje para estructurar información con objeto de encontrar de manera eficiente soluciones casi óptimas.”

En (Blum y Roli 2003) se puede encontrar una extensa clasificación y comparación de las principales estrategias de búsqueda metaheurísticas, que se muestra en la **figura 3-14**.



Fuente: https://commons.wikimedia.org/wiki/File:Metaheuristics_classification.svg [Obtenido en julio de 2015]

Figura 3-14: Clasificación de técnicas metaheurísticas.

En los últimos 25 años las metaheurísticas están siendo un ámbito de investigación clave dentro de la optimización combinatoria tal y como ilustran varias revisiones de la

literatura especializada: (Osman y Laporte 1996), (Blum y Roli 2003) o (Gendreau y Potvin 2005); y libros: (Corne, Dorigo y Glover 1999), (Glover y Kochenberger 2003) y (Gendreau y Potvin 2010).

Dada la relevancia y complejidad inherente a los problemas de tipo VRP, éstos se han convertido en una tipología de problema referentes para la aplicación y validación de técnicas metaheurísticas. Prueba de ello son las revisiones de la literatura específica de la aplicación de metaheurísticas a la resolución del VRP incluidas en (Gendreau, Laporte y Potvin 2002), (Cordeau, y otros 2005), (Gendreau, y otros 2008), (Laporte 2009) y (Potvin 2009).

3.3.1 Metaheurísticas basadas en vecindarios

En esta sección se describen genéricamente las principales familias de metaheurísticas basadas en vecindarios aplicadas de manera satisfactoria a problemas de tipo VRP. Además, se incluyen las referencias de las técnicas que han obtenido los resultados más satisfactorios para cada uno de las familias.

El recocido simulado (*Simulated Annealing* - SA) (Kirkpatrick, Gelatt y Vecchi 1983) (Cerny 1985) mejora la limitación que posee la búsqueda local en cuanto a quedarse bloqueada rápidamente en un óptimo local, aceptando soluciones vecinas sin mejora con una probabilidad que está gobernada por un proceso estadístico denominado programa de enfriamiento. Esta técnica toma su analogía de la industria metalúrgica, en concreto del proceso de fabricación del acero: en primer lugar los componentes del acero se someten a altas temperaturas con lo cual sus partículas se vuelven muy inestables y poseen una gran capacidad de movimiento; posteriormente el compuesto se va enfriando lentamente con lo cual la estabilidad de sus partículas se va reduciendo paulatinamente hasta que no pueden moverse en absoluto. Cuanto mayor es la temperatura mayor es la capacidad que tienen las partículas para moverse libremente; pero a medida que la temperatura decrece, las partículas pierden movilidad. En este criterio se basa el algoritmo para alterar la probabilidad de aceptar vecinos sin mejora durante el proceso del algoritmo: muy alta al principio (facilitando el proceso de exploración o diversificación) y baja al final del proceso (centrándose en la explotación o intensificación).

La búsqueda tabú (*Tabu Search* - TS) centra el proceso de búsqueda en torno al mejor vecino de la solución actual por medio de un mecanismo de aprendizaje basado en memorias de corto, medio y largo plazo. Se podría decir que reemplaza el comportamiento aleatorio que poseen otras metaheurísticas, por el aprendizaje basado en información obtenida durante el proceso de búsqueda. La técnica escapa de los óptimos locales aceptando la mejor solución del vecindario que explora en cada momento. El proceso de decisión a la hora de escoger un vecino se centra en dos

mecanismos fundamentales: el primero de ellos se basa en la memoria a corto plazo y sirve para evitar volver a evaluar soluciones recientemente analizadas (conocidas como tabú); y el segundo permite la aceptación de soluciones que cumplen un “criterio de aspiración” (por ejemplo, la mejor solución, o la solución que contiene una determinada característica que podría estar presente en una buena solución). Por último, la técnica utiliza las memorias a medio y largo plazo para controlar los procesos de “intensificación” (que centra la búsqueda en torno a las mejores soluciones o las que tienen alguna característica que podría incluir una buena solución) y “diversificación” (que intenta avanzar hacia áreas del espacio de búsqueda inexploradas buscando soluciones con características poco frecuentes). La clave del éxito de la búsqueda tabú está en el balance equilibrado entre los procesos de intensificación y diversificación.

La búsqueda tabú se ha aplicado con mucho éxito en la resolución de problemas de tipo VRP, destacando especialmente las técnicas TABUROUTE (Gendreau, Hertz y Laporte 1994), la búsqueda tabú unificada (*Unified Tabu Search* - UTS) (Cordeau, Gendreau y Laporte 1997), (Cordeau, Laporte y Mercier 2001) y la técnica de memoria adaptada (*Adaptive Memory* - AM) (Taillard 1993), (Tarantilis 2005) y (Rochat y Taillard 1995).

Los conceptos de la búsqueda tabú han inspirado otras metaheurísticas como la búsqueda local guiada (*Guided Local Search*) (Voudouris y Tsang 1999), que utiliza el concepto de memoria a largo plazo para penalizar características recurrentes que aparecen en varias soluciones, y ha sido aplicada satisfactoriamente en (Tarantilis, Zachariadis y Kiranoudis 2007) y (Zachariadis y Kiranoudis 2010), a problemas de tipo VRP. De igual manera, el concepto de “criterio de aspiración” juega un rol importante en la búsqueda escalar colinas basada en atributos (*Attribute Based Hill Climber* - ABHC) (Whittle y Smith 2004) y (Derigs y Kaiser 2007).

La búsqueda de vecindario variable (*Variable Neighborhood Search* - VNS) (Whittle y Smith 2004) y (Derigs y Kaiser 2007), analiza el hecho de que un óptimo local está definido para un determinado vecindario. Por lo tanto, la modificación del vecindario, o alguno de sus parámetros, durante el proceso de búsqueda podría ofrecer más posibilidades de encontrar una buena solución. Esta técnica se utiliza con frecuencia en las metaheurísticas híbridas.

Con una inspiración parecida a la anterior, la búsqueda de gran vecindario adaptativo (*Adaptive Large Neighbourhood Search* - ALNS) (Pisinger y Ropke 2007), utiliza los beneficios del uso de vecindarios variados basados en movimientos de tipo “ruina-y-reconstrucción” (*ruin-and-recreate*) (Shaw 1998). La frecuencia de uso de cada uno de los vecindarios se adapta de forma dinámica durante el proceso de búsqueda en base al rendimiento que han producido en el pasado.

Para finalizar, la búsqueda local iterativa (*Iterative Local Search* - ILS) (Lourenço, Martin y Stützle 2010) realiza un proceso iterativo que consta de 2 fases: en primer lugar

se aplica una búsqueda local que obtiene un óptimo local; posteriormente, se realiza un proceso de “perturbación” para escapar del óptimo local y se inicia de nuevo el proceso. Una aplicación de esta técnica puede encontrarse en (Prins 2009).

3.3.2 Metaheurísticas basadas en poblaciones de soluciones

Las técnicas basadas en poblaciones se inspiran en mecanismos de la naturaleza y la evolución de las especies. Los algoritmos genéticos (*Genetic Algorithms* - GA) y evolutivos (*Evolutionary Algorithms* - EA) surgieron en los años 50, pero su formulación actual se debe a Holland (Holland 1975). Estas técnicas interpretan las leyes de la genética y la selección natural para evolucionar una población de soluciones del problema mediante el uso de operadores de selección, cruce (*crossover*) y mutación. Además de las soluciones, los propios operadores evolucionan a medida que avanza el proceso de búsqueda mediante la modificación de parámetros. Tradicionalmente los algoritmos poblacionales avanzaban lentamente, por ese motivo, los esquemas generales han sido complementados con el uso de búsquedas locales generando una nueva familia técnicas de nominada “búsquedas genéticas locales” (Mühlenbein, Gorges-Schleuter y Krämer 1988) o “búsquedas meméticas” (Moscato y Cotta 2010). Referencias de algoritmos genéticos aplicados a problemas de tipo VRP pueden encontrarse en (Potvin 2009). Hay dos cuestiones muy relevantes a la hora de aplicar algoritmos genéticos para problemas de tipo VRP. La primera de ellas es la representación de la solución mediante una ruta-gigante formada por todos los clientes sin usar delimitadores (Prins 2004) junto a procesos de agrupación de clientes para dividir la ruta en un conjunto de rutas. Este enfoque está directamente relacionada con la heurística de construcción de rutas *route-first cluster-second* (ver sección 3.1 Heurísticas constructivas). La segunda, es la gestión de la diversidad de los individuos de la población (Prins 2004) y (Vidal, y otros 2012).

Otros dos métodos de algoritmos poblacionales basados en la recombinación de soluciones son: re-encaminamiento de trayectorias (*Path Relinking* - PR) y búsqueda por dispersión (*Scatter Search* - SS) (Glover 1977) y (Resende, y otros 2010). Estos métodos se diferencian de los algoritmos genéticos clásicos en la manera en que se cruzan los individuos y el tamaño de la población (que suele ser menor).

Los enfoques de optimización mediante colonias de hormigas (*Ant Colony Optimization* - ACO) (Dorigo y Stützle 2004) están inspirados en el comportamiento social que utilizan las hormigas para buscar comida, y son los métodos basados en partículas (*swarm-type*) más utilizados en optimización. Esta técnica ha sido aplicada a los problemas de tipo VRP por varios autores como por ejemplo: (Bullnheimer, Hartl y Strauss 1999), (Bell y McMullen 2004) y (Yu, Yang y Yao 2009).

Otros algoritmos basados en partículas aplicados a problemas de tipo VRP como las colonias de abejas (Marinakis y Marinaki 2010) y los enjambres de partículas (Marinakis y Marinaki 2011) utilizan mecanismos de aprendizaje como redes neuronales (Ghaziri 1996), (Creput y Koukam 2008); y sistemas inmunes artificiales (Masutti y De Castro 2008). Estas técnicas suelen combinarse con algoritmos de búsqueda local, y por ese motivo es complicado estimar de manera apropiada el impacto de los paradigmas de inteligencia cooperativa en el rendimiento de estas técnicas.

Las metaheurísticas poblacionales y bio-inspiradas (las que se basan en el comportamiento de organismos vivos o fenómenos físicos) son sin duda las técnicas de resolución que más interés están despertando en la comunidad científica debido a su versatilidad para resolver problemas de distinta naturaleza y/o restricciones (Pintea 2014). Este hecho es también extensivo en el contexto de los problemas de asignación de rutas a vehículos, tal y como se puede comprobar en la reciente reedición del libro de Toth y Vigo (Toth y Vigo 2015), las revisiones del estado del arte como (Vidal, y otros 2013).

3.3.3 Metaheurísticas híbridas

Las metaheurísticas híbridas combinan los conceptos de diferentes técnicas para aprovechar las bondades de cada una de ellas. La combinación se suele realizar de manera yuxtapuesta (los métodos se invocan de manera consecutiva) o de manera indisociable (los métodos se integran generando una nueva metaheurística completamente diferente). Las técnicas integradas suelen ser heurísticas y metaheurísticas, pero también se utilizan elementos de la programación de restricciones o las búsquedas de árboles entre otros. En (Raidl, Puchinger y Blum 2010) y (Blum, y otros 2011) puede encontrarse información genérica acerca del ámbito de las metaheurísticas híbridas, pero este ámbito que todavía está en proceso de desarrollo.

En la literatura se puede encontrar una amplia variedad de técnicas híbridas que combinan diferentes técnicas por ejemplo: diferentes tipos de técnicas basadas en vecindarios, técnicas basadas en vecindarios y técnicas poblacionales; o metaheurísticas que integran elementos de la programación entera y/o programación de restricciones.

Dos de las tres metaheurísticas más eficientes hoy en día para la resolución de problemas de tipo VRP (Nagata y Bräysy 2009b) y (Vidal, y otros 2012) combinan los algoritmos genéticos con la búsqueda local. También, dentro de las heurísticas más avanzadas se encuentran combinaciones de algoritmos genéticos con la búsqueda tabú (Perboli, Pezzella y Tadei 2008), los algoritmos genéticos con optimización basada en colonias de partículas (Marinakis y Marinaki 2010), o los algoritmos genéticos

combinados con técnicas multipoblacionales (Osaba, Díaz y Onieva 2014) y (Osaba, y otros 2014).

Por último, cabe destacar el trabajo realizado en el ámbito de los algoritmos genéticos híbridos por Vidal y otros en (Vidal, y otros 2014). Este trabajo es una continuación de un estudio previo (Vidal, y otros 2013) en el que los autores realizan un exhaustivo análisis y síntesis de las principales variantes y características del VRP acuñando el concepto de problema de asignación de rutas a vehículos multiatributo (MAVRP). En el citado trabajo se presenta el diseño modular de una metaheurística genérica para la resolución de diferentes variantes del problema de tipo VRP. La metaheurística propuesta combina el uso de un algoritmo genético combinado con una búsqueda local genérica y mecanismos para garantizar la diversidad de los individuos de la población. El trabajo contempla una extensa experimentación en la que se analizan 1099 instancias de 29 variantes del VRP en las que la técnica propuesta igual o mejora los mejores resultado conocidos para cada instancia concreta.

3.3.4 Metaheurísticas paralelas y cooperativas

Las metaheurísticas paralelas exploran el espacio de búsqueda de manera simultánea mediante distintos procesos en paralelo (Toulouse, Crainic y Gendreau 1996), (Alba 2005) y (Crainic y Toulouse 2010). Los distintos tipos de métodos de esta categoría se diferencian en base a cómo se realiza el paralelismo, cómo se comunican los procesos que están trabajando en paralelo y cómo se gestiona la búsqueda global. Una primera clasificación de estos métodos diferencia los métodos de “bajo-nivel” y los métodos de “alto-nivel”.

Los métodos de “bajo-nivel” se caracterizan por descomponer las partes del algoritmo principal en tareas independientes, pero sin alterar el comportamiento del método de resolución. Estos métodos se centran fundamentalmente en los “cuellos de botella” de las técnicas tradicionales: evaluación de vecindarios en las técnicas de búsqueda local, operadores de cruce o selección en los algoritmos genéticos, etc. Este tipo de enfoque ha tenido muy poca aplicación en el contexto de los problemas de tipo VRP.

Por otro lado se encuentran los métodos de “alto-nivel” cuya esencia es la descomposición del problema o la realización de múltiples búsquedas en diferentes espacios de búsquedas. Siguiendo el último de los enfoques, la búsqueda múltiple paralela independiente (*parallel independent multi-search*) implica la obtención de la mejor solución final de entre varios métodos independientes que trabajan en paralelo y no se comunican ni intercambian información. Pese al buen rendimiento que ofrecen este tipo de técnicas, las que verdaderamente ofrecen buenos resultados son las técnicas que utilizan algún “esquema de cooperación” a partir de la compartición o intercambio de información durante el proceso de búsqueda.

Las principales características de las metaheurísticas cooperativas se centran en qué información se intercambia, con qué frecuencia se produce el intercambio de información y la forma en que se cada una de las búsquedas paralelas utiliza la información intercambiada. Los enfoques más habituales utilizan un esquema de memoria centralizada a través del que los diferentes procesos intercambian (de manera asíncrona) las mejores soluciones encontradas o características recurrentes que forma parte de las buenas soluciones. Ejemplos de este tipo de técnicas aplicadas a problemas de tipo VRP pueden encontrarse en: (Rochat y Taillard 1995), (Badeau, y otros 1997), (Rego 2001), (Groër y Golden 2011), (Cordeau y Maischberger 2012) y (Jin, Crainic y Lokketagen 2012).

3.3.5 Metaheurísticas más eficientes para el VRPTW

La variantes del VRP con ventanas de tiempo es sin duda la más estudiada y prueba de ellos son las revisiones de la literatura especializadas como (Bräysy y Gendreau 2005a), (Bräysy y Gendreau 2005b) y (Gendreau y Tarantilis 2010).

Hoy en día existen métodos cuasi-exactos (métodos aproximados que obtienen soluciones muy buenas, las cuales se conjetura que son óptimos globales, pero sin asegurarlo) que son capaces de resolver la mayor parte de las instancias de hasta 100 clientes, e incluso algunas instancias con hasta 1000 clientes. No obstante los métodos exactos dependen en exceso de las características de la instancia concreta de problema y el tamaño de las ventanas de tiempo.

Hoy en día, las técnicas que ofrecen los mejores resultados son metaheurísticas que combinan las técnicas evolutivas con otros métodos. Entre las más eficientes destacan especialmente:

- El algoritmo evolutivo guiado de Repoussis y otros (Repoussis, Tarantilis y Ioannou 2009). Esta técnica combina una técnica evolutiva, mutación mediante una técnica de ruina-y-reconstrucción y una búsqueda local guiada.
- La búsqueda de gran vecindario de Prescott-Gagnon y otros. (Prescott-Gagnon, Desaulniers y Rousseau 2009) combina la búsqueda de gran vecindario con la técnica exacta *branch-and-price* para la reconstrucción de soluciones.
- El algoritmo genético híbrido de Nagata y otros (Nagata, Bräysy y Dullaert 2010) usa un operador de cruce muy eficiente y relaja las ventanas de durante el proceso de búsqueda.
- El algoritmo de camino re-encaminamiento de trayectorias de Hashimoto y Yagiura (Hashimoto y Yagiura 2008) también utiliza conceptos de computación evolutiva.

- El algoritmo genético con control adaptado de diversidad de Vidal y otros. (Vidal, y otros 2013) también combina el esquema evolutivo con la relajación en las ventanas de tiempo por medio de un mecanismo de garantía de la diversidad de las soluciones generadas.

3.4 Características clave del diseño de metaheurísticas para el VRP

Pese a que el objetivo del trabajo realizado en la presente tesis doctoral no es el desarrollo de una nueva técnica de resolución de problemas de tipo VRP que compita con las mejores técnicas existentes en la literatura, en esta sección se incluye una breve reseña al excelente trabajo de síntesis y organización realizado por Vidal y otros en (Vidal, y otros 2013) en el que se realiza una revisión del estado del arte en torno a problemas de asignación de rutas a vehículos con múltiples atributos y se analizan las características de 64 técnicas metaheurísticas aplicadas a 15 variantes del VRP, con objeto de identificar los elementos clave que incorporan dichas técnicas y que son relevantes en el diseño de nuevas técnicas.

El análisis de las diferentes técnicas se realiza en base a 19 características que se agrupan en torno a siete grupos de características:

- *Espacio de búsqueda*: Este grupo de categorías analiza la manera en que las técnicas representan la información del problema. En este sentido, la mayor parte de las técnicas utilizan una representación de “ruta gigante sin delimitadores”. Por otro lado, también es frecuente el uso de soluciones no válidas obtenidas a partir de la relajación de las restricciones del problema. Esto aumenta la robustez de la técnica y facilita la reducción del número de rutas, evitando la complejidad que entraña el proceso de reducción si el problema no se relaja.
- *Vecindario utilizado*: en relación al vecindario utilizado, la gran mayoría de las técnicas utilizan múltiples vecindarios bien de manera secuencial o combinada, porque está demostrado que la búsqueda con vecindario variable es uno de los elementos clave para garantizar el éxito de una técnica, especialmente cuando existen restricciones complejas. Otro aspecto destacable en relación a los vecindarios es el uso de listas de vecinos, y estructuras para almacenar cálculos previos, lo que supone un ahorro en el tiempo de cómputo necesario para el proceso del vecindario.
- *Trayectoria de la búsqueda*: la mayor parte de las técnicas analizadas utilizan elecciones aleatorias durante el proceso, para incrementar la diversidad de las soluciones y evitar el comportamiento cíclico. En relación a la trayectoria del

proceso de resolución, la mayor parte de las técnicas analizadas están basadas en vecindarios y utilizan procesos continuos en los que las modificaciones en una solución, generan soluciones próximas que comparten características con las soluciones previas. Por otro lado, las técnicas poblacionales presentan trayectoria discontinuas basadas en saltos, en las que las soluciones sucesivas se pueden diferenciar bastante de las soluciones anteriores. En algunos casos, se utiliza un enfoque híbrido, que combina la trayectoria continua, con saltos para escapar de óptimos locales.

- *Uso de memorias y estructuras de control:* prácticamente todas las técnicas que obtienen buenos resultados en problemas de tipo VRP incorporan algún tipo de estrategia para recolectar, gestionar y utilizar información que se va generando durante el proceso de búsqueda. Dicha información contempla la recolección de buenas soluciones, o características que presentan las buenas soluciones, que posteriormente utiliza la técnica para guiar el proceso mediante los conceptos de intensificación en torno a las soluciones de mayor calidad y la diversificación que permite la exploración de espacios de soluciones aún no analizados.
- *Hibridación:* esta característica está presente en muchas técnicas, combinando diferentes estrategias poblacionales con estrategias basadas en vecindarios.
- *Paralelismo:* en relación al paralelismo, salvo las técnicas que utilizan el reinicio múltiple, pocas de las técnicas se basan en paralelismo o mecanismos de cooperación.
- *Descomposición del problema:* A pesar de que alguna de las técnicas analizadas descomponen en problema en sub-problemas más simples o con menos elementos, la realidad es que todavía se necesita un mayor esfuerzo de investigación en la descomposición y posterior recombinación de las soluciones de los sub-problemas para configurar la solución al problema completo.

4

Heurística constructiva de inicialización y operadores transformativos de mejora para la resolución del problema VRPTW

Después de revisar los conceptos relativos a los problemas de asignación de rutas a vehículos y las técnicas de resolución heurísticas y metaheurísticas que permiten su resolución, en este capítulo se describe el aporte realizado en la presente tesis doctoral. Dicho aporte tiene por objeto la validación de las dos hipótesis planteadas en el capítulo inicial de este documento.

A modo de resumen, el aporte realizado consiste en una serie de técnicas y procedimientos que permiten mejorar las heurísticas aplicadas a problemas de asignación de rutas a vehículos con restricción de ventanas de tiempo (VRPTW). En ese sentido, Las contribuciones/aportaciones fundamentales de la tesis son:

- Una nueva heurística de construcción de la solución inicial para problemas de tipo VRPTW.
- Un conjunto de operadores de mejora para problemas de tipo VRPTW cuyo énfasis se centra en la reducción del número de rutas.

Por otro lado, a modo de resultado de investigación, también se ha desarrollado un entorno software para la visualización y simulación de instancias y soluciones de problemas de asignación de rutas a vehículos.

Este capítulo se ha estructurado en tres secciones: la sección 4.1 describe la nueva heurística de construcción, la sección 4.2 se centra en los operadores de mejora; y por último, la sección 4.3 en la que se detallan los aspectos fundamentales del entorno para la visualización simulación de instancias de problemas y soluciones.

4.1 Heurística de construcción para el VRPTW

En esta sección se describe el primero de los aportes desarrollado como parte del trabajo de la presente tesis doctoral. Dicho aporte lo constituye una nueva heurística de construcción que optimiza el proceso de generación de la solución inicial para problemas de asignación de rutas a vehículos con restricción de ventanas de tiempo.

El de tipo VRPTW impone una restricción adicional a la ordenación de los clientes que forman parte de una ruta. Por lo tanto, la generación de una solución inicial tiene una dificultad añadida a la variante clásica del VRP en la que únicamente hay que contemplar la capacidad del vehículo (o en algunos casos la duración/distancia máxima), restando importancia a la distancia recorrida puesto que ese objetivo se aborda en una fase posterior. En este sentido, la identificación del número inicial de rutas es un proceso relativamente sencillo puesto que únicamente debe analizarse la demanda total de los clientes y la capacidad de los vehículos. De esta forma, es altamente probable que la solución inicial contenga un número de vehículos cercano al valor mínimo. Por el contrario, la variante con restricción de ventanas de tiempo requiere un proceso de construcción de la solución inicial (también conocido como inicialización) más sofisticado puesto que las restricciones que imponen las ventanas de tiempo en cuanto al inicio más temprano y tardío del servicio en cada uno de los clientes provoca que incluso la definición del número de inicial de rutas, sea en sí mismo un problema de optimización combinatoria ya que podría ocurrir que varios clientes tengan ventanas temporales idénticas y muy pequeñas, lo que imposibilita su asignación a la misma ruta, situación que no ocurre en ningún caso en la variante clásica del VRP.

4.1.1 Bases formales de la heurística de construcción para el VRPTW

Motivado por la dificultad que entraña el proceso de generación de la solución inicial en problemas de tipo VRPTW, el autor de la presente tesis doctoral analizó la posibilidad de generar una nueva heurística de construcción que mejorase a las técnicas

más representativas que se encuentran en la literatura. Además, también se quería tener presente el tiempo de ejecución ya que normalmente el proceso de construcción representa un porcentaje muy pequeño del tiempo que emplean las técnicas heurísticas y metaheurísticas más avanzadas en encontrar una solución a un problema de optimización combinatoria. Por ese motivo, las soluciones iniciales deben generarse lo más rápido posible.

Tras analizar las mejores alternativas de heurísticas de construcción de la solución inicial para problemas de tipo VRPTW (ver sección 3.1.1) se escogieron tres heurísticas secuenciales en las que focalizar el trabajo a desarrollar: la heurística I₁ de Solomon (Solomon 1987), la heurística IMPACT (Ioannou, Kritikos y Prastacos 2001) y la heurística IRCI (Figliozzi 2010). Todas ellas presentan similitudes en cuanto al proceso general:

- Construyen las rutas una a una de manera secuencial, insertando en cada iteración el cliente que minimice una función que se conoce como criterio de inserción. Cada vez que no pueden insertarse más clientes a la ruta en construcción, se inicializa una nueva ruta y el proceso continúa.

Con esta filosofía común, las principales diferencias que existen entre cada una de las heurísticas son las siguientes:

- La heurística I₁ de Solomon analiza todas las posibilidades de inserción de los clientes no asignados a la ruta actual y escoge el cliente y posición que minimicen la función que representa al criterio de inserción.
- La heurística IMPACT tiene la misma filosofía que la heurística I₁, pero sofisticada el criterio de inserción contemplando el impacto de la inserción de un nuevo cliente respecto a sí mismo, respecto a los clientes que están en la ruta en construcción, y respecto al resto de clientes que todavía no forman parte de ninguna ruta.
- Por último, la heurística IRCI tiene un comportamiento diferente, ya que en este caso el proceso completo se realiza en dos fases:
 - En la primera fase se construye una primera solución a partir de la inserción de clientes al final de la ruta actual. Para seleccionar el siguiente cliente a insertar al final de la ruta actual se coge cada uno de los clientes no asignados a la ruta y se considera su inserción al final de la ruta actual, analizando posibles soluciones que se obtendrían tras esta inserción (utilizando la misma técnica para insertar en una ruta los clientes aún no asignados). El proceso de inserción de un nuevo cliente, finaliza seleccionando el cliente que generaría una solución de menor coste. Este proceso se repite tras cada nueva inserción hasta que todos los clientes están asociados a una ruta.

Si bien es cierto que este proceso permite obtener mejores resultados que las otras dos heurísticas analizadas para instancias de problemas con una distribución aleatoria de clientes (tal y como se podrá observar en el capítulo 5), el tiempo que tarda el algoritmo es varias veces superior al empleado por las heurísticas I₁ e IMPACT, llegando a ser la diferencia más de 200 veces mayor (según las pruebas experimentales realizadas por el autor de la presente tesis doctoral) entre la heurística I₁ y la heurística IRCI para cada una de las categorías del juego de ensayo de Solomon (SINTEF 2015).

- En la segunda fase, se realiza un proceso de “reconstrucción” utilizando las nociones de proximidad entre las rutas inicialmente generadas. Este proceso, selecciona las $n - 1$ rutas más próximas entre sí, extrae los clientes, y realiza un nuevo proceso de construcción tal y como se ha descrito en la primera fase. Las nuevas rutas generadas se combinan con la ruta que se había dejado fuera del proceso de reconstrucción, y se repite de nuevo el proceso de reconstrucción hasta que no puede reducirse el número de rutas. Si bien este nuevo proceso consigue mejorar la solución, otra vez los tiempos necesarios para lograr dicha mejora son muy superiores a los que necesitan las otras dos heurísticas analizadas.

Con las características de las heurísticas analizadas, el autor de la presente tesis doctoral propone el diseño de una nueva heurística que: (1) tomando la base de la heurística I₁ de Solomon, (que es la más utilizada en técnicas heurísticas y metaheurísticas avanzadas como se puede comprobar en (Bräysy y Gendreau 2005a), (Vidal, y otros 2013) y (Toth y Vigo 2015)), y (2) combinándola con el proceso de reconstrucción de la heurística IMPACT; obtenga unos mejores resultados que la heurística I₁ de Solomon, y en un tiempo sensiblemente menor que el empleado por la heurística IMPACT.

Con todo lo anterior, la nueva heurística de construcción de rutas propuesta es un proceso determinista⁷ de construcción de rutas secuencial y en dos fases, que posee las siguientes características:

- En primer lugar, se utiliza un proceso de construcción basado en la heurística I₁ de Solomon, mejorado con el uso de “listas de candidatos” para el análisis de las opciones de inserción de cada cliente en la ruta actualmente en construcción.
- En segundo lugar, se realiza un proceso de reconstrucción basado en la técnica utilizada por la heurísticas IRCI, pero utilizando la misma heurística I₁ de Solomon para realizar el proceso de reconstrucción. En este caso, se genera un nuevo sub-

⁷ Dada una instancia del problema y unos mismos parámetros de configuración de para la heurística de I₁ de Solomon, la heurísticas propuesta siempre obtiene los mismos resultados.

problema a partir de los clientes que formaban parte de las $n - 1$ rutas más próximas entre sí, generadas por el proceso inicial; y tras la resolución del sub-problema, se retorna la solución al problema global.

Esta nueva heurística de construcción para problemas de tipo VRPTW permite validar la primera de las hipótesis planteadas por el autor de esta tesis doctoral:

HIPÓTESIS 1: *Es posible definir una técnica de construcción de soluciones iniciales para problemas de asignación de rutas a vehículos con restricción fuerte de ventanas de tiempo que mejore a las principales heurísticas de construcción conocidas.*

Tal y como se analizará en el capítulo 5, esta hipótesis ha sido validada utilizando como referencia el juego de ensayo de problemas de Solomon (SINTEF 2015), obteniendo además mejores resultados que los obtenidos por las otras tres heurísticas de construcción tomadas como referencia durante la experimentación.

4.1.2 Diseño de la heurística de construcción para el VRPTW

En esta sección se describen las particularidades de la heurística de construcción de la solución inicial para problemas de tipo VRPTW. Tal y como se ha descrito anteriormente, esta heurística construye las rutas en dos fases denominadas: inicialización y reconstrucción.

A continuación, se describe el funcionamiento de cada una de las dos fases:

- La *fase de inicialización*, es un proceso similar a la heurística de I_1 de Solomon, utilizando los mismos criterios de selección e inserción de clientes, junto con una variación en la implementación del proceso de análisis de las opciones de inserción de cada cliente en la ruta actual. Esta variación se traduce en el uso del concepto de “listas de vecinos” introducido en la sección 3.2.3.2. De esta forma, se reduce sensiblemente el tiempo necesario para analizar las opciones de inserción de un cliente entre cada par de nodos en la ruta parcial en proceso de construcción.

En concreto, el proceso de análisis de la mejor ubicación de inserción de un cliente en la ruta actual, se realiza de acuerdo al proceso ilustrado en el **algoritmo 4-1**. Dicho proceso analiza únicamente la inserción del nuevo cliente justo antes, o justo después que los clientes cuya ventana de tiempo es compatible con la del cliente a insertar. Por ejemplo, la lista de vecinos NL^{PREV} la formarán los clientes de la ruta (actualmente en construcción) que permitan que el nuevo cliente se inserte delante de ellos de tal forma que se cumplan:

$$\circ \quad E_{c_{new}} + s_{c_{new}} + t_{c_{new}, c_{prev}} \leq L_{c_{prev}}$$

donde:

- $E_{c_{new}}$, es el inicio más temprano del nuevo cliente.
- $s_{c_{new}}$, es el tiempo de servicio del nuevo cliente.
- $t_{c_{new}, c_{prev}}$, es el tiempo de trayecto entre el cliente nuevo y el que quedará en la ruta después de él.
- $L_{c_{prev}}$, es el inicio más tardío para el cliente que quedará después del cliente nuevo.

Algoritmo 4-1: Pseudocódigo del proceso de selección de la mejor ubicación para un cliente

```

1. Entrada: La ruta actual  $r$  y un cliente  $c_{new}$  a insertar
2.  $NL^{NEXT} = NL^{NEXT}(c_{new}, r)$ 
3.  $NL^{PREV} = NL^{PREV}(c_{new}, r)$ 
4.  $p^* = -1$ 
5.  $cost^* = MAX\_VALUE$ 
6. LOOP  $c_{next} \in NL^{NEXT}$ 
7.   IF (isFeasible( $c_{next-1}, c_{new}, c_{next}$ ) &&  $c_1(c_{next-1}, c_{new}, c_{next}) < cost^*$ )
8.      $p^* = position(r, c_{next})$ 
9.      $cost^* = c_1(c_{next-1}, c_{new}, c_{next})$ 
10.  END IF
11. END LOOP
12. LOOP  $c_{prev} \in NL^{PREV}$ 
13.  IF (isFeasible( $c_{prev}, c_{new}, c_{prev+1}$ ) &&  $c_1(c_{prev}, c_{new}, c_{prev+1}) < cost^*$ )
14.     $p^* = position(r, c_{prev}) + 1$ 
15.     $cost^* = c_1(c_{prev}, c_{new}, c_{prev+1})$ 
16.  END IF
17. END LOOP
18. Salida: RETURN  $p^*$ 

```

De forma análoga, la lista de vecinos NL^{NEXT} se utiliza para analizar las opciones de insertar un nuevo cliente inmediatamente después de otro que ya forma parte de la ruta. Las listas NL^{PREV} y NL^{NEXT} se generan como un paso previo al proceso de inicialización y son estáticas, es decir, no se actualizan a medida que se van insertando nodos en la ruta actual, para evitar la sobrecarga que implica el proceso actualización. Por ese motivo, sigue siendo necesaria la comprobación de la inserción en las líneas 7 y 13 puesto que al insertar un cliente en una ruta, el inicio más temprano se sustituye por el inicio “real” (que puede ser superior a E), lo que

podría provocar una incompatibilidad de ventanas de tiempo y la imposibilidad de realizar la inserción. Además, otra particularidad que presentan las listas de vecinos es que, en cada momento, de todos los vecinos de cada cliente únicamente se analizan aquellos que forman parte de la ruta actual.

Con estas modificaciones, el proceso obtiene la misma solución que la heurística de I1 de Solomon, pero mejorando un poco el tiempo puesto que sólo en el peor de los casos tendrán que evaluarse todas las alternativas de inserción de cada cliente en la actual.

- Una vez finalizada la fase de inicialización, se realiza la *fase de reconstrucción*. Esta segunda fase, utiliza la misma filosofía que el algoritmo de mejora de rutas propuesto por Figliozzi como parte de la heurística IRCI (Figliozzi 2010). En este caso, el proceso de reconstrucción selecciona las $n - 1$ rutas más próximas entre sí y extrae sus clientes para generar un sub-problema, que posteriormente es resuelto utilizando de nuevo la heurística de la fase de inicialización anterior. Una vez resuelto el sub-problema, las rutas resultantes se unen a la ruta que inicialmente había quedado reservada, y todas ellas se retornan como la nueva solución.

Para finalizar la descripción de la nueva heurística de construcción de soluciones iniciales para problemas de tipo VRPTW diseñada como parte del trabajo de la presente tesis doctoral, se analizan las similitudes y diferencias de la nueva técnica propuesta respecto a las técnicas que han servido de inspiración para su diseño:

- La nueva heurística de construcción planteada toma como inspiración básica los criterios para la selección y ubicación de clientes propuestos por la heurística I1 de Solomon, por su simplicidad y por la rapidez de dicha heurística a la hora de generar una solución. No obstante, presenta dos diferencias respecto a dicha heurística:
 - Por un lado, el uso de “listas de vecinos” durante el análisis de la mejor ubicación para la inserción de un cliente en la ruta actual. Esta modificación permite reducir el tiempo global de la técnica puesto que se reduce el número de alternativas de inserción que deben comprobarse cada vez que se quiere insertar un cliente en la ruta parcial que está siendo construida en cada instante.
 - Y por otro lado, la incorporación de una segunda fase que tiene por objeto la mejora de la solución inicial a partir de la transformación del problema original en un sub-problema formado por los clientes pertenecientes a las rutas que están más próximas entre sí. En este caso, la modificación persigue la reducción del número de rutas generado por la versión original de la

heurística propuesta por Solomon, aislando los clientes pertenecientes a la ruta más alejada de los clientes de las rutas geográficamente más próximas.

- Además, la heurística definida, pese a compartir la filosofía de dos fases (construcción y mejora) de la heurística IRCI; utiliza procesos más simples (basados en la heurística I₁ de Solomon) lo que supone una notable reducción en el tiempo necesario para la generación de la solución inicial.

Con todo lo anterior, se puede concluir, que pese a compartir similitudes con las dos técnicas utilizadas como inspiración el resultado obtenido es una técnica que combina las bondades en cuanto a sencillez y rapidez de la heurística I₁ de Solomon, con la capacidad que aporta la heurística IRCI de Figliozzi para la reducción del número de rutas utilizando los conceptos de simplificación del problema y la proximidad geográfica entre las rutas.

4.2 Operadores de mejora basados en la reducción del número de rutas

En esta sección se describe el principal de los aportes desarrollados en la presente tesis doctoral. Dicho aporte consiste en un conjunto de operadores de mejora específicamente destinados a problemas de asignación de rutas a vehículos, con restricción de ventanas de tiempo (VRPTW).

En el contexto de los problemas de tipo VRP y sus variantes (ver secciones 2.2 y 2.3) existen una serie de operadores o métodos de transformación de soluciones cuyo objetivo se centra en la mejora de las rutas que constituyen la solución al problema, por medio del intercambio de nodos (clientes) o fragmentos de rutas (secuencias de clientes) tanto a nivel individual como entre pares de rutas. Dichos operadores representan uno de los elementos clave de las búsquedas de vecindarios o búsquedas locales (ver sección 3.2.1). Estos operadores realizan pequeñas modificaciones sobre la solución actual lo que permite controlar el tamaño del vecindario de nuevas soluciones generadas a partir de la solución actual, y por consiguiente el tiempo de ejecución. Si bien el tiempo de proceso es un elemento importante, estos operadores centran su análisis en soluciones cercanas a la solución actual limitando el espacio de búsqueda que son capaces de explorar. Por ese motivo, se dice que estos operadores intensifican el proceso de búsqueda en regiones específicas del espacio de soluciones del problema (próximas a la solución actual), lo que dificulta el proceso de exploración exhaustivo del espacio de búsqueda (a éste último proceso se le denomina “diversificación” de la solución actual) hacia regiones que pudieran contener soluciones más prometedoras. Además, estos operadores tienen una limitada o prácticamente nula capacidad para reducir el número de rutas puesto que sólo en contadas ocasiones el movimiento de un

conjunto de nodos entre dos rutas pueden dejar una de ellas vacía, lo que permitiría eliminarla y reducir el número de rutas de la solución actual.

Por otro lado, en la literatura pueden encontrarse heurísticas que focalizan sus esfuerzos en la reducción del número de rutas (ver sección 3.2.2). Estas técnicas, que tienen su origen en la técnica denominada “cadenas de expulsión”, realizan procesos de extracción y reinserción de nodos, o directamente eliminación de rutas completas y reinserción posterior de los clientes extraídos en las rutas que permanecen en la solución actual. En este último caso, el ejemplo más representativo y exitoso de heurística de reducción de rutas, es la heurística de Nagata y Bräysy (Nagata y Bräysy 2009a) cuyo funcionamiento se explicó con detalle en la sección 3.2.2.

Tomando como inspiración algunos de los conceptos relacionados con la técnica de “cadenas de expulsión”, el principal aporte realizado por la presente tesis doctoral es una familia de operadores de mejora de rutas cuyo objetivo es la reducción del número de rutas de la solución actual, combinado con la reducción de la distancia total recorrida por todas las rutas que forman una solución. En ese sentido, comparten similitudes con los operadores mencionados en el párrafo anterior, combinándolos con conceptos intuitivos como el tamaño de una ruta (número de clientes que posee una ruta) y estimaciones de lejanía y proximidad respecto al “centro de gravedad de una ruta”⁸. Los operadores propuestos están pensados inicialmente para ser integrados en procesos de búsqueda local, aportando a dichos procesos una mayor capacidad de exploración (diversificación) del espacio de soluciones que los operadores tradicionales ya que realizan modificaciones más sofisticadas sobre la solución actual que permiten aumentar la capacidad de diversificación del proceso de búsqueda.

Una vez introducida la noción básica de los operadores de mejora propuestos, se procede a revisar con más detalle el razonamiento seguido para su diseño, y seguidamente, se describen las particularidades de cada uno de ellos.

4.2.1 Motivación para el diseño de los operadores de mejora

Las soluciones iniciales generadas por las heurísticas secuenciales de construcción para problemas de tipo VRPTW (ver sección 3.1.1) focalizan sus esfuerzos en la generación de una solución inicial de una forma rápida y eficiente, lo que dificulta su capacidad de exploración del espacio de soluciones puesto que toman decisiones irreversibles a la hora de asignar los clientes a un vehículo, y ordenarlos dentro de una ruta. Por ese motivo, son necesarios procesos posteriores de mejora, que una vez generada una solución inicial pueden revisar las decisiones tomadas en cuanto a la asignación y

⁸ El “centro de gravedad de una ruta” es el punto “imaginario” que se encuentra en medio del grupo o clúster formado por todos los clientes que componen la ruta (incluido almacén central).

ordenación de los clientes. Este argumento es coherente debido a la naturaleza del problema, pero podría incluso confirmarse explícitamente si el proceso de construcción pudiese analizar con detalle la estructura de las rutas generadas y la ubicación de los clientes.

A continuación, se analiza cómo esta intuición puede confirmarse de forma gráfica. En la **figura 4-1** y la **figura 4-2** se muestran dos posibles soluciones a instancias de problemas pertenecientes a la Clase C del juego de ensayo de Solomon (SINTEF 2015). Esta clase de problema tiene la particularidad de que los clientes están agrupados ($C = clustered$), y que las limitaciones en cuanto a capacidad de los vehículos y tamaño de la ventana temporal del almacén central hacen que las rutas contengan un número reducido de clientes.

Tal y como puede apreciarse al comparar visualmente las dos soluciones, en la primera de ellas los clientes parecen estar más ordenados, ya que esta solución está compuesta por 10 vehículos, y cada uno de ellos atiende a los clientes que se encuentran geográficamente agrupados. Esta solución se corresponde con la solución óptima para la instancia C101, que es la más fácil de resolver de manera exacta debido a que las ventanas de tiempo de los clientes son muy grandes. Este hecho facilita que la mayor parte de técnicas de resolución consigan obtener dicha solución.

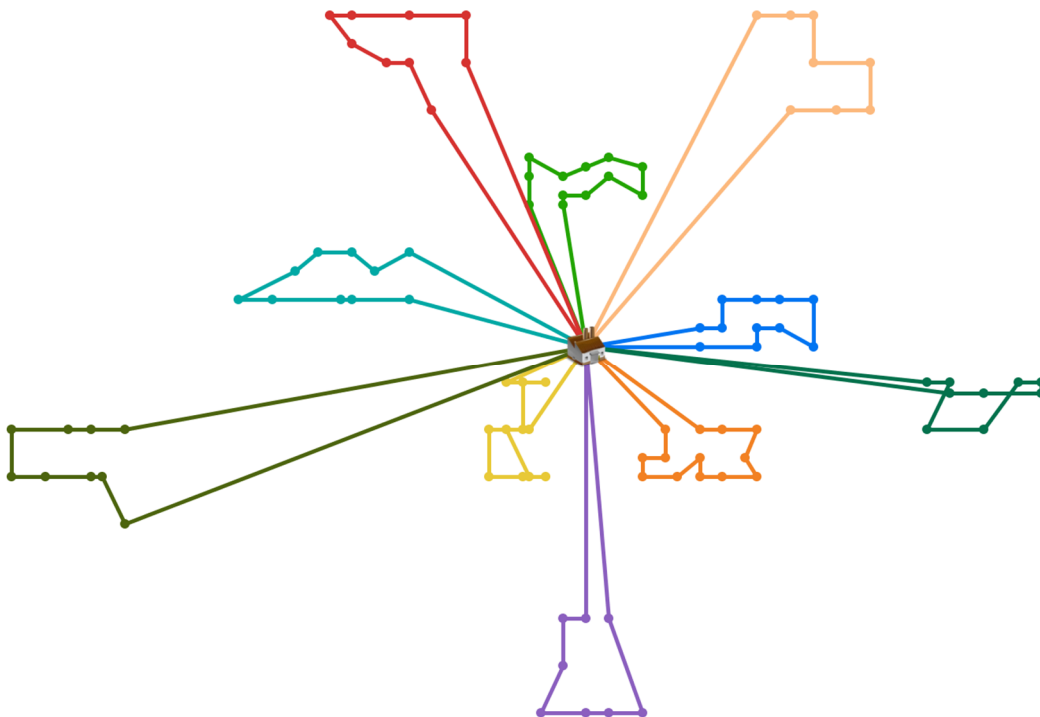


Figura 4-1: Solución a un problema de tipo VRPTW-1.

Por otro lado, la segunda solución permite apreciar que en este caso las rutas parece que han sido generadas sin ningún tipo de criterio ya que además de existir un vehículo más que en la solución anterior (11 frente a 10), los vehículos incluyen a clientes geográficamente muy dispersos, lo que notablemente influye de manera muy negativa en la distancia total recorrida.

Teniendo presente el ejemplo ilustrado, sería deseable el diseño de alguna técnica que permita el movimiento de clientes entre rutas de tal forma que los clientes se ubiquen en torno a los que están geográficamente más próximos a ellos. Este enfoque es el que utilizan los operadores de intercambio de nodos y arcos, pero tiene serias dificultades para reducir el número de rutas partiendo de una solución como la que se muestra en la **figura 4-2**. Este hecho es debido en gran medida a que las modificaciones afectan a un reducido número de rutas. Por ese motivo, es necesario la definición de operadores más sofisticados, que afecten a un mayor número de rutas, con lo cual las probabilidades de reducir su número aumenten. Este último enfoque es el que siguen los operadores de reducción del número de rutas.

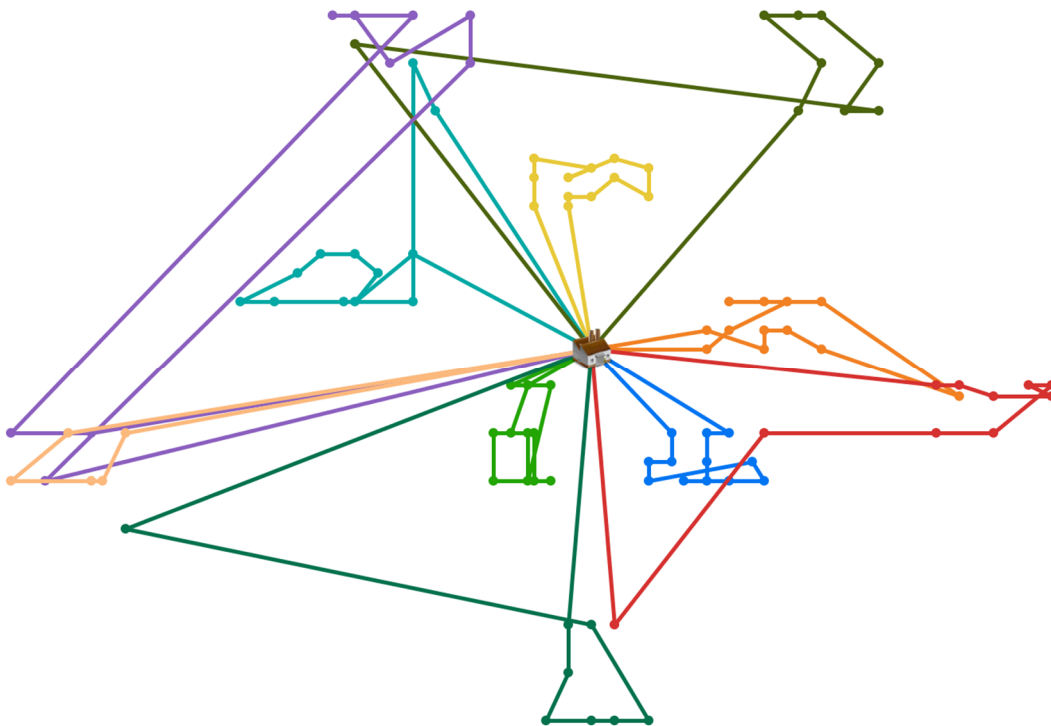


Figura 4-2: Solución a un problema de tipo VRPTW-2.

Los operadores propuestos como aporte de la presente tesis doctoral se engloban dentro de la categoría de operadores que afectan a un conjunto de rutas mayor que los

operadores tradicionales de intercambio de nodos o secuencias de nodos. Los nuevos operadores definidos ponen el foco en la generación de soluciones vecinas que se diferencien notablemente de la solución actual, con la esperanza de que el proceso de búsqueda permita explorar regiones del espacio de soluciones más prometedoras. Esta intuición, se complementa con las nociones intuitivas de tamaño de la ruta (número de clientes que posee una ruta); y proximidad entre los clientes que forman parte de una ruta y el “centro de gravedad” del grupo de clientes (*cluster*) que forman cada ruta. Con este enfoque, los nuevos operadores de intercambio de nodos propuestos complementan a los operadores tradicionales revisados en la sección 3.2.1 y las heurísticas de reducción del número de rutas descritas en la sección 3.2.2.

De esta forma, se ha diseñado una familia de operadores, todos ellos con una estructura similar, pero que se diferencian en el criterio inicial que toman para la reducción del número de rutas o reasignación de clientes entre rutas. En las siguientes secciones se describe en primer lugar la estructura base del operador genérico de reducción del número de rutas, y seguidamente se introducen las peculiaridades de las tres variantes de este operador propuestas: (1) el operador de reasignación de clientes alejados, (2) el operador de eliminación de rutas pequeñas y (3) el operador de eliminación aleatoria de rutas.

4.2.2 Estructura general del operador reducción del número de rutas

Tal y como se ha argumentado en la sección anterior, el objetivo fundamental de la familia de operadores propuestos es la reducción del número de rutas de la solución actual. Para ello, combinando la intuición de los “grupos de expulsión” con aspectos de proximidad entre los clientes y los “centros de gravedad” de las rutas, se propone un esquema básico para el operador de mejora cuya estructura base se muestra en el **algoritmo 4-2**, y puede resumirse en los siguientes puntos:

- En primer lugar (línea 2) se *inicializa el “grupo de expulsión” (ejection pool)*. El “grupo de expulsión” lo componen los clientes que se eliminan de su ubicación original en la solución actual, y que posteriormente serán reinsertados de nuevo para configurar la nueva solución. Este proceso de inicialización es el que aporta las diferencias fundamentales a los tres operadores propuestos en las siguientes secciones. En concreto, los criterios de inicialización del “grupo de expulsión” tomados como base en el diseño inicial y la experimentación son:
 - Los clientes más alejados del centro de gravedad de su ruta actual.
 - Los clientes pertenecientes a las rutas más pequeñas.
 - Los clientes pertenecientes a una ruta seleccionada aleatoriamente.

Lógicamente podrían definirse más criterios de inicialización, pero los tres criterios planteados son sencillos e intuitivos, y permiten obtener unos buenos resultados.

- Una vez inicializado el “grupo de expulsión”, se pasaría a la fase de *eliminación de las rutas vacías* (línea 3). En esta fase, se analizan las rutas de la solución actual, y se eliminan aquellas que se han quedado sin clientes después de inicializar el “grupo de expulsión”. Dependiendo del número de clientes que se hayan eliminado de cada ruta, en esta fase podría ocurrir que ninguna ruta se haya quedado vacía. En ese caso, el operador no tendrá capacidad de eliminar ninguna ruta, pero aun así, podría mejorar la solución actual mediante la reducción de la distancia total recorrida (al igual que haría un operador de mejora intraruta).

Algoritmo 4-2: Pseudocódigo del operador de reducción del número de rutas.

```

1. Entrada:  $Solución_{actual}$ ,  $optimizarRutas$  y  $reinserciónPorProximidad$ 
2.  $EjectionPool = inicializarGrupoExpulsión(Solución_{actual})$ 
3.  $Solución_{Nueva} = eliminarRutasVacías(Solución_{actual})$ 
4. IF ( $optimizarRutas$ ) THEN
5.    $optimizarRutas(Solución_{nueva})$ 
6. END IF
7. IF ( $reinserciónPorProximidad$ ) THEN
8.    $reinserter(EjectionPool, Solución_{nueva})$ 
9. END IF
10. IF ( $EjectionPool \neq \emptyset$ ) THEN
11.    $Solución_{Nueva} = HeuristicaConstructiva.reconstruir(EjectionPool, Solución_{nueva})$ 
12. END IF
13. IF ( $Solución_{Nueva}$  mejor que  $Solución_{actual}$ ) THEN
14.    $Solución_{actual} = Solución_{nueva}$ 
15. END IF
16. Salida: RETURN  $Solución_{actual}$ 

```

- Tras la fase de eliminación de rutas, y de manera opcional (definida en el proceso de inicialización del operador), se realiza un *proceso de optimización de la solución en su estado actual* (línea 5). De esta forma, se modifica la estructura original de las rutas tras extraer los nodos que han sido insertados en el “grupo de expulsión” aumentando así la capacidad de diversificación del operador.

En los operadores propuestos se sugiere el empleo del operador intraruta Or-opt, aunque podría utilizarse otro tipo de operador de mejora, incluso un operador de mejora interruta.

- Después de la optimización de las rutas en su estado actual, se continúa con la fase de *reinserción de clientes* (línea 8). Este proceso, que también puede aplicarse de forma opcional, procesa todos los clientes que están en el “grupo de expulsión”.

Cada uno de ellos intenta ser reinsertado en la ubicación que aporte un menor incremento en la distancia total de la recorrida. Este proceso analiza todas las rutas, y para cada una de ellas se hace uso de las listas de vecinos (NL^{PREV} y NL^{NEXT}) descritas en la sección 4.1.2 con objeto de acelerar el proceso. Al finalizar este proceso, podría ocurrir que alguno de los clientes no se haya reinsertado satisfactoriamente en ninguna de las rutas actuales.

- Como último paso del proceso, se realiza la fase de *reconstrucción final de la nueva solución* (línea 11). En esta fase, se utiliza una heurística de construcción paralela que toma como base las rutas en su estado actual y los clientes que todavía se encuentran en el “grupo de expulsión” (que pudieran ser todos los extraídos, si no se realiza la fase de reinserción de clientes). Después de invocar a la heurística de construcción paralela, la nueva solución ha sido generada por completo, y será devuelta siempre que mejore a la solución actual de acuerdo a la función objetivo jerárquica del VRP que contempla en primer lugar el número de rutas y posteriormente la distancia total recorrida (línea 13).

En este caso, el proceso de reconstrucción podría utilizar cualquier otra técnica para la reinserción de los clientes del “grupo de expulsión”, pero en el caso de los operadores propuestos se ha optado por el uso de la heurística propuesta por (Campbell y Savelsbergh 2004), que es especialmente interesante por su rapidez y sencillez.

Tal y como puede observarse en el **algoritmo 4-3**, el proceso de reconstrucción de la heurística paralela seleccionada comienza con un conjunto de clientes sin asignar y el conjunto de rutas iniciales (en este caso, las rutas de la nueva solución en su estado tras el proceso de reinserción). Posteriormente se realiza un proceso iterativo insertando en cada paso el cliente que aporte un mayor beneficio (que en este caso es el valor negativo del incremento de distancia derivado de la inserción). El proceso finaliza cuando todos los clientes no asignados han sido insertados satisfactoriamente en una ruta. Durante dicho proceso, podría ser necesario crear alguna ruta adicional; cuando un cliente no pueda ser insertado en ninguna de las rutas existentes.

Con el esquema descrito, se puede apreciar que este nuevo tipo de operador realiza un proceso más complejo que los operadores tradicionales de intercambio de nodos y arcos vistos en la sección 3.2.1, con lo cual el tiempo de aplicación será mayor, pero por otro lado, la técnica propuesta posee una buena capacidad para la reducción del número de rutas de la solución actual.

Por otro lado, si bien cierto que el operador propuesto es similar a las heurísticas de reducción del número de rutas revisadas en la sección 3.2.2; en este caso, el “grupo de reinserción” se inicializa al comenzar el proceso y posteriormente su tamaño no crece de nuevo en ningún momento. Es decir, simplemente se extraen clientes de las rutas

una única vez, y no varias veces de forma encadenada, como sugiere la técnica de “cadenas de expulsión” (*ejection chains*) en la que se encadenan secuencias de extracción e inserción de tal forma que para reinsertar los nodos extraídos pueden generarse nuevas “expulsiones” de nodos de su ubicación original.

Finalmente, otra diferencia respecto a técnicas de características similares es el hecho de utilizar un proceso simple para la reconstrucción final de la nueva solución por medio de una heurística constructiva paralela, porque otras alternativas parecidas como la heurística de Nagata y Bräysy (Nagata y Bräysy 2009a) utilizan costosos procesos que se basan en la aceptación de soluciones parciales no válidas (que incumplen alguna restricción de capacidad o ventana temporal) y su posterior “reparación” mediante técnicas de búsqueda local.

Algoritmo 4-3: Pseudocódigo de la heurística de construcción paralela de Campbell y Savelsbergh

```

1.   $N = \text{clientes no asignados}$ 
2.   $R = \text{rutas iniciales}$ 
3.  WHILE  $N \neq \emptyset$  DO
4.     $p^* = -\infty$ 
5.    FOR  $j \in N$  DO                                //FOR-1
6.      FOR  $r \in R$  DO                                //FOR-2
7.        FOR  $(i-1, i) \in r$  DO                        //FOR-3
8.          IF ( $\text{inserciónFactible}(i, j) \ \&\& \ \text{beneficio}(i, j) > P^*$ ) THEN
9.             $r^* = r$ 
10.            $i^* = i$ 
11.            $j^* = j$ 
12.            $p^* = \text{beneficio}(i, j)$ 
13.         END IF
14.       END FOR                                //FOR-3
15.     END FOR                                //FOR-2
16.   END FOR                                //FOR-1
17.    $\text{insertar}(r^*, i^*, j^*)$ 
18.    $N = N \setminus j^*$ 
19. END WHILE
20. RETURN  $R$ 

```

Con todo lo anterior, la nueva familia de operadores de reducción del número de rutas propuesto representa un buen complemento a los vecindarios tradicionales de intercambio de nodos y arcos puesto que combinan la capacidad de reducción del número de rutas, con un proceso sencillo que permite controlar el tiempo de ejecución global. De esta forma, como se analizará con detalle en el capítulo 5, los operadores propuestos permiten la validación satisfactoria de la segunda hipótesis propuesta en la presente tesis doctoral:

HIPÓTESIS 2: *Es posible definir un nuevo operador de mejora para problemas de asignación de rutas a vehículos con restricción fuerte de ventanas de tiempo que facilite la reducción del número de rutas de una solución.*

En las siguientes secciones se describen los tres operadores de reducción de número de rutas propuestos para la validación de la segunda hipótesis de la presente tesis doctoral.

4.2.3 Operador de reasignación de clientes alejados (RcR-opt)

El primero de los operadores propuesto se denomina “Operador de reasignación de clientes alejados” (*Remote Customers Reallocation Operator - RcR-opt*). Este operador intenta mejorar la solución actual a partir de la eliminación de cada una de las rutas de los n clientes que están más alejados del “centro de gravedad de la ruta”. El proceso afecta a todas las rutas, e intenta reducir la distancia de todas ellas, pudiendo dejar alguna de las rutas vacías, si n es igual o superior al número de clientes que posee una determinada ruta.

La **figura 4-3** ilustra gráficamente el comportamiento del operador aplicado a una solución inicial con tres rutas, siendo el número de clientes eliminados $n = 1$. Como se puede observar, se elimina de cada una de las rutas el cliente más alejado, y posteriormente los clientes extraídos (que forman parte del “grupo de expulsión”) se reinsertan en el resto de rutas. En este ejemplo, como el número de clientes extraídos de cada ruta es menor al número de clientes de cada una de las rutas, no se elimina ninguna ruta; pero si se aplicase de nuevo el operador sobre la “nueva solución” con $n = 4$, podría eliminarse la ruta de la izquierda.

Este operador es una particularización del operador de reducción del número de rutas definido en la sección 4.2.2 mediante el **algoritmo 4-2** con una modificación que afecta únicamente a la fase de fase de inicialización del “grupo de expulsión”:

- El “grupo de expulsión” se inicializa con los n clientes más alejados del “centro de gravedad” de cada ruta.

Para lograr que después de inicializar el “grupo de expulsión” una ruta se quede vacía, ha de cumplirse la siguiente desigualdad: $n \geq |V_r| - 2$ siendo $|V_r|$ en número de clientes que forman parte de una ruta, incluidas las dos instancias del almacén central.

A parte de la particularización en cuanto a la fase de inicialización del “grupo de expulsión”, este operador no posee ninguna peculiaridad adicional, pudiendo definirse hasta cuatro variantes principales en función de que se realice o no la optimización de las rutas después de la extracción y el proceso de inserción en la ruta más cercana.

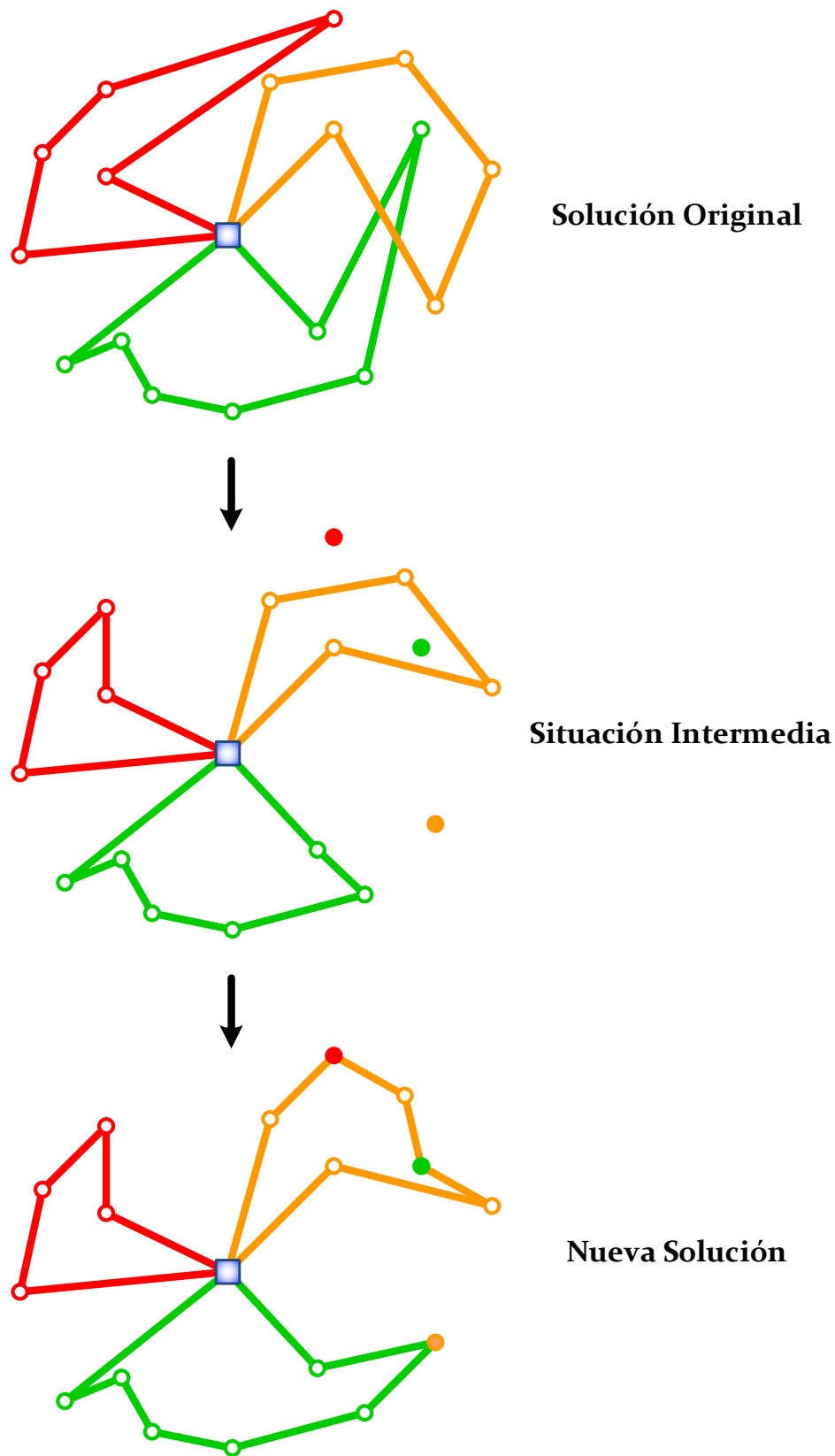


Figura 4-3: Funcionamiento del operador de reasignación de clientes alejados.

De esta forma, las cuatro variantes para este operador se denominan:

- *Basic Remote Customers Reallocation Operator* (**no** incluye ni optimización de las rutas ni reinserción en la ruta más cercana).
- *Full Remote Customers Reallocation Operator* (**si** incluye optimización de las rutas y reinserción en la ruta más cercana).
- *Customers Reallocation Operator with Route Optimization* (**no** incluye optimización de las rutas pero **no** incluye reinserción en la ruta más cercana).
- *Remote Customers Reallocation Operator with Nearest Reinsertion* (**no** incluye optimización de las rutas pero **si** incluye reinserción en la ruta más cercana).

Este operador nace con la motivación de mejorar la solución a partir de la extracción de los clientes más alejados respecto al “centro de gravedad” de una ruta. Esto conduciría al diseño de un operador que simplemente extraiga un número pequeño de nodos de cada una de las rutas, o sólo aquellos cuya distancia respecto al “centro de gravedad de la ruta” sea superior a la media.

No obstante, el planteamiento de este operador es más ambicioso, porque combina la mejora en distancia recorrida que puede proporcionar la reubicación de los clientes alejados con la reducción del número de rutas. Por ese motivo, el operador finalmente propuesto define el valor n (clientes a eliminar de cada una de las rutas) como el tamaño de la ruta más pequeña. De esta forma, una invocación al operador podría llegar a eliminar más de una ruta; si hay varias rutas cuyo tamaño sea igual a la ruta más pequeña.

4.2.4 Operador de eliminación de rutas pequeñas (SrE-opt)

El segundo de los operadores propuestos se denomina “Operador de eliminación de rutas pequeñas” (*Small Routes Elimination Operator - SrE-opt*). Este operador, al igual que el resto de los operadores propuestos intenta mejorar la solución actual a partir de la eliminación de alguna de las rutas existentes. Para ello intenta eliminar la/s ruta/s más pequeña/s, con la esperanza de que los clientes pertenecientes a esa/s ruta/s puedan ser reinsertados en el resto de rutas de la solución actual.

La **figura 4-4** ilustra gráficamente el comportamiento del operador aplicado a una solución inicial con cinco rutas, siendo el tamaño de la ruta más pequeña igual a uno. Como se puede observar, en este caso se eliminan las dos rutas más pequeñas (insertando sus clientes en el “grupo de expulsión”). Posteriormente, los clientes de las rutas eliminadas se reinsertan en el resto de rutas. En este ejemplo concreto, puede apreciarse que el operador consigue reducir el número inicial de rutas en dos unidades.

Este operador es otra particularización del operador de reducción del número de rutas definido en la sección 4.2.2 mediante el **algoritmo 4-2** con una modificación que afecta únicamente a la fase de inicialización del “grupo de expulsión”:

- El “grupo de expulsión” se inicializa con los clientes pertenecientes a las rutas más pequeñas de la solución actual. Para ello, se calcula el tamaño de la ruta más pequeña y se insertan en el “grupo de expulsión” los clientes de la/s ruta/s con menor tamaño.

Este operador no posee ninguna particularidad adicional, a parte de la modificación de la fase de inicialización del “grupo de expulsión”. Además, también pueden definirse cuatro variantes para este operador:

- *Basic Small Routes Elimination Operator* (**no** incluye ni optimización de las rutas ni reinserción en la ruta más cercana).
- *Full Small Routes Elimination Operator* (**si** incluye optimización de las rutas y reinserción en la ruta más cercana).
- *Small Routes Elimination Operator with Route Optimization* (**si** incluye optimización de las rutas pero **no** incluye reinserción en la ruta más cercana).
- *Small Routes Elimination Operator with Nearest Reinsertion* (**no** incluye optimización de las rutas pero **si** incluye reinserción en la ruta más cercana).

Este operador nace con la motivación de mejorar la solución a partir de la eliminación de las rutas más pequeñas, con la esperanza de que si una ruta tiene pocos clientes, podría ser “relativamente” fácil reinsertar sus clientes entre el resto de rutas de la solución actual. Al igual que en el caso del operador anterior, esta variante del operador de reducción del número de rutas podría eliminar más de una ruta, si existen varias rutas pequeñas y con el mismo tamaño en una solución.

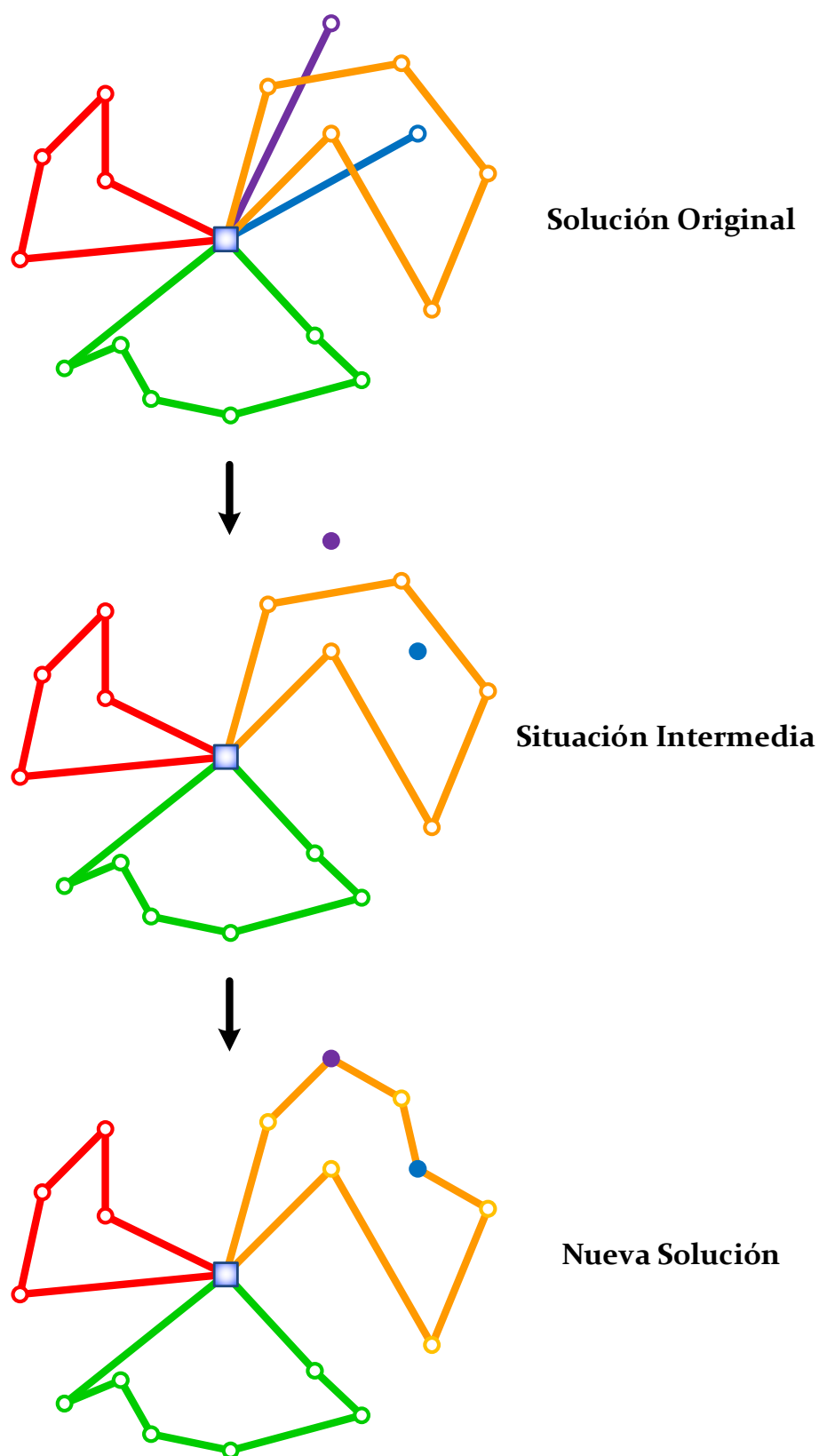


Figura 4-4: Funcionamiento del operador de eliminación de rutas pequeñas.

4.2.5 Operador aleatorio de eliminación de rutas (RrE-opt)

El tercer y último operador propuesto se denomina “Operador de eliminación aleatoria de rutas (*Random Route Elimination Operator - RrE-opt*)”. Como su propio nombre sugiere, este operador se basa en la eliminación aleatoria de una ruta, con la esperanza de que los clientes que formaban dicha ruta puedan ser reinsertados satisfactoriamente en el resto de rutas de la solución actual.

Este operador es la última particularización propuesta, y al igual que las anteriores, la única diferencia que aporta al **algoritmo 4-2** definido en la sección 4.2.2, es una variación en la fase de inicialización del “grupo de expulsión”:

- El “grupo de expulsión” se inicializa con los clientes pertenecientes a una ruta que es seleccionada al azar entre las que forman parte de la solución actual.

Al igual que en los dos casos anteriores, también se definen cuatro variantes para este operador:

- *Basic Random Route Elimination Operator* (**no** incluye ni optimización de las rutas ni reinsertión en la ruta más cercana).
- *Full Random Route Elimination Operator* (**si** incluye optimización de las rutas y reinsertión en la ruta más cercana).
- *Random Route Elimination Operator with Route Optimization* (**si** incluye optimización de las rutas pero **no** incluye reinsertión en la ruta más cercana).
- *Random Route Elimination Operator with Nearest Reinsertion* (**no** incluye optimización de las rutas pero **si** incluye reinsertión en la ruta más cercana).

El propósito de este operador es la introducción de un comportamiento aleatorio, cuyo efecto está ampliamente contrastado en la mayor parte de las técnicas heurísticas y metaheurísticas aplicadas a la resolución de cualquier problema de optimización combinatoria, siendo la heurística de Nagata y Bräysy (Nagata y Bräysy 2009a), la cual se basa en la reducción del número de rutas a partir de la eliminación de una ruta seleccionada de forma aleatoria, un ejemplo claro.

4.3 Entorno de visualización y simulación de problemas de tipo VRP

Para finalizar este capítulo en el que se presentan las aportaciones y resultados principales de la investigación realizada en la presente tesis doctoral, se describe en

esta sección un entorno gráfico que permite la visualización y simulación de las soluciones a problemas de asignación de rutas a vehículos.

El desarrollo de este entorno gráfico viene motivado por el hecho de realizar un contraste “visual” de las particularidades de una determinada instancia del problema y sus soluciones (que lógicamente podrían ser varias). De esta forma, se puede observar con claridad la naturaleza de la ubicación de los clientes en torno al almacén central (agrupada o aleatoria), así como el aspecto que tiene cada una de rutas que configuran la solución de una determinada instancia del problema, si está más o menos “ordenada” o si existen cruces entre segmentos, lo que indica que la distancia recorrida por la ruta no es del todo óptima. Además, permite la rápida identificación rutas en las que existe algún cliente que claramente está alejado del resto de clientes que configuran la ruta. Esta última situación ha representado sin duda una de las principales motivaciones para abordar el diseño de los operadores de mejora propuestos en el presente trabajo de tesis doctoral.

Para la resolución de una instancia de un problema no es necesario ninguna representación visual, puesto que todas las validaciones en cuanto al cumplimiento de las restricciones y el cálculo de la función objetivo son procesos que se pueden realizar sin el soporte de ninguna representación espacial. No obstante, el autor de la presente tesis doctoral considera importante tener una representación visual del problema y sus soluciones, puesto que facilita la comprensión de la naturaleza del problema y el análisis de las estrategias más adecuadas para la mejora u optimización de una determinada solución. De hecho, esta representación visual representó el punto de partida en el planteamiento de estrategias de mejora alternativas a las predominantes en la literatura especializada, intentando aplicar alguna de las nociones intuitivas que cualquier persona podría utilizar para mejorar la solución de un determinado problema.

El entorno de visualización y simulación es una aplicación de escritorio sencilla en cuyo diseño ha primado la capacidad para representar la información que configura la instancia de un problema y su solución. Dicha información se representa en forma tabular y de forma gráfica, existiendo también un panel para simular el recorrido que realizan los vehículos asociados a cada una de las rutas que forman parte de la solución a un problema.

En la **figura 4-5** se muestra la vista principal del entorno de visualización y simulación con la información de las soluciones a todas las instancias de problemas del juego de ensayo de (SINTEF 2015). Cada problema se representa en una pestaña, y cada una de ellas se divide en dos bloques: la parte izquierda muestra un resumen de las rutas, con el detalle de la distancia, el número de clientes, la demanda y la duración de cada una de las rutas, así como el total y valores medios de cada uno de los parámetros; por otro lado, en la parte central se muestra la solución de forma visual, diferenciando por

colores cada una de las rutas. Además, se pueden filtrar los vehículos que se muestran en cada momento, para focalizar la visualización en una única ruta o en un subconjunto de las mismas.

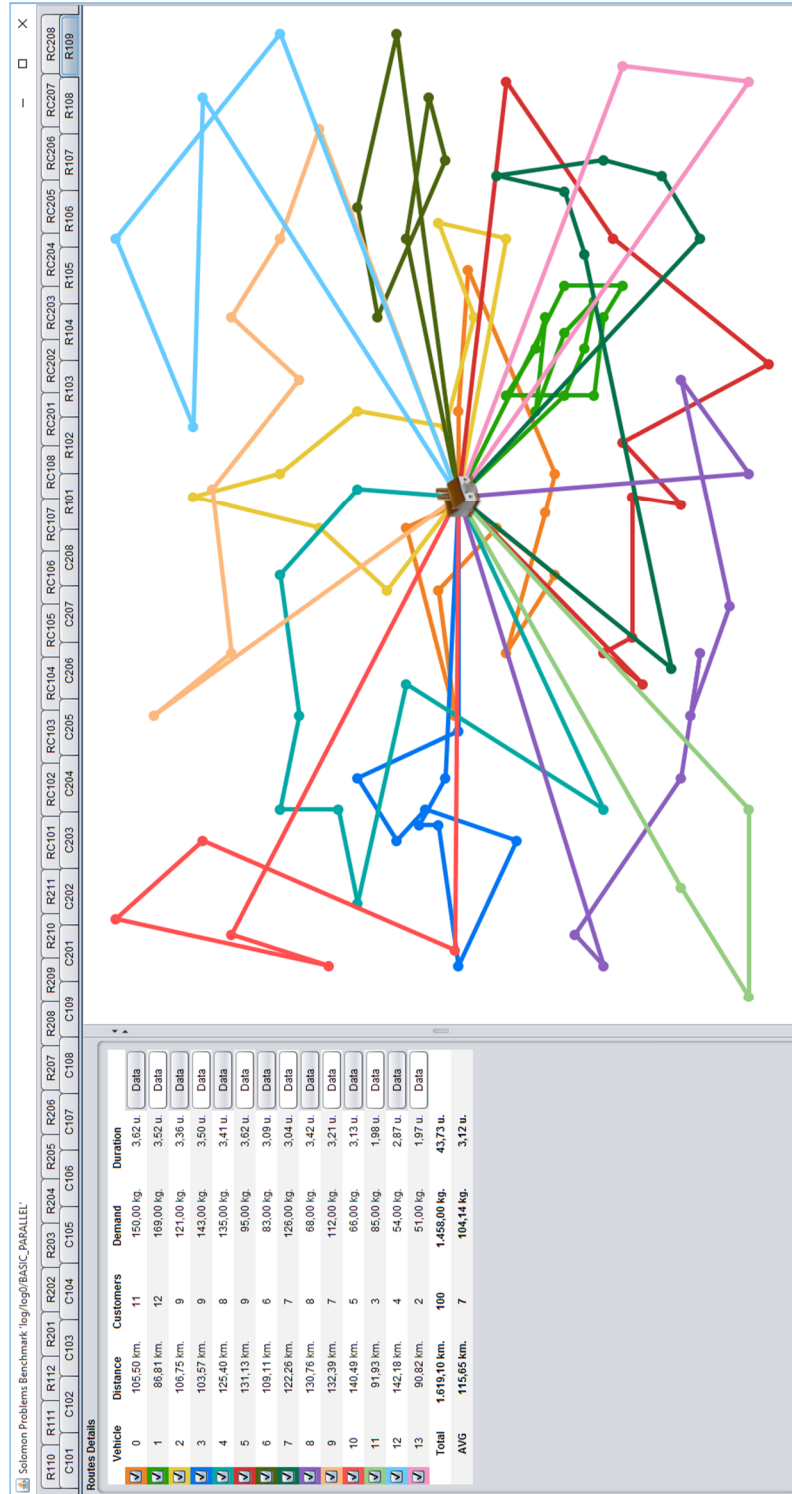


Figura 4-5: Vista principal del entorno de visualización y simulación.

Con el fin de analizar con más detalle una ruta concreta, se puede focalizar la visualización en una única ruta, mostrando su información en forma tabular, tal y como se observa en la **figura 4-6**; o en forma de simulación del recorrido que realiza el vehículo a medida que evoluciona el tiempo. Esta última vista de la ruta es especialmente interesante para detectar “problemas” que posee la ruta en cuanto a estructura, lo que ayuda a detectar las estrategias de mejora que pudieran optimizar la ruta.

Position	Customer	Demand	Service	Travel	Earliest Beginning	Latest Beginning	Arrival	Beginning	Waiting	Window Size
0	0	0,00 kg.	0,00	0,00 km.	0,00	1.236,00	0,00	0,00	0,00	1.236,00
1	52	10,00 kg.	90,00	21,21 km.	0,00	1.124,00	21,21	21,21	0,00	1.124,00
2	42	20,00 kg.	90,00	8,54 km.	68,00	149,00	119,76	119,76	0,00	81,00
3	41	10,00 kg.	90,00	2,00 km.	166,00	235,00	211,76	211,76	0,00	69,00
4	40	10,00 kg.	90,00	2,00 km.	264,00	321,00	303,76	303,76	0,00	57,00
5	44	10,00 kg.	90,00	3,00 km.	359,00	412,00	396,76	396,76	0,00	53,00
6	45	10,00 kg.	90,00	2,00 km.	541,00	600,00	488,76	541,00	52,24	59,00
7	48	10,00 kg.	90,00	2,00 km.	0,00	1.122,00	633,00	633,00	0,00	1.122,00
8	51	10,00 kg.	90,00	3,00 km.	0,00	1.121,00	726,00	726,00	0,00	1.121,00
9	50	10,00 kg.	90,00	2,24 km.	0,00	1.123,00	818,24	818,24	0,00	1.123,00
10	46	30,00 kg.	90,00	4,00 km.	0,00	1.125,00	912,24	912,24	0,00	1.125,00
11	49	10,00 kg.	90,00	3,61 km.	1.001,00	1.066,00	1.005,84	1.005,84	0,00	65,00
12	47	10,00 kg.	90,00	2,00 km.	0,00	1.127,00	1.097,84	1.097,84	0,00	1.127,00
13	0	0,00 kg.	0,00	18,03 km.	0,00	1.236,00	0,00	1.115,87	0,00	1.236,00

View Simulation

Figura 4-6: Vista tabular de la información de una ruta.

A modo de ejemplo, en la **figura 4-7** se muestra la simulación de una ruta en la que el vehículo se desplaza en primer lugar al cliente etiquetado como 51, a continuación se desplaza al cliente 41 y posteriormente al cliente 40. A simple vista se puede intuir que podría conseguirse una mejora en la distancia total recorrida por el vehículo si después de visitar el cliente 51 se visitase el cliente 49. Esto sería viable en la variante clásica del VRP, pero en el caso del problema de tipo VRPTW quizá la ventana temporal del cliente 41 (y los clientes que le suceden en la ruta) finalice antes que la ventana temporal del cliente 49, lo que imposibilita la ordenación intuitiva basada únicamente en distancias.

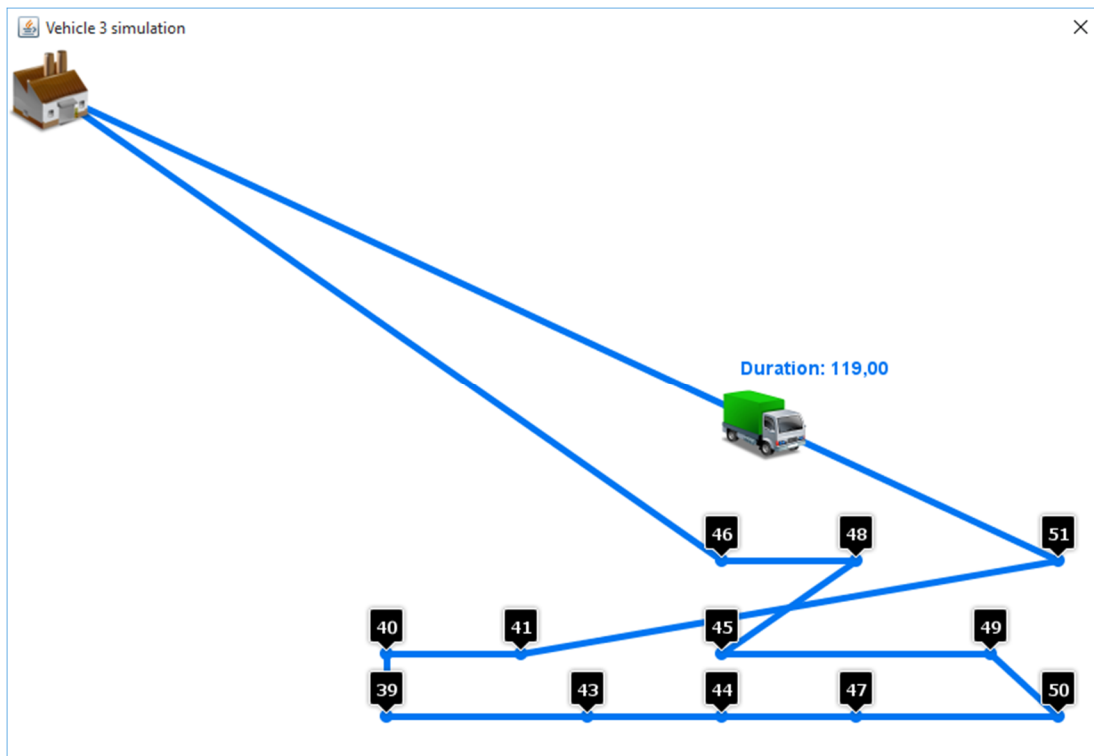


Figura 4-7: Vista de la simulación de una ruta.

Por último, para finalizar esta sección, simplemente destacar que el entorno de visualización y simulación se inicializa a partir de documentos en formato XML que pueden contener información de la solución a una instancia del problema o únicamente la configuración del problema (para analizar la distribución de los clientes en). A modo de ejemplo, en el Apéndice C se muestra el fragmento de un documento XML con la solución a una instancia de problema del juego de ensayo de Solomon.

5

Experimentación y validación de resultados

Para comprobar que tanto el aporte realizado por la nueva heurística de construcción, como los operadores de mejora, permiten la obtención de unos resultados competitivos, se ha llevado a cabo un proceso de experimentación utilizando como referencia las instancias de problemas del juego de ensayo más representativo de problemas de tipo VRPTW: el juego de ensayo de Solomon (SINTEF 2015).

La experimentación realizada en el presente trabajo de tesis doctoral tiene por objeto la validación de las dos hipótesis definidas en el capítulo inicial. Cada una de las hipótesis será validada de forma independiente mediante dos procesos de experimentación. Para ello, se contrastarán cada uno de los dos principales aportes desarrollados en la presente tesis doctoral con técnicas de características similares, siendo los resultados obtenidos en dichos contrastes los que soporten la validez de cada una de las hipótesis planteadas. En concreto, los dos procesos de experimentación realizados serán los siguientes:

- Por un lado, se compararán los resultados obtenidos por la nueva heurística de construcción con tres de las heurísticas de construcción más representativas en relación a los resultados obtenidos y su frecuencia de uso en la literatura. En este caso, la comparación será directa, es decir, se compararán los resultados obtenidos por las cuatro técnicas: la propuesta y las tres referentes de la literatura. Como las cuatro técnicas comparadas son deterministas la comparación se hará en base a la diferencia entre los resultados obtenidos, sin hacer uso de ninguna prueba estadística.

- Por otro lado, para validar los operadores de mejora basados en la reducción del número de rutas, se comparará el rendimiento de los operadores propuestos con el obtenido por los operadores de mejora más referenciados en la literatura. En este caso, la comparación se realizará integrando los diferentes operadores (los nuevos y los tradicionales) en un proceso de búsqueda heurístico basado en una búsqueda paralela con vecindario variable (*Parallel Variable Neighborhood Search*). En esta ocasión, como la técnica utilizada para la validación no es determinista, las comparaciones se realizarán por medio del análisis estadístico de los resultados obtenidos.

Lo que resta del capítulo se ha organizado en cuatro secciones: la sección 5.1 introduce cuestiones relacionadas con la comparación de técnicas de resolución para problemas de optimización combinatoria; la sección 5.2 describe en detalle las cuestiones relativas a la configuración de la fase de experimentación; la sección 5.3 presenta los resultados; y finalmente la sección 5.4 incorpora las conclusiones extraídas tras analizar los resultados obtenidos.

5.1 Comparación de técnicas de resolución de problemas de optimización combinatoria

Antes de exponer los aspectos relativos a la experimentación y la validación de los resultados obtenidos, se ha decidido incorporar en este punto información relativa a criterios de comparación, además de una serie de buenas prácticas que debieran contemplarse a la hora de implementar y presentar resultados de nuevas técnicas de resolución de problemas de optimización combinatoria. Esta sección pretende enfatizar la importancia que tiene la manera de presentar tanto los trabajos realizados como las comparaciones y los resultados obtenidos, para reducir la ambigüedad y facilitar la reproducción de los experimentos.

5.1.1 Criterios generales de comparación de técnicas de resolución del VRP

La evaluación de un método de resolución heurístico o meta-heurístico se basa en el análisis de varios parámetros que permiten medir el rendimiento de dicho método. Ejemplos de dichos parámetros son: el tiempo de ejecución, la calidad de la solución obtenida, la facilidad y sencillez de la implementación, la robustez y la flexibilidad (Barr, y otros 1995) y (Cordeau, y otros 2002).

La *flexibilidad* hace referencia a la facilidad del método para adaptarse a los cambios que se produzcan en el problema a resolver; ya sea en las restricciones o en la función

objetivo. Esto es especialmente importante por el hecho de que los métodos heurísticos y meta-heurísticos se diseñan para resolver problemas del mundo real donde los cambios o modificaciones de la tarea a resolver son una constante muy habitual. Por lo tanto, una técnica flexible podría ser aplicada a diferentes tipos de problemas.

La *robustez* está relacionada con la capacidad de obtener buenas soluciones todas las veces que se ejecuta el método sobre una misma instancia de un problema. La robustez de un método no determinista viene dada directamente por la desviación típica de la calidad de las soluciones (robustez de optimización) y del tiempo de ejecución (robustez temporal). Teniendo en cuenta esto, menor desviación típica supone que un método es más robusto. El carácter no determinista de las técnicas heurísticas y metaheurísticas (debido a la incorporación de parámetros de carácter aleatorio y/o probabilístico) provoca que el resultado de varias ejecuciones del mismo método sobre una instancia de un problema, no sea siempre el mismo. Pese a que habitualmente en la literatura se reportan únicamente los mejores resultados obtenidos como medida del rendimiento de los métodos no deterministas, esto puede crear una imagen distorsionada del rendimiento real. Lo adecuado sería realizar un promedio de los resultados obtenidos en varias ejecuciones, reportando también los peores valores, e incluso la desviación estándar. De esta forma, se puede obtener una visión adecuada del rendimiento del método (Cordeau, y otros 2002).

El *tiempo* que necesita un método para obtener buenas soluciones es un criterio importante a la hora de elegir una técnica de resolución, pero para su cálculo sería adecuado realizar varias ejecuciones y obtener la media. Cuanto mayor sea el número de ejecuciones, más fiable será el valor de tiempo reportado.

Al igual que el tiempo de ejecución, la *calidad de la solución* es otro criterio importante. De hecho, muchos de los autores consideran éste como el criterio “más importante”. En relación a la calidad de la solución, normalmente se utiliza como referencia lo cerca que está la solución obtenida de la solución óptima o la mejor solución conocida para una determinada instancia de un problema, aunque también es habitual realizar rankings o análisis estadísticos que van más allá de la diferencia absoluta, analizando si dicha diferencia es significativa o no. Normalmente, existe una relación directa entre el tiempo de ejecución y la calidad de la solución; a mayor tiempo de ejecución, mayor calidad de la solución, y viceversa. La gran dificultad que existe en este binomio es encontrar métodos de resolución que encuentren soluciones de gran calidad en un tiempo de ejecución reducido.

Una vez definidos los principales parámetros que permiten analizar el rendimiento de un método heurístico o meta-heurístico, es importante revisar la forma de obtener o recopilar dichos parámetros. La técnica más utilizada para la evaluación del rendimiento de un método heurístico o meta-heurístico es el análisis empírico. Este análisis empírico tiene por objeto la recopilación de los parámetros de referencia a

partir de la ejecución del método heurístico sobre un conjunto de instancias de un problema que sea suficientemente representativo de la realidad del tipo de problema a resolver. Tras recopilar los parámetros, mediante la hipótesis de inducción, se pueden generalizar los resultados obtenidos a la globalidad de instancias del tipo de problema que se quiere resolver. A pesar de que la mayor parte de los autores reportan los valores de los parámetros mencionados anteriormente, existen muchas dificultades para comparar métodos heurísticos y meta-heurísticos de resolución de problemas de optimización combinatoria. Estas dificultades hacen que, en algunas situaciones, sea difícil realizar una comparación justa entre diferentes técnicas.

La primera dificultad existente a la hora de comparar diferentes técnicas es la disparidad en cuanto al hardware utilizado para realizar el análisis; este hecho afecta directamente al tiempo de ejecución. En este sentido (Dongarra 1998) propone una serie de factores correctivos para ajustar y uniformizar los tiempos de ejecución pese a que existan diferencias en el hardware utilizado.

Otra cuestión controvertida son las diferencias derivadas de la codificación, tanto del lenguaje de programación, como las precisión de los redondeos de los valores numéricos, como del estilo de codificación o la cantidad de recursos y tiempo empleados por una determinada implementación (Hooker 1995).

Por último, una dificultad adicional que es específica de los métodos heurísticos y meta-heurísticos de resolución de problemas de tipo VRPTW es el hecho de que normalmente sólo se reportan los mejores resultados obtenidos. Además, en muchos casos, los autores no reportan el número de ejecuciones o el tiempo necesario para obtener los resultados reportados. En estos casos, es imposible sacar conclusiones consistentes acerca de la eficiencia de los métodos, o comparar unos métodos con otros. Para poder obtener conclusiones apropiadas, es necesario conocer el número de ejecuciones y el tiempo necesario para obtener los resultados reportados. Lógicamente, esto no es posible si los autores no han reportado la información necesaria. Por este motivo, en muchos casos, si se desea realizar una comparativa *formal*, es necesario implementar y probar los diferentes métodos de nuevo utilizando criterios de codificación y validación uniformes.

Todas las cuestiones esbozadas en esta sección han dado pie a que el autor, junto con un grupo de compañeros que trabajan en el mismo ámbito de investigación, haya elaborado un breve decálogo de buenas prácticas que debieran ser contempladas a la hora de implementar y presentar resultados de técnicas de resolución de problemas de optimización combinatoria (Osaba y Carballedo 2012) y (Osaba, y otros 2014). La siguiente sección incorpora un resumen de dicho trabajo.

5.1.2 Buenas prácticas para la implementación y comparación de técnicas de resolución de problemas de optimización combinatoria

Una vez presentados los aspectos clave que deben ser contemplados a la hora de comparar técnicas de resolución para problema de optimización combinatoria, y tras las dificultades encontradas tanto por el autor de la presente tesis doctoral como por sus compañeros a la hora de replicar o comparar sus trabajos con otros trabajos existentes en la literatura, se muestra un pequeño decálogo de buenas prácticas que debieran ser contempladas a la hora de implementar y comparar técnicas de resolución de problemas de optimización combinatoria. Dicho decálogo se enmarca en los trabajos realizados en (Osaba y Carballedo 2012) y (Osaba, y otros 2014), que pueden ser consultados para ampliar la información aquí descrita y revisar ejemplos de malas prácticas que pueden ser encontrados en la literatura.

Las buenas prácticas propuestas se agrupan en dos categorías: por un lado, las buenas prácticas relacionadas con la implementación de nuevas técnicas de resolución de problemas de optimización combinatoria; y por otro lado, las buenas prácticas relacionadas con la presentación de resultados con objeto de realizar una comparación objetiva y justa entre diferentes técnicas.

El primer grupo de buenas prácticas, forman parte de una pequeña metodología que marca una serie de pasos que debieran contemplarse a la hora de implementar una nueva técnica:

- Especificar detalladamente las características y restricciones del problema concretando si dichas restricciones son estrictas (*hard*), que no pueden incumplirse en ningún momento; o ligeras (*soft*), que por el contrario pueden ser incumplidas en algún momento (contemplando una penalización en la función objetivo), tanto durante el proceso de resolución, como al generar una solución final.
- Describir con claridad la función objetivo utilizada, identificando los diferentes objetivos que deben satisfacerse (si es que existen varios), así como la jerarquización de los mismos.
- Utilizar un nombre claro y representativo de la técnica concreta evitando ambigüedades que puedan crear una idea equivocada al lector. De la misma forma, el resumen (*abstract*) que acompañe a la publicación del trabajo realizado debe ser concreto, claro y conciso.
- Describir con el mayor detalle posible cada uno de los aspectos relacionados con la implementación, especialmente en cuanto a parámetros de la técnica y

operadores/vecindarios indicando claramente si el trabajo es complemente original o está basado en una adaptación de algún trabajo previo existente. Esta descripción debe ser lo más detallada posible para permitir replicar los resultados obtenidos, puesto que este hecho es sin duda el que más dificultades y pérdida de tiempo genera cuando se pretende generar un nuevo aporte que actualice el conocimiento existente.

Después de presentar brevemente las pautas o buenas prácticas que debieran contemplarse a la hora de implementar y describir una nueva técnica de resolución de problemas de optimización combinatoria, se incorporan algunas cuestiones que deben cuidarse especialmente cuando se presentan los resultados obtenidos y éstos son comparados con los reportados por otras técnicas existentes. Dichas pautas se describen a continuación:

- Si el problema lo permite, las pruebas debieran realizarse utilizando instancias de problemas conocidas pertenecientes a un juego de ensayo (*benchmark*) que sea referente para la tipología de problema tratado. Todas las instancias deben ser referenciadas de manera adecuada; y si fuese necesario la definición de alguna instancia nueva (que puede ser perfectamente válido en algunos contextos) esta ha de ser descrita de forma detallada.
- Un aspecto de importancia capital es el hecho de mostrar los tiempos de ejecución así como la unidad temporal, junto con una explicación de las características de los recursos computacionales utilizados para llevar a cabo la experimentación.
- Además del tiempo de ejecución, con objeto de evaluar la capacidad de convergencia que posee la nueva técnica, es aconsejable indicar el número de iteraciones o número de evaluaciones de la función objetivo que realiza la técnica para obtener los resultados en cada ejecución.
- Es recomendable mostrar la mayor cantidad posible de datos relacionados con los resultados, como por ejemplo: número de ejecuciones, la *mejor* y la *peor* solución, la media, la desviación estándar y el tiempo de ejecución; de esta forma, la comparación entre diferentes técnicas es más fiable.
- Por último, para que las comparaciones entre diferentes técnicas sean estrictas y objetivas, se recomienda utilizar alguna prueba estadística como podrían ser la prueba *t* de *Student*, el test normal *z*, el test no paramétrico de Friedman (Friedman 1937) y (Friedman 1940) o el test post-hoc de Holm (Holm 1979).

En las siguientes secciones, cuando se presente tanto la experimentación como la validación y comparación de los resultados obtenidos, se hará un especial énfasis en las buenas prácticas presentadas en esta sección.

5.1.3 Conjuntos de instancias de problemas de tipo VRPTW

En esta sección se describen los principales conjuntos de instancias de problemas de tipo VRPTW que pueden encontrarse en la literatura, y que sirven como referencia a la hora de comparar y demostrar las bondades de las técnicas de resolución de esta tipología de problema. Dichos juegos de ensayo, que reciben el nombre de los científicos que los definieron en su día, son: el juego de ensayo de Solomon (*Solomon's Benchmark*) y el juego de ensayo de Gehring y Homberger (*Gehring & Homberger's benchmark*).

Además de diferentes heurísticas para la construcción inicial de rutas, en su trabajo de 1987 Solomon (Solomon 1987) definió un conjunto de instancias de problemas de tipo VRPTW que se han convertido en un referente para la evaluación de técnicas de resolución de problemas de tipo VRPTW. Dicho conjunto de problemas se conoce con el nombre “juego de ensayo de Solomon” (*Solomon's Benchmark*).

Este conjunto de problemas está compuesto por 56 instancias cada una de ellas con 100 clientes, un único almacén central, restricciones de capacidad, ventanas de tiempo, y un límite máximo en la duración de cada ruta. Los problemas están agrupados en seis clases/categorías de problemas denominados C₁, C₂, R₁, R₂, RC₁ y RC₂. La C indica que los clientes están agrupados formando clústeres o grupos, la R representa que los clientes están distribuidos de forma aleatoria; y por último, la RC indica que existe una mezcla entre clientes agrupados y distribuidos de forma aleatoria.

Cada una de las clases de problemas contiene entre 8 y 12 instancias de problemas. En todas las instancias de una misma clase los clientes poseen la misma ubicación y demanda, lo único que cambia de una instancia a otra son las ventanas de tiempo. En cuanto a la densidad de las ventanas de tiempo (porcentaje de clientes que poseen ventana de tiempo), existen instancias con 25%, 50%, 75% y 100% de clientes con ventanas de tiempo restrictivas; el resto de clientes tienen ventanas de tiempo iguales a la del almacén central por lo que no tienen una limitación temporal a la hora de ser asignados a una ruta.

A partir de las soluciones óptimas de las diferentes instancias, disponibles en (Solomon, 2005), se puede observar que las clases C₁, R₁ y RC₁ tienen un horizonte de planificación corto (los vehículos tienen poca capacidad y los límites de tiempo de las rutas son cortos), con lo que cada vehículo puede atender a pocos clientes. Por este motivo, para la resolución de cada una de las instancias de estas clases se necesitan entre 9 y 19 vehículos. Por otro lado, las clases C₂, R₂ y RC₂ poseen un horizonte de planificación más largo, estando en este caso el número de vehículos necesarios acotado entre 2 y 4.

El conjunto de problemas de Solomon está accesible on-line en varias ubicaciones. De entre todas ellas, las más interesantes son la página personal de profesor Solomon en la Universidad de Northeastern (Northeastern University 2005) y la página del centro de investigación SINTEF (SINTEF 2015). En ambas ubicaciones se pueden encontrar tanto las instancias de los problemas como las referencias a las técnicas que han conseguido obtener los mejores resultados. Lamentablemente estas referencias no se actualizan con frecuencia, y por ese motivo, no existen garantías de que los resultados reportados en dichas páginas sean los mejores conocidos a día de hoy. Una referencia un poco más actualizada, puede encontrarse en el trabajo de Vidal y otros (Vidal, Crainic, y otros 2014).

Además del *benchmark* de Solomon, Gehring y Homberger (*Gehring & Homberger's benchmark*) definieron un nuevo juego de ensayo de instancias de problemas de tipo VRPTW. En este caso, los problemas definidos son una extensión de los problemas de Solomon en los que se incrementa el número de clientes, creando instancias de 200, 400, 600, 800 y 1000 clientes. El incremento del número de clientes responde a la realidad de los problemas que surgen en el mundo real dónde los problemas de transporte y logística pueden superar el millar de clientes. El resto de elementos de las instancias de los problemas tanto las clases (C, R y RC), como las capacidades de los vehículos, ventanas de tiempo y límites de distancia por ruta, son similares a los propuestos por el juego de ensayo de Solomon. Este conjunto de problemas se conoce con el nombre de “instancias de gran escala” (*Large-Scale instances*) y son sin duda alguna la referencia para la evaluación de las técnicas metaheurísticas más avanzadas. Para este conjunto de problemas también existen referencias de las mejores soluciones conocidas (SINTEF 2015); en este caso, se pueden encontrar datos actualizados en mayo de 2015.

A la hora de evaluar la calidad de las soluciones obtenidas por una técnica de resolución para estos dos juegos de ensayo, debiera tenerse en cuenta la función objetivo jerárquica de tres niveles del VRPTW:

- El primer objetivo es minimizar el número de vehículos.
- Para el mismo número de vehículos, el siguiente objetivo es minimizar la distancia total recorrida.
- Por último, para la misma distancia recorrida, es preferible una solución con menor tiempo total de espera, es decir, el tiempo que tiene que esperar un vehículo en la ubicación de un cliente hasta que se inicie su ventana de tiempo.

Pese a que ésta debiera ser la función objetivo utilizada como referencia, en (SINTEF 2015), que recoge los mejores resultados conocidos para los dos juegos de ensayo descritos, únicamente se hace referencia a los dos primeros objetivos de la función

descrita en el párrafo anterior. Por ese motivo, en algunos casos, puede existir cierta controversia a la hora de presentar los resultados, tal y como se ha comentado en la sección 5.1.2.

5.2 Configuración de la experimentación

Tal y como lo enuncia el título, en esta sección se describen las características y aspectos fundamentales que se han tenido en cuenta a la hora de configurar la experimentación realizada.

5.2.1 Descripción de la función objetivo

La experimentación realizada como parte del presente trabajo de tesis doctoral se centra en la evaluación del aporte propuesto para el problema de tipo VRPTW. Por ese motivo, siguiendo las pautas en cuanto a buenas prácticas esbozadas en la sección 5.1.2, en esta sección se clarifica tanto la función objetivo, como las restricciones que se contemplarán en la fase de experimentación.

En este sentido, la experimentación se centra en la formulación clásica del VRPTW introducida en las secciones 2.2.2 y 2.3.10, recogiendo a continuación las restricciones que deben ser respetadas para la obtención de una solución:

- Se dispone de una flota fija (cuyo número no puede ser sobrepasado) y homogénea de vehículos, todos ellos con la misma capacidad (que no debe ser superada en ningún momento).
- Todos los clientes deben formar parte de una ruta, y su demanda debe ser satisfecha completamente por un único vehículo.
- Todas las rutas deben iniciar y finalizar su recorrido en el almacén central.
- No se puede incumplir ninguna ventana de tiempo (no se puede iniciar el servicio antes del inicio más temprano E_i ni después del inicio más tardío L_i). En ese sentido, se puede decir que en el presente trabajo se abordará la versión estricta del VRPTW. El almacén también dispone de una ventana de tiempo

En relación a la función objetivo, se utilizará el esquema jerárquico de dos niveles sugerido en (SINTEF 2015), que es el habitual cuando se usa como referencia el juego de ensayo de Solomon (ver sección 5.1.3):

- En primer lugar, se intenta minimizar el número de rutas o vehículos necesarios.
- A igual número de vehículos, será mejor la solución que implique una menor distancia total recorrida.

Por último, unas últimas cuestiones a destacar, (y que rara vez se indican explícitamente):

- Las distancias entre los nodos son simétricas.
- La distancia y el tiempo son equivalentes.
- Para la representación de los valores de distancia, capacidad y tiempo se utiliza la precisión decimal doble⁹.
- Al agrupar los resultados en cuanto a número de rutas y distancia total recorrida para todas las instancias de una categoría de problemas se utiliza un redondeo con tres decimales.

5.2.2 Representación de las soluciones

Un aspecto importante a la hora de aplicar una técnica de resolución a un problema de optimización combinatoria, es la forma en que se representa la información que configura la solución a un problema. Dicha representación es especialmente importante porque condiciona en gran medida la manera de manipular los elementos que configuran el problema durante el proceso de resolución. Por ejemplo, al aplicar un operador de mejora en un búsqueda local, o al realizar un proceso de cruce en un algoritmo poblacional.

En el presente trabajo de tesis doctoral, y con objeto de facilitar la comprensión del trabajo realizado y la reutilización el código fuente de forma que pueda ser integrado posteriormente en un sistema de información, se ha optado por utilizar una representación (tanto del problema como de la solución) guiada por las pautas y buenas prácticas del diseño orientado a objetos. De esta forma, todos los elementos que configuran el problema son objetos y cada uno de ellos contiene a modo de atributos las características o parámetros que lo describen conceptualmente. Este mismo enfoque también es seguido a la hora de implementar los diferentes operadores, algoritmos y comprobaciones para la validación de las restricciones del problema.

Para el caso del problema tratado: el VRPTW, este enfoque, no es el más ortodoxo desde el punto de vista de la literatura, puesto que lo habitual es utilizar una representación de “camino gigante” (*giant tour*) (Toth y Vigo 2015) formada a partir de una

⁹ https://es.wikipedia.org/wiki/Coma_flotante [consultado en septiembre de 2015]

permutación de números siendo cada uno de ellos el identificador de un cliente (reservando el identificador 0 para el almacén central). Este tipo de representación ofrece un rendimiento contrastado en relación al espacio de almacenamiento ya que puede representarse mediante una lista (o *array*) de números. Además, esta representación es muy flexible para realizar operaciones de cruce o mutación en el contexto de los algoritmos genéticos que se basan en la recombinación de secuencias de elementos que simulan las cadenas de ADN de los individuos que forman una población de soluciones.

Pese a las bondades de la representación de “camino gigante”, que tienen su origen en el problema del TSP, para el presente trabajo se ha decidido utilizar una representación orientada a objetos con la siguiente estructura:

- El Problema (*Problem*) es un objeto compuesto básicamente por un almacén central (*Depot*), una lista de clientes (*List<Customer>*), una lista de vehículos (*List<Vehicle>*) y una lista de rutas (*List<Route>*).
- La Ruta (*Route*) se representa básicamente mediante una lista ordenada de clientes, y además contiene atributos precalculados (ver sección 3.2.3.3) que almacenan la demanda, la distancia, el tiempo de servicio y el tiempo de espera.
- El Cliente (*Customer*) es el elemento más importante de la representación, y además de la identificación y su ubicación, almacena la información de la ventana de tiempo (inicio más temprano e inicio más tardío), la demanda, el tiempo de servicio, la distancia respecto al almacén central, y varios atributos para representar instantes de tiempo, (llegada, inicio, salida y tiempo de espera) que son utilizados para la ordenación de los clientes a la hora de definir las rutas.

Con esta representación, que es más expresiva que las representaciones tradicionales, todas las operaciones de manipulación de clientes y rutas se realizan de una forma muy intuitiva para cualquier persona que tenga unos mínimos conocimientos de orientación a objetos, con lo cual, retomar el trabajo realizado para mejorarlo y evolucionarlo será una tarea relativamente sencilla de abordar.

Este último aspecto, es sin duda una de las principales motivaciones que ha movido al autor de esta tesis doctoral a diferenciarse de la tendencia habitual en el contexto de la algoritmia para la resolución de problemas de optimización combinatoria. En este contexto, muchas veces se prima el resultado final, y no tanto el hecho de que la solución planteada (especialmente los detalles específicos de la implementación e incluso el código fuente) se ponga a disposición de la comunidad para que libremente cualquier persona pueda realizar un aporte, promoviendo así la mejora y evolución del trabajo realizado. Teniendo en cuenta lo anterior, podría reducirse considerablemente

el tiempo y la curva de aprendizaje necesarios para afrontar una nueva tipología de problema o técnica de resolución.

5.2.3 Implementación de las heurísticas de construcción

Como se ha descrito en la parte introductoria de este capítulo, la experimentación llevada a cabo en la presente tesis doctoral se compone de dos procesos. En el primero de ellos se compara el rendimiento de la heurística de construcción propuesta como aporte de la presente tesis doctoral con las tres heurísticas de construcción más representativas en el contexto del VRPTW (ver sección 3.1.1).

Tal y como se ha descrito en apartados anteriores, con objeto de minimizar el impacto de la implementación en los resultados y su comparación, todas las heurísticas comparadas han sido codificadas por el autor del presente trabajo. La implementación de cada una de las heurísticas respeta al máximo los detalles que fueron reportados por los autores en sus trabajos originales; con las adaptaciones pertinentes en relación a la codificación del problema y la solución.

A continuación, se describen las diferentes variantes y combinaciones de heurísticas de inicialización que se presentarán en las secciones 5.3 y 5.4 para validar la primera de las hipótesis planteadas en la presente tesis doctoral.

El objetivo fundamental del primer proceso de experimentación consiste en contrastar la efectividad de la nueva heurística de construcción propuesta, que en adelante se identificará como $I1^{RC}$ (RC se corresponde con el acrónimo del nombre y primer apellido del autor de la presente tesis doctoral). Para ello se ejecutarán las diferentes heurísticas a comparar sobre las instancias de problemas del juego de ensayo de Solomon (Solomon 1987) presentando como resultados los valores obtenidos para cada una de las clases de problemas, que es la manera más habitual de presentar los resultados para este juego de ensayo, tal y como se puede observar en la **tabla 3-1**.

En concreto, la experimentación analizará nueve variantes de procesos de inicialización que se corresponden con las tres heurísticas básicas (con la particularidad de que para la heurística IRCI sólo se ejecutará el proceso de construcción inicial), las tres heurísticas básicas junto con el proceso de reconstrucción propuesto en la heurística IRCI; y finalmente las tres heurísticas básicas junto con el proceso de reconstrucción definido como aporte de la presente tesis doctoral.

A modo de resumen, se detallan los acrónimos que se corresponden con cada una de las diferentes heurísticas analizadas:

- $I1$: heurística $I1$ de Solomon.
- $IMPACT$: heurística $IMPACT$.
- $IRCI$: proceso de inicialización de la heurística $IRCI$ (sin la mejora basada en reconstrucción).
- $I1_{IRCI}$: heurística $I1$ + proceso de reconstrucción propuesto en $IRCI$.
- $IMPACT_{IRCI}$: heurística $IMPACT$ + proceso de reconstrucción propuesto en $IRCI$.
- $IRCI_{IRCI}$: heurística $IRCI$ completa (combinando inicialización + reconstrucción).
- $I1^{RC}$: heurística de dos fases propuesta por el autor como uno de los aportes de la presente tesis doctoral.
- $IMPACT_{I1^{RC}}$: heurística $IMPACT$ + proceso de reconstrucción basado en $I1^{RC}$.
- $IRCI_{I1^{RC}}$: heurística $IRCI$ + proceso de reconstrucción basado en $I1^{RC}$.

Para finalizar esta descripción, simplemente destacar que, pese a que las técnicas de construcción son todas ellas deterministas, durante la experimentación se ha ejecutado cada una de ellas diez veces con objeto de que los valores de tiempo de ejecución sean consistentes.

5.2.4 Ajuste de los parámetros de las heurísticas de construcción

Una de las cuestiones más complejas a la hora de ajustar el funcionamiento de una heurística de construcción es sin duda la configuración de los parámetros asociados a las funciones que determinan la ubicación más adecuada de cada cliente en el proceso de construcción de las rutas.

Tal y como se revisó en la sección 3.1.1, en la que se describió con detalle el funcionamiento de las principales heurísticas de construcción para el problema del VRPTW, las diferentes heurísticas de construcción presentadas utilizan un conjunto de parámetros que permiten ponderar los diferentes criterios que se usan para determinar en cada paso el cliente y su posición más adecuada de inserción en la ruta actual. Dichos criterios contemplan aspectos como el incremento en distancia, el desfase en el inicio, la capacidad del vehículo, o incluso el impacto que supone la incorporación de un nuevo vehículo a la solución actual.

El ajuste fino de los parámetros es una tarea costosa que se realiza de forma empírica mediante la experimentación o tras un proceso de análisis estadístico. En el caso que nos ocupa, el proceso de ajuste de los parámetros de cada una de las heurísticas de construcción se ha realizado tomando como referencia dos instancias de cada una de

las clases de problemas de Solomon. En concreto se ha elegido para cada una de las clases la instancia “más sencilla”: aquella que prácticamente no tiene colisiones en las ventanas de tiempo (C101, C201, R112, R208, RC104 y RC208); y una de las instancias “más difíciles”: la que necesita un mayor número de vehículos (C103, C204, R101, R201, RC101, RC201) (SINTEF 2015). De esta forma, los mejores parámetros obtenidos de entre el conjunto de variaciones de los parámetros con pequeños incrementos, son los que se utilizarán para cada uno de los métodos durante el proceso de experimentación. Parece lógico suponer que con el uso de estos parámetros no se obtienen los resultados reportados por los autores de las tres heurísticas que se utilizarán como referencia en la experimentación (recogidos en la **tabla 3-1**). La justificación de su elección viene motivada por las dificultades encontradas a la hora de intentar reproducir estos resultados, utilizando los parámetros que se reportaban en los trabajos originales. No obstante, el objetivo del trabajo realizado en la presente tesis doctoral, no es la obtención de una nueva técnica que supere a las mejores técnicas existentes en cuanto a resultado final; sino el desarrollo de un aporte que obtenga unos resultados prometedores y comparables a los de las técnicas más representativas.

A continuación, se muestra la configuración elegida para la generación de los parámetros iniciales de cada una de las heurísticas utilizadas:

- *Heurística I1* (Solomon 1987)
 - Criterios para la elección del primer cliente de la ruta:
 - El cliente más alejado del almacén central
 - El cliente con el inicio tardío más pequeño
 - Parámetros utilizados por la función de selección de clientes:
 - α_1 y α_2 : valores entre 0 y 1 con incrementos de 0,10.
 - λ : valores entre 1 y 2 con incrementos de 0,25.
 - $\mu = 1$: este es el mejor valor propuesto en (Solomon 1987).
 - Combinaciones de parámetros analizadas: 110
 - Combinaciones de parámetros elegidas: 12
- *Heurística IMPACT* (Ioannou, Kritikos y Prastacos 2001)
 - Criterios para la elección del primer cliente de la ruta:
 - El cliente más alejado del almacén central
 - Parámetros utilizados por la función de selección de clientes:

- $b_s + b_e + b_r = 1$: valores entre 0 y 1 con incrementos de 0,10.
- $b_1 + b_2 + b_3 = 1$: valores entre 0 y 1 con incrementos de 0,10.
- Combinaciones de parámetros analizadas: 4356
- Combinaciones de parámetros elegidas: 12
- *Heurística IRCI* (Figliozzi 2010)
 - Parámetros utilizados por la función de selección de clientes:
 - $\delta_0 \geq 0$: valores entre 100 y 700 con incrementos de 100.
 - $\delta_1 + \delta_2 + \delta_3 = 1$: valores entre 0 y 1 con incrementos de 0,10.
 - Combinaciones de parámetros analizados: 1848
 - Combinaciones de parámetros escogidas: 12

El detalle de los parámetros concretos se incorpora en el Apéndice A del presente documento.

5.2.5 Operadores de intercambio de arcos y nodos

Como se ha esbozado ya en la parte introductoria de este capítulo, la experimentación realizada analizará por un lado el comportamiento de la nueva heurística de construcción de soluciones iniciales para problemas de tipo VRPTW; y por otro lado el rendimiento de los operadores de mejora para el mismo tipo de problema. En ese sentido, los operadores propuestos serán comparados con algunos de los operadores de mejora de rutas clásicos basados en el intercambio de nodos y aristas, que fueron introducidos en la sección 3.2.1. En concreto, se utilizarán siete operadores. Por un lado, cinco operadores interruta: 2-opt*, recolocación, intercambio individual, intercambio de cadenas sin inversión, e intercambio de cadenas con inversión; y por otro lado, dos operadores intraruta: 2-opt y Or-opt. Las cuestiones generales en cuanto a la implementación y comportamiento de todos los operadores son las siguientes:

- Para garantizar la equidad y evitar el impacto que puede suponer la implementación de los operadores en los resultados, todos ellos han sido codificados por el autor del presente trabajo de tesis doctoral usando las mismas convenciones en cuanto a estilo de programación y estructuras de datos utilizadas. En este sentido, cabe prestar una especial atención al uso de los patrones de diseño

de software *Strategy*¹⁰ y *Visitor*¹¹ a la hora de analizar la idoneidad de las rutas y realizar los cambios en las mismas. Estos patrones particularizan las comprobaciones a la naturaleza del problema a resolver, con lo cual, la adaptación de los operadores implementados a una nueva variante del problema de tipo VRP sería una labor sencilla de abordar.

- Todos los operadores han sido adaptados específicamente para el problema de tipo VRPTW respetando en todo momento las restricciones impuestas por las ventanas de tiempo. Por lo tanto todos ellos, a partir de una solución previa, generan una solución válida que podrá ser mejor o peor que la solución previa.
- Los operadores poseen un comportamiento determinista, generando siempre el mismo resultado para la misma entrada.
- Para optimizar la eficiencia en cuanto al tiempo de aplicación de los operadores, todos se han implementado siguiendo el enfoque de búsqueda secuencial propuesto por en (Irnich, Funke y Grünert 2006) (véase sección 3.2.3.1), listas de vecinos (sección 3.2.3.2), las optimizaciones propuestas en cuanto al uso de variables globales y el análisis eficiente de la validez de las modificaciones realizadas en una ruta (sección 3.2.3.3).
- Los operadores intraruta se aplican de acuerdo a la estructura del **algoritmo 5-1**, en la cual puede observarse que el operador se integra en un proceso iterativo que intenta aplicar el operador a todas las rutas del problema hasta que no se encuentra ninguna mejora. En cada iteración, el operador utiliza el criterio “primera mejora encontrada” (*first improvement*) para finalizar su proceso. De esta forma, se consigue optimizar el tiempo de ejecución global.
- Una cuestión adicional a destacar, es el uso de la constante MIN_GAIN, que se utiliza como referencia para determinar la ganancia mínima que debe tener la nueva solución para que sustituya a la previa.
- En el caso de los operadores interruta el proceso se realiza de acuerdo al **algoritmo 5-2**. Aquí también se utiliza un proceso iterativo que aplica el operador a cada par de rutas del problema usando el mismo criterio de “primera mejora encontrada” cada vez que se encuentra una mejora, tanto internamente en el operador, como en el propio proceso iterativo, tal y como se puede observar en la sentencia de la línea 34, que provoca la finalización del bucle externo cada vez que se mejora alguna ruta.

¹⁰ https://en.wikipedia.org/wiki/Strategy_pattern [consultado en septiembre de 2015]

¹¹ https://en.wikipedia.org/wiki/Visitor_pattern [consultado en septiembre de 2015]

Algoritmo 5-1: Pseudocódigo de la aplicación de un operador intraruta.

```

1.  intraRouteOptimization(operator, problem, MIN_GAIN)
2.    oldRoute =  $\emptyset$ 
3.    newRoute =  $\emptyset$ 
4.    bestRoute =  $\emptyset$ 
5.    updated = false
6.    oldDistance = -1
7.    newDistance = -1
8.
9.    DO {
10.     updated = false
11.
12.     FOR (oldRoute  $\in$  problem) DO
13.       oldDistance = oldRoute.getDistance()
14.       newRoute = operator.apply(oldRoute, MIN_GAIN)
15.       newDistance = newRoute.getDistance()
16.
17.       IF ((preDistance - newDistance) > MIN_GAIN) THEN
18.         problem.updateRoute(oldRoute, newRoute)
19.         updated = true
20.         BREAK
21.       END IF
22.     END FOR
23.   WHILE (updated)
24.   RETURN: problem

```

- Por último, se recoge aquí también la estructura utilizada para la aplicación de los operadores de reducción del número de rutas que forman parte de la mejora propuesta por el presente trabajo de tesis doctoral: RcR-opt, SrE-opt y RrE-opt. En este caso, también se sigue el mismo enfoque que en los operadores intraruta e interruta, (**algoritmo 5-4**). El operador se incrusta en un proceso iterativo, que se detiene cuanto no se encuentra ninguna mejora adicional para la solución actual. Dicha mejora se analiza en relación a la función objetivo jerárquica, descrita en la sección 5.2.1.

Una vez descritas las cuestiones generales en relación a los operadores de mejora, a continuación, se detallan las particularidades específicas de alguno de los operadores, con el fin de simplificar la reproducción de la experimentación realizada:

- *Operador interruta de recolocación:* este operador está basado en el operador de recolocación propuesto por Savelsbergh en (Savelsbergh 1992), y posteriormente adaptado a la búsqueda local secuencial en (Irnich, Funke y Grünert 2006).

Algoritmo 5-2: Pseudocódigo de la aplicación de un operador interruta.

```

1.  interRouteOptimization(operator, problem, MIN_GAIN)
2.    List newRoutes =  $\emptyset$ 
3.    oldR1 =  $\emptyset$ 
4.    oldR2 =  $\emptyset$ 
5.    newR1 =  $\emptyset$ 
6.    newR2 =  $\emptyset$ 
7.    bestRoute =  $\emptyset$ 
8.    updated = false
9.    oldDistance = -1
10.   newDistance = -1
11.
12.   DO
13.     updated = false
14.     outterFor:
15.
16.     FOR (oldR1  $\in$  problem) DO
17.       FOR (oldR2  $\in$  problem) DO
18.         IF (oldR1  $\neq$  oldR2) THEN
19.           oldDistance = oldR1.getDistance() + oldR2.getDistance()
20.           newRoutes = operator.apply(oldR1,oldR2,MIN_GAIN)
21.           newDistance = newRoutes.getDistance()
22.           IF ((preDistance - newDistance) > MIN_GAIN) THEN
23.             problem.updateRoute(oldR1,newRoute[0])
24.             problem.updateRoute(oldR2,newRoute[1])
25.             updated = true
26.             BREAK outterFor
27.           END IF
28.         END IF
29.       END FOR
30.     END FOR
31.   WHILE (updated)
32. RETURN: problem

```

- *Operador interruta de intercambio individual:* este operador, al igual que el anterior, está basado en el operador de intercambio individual propuesto por Savelsbergh (Savelsbergh 1992) y posteriormente adaptado a la búsqueda local secuencial en (Irnich, Funke y Grünert 2006).
- *Operadores interruta de intercambio de cadenas con/sin inversión:* estos operadores se han implementado como una generalización del operador interruta de intercambio individual, pero siendo el número de nodos intercambiado mayor que uno. En concreto, para controlar el tamaño del vecindario generado por este operador, se limita el número máximo de nodos intercambiados a cuatro; probando en cada caso segmentos (secuencias de nodos consecutivos) de dos, tres y cuatro. Por último, la diferencia entre la variante con y sin inversión, se basa en

el hecho de que el segmento de nodos intercambiados mantiene o invierte el orden que tenían los nodos en la ruta original.

- Por último, los operadores de mejora basados en la reducción del número de rutas han sido implementados de acuerdo a las pautas descritas en la sección 4.2.

Algoritmo 5-3: Pseudocódigo de la aplicación de un operador de mejora basado en la reducción del número de rutas.

```

1.  minNumRoutesOptimization(operator, problem, MIN_GAIN)
2.  List newRoutes =  $\emptyset$ 
3.  List oldRoutes =  $\emptyset$ 
4.  updated = false
5.
6.  DO
7.    updated = false
8.    oldRoutes = problem.getRoutes()
9.    newRoutes = operator.apply(oldRoutes, MIN_GAIN)
10.
11.   IF (newRoutes better than oldRoutes) THEN
12.     problema.updateRoutes(oldRoutes, newRoutes)
13.     updated = true
14.   END IF
15.  WHILE (updated)
16.  RETURN: problem

```

5.2.6 Implementación algoritmo de búsqueda local de referencia

El segundo proceso de experimentación realizado tiene por objeto la validación de los operadores de mejora propuestos en el presente trabajo de tesis doctoral. Para ello, se comparará el rendimiento de los nuevos operadores con el obtenido por los operadores clásicos de intercambio de nodos y arcos; todo ellos integrados en una heurística de búsqueda local paralela con vecindario variable. La elección de esta heurística como base para la experimentación viene motivada por los siguientes aspectos:

- En primer lugar, la búsqueda elegida es una búsqueda local que avanza siempre que exista una mejora en la solución actual. Este hecho es especialmente importante en el contexto de la presente tesis doctoral, que aborda el desarrollo de una mejora en el proceso de construcción para la generación de una “buena” solución inicial. Por ese motivo, el uso de una heurística o metaheurística que acepte soluciones peores a la inicial desvirtuaría completamente la importancia que se le ha dado al proceso de inicialización.
- Por otro lado, pese a ser una búsqueda local, se ha optado por utilizar un vecindario variable con objeto de incorporar un comportamiento no determinista en base a la combinación de los diferentes operadores. Esto permite, obtener mejores

resultados que un proceso de búsqueda local simple; y además, facilita la realización de un análisis más exhaustivo del aporte de los nuevos operadores definidos.

El **algoritmo 5-4** muestra el esquema general de la heurística de búsqueda local paralela con vecindario variable utilizada durante la experimentación para la validación de la segunda de las hipótesis definidas en la presente tesis doctoral. Los pasos más relevantes del algoritmo propuesto son:

- *Fase de inicialización de los parámetros y criterios que controlan el flujo del algoritmo* (paso previo al inicio del proceso):
 - Número de soluciones actuales *NumSoluciones*. Este valor determina el número de soluciones diferentes que se analizan en cada iteración y marca la amplitud del proceso “paralelo”.
 - Número máximo de iteraciones MAX_{it} . Este valor se combina con el número máximo de iteraciones sin mejora para determinar el instante en que finaliza la ejecución del algoritmo.
 - Número máximo de iteraciones sin mejora $MAX_{it_sinMejora}$. Este valor se utiliza para complementar el criterio de fin del algoritmo junto con el número máximo de iteraciones. De esta forma, el proceso del algoritmo finaliza cuando se supere el número máximo de iteraciones o la solución actual no se haya actualizado durante un determinado número de iteraciones.
 - Variables booleanas que controlan la aplicación o no de cada uno de los tipos de operadores de mejora: *aplicarOperador_{InterRuta}*, *aplicarOperador_{MinNumRutas}*, y *aplicarOperador_{IntraRuta}*. Estas variables se utilizarán para generar variantes del algoritmo que se diferencien en los tipos de operadores utilizados.
 - Conjunto de operadores interruta: contiene los cinco operadores descritos anteriormente: 2-opt*, recolocación, intercambio individual, intercambio de cadenas sin inversión e intercambio de cadenas con inversión. La función *seleccionarOperadorInterRuta()* elige (de forma aleatoria) el operador que se utiliza en cada iteración. El operador seleccionado en cada iteración se aplicará a todas las soluciones que están en la lista *l_Soluciones_{actual}*.
 - Conjunto de operadores de reducción del número de rutas: contiene los tres operadores propuestos: RcR-opt, SrE-opt y RrE-opt (usando las cuatro variantes propuestas para cada uno de ellos presentadas en la sección 4.2). Al igual que en el caso anterior la selección del operador se realizará de forma aleatoria.

- Conjunto de operadores intraruta: contiene los operadores 2-opt y Or-opt. La utilidad de esta lista es similar a la de las listas de los otros dos tipos de operadores.

Algoritmo 5-4: Pseudocódigo del algoritmo de búsqueda local paralela con vecindario variable.

```

1. Entrada: Problema
2.   $it = 0$ 
3.   $it\_sinMejora = 0$ 
4.   $l\_Soluciones_{actual} = inicializarSolucionesActuales(NumSoluciones)$ 
5.   $l\_Soluciones_{nueva} = \emptyset$ 
6.   $solucion_{nueva} = \emptyset$ 
7.
8.  DO
9.     $l\_Soluciones_{nueva} = \emptyset$ 
10.    $opeador_{InterRuta} = seleccionarOperadorInterRuta()$ 
11.    $opeador_{MinNumRutas} = seleccionarOperadorMinNumRutas()$ 
12.    $opeador_{IntraRuta} = seleccionarOperadorIntraRuta()$ 
13.
14.   FOR ( $solucion_{actual} \in l\_Soluciones_{actual}$ )
15.     IF ( $aplicarOperador_{InterRuta}$ ) THEN
16.        $solucion_{nueva} = opeador_{InterRuta}.aplicar(solucion_{actual})$ 
17.     END IF
18.     IF ( $aplicarOperador_{MinNumRutas}$ ) THEN
19.        $solucion_{nueva} = opeador_{MinNumRutas}.aplicar(solucion_{nueva})$ 
20.     END IF
21.     IF ( $aplicarOperador_{IntraRuta}$ ) THEN
22.        $solucion_{nueva} = opeador_{IntraRuta}.aplicar(solucion_{nueva})$ 
23.     END IF
24.      $l\_Soluciones_{nueva}.add(solucion_{nueva})$ 
25.   END FOR
26.
27.    $l\_Soluciones_{nueva} = seleccMejoresSoluciones(l\_Soluciones_{nueva}, l\_Soluciones_{actual})$ 
28.
29.   IF ( $l\_Soluciones_{nueva}$  mejor que  $l\_Soluciones_{actual}$ ) THEN
30.      $l\_Soluciones_{actual} = l\_Soluciones_{nueva}$ 
31.      $it\_sinMejora = 0$ 
32.   ELSE
33.      $it\_sinMejora ++$ 
34.   END IF
35.
36.    $it ++$ 
37.   WHILE ( $it < MAX_{it}$  &&  $it\_sinMejora < MAX_{it\_sinMejora}$ )
38. Salida: RETURN mejor solución  $\in l\_Soluciones_{actual}$ 

```

- Ganancia mínima de los operadores de mejora: este parámetro especifica la diferencia mínima que debe existir (medida en coste de la solución) entre la solución previa y la solución después de aplicar un operador, para considerar que existe una mejora. Si la diferencia entre las dos soluciones es inferior a la ganancia mínima, el operador devuelve la solución previa como resultado.
- Criterio de selección de mejores soluciones: define el comportamiento de la función *seleccMejoresSoluciones()*. Esta función escoge el conjunto de soluciones que se analizan para determinar si existe una mejora en la iteración actual. En este sentido, se han analizado tres criterios de selección de las nuevas soluciones:
 - *ELITIST*: este criterio selecciona las mejores soluciones combinando las actuales y las nuevas.
 - *NEW_SOLUTIONS*: este criterio selecciona directamente las nuevas soluciones.
 - *HALF_RANDOM*: este criterio combina las soluciones actuales y las nuevas; seleccionando finalmente el 50% de las mejores soluciones y el otro 50% al azar (y sin repetición).
- Criterio de análisis de la existencia de mejora de la nueva solución respecto a la solución actual. Este criterio analiza la existencia de mejora actualizando contador $MAX_{it_sinMejora}$ de forma apropiada. En este sentido, se han utilizado dos criterios para la determinación de la existencia de mejora, ambos basados en la función objetivo descrita previamente en la sección 5.2.1:
 - *BEST_IMPROVED*: considera que existe mejora siempre que “la mejor” solución de la lista $l_Soluciones_{nueva}$ sea mejor que “la mejor” solución de la lista $l_Soluciones_{actual}$.
 - *WORST_IMPROVED*: determina la existencia de mejora siempre que “la mejor” solución de la lista $l_Soluciones_{nueva}$ sea mejor que “la peor” solución de la lista $l_Soluciones_{actual}$.
- *Generación del conjunto de soluciones iniciales* (línea 4), utilizando una heurística constructiva. Este proceso inicializa tantas soluciones iniciales como especifique el contador *NumSoluciones*.
- *Selección de los operadores de mejora utilizados en la iteración actual* (líneas 10-12). La selección afecta a los tres tipos de operadores de mejora descritos con anterioridad: operadores interruta, operadores intraruta y operadores de

reducción del número de rutas. La selección se realiza de forma totalmente aleatoria.

- *Aplicación de los operadores y generación de las nuevas soluciones* (líneas 14-25). Este proceso se corresponde con el bucle *FOR*. En él se procesa cada una de las soluciones que está en la lista $l_Soluciones_{actual}$ y se aplican los operadores seleccionados teniendo en cuenta las variables booleanas que determinan la aplicación o no de cada uno de los tipos de operadores.
- *Selección de las nuevas soluciones* (línea 27), utilizando el criterio de selección de las mejores soluciones
- *Análisis de la existencia de mejora* (línea 29), a partir del criterio de determinación de la mejora.
- *Comprobación del fin del proceso* (línea 37), en base al análisis de las variables de control it e $it_sinMejora$ y sus límites: MAX_{it} y $MAX_{it_sinMejora}$ respectivamente.
- *Retorno de la mejor solución* (línea 38), que será la “mejor” de las soluciones que se encuentren en la lista $l_Soluciones_{actual}$, una vez finalizado el proceso de búsqueda.

Acrónimo	Operadores utilizados
$PVNS_{Intra}$	- Intraruta
$PVNS_{Inter}$	- Interruta
$PVNS_{Min}$	- MinNumRutas (todos)
$PVNS_{RrE}$	- RrE-opt
$PVNS_{Inter-Intra}$	- Interruta - Intraruta
$PVNS_{Min-Intra}$	- MinNumRutas (todos) - Intraruta
$PVNS_{RrE-Intra}$	- RrE-opt - Intraruta
$PVNS_{All}$	- Interruta - MinNumRutas (todos) - Intraruta
$PVNS_{All-RrE}$	- Interruta - RrE-opt - Intraruta

Tabla 5-1: Variantes del algoritmo de búsqueda paralela con vecindario variable.

Después de ejecutar y analizar los resultados obtenidos por las diferentes alternativas de estructura de la heurística paralela de vecindario variable (*Parallel Variable Neighborhood Search* - PVNS) definida en esta sección, de cara a la experimentación final, se presentan únicamente los resultados de las diez variantes más representativas (ver **tabla 5-1**). Estas nueve variantes, que se identifican con el acrónimo ($PVNS_{XX}$), se diferencian básicamente en los operadores de mejora utilizados en cada una de ellas. El resto de alternativas y variantes del algoritmo base no se han incorporado en el documento ni en el análisis porque no reportan diferencias significativas en los resultados obtenidos.

Aparte de los operadores utilizados, el resto de parámetros de configuración de las nueve variantes del algoritmo PVNS son los mismos en todos los casos:

- Número total de ejecuciones realizado para cada variante del algoritmo: 30
- Número de soluciones actuales: 12. Las soluciones iniciales se han generado mediante la heurística propuesta como aporte de la presente tesis doctoral ($I1^{RC}$) y los parámetros para la heurística I1 de Solomon adjuntados en el Apéndice A. Debido a que el proceso de construcción es determinista, y para controlar el tiempo necesario para llevar a cabo cada una de las ejecuciones, las soluciones iniciales se han generado una única vez, y posteriormente se han reutilizado durante toda la experimentación. Teniendo en cuenta este hecho, los tiempos de ejecución que se mostrarán para cada una de las variantes del algoritmo $PVNS_{XX}$ incluyen únicamente el proceso de lectura de las soluciones iniciales, y no el tiempo necesario para generarlas completamente.
- Número máximo de iteraciones: 100
- Número máximo de iteraciones sin mejora: 20
- Ganancia mínima de los operadores de mejora: 5,0
- Criterio de selección de mejores soluciones: *ELITIST*
- Criterio de determinación de la existencia de mejora: *BEST_IMPROVED*

5.2.7 Entorno de ejecución y lenguaje de desarrollo

Una cuestión adicional que toma una especial relevancia a la hora de presentar los resultados de una experimentación, son las características del entorno computacional en el que ésta ha sido realizada. En ese sentido, tanto la implementación como las pruebas se han realizado en un ordenador portátil MacBook Air con procesador Intel Core i5 4250U a 1,3 GHz. y 8 GB de memoria RAM; sobre plataforma OS X Yosemite.

Por último, la implementación se ha realizado utilizando el lenguaje de programación Java, con convenciones de código de la versión 8; siendo la versión de la máquina virtual sobre la que se han ejecutado las pruebas la 1.8 *update 60*.

5.3 Resultados y pruebas estadísticas

Una vez descritas de manera exhaustiva las cuestiones relevantes en cuanto a la configuración de la experimentación en esta sección se presentan los resultados obtenidos.

La sección se organiza de la siguiente forma: la sección 5.3.1 muestra los resultados obtenidos en la comparación de las diferentes heurísticas de construcción descritas en la sección 5.2.3; por otro lado, la sección 5.3.2 presenta los resultados de la experimentación llevada a cabo para analizar el rendimiento de los operadores de mejora propuestos.

5.3.1 Resultados de la experimentación en torno a la heurística de construcción para el VRPTW

En esta sección se muestran los resultados de la experimentación llevada a cabo en torno a las heurísticas de construcción de soluciones iniciales para problemas de tipo VRPTW.

Tal y como se ha descrito en la sección 5.2.3, durante la experimentación se han realizado pruebas con un total de nueve heurísticas de construcción (tres variantes para cada una de las heurísticas utilizadas como referencia), todas ellas aplicadas al juego de ensayo de problemas de tipo VRPTW de Solomon completo. Los resultados obtenidos se recogen en dos bloques de tablas para facilitar su comparación y análisis:

- Las tablas **tabla 5-2**, **tabla 5-3** y **tabla 5-4** se corresponden respectivamente con:
 - La variante básica de cada una de las tres heurísticas de referencia.
 - La variante básica de las tres heurísticas de referencia junto con la mejora propuesta por la heurística *IRCI*.
 - La variante básica combinada con la mejora propuesta como aporte de la presente tesis doctoral (ver sección 4.1).
- En las tablas **tabla 5-5**, **tabla 5-6** y **tabla 5-7** se agrupan las variantes de cada heurística de referencia, para facilitar el análisis de cada una de forma individual:

- Las tres variantes de la heurística *I1* de Solomon.
- Las tres variantes de la heurística *IRCI*.
- Las tres variantes de la heurística *IMPACT*.

Todas las tablas presentan una estructura similar. Dicha estructura es la más utilizada en la literatura para mostrar los resultados de técnicas que se aplican al juego de ensayo de Solomon completo. A modo de ejemplo, se sugiere consultar (Bräysy y Gendreau 2005a) o (Vidal, y otros 2014).

Las filas representan las clases de problemas; y para cada una de ellas, en columnas, se muestran los valores medios de número de vehículos (NV) y distancia (D) calculados a partir de los resultados obtenidos en todas las instancias de una misma clase. Adicionalmente, para facilitar y clarificar la comparación de los resultados, se han incorporado dos columnas adicionales $\%_{NV}$ y $\%_D$ que muestran la diferencia (en porcentaje) respecto a los mejores valores que aparecen en cada tabla (los mejores valores aparecen sombreados en color verde).

Finalmente, se incorpora una columna adicional (T) con el tiempo (en segundos) utilizado para resolver todas las instancias de una determinada clase. El tiempo se ha obtenido a partir de los datos obtenidos tras ejecutar cada una de las heurísticas en diez ocasiones.

A continuación, se incluye una breve descripción del contenido de los resultados obtenidos, que será ampliado en la sección 5.4.1 con un análisis más pormenorizado.

Clase	<i>I1</i>					<i>IRCI</i>					<i>IMPACT</i>				
	NV	$\%_{ND}$	D	$\%_D$	T	NV	$\%_{ND}$	D	$\%_D$	T	NV	$\%_{ND}$	D	$\%_D$	T
C1	10,00	-	945,13	-	0	10,00	-	1015,10	7%	224	10,33	3%	1375,64	46%	7
C2	3,38	-	753,58	-	1	4,75	41%	910,85	21%	100	3,88	15%	1109,54	47%	9
R1	14,33	5%	1624,13	11%	1	13,67	-	1458,38	-	195	13,75	1%	1678,82	15%	9
R2	3,36	6%	1419,38	10%	2	3,18	-	1293,33	-	356	3,55	11%	1676,12	30%	14
RC1	14,38	7%	1760,14	8%	0	13,38	-	1634,16	-	120	13,75	3%	1876,07	15%	6
RC2	3,75	3%	1636,52	7%	1	3,63	-	1523,82	-	262	4,13	14%	2035,64	34%	9

Tabla 5-2: Resultados obtenidos por la versión básica de cada una de las heurísticas de construcción.

La **tabla 5-2** muestra los resultados de las versiones básicas de las tres heurísticas analizadas. Como se puede observar, y siempre de acuerdo a la configuración de la experimentación realizada, la heurística *I1* de Solomon es la que obtiene mejores resultados para los problemas de las clases C1 y C2; que se corresponden con problemas cuyos clientes están agrupados. Por otro lado, la heurística *IRCI* es la que domina, es

decir, es la que obtiene mejores resultados en el resto de clases de problemas: los que tienen una distribución totalmente aleatoria de clientes (R_1 y R_2), y los que tienen una distribución mixta (RC_1 y RC_2). No obstante, pese a la superioridad de la heurística $IRCI$, los tiempos necesarios para la obtención de los resultados son notablemente mayores (llegando a ser más de 350 veces superiores respecto a la heurística $I1$ para la clase R_2). Este hecho, justificaría la elección de cualquiera de las otras dos heurísticas ($I1$ o $IMPACT$) como proceso de inicialización de una técnica heurística o metaheurística de mejora, ya que dicho proceso debiera representar un pequeño porcentaje del tiempo empleado por la técnica. Pese a ello, dependiendo de la técnica utilizada a posteriori, una buena solución inicial puede condicionar mucho el resultado de una técnica metaheurística (Bräysy y Gendreau 2005a). Por último, un dato adicional en relación a la heurística $IMPACT$ es que la diferencia en número de vehículos está siempre por debajo del 15%; mientras que la diferencia en relación a la distancia no es en ningún caso inferior al 15%.

Clase	$I1_{IRCI}$					$IRCI_{IRCI}$					$IMPACT_{IRCI}$				
	NV	%ND	D	%D	T	NV	%ND	D	%D	T	NV	%ND	D	%D	T
C1	10,00	-	945,00	-	670	10,00	-	1004,47	6%	877	10,00	-	1116,25	18%	696
C2	3,38	-	726,65	-	166	4,13	22%	835,94	15%	368	3,75	11%	857,11	18%	232
R1	13,75	3%	1499,33	-2%	600	13,50	1%	1423,68	-7%	727	13,33	-	1533,72	-	684
R2	3,27	3%	1333,38	4%	230	3,18	-	1279,08	-	660	3,18	-	1432,40	12%	340
RC1	13,25	-	1645,19	2%	445	13,25	-	1619,00	-	536	13,25	-	1702,69	5%	490
RC2	3,63	1%	1570,28	4%	197	3,50	-	1514,21	-	499	3,88	3%	1635,58	8%	352

Tabla 5-3: Resultados obtenidos por cada una de las heurísticas de construcción utilizando la mejora $IRCI$.

En la **tabla 5-3** se muestran los resultados obtenidos por las variantes básicas de las tres heurísticas junto con el proceso de mejora propuesto en la heurística $IRCI$. En este caso, el primer dato que salta a la vista son los tiempos de ejecución. Dichos valores son, en todos los casos, notablemente superiores a los empleados por las versiones simples de los algoritmos. No obstante, siempre se aporta una mejora tanto en número de vehículos como en distancia recorrida para las tres heurísticas analizadas (tal y como puede confirmarse de manera un poco más clara al revisar las tablas **tabla 5-5**, **tabla 5-6** y **tabla 5-7**). Analizando los mejores resultados para cada una de las clases de problema, la variante $I1_{IRCI}$ sigue dominando en las clases C_1 y C_2 ; pero ahora, el dominio de las clases con distribución aleatoria y mixta se reparte entre la heurística $IRCI_{IRCI}$ (que obtiene los mejores resultados para las clases R_2 , RC_1 y RC_2) y la heurística $IMPACT_{IRCI}$ (que obtiene mejores resultados para la clase R_1).

Clase	$I1^{RC}$					$IRCI_{I1^{RC}}$					$IMPACT_{I1^{RC}}$				
	NV	% _{ND}	D	% _D	T	NV	% _{ND}	D	% _D	T	NV	% _{ND}	D	% _D	T
C ₁	10,00	-	930,38	-	4	10,00	-	941,93	1%	236	10,11	1%	1020,08	10%	10
C ₂	3,13	-	689,18	-	3	3,88	24%	781,95	13%	103	3,25	4%	811,75	18%	12
R ₁	14,00	4%	1502,45	-5%	6	13,67	-	1458,38	-8%	204	13,50	-	1578,84	-	14
R ₂	3,27	3%	1371,81	6%	5	3,18	-	1290,51	-	391	3,18	-	1514,85	17%	18
RC ₁	14,00	5%	1702,97	5%	3	13,38	-	1625,04	-	130	13,50	1%	1777,75	9%	9
RC ₂	3,63	-	1618,63	7%	3	3,63	-	1515,15	-	280	3,75	3%	1756,50	16%	12

Tabla 5-4: Resultados obtenidos por cada una de las heurísticas de construcción utilizando la mejora $I1^{RC}$.

La **tabla 5-4** muestra los resultados de las tres heurísticas utilizadas como referencia junto con la mejora $I1^{RC}$ propuesta en la presente tesis doctoral. En esta ocasión, la tendencia en cuanto a dominio en las diferentes clases es la misma que en las variantes que incorporan la mejora $IRCI$: la heurística $I1^{RC}$ obtiene mejores resultados en las clases C₁ y C₂; la heurística $IMPACT_{I1^{RC}}$ domina en la clase R₁; y por último, la heurística $IRCI_{I1^{RC}}$ es la que mejor se comporta en las clases R₂, RC₁ y RC₂. Además, otro dato que tiene especial relevancia es que la diferencia en número de rutas para las clases con distribución aleatoria y mixta nunca supera el 5% entre la mejor solución y el resto. En relación a los tiempos de ejecución, la heurística $I1^{RC}$ es la que obtiene los valores más pequeños (al igual que lo hacen en resto de variantes de la heurística I₁ de Solomon) en comparación con las otras dos heurísticas; siendo los datos de la heurística $IRCI_{I1^{RC}}$ los peores.

Clase	$I1$					$I1_{IRCI}$					$I1^{RC}$				
	NV	% _{ND}	D	% _D	T	NV	% _{ND}	D	% _D	T	NV	% _{ND}	D	% _D	T
C ₁	10,00	-	945,13	2%	0	10,00	-	945,00	2%	670	10,00	-	930,38	-	4
C ₂	3,38	8%	753,58	9%	1	3,38	8%	726,65	5%	166	3,13	-	689,18	-	3
R ₁	14,33	4%	1624,13	8%	1	13,75	-	1499,33	-	600	14,00	2%	1502,45	-	6
R ₂	3,36	3%	1419,38	6%	2	3,27	-	1333,38	-	230	3,27	-	1371,81	3%	5
RC ₁	14,38	8%	1760,14	7%	0	13,25	-	1645,19	-	445	14,00	-	1702,97	4%	3
RC ₂	3,75	3%	1636,52	4%	1	3,63	-	1570,28	-	197	3,63	-	1618,63	3%	3

Tabla 5-5: Resultados obtenidos por las tres variantes de la heurística I₁.

Si realizamos una comparación de cada una de las variantes de una misma heurística (**tabla 5-5**, **tabla 5-6** y **tabla 5-7**) podemos observar claramente como las variantes que incorporan un segundo proceso de mejora obtienen mejores resultados que las versiones básicas, o simples, tanto en número de vehículos como distancia recorrida; pero con un incremento en el tiempo de ejecución. Además, en relación a los tiempos

de ejecución, las versiones que incorporan la mejora propuesta por la heurística $IRCI$ son las que obtienen peores resultados.

Clase	$IRCI$					$IRCI_{IRCI}$					$IRCI_{I1^{RC}}$				
	NV	%ND	D	%D	T	NV	%ND	D	%D	T	NV	%ND	D	%D	T
C1	10,00	-	1015,10	8%	224	10,00	-	1004,47	7%	930	10,00	-	941,93	-	236
C2	4,75	23%	910,85	16%	100	4,13	6%	835,94	7%	407	3,88	-	781,95	-	103
R1	13,67	1%	1458,38	2%	195	13,50	-	1423,68	-	824	13,67	1%	1458,38	2%	204
R2	3,18	-	1293,33	1%	356	3,18	-	1279,08	-	750	3,18	-	1290,51	1%	391
RC1	13,38	1%	1634,16	1%	120	13,25	-	1619,00	-	601	13,38	1%	1625,04	-	130
RC2	3,63	4%	1523,82	1%	262	3,50	-	1514,21	-	561	3,63	4%	1515,15	-	280

Tabla 5-6: Resultados obtenidos por las tres variantes de la heurística $IRCI$.

Por otro lado, focalizando el análisis en las dos variantes que incorporan mejora, se pueden observar que la mejora $I1^{RC}$ obtiene mejores resultados en las clases C1 y C2; mientras que la mejora $IRCI$ es la que domina en las clases con distribución de clientes aleatoria (R1 y R2) y mixta (RC1 y RC2).

Clase	$IMPACT$					$IMPACT_{IRCI}$					$IMPACT_{I1^{RC}}$				
	NV	%ND	D	%D	T	NV	%ND	D	%D	T	NV	%ND	D	%D	T
C1	10,33	3%	1375,64	23%	7	10,00	-	1116,25	-	696	10,11	1%	1020,08	-9%	10
C2	3,88	19%	1109,54	37%	9	3,75	15%	857,11	6%	232	3,25	-	811,75	-	12
R1	13,75	3%	1678,82	9%	9	13,33	-	1533,72	-	684	13,50	1%	1578,84	3%	14
R2	3,55	11%	1676,12	17%	14	3,18	-	1432,40	-	340	3,18	-	1514,85	6%	18
RC1	13,75	4%	1876,07	10%	6	13,25	-	1702,69	-	490	13,50	2%	1777,75	4%	9
RC2	4,13	10%	2035,64	16%	9	3,88	3%	1635,58	-7%	352	3,75	-	1756,50	-	12

Tabla 5-7: Resultados obtenidos por las tres variantes de la heurística $IMPACT$.

No obstante, las diferencias en los resultados obtenidos por las dos mejoras están en todos los casos bastante controladas, oscilando entre el 1% y el 7% en todos los casos (tanto en número de vehículos como en distancia).

A modo de resumen final, se incluye la **tabla 5-8**, que muestra los mejores resultados obtenidos. Estos datos representan las soluciones de mayor calidad, las cuales formarán parte del estado inicial de la búsqueda local utilizada en la experimentación en torno a los operadores de mejora. Como se puede observar (en la citada tabla), la heurística $I1^{RC}$ obtiene los mejores resultados en las clases C1 y C2; la heurística $IMPACT_{IRCI}$ domina en la clase R1; y finalmente, la heurística $IRCI_{IRCI}$ es la que reporta los mejores

valores para las clases R2, RC1 y RC2. Pese al dominio de la mejora $IRCI$ en las clases con distribución aleatoria y mixta de clientes, cabe destacar que las diferencias porcentuales (tanto en número de vehículos como en distancia) respecto a las variantes que incorporan la mejora $I1^{RC}$ no supera en ningún caso el 4%.

Clase	Heurística	NV	D	T
C1	$I1^{RC}$	10,000	930,377	4
C2	$I1^{RC}$	3,125	689,181	3
R1	$IMPACT_{IRCI}$	13,333	1533,718	684
R2	$IRCI_{IRCI}$	3,182	1279,084	750
RC1	$IRCI_{IRCI}$	13,250	1619,001	601
RC2	$IRCI_{IRCI}$	3,500	1514,212	561

Tabla 5-8: Mejores resultados obtenidos por las heurísticas de construcción.

Una última cuestión a destacar de los resultados obtenidos está relacionada con los tiempos de ejecución. En este sentido, las variantes que incorporan la mejora $I1^{RC}$ superan ampliamente los resultados obtenidos por las variantes que incorporan la mejora $IRCI$.

5.3.2 Resultados de la experimentación en torno a los operadores de mejora basados en la reducción del número de rutas

En esta sección se recogen los datos correspondientes a la experimentación llevada a cabo para validar la idoneidad de los operadores de mejora basados en la reducción del número de rutas.

Durante las pruebas se han ejecutado múltiples variantes del algoritmo de búsqueda local especialmente diseñado para la experimentación (ver sección 5.2.6). En este documento se presentan únicamente los resultados obtenidos por las nueve variantes más representativas, tal y como se ha expuesto anteriormente, y queda recogido en la **tabla 5-1**.

Antes de mostrar las tablas correspondientes a los resultados obtenidos por cada una de las variantes del algoritmo de búsqueda local, se añade la **tabla 5-8**, en la que se muestra, a modo de referencia, la mejor solución, la cual es utilizada para la inicialización del algoritmo de búsqueda local propuesto. Estos valores se obtienen al combinar las mejores soluciones obtenidas por las variantes de las heurísticas de construcción utilizadas en la primera fase de la experimentación (ver sección B.1 del Anexo B).

Clase	NV	D
C1	10,000	919,269
C2	3,125	694,764
R1	13,333	1523,614
R2	3,182	1289,546
RC1	13,125	1641,296
RC2	3,500	1524,661

Tabla 5-9: Mejor solución inicial previa a la ejecución de la búsqueda local.

Los resultados presentados en esta sección se muestran mediante nueve tablas. Cada una de ellas se corresponde con una variante del algoritmo de búsqueda local utilizado. Todas las tablas tienen una estructura similar: cada una de las filas representa una clase del juego de ensayo de Solomon; y en columnas, se incorpora información relativa al número total de vehículos (NV), distancia (D) y tiempo de ejecución en segundos (T). Además, para el número de vehículos y la distancia, se añade el valor máximo (NV_{\max} y D_{\max}), el mínimo (NV_{\min} y D_{\min}), la media y la desviación estándar (NV_{avg} y D_{avg}).

A continuación, se presentan los resultados obtenidos por cada una de las variantes y se realiza una breve descripción de los valores medios de vehículos, distancia total y tiempo de ejecución. Posteriormente, estos datos serán analizados con más detalle en la sección 5.4.2.

La **tabla 5-10** presenta los resultados obtenidos al ejecutar la variante $PVNS_{\text{Inter}}$, que utiliza únicamente operadores interruta. En este caso, los operadores interruta sólo consiguen mejorar la distancia recorrida; manteniendo el número de rutas que tenía la mejor solución inicial (ver **tabla 5-9**). Esto es debido a la limitada capacidad que poseen los operadores interruta para la reducción del número de rutas.

$PVNS_{\text{Inter}}$									
Clase	T	NV_{\min}	NV_{\max}	NV_{avg}		D_{\min}	D_{\max}	D_{avg}	
C1	38	10,000	10,000	10,000	$\pm 0,000$	908,599	908,599	908,599	$\pm 0,000$
C2	35	3,125	3,125	3,125	$\pm 0,000$	689,181	689,181	689,181	$\pm 0,000$
R1	52	13,333	13,333	13,333	$\pm 0,000$	1477,151	1483,098	1479,800	$\pm 2,715$
R2	68	3,182	3,182	3,182	$\pm 0,000$	1226,860	1228,905	1227,510	$\pm 0,938$
RC1	26	13,125	13,125	13,125	$\pm 0,000$	1622,369	1625,657	1623,778	$\pm 1,182$
RC2	48	3,500	3,500	3,500	$\pm 0,000$	1475,695	1478,765	1476,677	$\pm 1,414$

Tabla 5-10: Resultados obtenidos por la variante $PVNS_{\text{Inter}}$

En la **tabla 5-11** se recogen los resultados de la ejecución de la variante $PVNS_{Intra}$. Al igual que la variante anterior, en este caso tampoco se reduce el número de rutas; y la única mejora que aporta esta variante es la reducción de la distancia total recorrida. Comparando las variantes $PVNS_{Inter}$ y $PVNS_{Intra}$, se puede apreciar que las diferencias en distancias no son muy elevadas, dominando la versión que utiliza únicamente los operadores interruta; salvo en la clase C2. Por otro lado, en relación a los tiempos de ejecución, los valores son parecidos, aunque la variante $PVNS_{Intra}$ siempre tarda menos que la variante $PVNS_{Inter}$.

$PVNS_{Intra}$									
Clase	T	NV _{min}	NV _{max}	NV _{avg}		D _{min}	D _{max}	D _{avg}	
C1	23	10,000	10,000	10,000	± 0,000	912,703	912,703	912,703	± 0,000
C2	28	3,125	3,125	3,125	± 0,000	685,240	685,240	685,240	± 0,000
R1	24	13,333	13,333	13,333	± 0,000	1511,059	1511,789	1511,497	± 0,400
R2	44	3,182	3,182	3,182	± 0,000	1250,894	1250,899	1250,897	± 0,003
RC1	13	13,125	13,125	13,125	± 0,000	1635,053	1635,420	1635,200	± 0,201
RC2	31	3,500	3,500	3,500	± 0,000	1477,608	1478,793	1478,082	± 0,649

Tabla 5-11: Resultados obtenidos por la variante $PVNS_{Intra}$

La **tabla 5-12** muestra los resultados de la primera variante que incorpora los operadores de mejora de rutas propuestos en la presente tesis doctoral. En este caso, la variante $PVNS_{Min}$ combina el uso de los tres operadores de mejora definidos. Esta variante consigue reducir el número de rutas para las clases C2, R1 y RC1, manteniendo los valores iniciales para las clases C1 (cuyo número era inicialmente igual al mejor conocido), R2 y RC2. No obstante, pese a no mejorar el número de rutas en tres clases, los valores de distancia obtenidos son mejores que los reportados por las variantes $PVNS_{Inter}$ y $PVNS_{Intra}$ (aunque la desviación estándar es mayor que en las dos variantes anteriores).

$PVNS_{Min}$									
Clase	T	NV _{min}	NV _{max}	NV _{avg}		D _{min}	D _{max}	D _{avg}	
C1	71	10,000	10,000	10,000	± 0,000	899,722	904,519	902,165	± 2,276
C2	116	3,000	3,000	3,000	± 0,000	657,431	663,848	660,707	± 2,954
R1	150	13,167	13,250	13,217	± 0,045	1415,028	1439,635	1430,554	± 9,431
R2	201	3,182	3,182	3,182	± 0,000	1213,114	1219,907	1215,626	± 2,804
RC1	82	12,750	13,000	12,900	± 0,105	1583,040	1634,334	1597,327	± 21,536
RC2	125	3,500	3,500	3,500	± 0,000	1451,383	1460,713	1457,632	± 3,629

Tabla 5-12: Resultados obtenidos por la variante $PVNS_{Min}$

Por último, los tiempos empleados por esta variante son superiores a los de las dos variantes iniciales. Esto es debido a que los nuevos operadores realizan un proceso más complejo que los operadores tradicionales de intercambio de nodos y arcos.

La **tabla 5-13** muestra los resultados obtenidos por la variante $PVNS_{RrE}$. Dicha variante utiliza únicamente el operador RrE-opt. El objetivo en este caso consiste en analizar el comportamiento específico del único operador nuevo que tiene un comportamiento no determinista. A tenor de los resultados obtenidos, se puede confirmar que esta variante permite igualar a la anterior en cuanto a número de rutas en todas las clases, pero obtiene peores resultados en distancia en todas las clases salvo en la clase RC1. Además, los tiempos de proceso son sensiblemente superiores a los empleados por la variante $PVNS_{Min}$.

$PVNS_{RrE}$									
Clase	T	NV _{min}	NV _{max}	NV _{avg}		D _{min}	D _{max}	D _{avg}	
C1	77	10,000	10,000	10,000	± 0,000	900,873	904,919	903,224	± 1,549
C2	126	3,000	3,000	3,000	± 0,000	656,365	662,698	659,080	± 2,958
R1	157	13,167	13,250	13,184	± 0,037	1400,455	1432,692	1416,379	± 12,994
R2	249	3,091	3,182	3,164	± 0,041	1219,923	1233,117	1227,850	± 5,246
RC1	87	12,750	13,000	12,900	± 0,105	1564,251	1583,369	1573,650	± 7,743
RC2	138	3,500	3,500	3,500	± 0,000	1467,072	1491,324	1480,405	± 9,558

Tabla 5-13: Resultados obtenidos por la variante $PVNS_{RrE}$

En la **tabla 5-14** se muestran los resultados de la variante $PVNS_{Inter-Intra}$. En este caso se combina el uso de operadores interruta e intraruta. En cada iteración, primero se aplica un operador interruta y, posteriormente, un operador intraruta. Al igual que ocurre en las situaciones en que no se utilizan operadores de reducción del número de rutas, esta variante tampoco consigue reducir el número de rutas para ninguna de las clases de problemas.

$PVNS_{Inter-Intra}$									
Clase	T	NV _{min}	NV _{max}	NV _{avg}		D _{min}	D _{max}	D _{avg}	
C1	64	10,000	10,000	10,000	± 0,000	903,645	903,645	903,645	± 0,000
C2	70	3,125	3,125	3,125	± 0,000	678,771	678,771	678,771	± 0,000
R1	83	13,333	13,333	13,333	± 0,000	1466,902	1480,108	1474,467	± 4,596
R2	139	3,182	3,182	3,182	± 0,000	1180,805	1191,074	1186,407	± 3,526
RC1	44	13,125	13,125	13,125	± 0,000	1614,294	1618,911	1617,395	± 1,261
RC2	92	3,500	3,500	3,500	± 0,000	1408,727	1430,895	1416,322	± 7,725

Tabla 5-14: Resultados obtenidos por la variante $PVNS_{Inter-Intra}$

En relación a la distancia, los resultados obtenidos son mejores que los obtenidos por $PVNS_{Inter}$ y $PVNS_{Intra}$; pero no son mejores que los obtenidos cuando se utiliza algún operador de reducción del número de rutas. En relación al tiempo de ejecución, los tiempos siguen siendo inferiores a los que emplean las variantes que utilizan operadores de reducción del número de rutas.

La **tabla 5-16** muestra los resultados de la variante que combina los operadores de reducción del número de rutas junto con los operadores de mejora intraruta. Esta variante iguala o mejora en cuanto al número de rutas a las variantes que únicamente utilizan operadores de reducción del número de rutas ($PVNS_{Min}$ y $PVNS_{RrE}$) salvo en el caso de la clase R1 en la que la variante $PVNS_{RrE}$ obtiene un número de rutas inferior. En relación a la distancia, esta nueva variante supera a todas las analizadas hasta el momento. Los tiempos de ejecución de la variante $PVNS_{Min-Intra}$ son sensiblemente superiores, pero las diferencias no son notables respecto a las variantes $PVNS_{Min}$ y $PVNS_{RrE}$.

$PVNS_{Min-Intra}$									
Clase	T	NV _{min}	NV _{max}	NV _{avg}		D _{min}	D _{max}	D _{avg}	
C1	95	10,000	10,000	10,000	± 0,000	893,913	898,807	897,007	± 1,859
C2	139	3,000	3,000	3,000	± 0,000	648,972	662,408	653,798	± 5,768
R1	193	13,167	13,250	13,217	± 0,045	1405,429	1425,898	1417,730	± 8,399
R2	253	3,091	3,182	3,164	± 0,041	1191,694	1206,009	1197,608	± 6,516
RC1	87	12,750	12,875	12,850	± 0,056	1563,967	1595,861	1581,584	± 13,297
RC2	135	3,500	3,500	3,500	± 0,000	1415,807	1443,887	1431,892	± 11,175

Tabla 5-15: Resultados obtenidos por la variante $PVNS_{Min-Intra}$

Al igual que se hizo en el caso de la variante sencilla que sólo incluía operadores de reducción del número de rutas ($PVNS_{Min}$), la **tabla 5-17** muestra una variante muy parecida a la anterior ($PVNS_{Min-Intra}$), pero en este caso se utiliza únicamente el operador RrE-opt para analizar otra vez el impacto del comportamiento aleatorio de dicho operador. En ese sentido, se puede apreciar que el rendimiento de esta variante es comparable al de la variante anterior que utilizaba los tres operadores definidos. De hecho, la variante $PVNS_{Min-RrE}$ supera a la variante $PVNS_{Min-Intra}$ en cuanto a número de rutas en las clases R1 y RC1 igualando los resultados en las otras cuatro clases de problemas. Por otro lado, en cuanto a la distancia, los datos son prácticamente similares, siendo la diferencia más grande en torno al 1% en favor de la variante $PVNS_{Min-Intra}$. Finalmente, en relación a los tiempos de ejecución, al igual que ocurría al comparar las variantes $PVNS_{Min}$ y $PVNS_{RrE}$, la variante $PVNS_{RrE-Intra}$ requiere un tiempo sensiblemente superior al empleado por la variante $PVNS_{Min-Intra}$ para obtener los resultados.

$PVNS_{RrE-Intra}$									
Clase	T	NV _{min}	NV _{max}	NV _{avg}		D _{min}	D _{max}	D _{avg}	
C ₁	110	10,000	10,000	10,000	± 0,000	897,414	900,676	898,641	± 1,215
C ₂	181	3,000	3,000	3,000	± 0,000	650,213	657,088	654,221	± 2,492
R ₁	218	13,083	13,250	13,167	± 0,059	1403,462	1422,110	1411,899	± 7,744
R ₂	322	3,091	3,182	3,164	± 0,041	1213,910	1230,450	1220,848	± 7,145
RC ₁	145	12,750	12,875	12,800	± 0,068	1535,918	1560,996	1547,203	± 11,093
RC ₂	185	3,500	3,500	3,500	± 0,000	1443,033	1450,277	1447,155	± 3,765

Tabla 5-16: Resultados obtenidos por la variante $PVNS_{RrE-Intra}$

La **tabla 5-17** muestra los resultados de una de las últimas variantes. Esta variante combina todos los operadores, tanto los interruta, como los intraruta y los de reducción del número de rutas. Al igual que se explicó en la sección 5.2.6 el proceso completo del algoritmo incluye en cada iteración el uso de un operador interruta, un operador de reducción del número de rutas y, finalmente, un operador intraruta. Esta ordenación viene motivada por el hecho de focalizar primero el proceso en la reasignación de clientes (intentando reducir el número de rutas). Esta variante consigue unos buenos resultados, siendo la que domina tanto en número de rutas como en distancia en las clases C₁, C₂ y RC₂ (sombreadas en verde). En las otras tres clases (R₁, R₂ y RC₁) obtiene peores resultados que $PVNS_{All-RrE}$ tanto en número de vehículos como en distancia; siendo la diferencias siempre inferior a 0,6% en cuanto al número de vehículos. Finalmente, en relación a los tiempos de ejecución, los valores son similares a los que obtiene $PVNS_{All-RrE}$, y sensiblemente superiores a los de $PVNS_{Min-Intra}$ y $PVNS_{RrE-Intra}$.

$PVNS_{All}$									
Clase	T	NV _{min}	NV _{max}	NV _{avg}		D _{min}	D _{max}	D _{avg}	
C ₁	134	10,000	10,000	10,000	± 0,000	880,375	885,301	882,330	± 1,969
C ₂	192	3,000	3,000	3,000	± 0,000	642,619	653,125	648,362	± 4,324
R ₁	284	13,167	13,250	13,217	± 0,045	1373,873	1397,027	1385,989	± 10,548
R ₂	350	3,182	3,182	3,182	± 0,000	1126,619	1143,278	1138,364	± 6,714
RC ₁	143	12,625	12,875	12,825	± 0,112	1507,355	1573,081	1540,354	± 23,384
RC ₂	238	3,375	3,500	3,450	± 0,068	1349,531	1377,840	1363,978	± 11,228

Tabla 5-17: Resultados obtenidos por la variante $PVNS_{All}$

Para finalizar la presentación de los resultados obtenidos por las diferentes variantes del algoritmo de búsqueda local utilizado en la experimentación, la **tabla 5-18** muestra los valores de la última variante analizada. En esta caso se utilizan todos los tipos de

operadores (al igual que en $PVNS_{All}$), pero en relación a los operadores de reducción del número de rutas, se emplea únicamente el que tiene un comportamiento aleatorio (RrE-opt). Esta variante obtiene unos buenos resultados, superando a $PVNS_{All}$ en las clases R1, R2 y RC1, con unos tiempos de ejecución muy similares.

$PVNS_{All-RrE}$									
Clase	T	NV _{min}	NV _{max}	NV _{avg}		D _{min}	D _{max}	D _{avg}	
C1	134	10,000	10,000	10,000	± 0,000	882,173	886,522	883,588	± 1,737
C2	190	3,000	3,000	3,000	± 0,000	638,462	657,886	650,658	± 8,422
R1	260	13,083	13,250	13,167	± 0,059	1373,026	1390,708	1383,364	± 7,103
R2	350	3,091	3,182	3,164	± 0,041	1157,931	1175,218	1165,396	± 6,210
RC1	146	12,750	12,875	12,800	± 0,068	1512,671	1541,686	1526,135	± 10,794
RC2	229	3,500	3,500	3,500	± 0,000	1347,134	1413,412	1382,777	± 31,255

Tabla 5-18: Resultados obtenidos para la variante $PVNS_{All-RrE}$

Tras presentar los datos obtenidos por todas las variantes del algoritmo de búsqueda local utilizado para la validación de la idoneidad de los operadores de reducción del número de rutas, se incorporan dos tablas adicionales: **tabla 5-19** y **tabla 5-20**. Estas nuevas tablas muestran de forma agrupada los resultados obtenidos por las cuatro variantes que permiten reducir el número de rutas respecto a la mejor solución inicial.

	Inicial	$PVNS_{Min-Intra}$		$PVNS_{RrE-Intra}$		$PVNS_{All}$		$PVNS_{All-RrE}$	
C1	10,000	10,000	0,0%	10,000	0,0%	10,000	0,0%	10,000	0,0%
C2	3,125	3,000	-4,0%	3,000	-4,0%	3,000	-4,0%	3,000	-4,0%
R1	13,333	13,217	-0,9%	13,167	-1,2%	13,217	-0,9%	13,167	-1,2%
R2	3,182	3,164	-0,6%	3,164	-0,6%	3,182	0,0%	3,164	-0,6%
RC1	13,125	12,850	-2,1%	12,800	-2,5%	12,825	-2,3%	12,800	-2,5%
RC2	3,500	3,500	0,0%	3,500	0,0%	3,450	-1,4%	3,500	0,0%

Tabla 5-19: Resumen del número de vehículos obtenido por las mejores variantes.

Para cada una de las variantes, se incorpora el porcentaje de mejora respecto a los valores iniciales de número de vehículos y distancia (apareciendo sombreados en verde los mejores valores obtenidos).

	Inicial	$PVNS_{Min-Intra}$		$PVNS_{RrE-Intra}$		$PVNS_{All}$		$PVNS_{All-RrE}$	
C1	919,269	897,007	-2,4%	898,641	-2,2%	882,330	-4,0%	883,588	-3,9%
C2	694,764	653,798	-5,9%	654,221	-5,8%	648,362	-6,7%	650,658	-6,3%
R1	1523,614	1417,730	-6,9%	1411,899	-7,3%	1385,989	-9,0%	1383,364	-9,2%
R2	1289,546	1197,608	-7,1%	1220,848	-5,3%	1138,364	-11,7%	1165,396	-9,6%
RC1	1641,296	1581,584	-3,6%	1547,203	-5,7%	1540,354	-6,2%	1526,135	-7,0%
RC2	1524,661	1431,892	-6,1%	1447,155	-5,1%	1363,978	-10,5%	1382,777	-9,3%

Tabla 5-20: Resumen de la distancia obtenida por las mejores variantes.

Una vez presentados los resultados de la experimentación, y para que las conclusiones extraídas en la siguiente sección sean rigurosas y justas, se han realizado dos exámenes estadísticos (utilizando el número medio de vehículos y la distancia media como objetos de estudio) tomando como referencia las pautas marcadas en el trabajo realizado por Derrac y otros (Derrac, y otros 2011). En primer lugar, se ha realizado el test no paramétrico de Friedman para comparación múltiple. Con este test se ha querido comprobar si existen diferencias significativas entre las nueve variantes del algoritmo de búsqueda local analizadas ($PVNS_{XXX}$).

En la **tabla 5-21** se muestran los datos del ranking promedio obtenidos por cada una de las variantes analizadas (cuanto menor sea el valor obtenido, mejor es el rendimiento). Se ha realizado la prueba para los dos parámetros de la función objetivo (ver sección 5.2.1) debido a que las diferencias en cuanto al número de rutas no eran excesivas, tal y como ha podido observarse en las tablas que se han revisado anteriormente en esta misma sección.

Variante	Ranking promedio	Ranking promedio
	NV_{avg}	D_{avg}
$PVNS_{All-RrE}$	3,2500	1,6667
$PVNS_{RrE-Intra}$	3,2500	4,1667
$PVNS_{All}$	4,0833	1,3333
$PVNS_{RrE}$	4,1667	6,0000
$PVNS_{Min-Intra}$	4,2500	4,0000
$PVNS_{Min}$	5,2500	5,6667
$PVNS_{Inter-Intra}$	6,9167	5,6667
$PVNS_{Inter}$	6,9167	7,8333
$PVNS_{Intra}$	6,9167	8,6667

Tabla 5-21: Ranking promedio obtenido mediante el test de Friedman para todas las variantes.

En relación al examen realizado para la cantidad de vehículos, el estadístico resultante del test de Friedman ha sido 15,4444. Con este valor, y analizando una distribución χ^2 con ocho grados de libertad, se puede asegurar, con un nivel de confianza del 90%, que existen diferencias significativas entre las variantes estudiadas. No obstante, como se puede comprobar en el ranking, hay un empate tanto en el primer lugar ($PVNS_{All-RrE}$ y $PVNS_{RrE-Intra}$), como en el último ($PVNS_{Inter-Intra}$, $PVNS_{Inter}$ y $PVNS_{Intra}$). A modo de complemento de este primer test estadístico, faltaría indicar que el valor p computado del test de Friedman ha sido 0,051058.

Una vez descubierto que existen diferencias significativas entre las distintas variables, es adecuado realizar un examen, técnica por técnica, en relación al número de vehículos. Por ese motivo, se ha realizado un test post-hoc de Holm usando como referencia la variante $PVNS_{All-RrE}$ (que encabeza el ranking en cuanto a número de vehículos). Los resultados de este test se muestran en la **tabla 5-22**. Como se puede apreciar, en ningún caso los valores p ajustado y no ajustado son simultáneamente inferiores o iguales a 0,05. Por lo tanto, no se puede confirmar estadísticamente que las diferencias en cuanto al número de rutas sean significativas respecto a la variante $PVNS_{All-RrE}$.

Después de analizar los test estadísticos realizados para el número medio de vehículos, viendo que las diferencias no eran significativas, y que además existía un empate en la parte superior del ranking; se ha realizado un nuevo proceso de análisis estadístico centrado en la distancia total recorrida (el segundo nivel de la función objetivo). Este nuevo proceso incluye nuevamente la realización del test de Friedman y el test de Holm.

Variante	p no ajustado	p ajustado
$PVNS_{Inter}$	0,020395	0,163159
$PVNS_{Intra}$	0,020395	0,163159
$PVNS_{Inter-Intra}$	0,020395	0,163159
$PVNS_{Min}$	0,205903	1,029516
$PVNS_{Min-Intra}$	0,527089	2,108357
$PVNS_{RrE}$	0,562083	2,108357
$PVNS_{All}$	0,598161	2,108357
$PVNS_{RrE-Intra}$	1,000000	2,108357

Tabla 5-22: Valores p ajustados y no ajustados del test de Holm para el número de vehículos.

En relación al nuevo test de Friedman asociado a la distancia (los valores del ranking se recogen en la **tabla 5-21**), el resultado estadístico del nuevo test ha sido 39,688889.

Con este valor, y tomando con referencia la misma distribución χ^2 con ocho grados de libertad utilizada anteriormente, se podría elevar el nivel de confianza hasta un 99,5%, ya que el punto crítico en ese caso es 21,955. En este caso, el valor p computado ha sido 0,000004. Por lo tanto, pese a existir un “empate” en relación al número de vehículos, después de realizar el segundo test de Friedman se puede concluir que la variante $PVNS_{All_RrE}$ domina al resto, combinando el número de vehículos y la distancia recorrida.

Una vez realizado el nuevo test de Friedman, para evaluar la significancia estadística de las diferencias en relación a la distancia recorrida, se ha realizado un nuevo test post-hoc de Holm tomando ahora como control a la variante $PVNS_{All}$ (que es la que obtiene los mejores resultados en el ranking de distancia). Analizando los valores p ajustado y no ajustado de la **tabla 5-23**, se puede confirmar que las diferencias son significativas cuando dichos valores sean inferiores o iguales a 0,05. Teniendo en cuenta lo anterior, podría afirmarse con un nivel de confianza del 99,5% que la variante $PVNS_{All}$ domina de manera significativa (en relación a la distancia) a las variantes $PVNS_{Intra}$, $PVNS_{Inter}$, $PVNS_{RrE}$, $PVNS_{Min}$ y $PVNS_{Inter-Intra}$; no ocurriendo lo mismo en el caso de las variantes $PVNS_{RrE-Intra}$, $PVNS_{Min-Intra}$ y $PVNS_{All-RrE}$.

Para finalizar esta sección, y siempre de acuerdo a la experimentación realizada, podría concluirse que pese a existir diferencias en relación al número de rutas obtenidas por las variantes del algoritmo de búsqueda local analizadas, dichas diferencias no son estadísticamente significativas. Por otro lado, las diferencias en relación a la distancia total recorrida entre las variantes que combinan el uso de los dos tipos de operadores (tradicionales y de reducción del número de rutas) y las variantes que no utilizan operadores de reducción del número de rutas sí son significativas. No confirmándose este hecho al comparar entre sí todas las variantes que simultáneamente los dos tipos de operadores.

Variante	p no ajustado	p ajustado
$PVNS_{Intra}$	0,000004	0,000028
$PVNS_{Inter}$	0,000039	0,000276
$PVNS_{RrE}$	0,003163	0,018977
$PVNS_{Min}$	0,006132	0,030660
$PVNS_{Inter-Intra}$	0,006132	0,030660
$PVNS_{RrE-Intra}$	0,073140	0,219419
$PVNS_{Min-Intra}$	0,091690	0,219419
$PVNS_{All-RrE}$	0,833029	0,833029

Tabla 5-23: Valores p ajustados y no ajustados del test de Holm para la distancia total recorrida.

Por último, después de combinar los resultados de los rankings de número de rutas y distancia total recorrida, podría asegurarse que la variante $PVNS_{All-RrE}$ es la que obtiene los mejores resultados para la experimentación realizada.

5.4 Análisis de los resultados

Continuando con la revisión de los resultados obtenidos durante los dos procesos de experimentación, en esta sección se realiza un análisis más detallado de dichos resultados. Al igual que la sección anterior, esta sección divide el análisis de los resultados en dos secciones independientes: la primera de ellas hace referencia a la mejora en torno a la heurística de construcción para el VRPTW; y la segunda profundiza en el análisis de los resultados en torno a los operadores de mejora basados en la reducción del número de rutas.

5.4.1 Análisis de resultados de la heurística de construcción para el VRPTW

Después de mostrar los resultados obtenidos en la fase de experimentación en torno a las heurísticas de construcción para el VRPTW, en esta sección se analizan dichos resultados con un poco más de detalle. Este análisis se centrará únicamente en la calidad de la solución (de acuerdo a la función objetivo definida en la sección 5.2.1) y los tiempos de ejecución, ya que las técnicas utilizadas en la experimentación presentan un comportamiento determinista.

Comenzando por la calidad de la solución, tal y como se describió en la sección 5.3.1, las variantes de las heurísticas que mejores resultados obtienen son las que incorporan dos fases en el proceso de construcción de las rutas: la fase inicial de construcción y la fase final de mejora. Este resultado se justifica por el hecho de que todas las heurísticas analizadas son secuenciales, con lo cual, las decisiones que se toman durante el proceso de construcción se centran únicamente en una ruta, lo que limita la capacidad de exploración del espacio de soluciones. No obstante, de esta manera se puede controlar de una manera adecuada el tiempo necesario para generar la solución inicial.

Como se puede comprobar en las tablas que recogen los resultados de las diferentes variantes (**tabla 5-2**, **tabla 5-3**, **tabla 5-4**, **tabla 5-5**, **tabla 5-6** y **tabla 5-7**), y en la tabla resumen de los mejores resultados (**tabla 5-8**), la heurística $I1^{RC}$ es la que obtiene los mejores resultados para las clases C1 y C2. Además, analizando las otras dos heurísticas junto con la mejora propuesta en la presente tesis doctoral, se puede apreciar que dicha mejora también permite que la heurística $IRCI$ obtenga los mejores resultados en las dos clases con distribución agrupada de clientes. Esto está justificado por la naturaleza

propia de la heurística $I1$ de Solomon, que obtiene un buen rendimiento en problemas con distribución agrupada de clientes.

Por otro lado, en las clases de problemas con una distribución totalmente aleatoria y mixta, dominan las variantes que incorporan la mejora $IRCI$. En este sentido, la variante $IRCI_{IRCI}$ es la que mejores resultados obtiene para las clases $R2$, $RC1$ y $RC2$; mientras que la variante $IMPACT_{IRCI}$ es la que domina en la clase $R1$. Para estas clases, la mejora $I1^{RC}$ sí que permite igualar el número de rutas en la clase $R2$, pero en las otras tres clases, las diferencias se encuentran entre el 1% y el 4% (tanto en número de vehículos como en distancia total recorrida). Teniendo en cuenta estas diferencias, podría decirse que la mejora $I1^{RC}$ es capaz de obtener unos resultados aceptables en cuanto a calidad de la solución.

Continuando ahora con el análisis de los tiempos de ejecución, los mejores resultados los obtienen las variantes básicas o las que incorporan el proceso de $I1^{RC}$. De hecho, el proceso de mejora $I1^{RC}$ supone un incremento de unos tres segundos (de media en todas las clases) a las versiones simples de las heurísticas $I1$ e $IMPACT$; siendo el incremento respecto a la versión simple de la heurística $IRCI$ igual a 15 segundos. Por otro lado, el proceso de mejora basado en la heurística $IRCI$ supone un incremento (respecto a los tiempos de las versiones simples) de 384, 470 y 460 segundos a las versiones sencillas de las heurísticas $I1$, $IRCI$ e $IMPACT$ respectivamente. Estos incrementos resultan un tanto elevados, pensando que las variantes $I1_{IRCI}$, $IRCI_{IRCI}$ e $IMPACT_{IRCI}$ requieren un tiempo de ejecución que es incluso superior a los tiempos necesarios por cualquiera de las variantes de búsqueda local utilizadas en la experimentación descrita en la sección 5.3.2. Estos tiempos tan elevados están derivados de la complejidad que entraña el proceso de la heurística $IRCI$.

A modo de conclusión final, podría afirmarse que el proceso de construcción propuesto como aporte de la presente tesis doctoral ($I1^{RC}$) representa una alternativa prometedora como proceso de inicialización para una técnica heurística o metaheurística, ya que es capaz de mejorar los resultados obtenidos por la heurística $I1$ de Solomon (que es utilizada con frecuencia en las técnicas más avanzadas (Vidal, y otros 2013)) con un incremento de tiempo controlado. Además, esto se reafirma al comparar la heurística propuesta con otras heurísticas más complejas y que requieren un mayor tiempo de proceso, como es el caso de la heurística $IRCI$.

5.4.2 Análisis de resultados de los operadores de mejora para el VRPTW

En esta sección, y para no repetir de nuevo la información presentada anteriormente, se realizará únicamente el análisis de los resultados en cuanto a la calidad de la solución, el tiempo de ejecución y la robustez (entendida como la capacidad que tiene una técnica de resolución para obtener soluciones similares en todas las ejecuciones)

que presentan las diferentes variantes del algoritmo de búsqueda local utilizado en la experimentación. Este análisis pretende determinar si los operadores de mejora basados en la reducción del número de rutas presentan un comportamiento prometedor para la mejora de soluciones de problemas de tipo VRPTW.

Comenzando por la calidad de la solución, y de acuerdo a los dos criterios definidos en la función objetivo utilizada (ver sección 5.2.1), todas las variantes del algoritmo que utilizan operadores de reducción del número de rutas mejoran la solución inicial; y obtienen siempre mejores resultados que las variantes que no usan los nuevos operadores propuestos: $PVNS_{Inter}$, $PVNS_{Intra}$, y $PVNS_{Inter-Intra}$. Como es lógico, este hecho se debe a que los operadores tradicionales de intercambio de nodos y arcos no focalizan sus esfuerzos en la reducción del número de rutas. Por el contrario, como los nuevos operadores propuestos tienen como primer objetivo la reducción del número de rutas, siempre superan o igualan a los primeros en base al primer criterio de la función objetivo (número de rutas). Además, debido a las características inherentes del problema de tipo VRPTW, una solución con un menor número de rutas suele llevar asociada una reducción en la distancia recorrida. Por ese motivo, las soluciones generadas por las variantes que emplean alguno de los nuevos operadores propuestos obtienen mejores resultados también en relación a la distancia total recorrida (el segundo criterio de la función objetivo).

Continuando con el análisis de la calidad de la solución, los mejores resultados los obtienen las variantes que utilizan los tres tipos de operadores: $PVNS_{All-RrE}$ y $PVNS_{All}$. La justificación en este caso viene derivada de la combinación de la capacidad que ofrecen los diferentes operadores en cuanto a reasignación y reordenación de clientes.

Por último, se puede apreciar la superioridad del operador de reducción del número de rutas basado en la eliminación aleatoria de una ruta (RrE-opt) frente a los otros dos operadores propuestos. Este hecho se confirma al comparar las versiones que utilizan todos los operadores de reducción del número de rutas ($PVNS_{Min}$, $PVNS_{Min-Intra}$ y $PVNS_{All}$) con las versiones que utilizan únicamente el operador RrE-opt ($PVNS_{RrE}$, $PVNS_{RrE-Intra}$ y $PVNS_{All-RrE}$). En este caso, la justificación viene motivada por la bondad que tienen los procesos aleatorios en el campo de la optimización combinatoria.

En relación a los tiempos de ejecución, se puede apreciar claramente que todas las variantes que incorporan alguno de los operadores de reducción del número de rutas requieren un mayor tiempo de ejecución para la obtención de los resultados, llegando a ser las diferencias máximas de 23 a 134 entre las variantes $PVNS_{Intra}$ y $PVNS_{All-RrE}$ para la clase C1 en la que no se mejora el número de rutas. Esto supondría una diferencia del orden de 12 segundos (2,5 frente a 14,8) para la resolución de un único problema de la clase C1; siendo la mejora en distancia obtenida de un 4%. Con estos datos, y

pensando en la aplicación de los operadores propuestos a un problema del “mundo real”, la mejora en calidad podría compensar el incremento de tiempo.

Analizando la robustez de las diferentes variantes utilizadas, se puede observar que todas las variantes tienen un comportamiento robusto en relación al número de vehículos, puesto que la desviación típica es cero o prácticamente nula (inferior a 0,1) para todas las variantes y clases de problemas (en el 94% de las situaciones es cero o inferior a 0,1). Por lo tanto, se puede concluir que los operadores de reducción del número de rutas no afectan a la robustez de una técnica de búsqueda en relación al número de vehículos.

Por otro lado, si se analiza la robustez a partir de la observación de la desviación típica de la distancia recorrida, los valores obtenidos son claramente superiores en todas las variantes que utilizan operadores de reducción del número de rutas. En concreto, estas variantes presentan una desviación típica en la distancia total siempre superior a cero, llegando en algunos casos a 31,2 unidades (la variante $PVNS_{All-RrE}$ en la clase RC2); siendo 7,8 el valor más alto obtenido por las variantes que utilizan únicamente operadores tradicionales (la variante $PVNS_{Inter-Intra}$ en la clase RC2). Con todo lo anterior, podría afirmarse que los operadores de reducción del número de rutas pueden afectar negativamente a la robustez de la técnica de resolución (analizando esta desde el punto de vista de la distancia recorrida); en comparación con los operadores tradicionales de intercambio de nodos y arcos.

Poniendo el foco en el comportamiento de convergencia, se analizará el número de iteraciones que realiza cada una de las variantes para encontrar la mejor solución. La **tabla 5-24** recoge el número de iteraciones que utiliza cada una de las variantes durante su ejecución. Como se puede observar en esta tabla, en general no existen grandes diferencias en cuanto al número de iteraciones utilizadas por cada una de las variantes, si bien es cierto que las variantes que obtienen las mejores soluciones son las que necesitan más iteraciones para finalizar su proceso. Este hecho es razonable, y ayuda a justificar también las diferencias en cuanto al tiempo de ejecución ya que a mayor número de iteraciones, mayor será el tiempo que tarde la técnica de resolución.

Por último, como conclusión final, puede destacarse que los operadores de mejora propuestos como aporte de la presente tesis doctoral superan a los operadores clásicos de intercambio nodos y arcos al aplicarse a problemas de tipo VRPTW. De este modo, permiten la obtención de mejores soluciones en comparación con los anteriores, manteniendo el incremento del tiempo necesario en unos márgenes razonables. Por ese motivo, se puede confirmar que los nuevos operadores pueden ser un complemento perfecto a los operadores clásicos de intercambio de nodos y arcos para ser integrados en cualquiera de las heurísticas y metaheurísticas más avanzadas diseñadas específicamente para problemas de tipo VRPTW, como las que se recogen en (Vidal, y otros 2013) y (Vidal, y otros 2014).

Variante	C ₁	C ₂	R ₁	R ₂	RC ₁	RC ₂
<i>PVNS_{Intra}</i>	21,38	21,05	25,65	29,29	23,75	28,33
<i>PVNS_{Inter}</i>	21,07	21,20	20,72	22,05	21,03	22,45
<i>PVNS_{RrE}</i>	24,13	24,60	40,03	29,45	40,10	26,68
<i>PVNS_{Min}</i>	25,84	25,00	43,47	28,09	41,58	22,83
<i>PVNS_{Inter-Intra}</i>	22,26	22,60	24,98	24,98	30,77	29,23
<i>PVNS_{RrE-Intra}</i>	25,69	26,00	45,55	32,55	36,90	25,65
<i>PVNS_{Min-Intra}</i>	24,96	25,83	40,53	26,45	47,10	23,65
<i>PVNS_{All-RrE}</i>	25,47	27,20	42,85	32,02	44,80	31,00
<i>PVNS_{All}</i>	26,29	27,83	48,07	35,65	44,15	35,53
Promedio	24,12	24,59	36,87	28,95	36,69	27,26

Tabla 5-24: Número de iteraciones realizadas por cada una de las variantes.

6

Conclusiones y líneas de trabajo futuras

En los capítulos anteriores se han descrito todas las actividades llevadas a cabo durante la realización de la presente tesis doctoral. En primer lugar, en el capítulo introductorio se ha contextualizado la problemática a abordar, se ha incorporado la definición de las hipótesis y los objetivos que han guiado el trabajo realizado, y se ha expuesto la metodología utilizada. Seguidamente, en los capítulos 2 y 3 se ha ahondado en los problemas de generación de rutas de vehículos y las técnicas utilizadas para su resolución respectivamente. Una vez revisada la literatura de referencia, se han descrito de manera detallada los algoritmos que han servido para la verificación de las hipótesis y la consecución de los objetivos. En el quinto capítulo se ha realizado una pormenorizada descripción de la experimentación y un análisis de los resultados. Además, se han identificado las conclusiones que justifican la consecución de los objetivos y la verificación de las hipótesis.

Finalmente, después de completar el trabajo de investigación, en este último capítulo se incorporan las conclusiones y las líneas de trabajo futuras que han sido extraídas tras la realización de la presente tesis doctoral. El capítulo se ha organizado en torno a dos secciones. La sección 6.1 recoge las conclusiones obtenidas tras finalizar el trabajo. Dichas conclusiones están relacionadas con los aportes científicos realizados, y con la propia actividad de investigación en su concepto general. Por último, la sección 6.2 esboza las líneas de trabajo futuras, cuya realización se prevé abordar en un horizonte temporal cercano.

6.1 Conclusiones

La presente tesis doctoral posee dos contribuciones fundamentales: por un lado, la nueva heurística de construcción de soluciones iniciales para problemas de tipo VRPTW; y por otro lado, los operadores de mejora basados en la reducción del número de rutas.

La nueva heurística de construcción de soluciones iniciales para problemas de tipo VRPTW ($I1^{RC}$) es una heurística secuencial de dos fases: inicialización y mejora.

- La primera fase está basada en una adaptación de la heurística de $I1$ de Solomon (Solomon 1987), que es una de las más utilizadas en los procesos de construcción de la solución inicial por heurísticas y metaheurísticas aplicadas a problemas de tipo VRPTW. La modificación propuesta para la fase de inicialización consiste básicamente en la optimización del proceso de análisis de las opciones de inserción de cada cliente en base al uso de las listas de vecinos (ver sección 3.2.3.2) y conceptos de la búsqueda secuencial (ver sección 3.2.3.1). De esta forma, se limita el análisis de alternativas de inserción de un nuevo cliente en la ruta en construcción a las opciones en las que las ventanas de tiempo del nuevo cliente y el cliente (inmediatamente anterior o inmediatamente posterior) junto al que pretender ser insertado, son compatibles. Así, se reduce el tiempo de ejecución en instancias de problemas cuyos clientes poseen un elevado porcentaje de incompatibilidad de ventanas de tiempo, o con soluciones en las que las rutas poseen un número considerable de clientes.
- La segunda fase se denomina mejora y está inspirada en la heurística $IRCI$ (Figliozzi 2010). Dicha fase se basa en la reconstrucción de las rutas que están próximas entre sí. Para ello, se selecciona un conjunto de rutas en base a su proximidad, se extraen todos los clientes de las rutas seleccionadas, y posteriormente se realiza un nuevo proceso de construcción utilizando únicamente los clientes que han sido extraídos de las rutas seleccionadas. Así, se puede reducir el número inicial de rutas o mejorar la calidad de la solución. La novedad que aporta la heurística $I1^{RC}$ consiste en generar un sub-problema a partir de los clientes extraídos de las rutas seleccionadas y aplicar de nuevo la versión mejorada de la heurística $I1$ de Solomon a dicho sub-problema (usando un enfoque de “*divide y vencerás*”).

Tras un exhaustivo proceso de experimentación en el que se compara la heurística $I1^{RC}$ con las heurísticas de construcción de soluciones iniciales para problemas de tipo VRPTW más exitosas de la literatura (ver sección 5.3.1), los principales resultados que se han extraído en relación a la heurística de construcción $I1^{RC}$ son las siguientes:

- De acuerdo a la experimentación realizada, y atendiendo a la calidad de la solución obtenida, puede observarse que la heurística $I1^{RC}$ supera a la heurística $I1$ de Solomon obteniendo unos resultados comparables a los obtenidos por la heurística $IRCI$ (con diferencias que no superan en ningún caso el 4%).
- Por otro lado, en relación a los tiempos de ejecución, la nueva heurística propuesta supera ampliamente a la heurística $IRCI$; llegando a ser la relación de 1 a 350 en el mejor de los casos (a favor de la heurística $I1^{RC}$).

En conclusión, puede afirmarse que la heurística $I1^{RC}$ representa una alternativa prometedora como proceso de inicialización para una técnica heurística o metaheurística, ya que es capaz de mejorar los resultados obtenidos por la heurística $I1$ de Solomon, con un incremento de tiempo controlado. Además, esto último se reafirma al comparar la heurística propuesta con otras alternativas que pueden encontrarse en la literatura, como es el caso de la heurística $IRCI$. Esto hace que la heurística $I1^{RC}$ pueda convertirse en un complemento idóneo para las técnicas que basan el proceso de generación de la solución inicial en la heurística original de Solomon; como puede comprobarse al revisar alguna heurísticas y metaheurísticas que reportan los mejores resultados conocidos para los juegos de ensayo de referencia (SINTEF 2015).

Los operadores de mejora propuestos nacen con el objetivo de superar una de las limitaciones fundamentales que poseen los operadores tradicionales utilizados por las técnicas de búsqueda local: la capacidad para reducir el número de rutas. En este sentido, los operadores tradicionales basan su proceso en el intercambio de nodos o arcos dentro de una misma ruta, o entre pares de rutas. Este enfoque presenta una gran dificultad para reducir del número de rutas de una solución. Por ese motivo, los procesos de reducción del número de rutas suelen implementarse como técnicas de resolución específicas (Nagata y Bräysy 2009a) o como pasos adicionales que se llevan a cabo después de obtener una solución, tal y como se recoge en (Bräysy y Gendreau 2005a), (Bräysy y Gendreau 2005b) y (Gendreau y Tarantilis 2010). Los tres operadores propuestos (RcR-opt, SrE-opt y RrE-opt) responden, todos ellos, a la citada limitación que poseen los operadores tradicionales (ver sección 3.2.1). Para ello, se utilizan nociones sencillas de número de clientes que existen en una ruta, y conceptos de proximidad y lejanía respecto al “centro de gravedad” de una ruta. Los nuevos operadores propuestos basan su funcionamiento en la extracción de clientes de una ruta (utilizando el concepto de “grupos de expulsión”) para su reinserción posterior. La capacidad de reducción del número de rutas de estos operadores viene justificada por situaciones en la que al extraer clientes de una ruta, ésta se queda vacía. Siguiendo este enfoque se ha creado una familia o clase de operadores, que en el presente trabajo, se

materializa en tres operadores cuyas características fundamentales se detallan a continuación:

- El primero de los operadores definidos se denomina *Operador de reasignación de clientes alejados* (RcR-opt). Este operador se caracteriza por extraer de cada una de las rutas los n clientes más alejados de su centro de gravedad. De esta forma, si el número de clientes de una ruta es menor o igual al número de clientes extraídos, la ruta será eliminada. Una vez extraídos los clientes de todas las rutas, se realiza un proceso de reinserción, que culminará en cuanto todos los clientes estén asignados a una ruta.
- El segundo operador se diferencia del primero únicamente en el criterio utilizado para extraer los clientes de sus rutas originales. Este operador recibe el nombre de *Operador de Eliminación de Rutas Pequeñas* (SrE-opt), y su nombre se debe a que el proceso de extracción de clientes comienza eliminando la ruta más pequeña (con menor número de clientes). Posteriormente, al igual que los otros dos operadores propuestos, se procesan los clientes extraídos para insertarlos de nuevo en alguna ruta.
- El último de los operadores combina la eliminación de una ruta que propone el operador SrE-opt con un proceso aleatorio para la selección de la ruta a eliminar. Este operador recibe el nombre de *Operador Aleatorio de Eliminación de Rutas* (RrE-opt), y está inspirado en la técnica metaheurística de Nagata y Bräysy (Nagata y Bräysy 2009a), que es especialmente relevante por los resultados obtenidos. Una vez extraídos los clientes de la ruta eliminada, el proceso continúa de manera similar al realizado por los otros dos operadores.

Los tres operadores definidos han sido diseñados de forma totalmente modular, de tal forma que sus procesos fundamentales: extracción de clientes, reinserción y reconstrucción final (ver sección 4.2), puedan ser reemplazados de manera sencilla; y así generar variantes adicionales a las descritas en el presente documento.

Al igual que en el caso de la heurística $I1^{RC}$, los operadores han sido sometidos a un extenso proceso de experimentación para validar su bondad al ser integrados en una búsqueda local (ver secciones 5.2 y 5.3.2). En concreto se ha utilizado una variante de la búsqueda local paralela de vecindario variable en la que se ha combinado el uso de los nuevos operadores junto con operadores tradicionales de mejora inter e intraruta. A continuación se describen los resultados obtenidos en relación a los operadores de mejora:

- En base a la experimentación realizada, los nuevos operadores de mejora basados en la reducción del número de rutas han demostrado un comportamiento

adecuado en cuanto a capacidad de mejora y robustez al ser integrados en un proceso de búsqueda local.

- De los tres operadores, el operador RrE-opt es que obtiene mejores resultados, llegando a superar a las variantes de la búsqueda local que utilizaban los tres operadores de forma combinada. Este hecho, ratifica que el comportamiento aleatorio es especialmente interesante en el ámbito de la resolución de problemas de optimización combinatoria.
- Por otro lado, los tiempos de ejecución que aportan los nuevos operadores de mejora son superiores a los de los operadores clásicos basados en el intercambio de nodos y arcos. No obstante, el incremento en los tiempos de ejecución se mantiene siempre dentro de unos márgenes razonables.

Con todo lo anterior, es prudente concluir que los nuevos operadores de mejora basados en la reducción del número de rutas representan un complemento adecuado a los operadores tradicionales de intercambio de nodos y arcos. De esta forma, la reducción del número de rutas puede integrarse implícitamente, y de manera sencilla, en el proceso de resolución de cualquier heurística o metaheurística.

Un resultado de la experimentación sería el entorno de visualización y simulación de soluciones para problemas de asignación de rutas a vehículos (ver sección 4.3). Esta sencilla herramienta permite representar de manera visual tanto la distribución de los clientes que componen un problema como las rutas que forman parte de una solución. En ese sentido, puede servir de ayuda para abordar nuevas estrategias de resolución ya que permite visualizar de una forma clara e intuitiva los principales elementos que configuran una instancia de un problema y su solución, facilitando la identificación de nuevas estrategias de mejora.

A modo de cierre de esta sección, podría afirmarse que los algoritmos realizados por el presente trabajo de tesis doctoral representan unos complementos adecuados a las técnicas heurísticas y metaheurísticas de resolución de problemas de asignación de rutas de vehículos con restricción de ventanas de tiempo.

6.2 Líneas de trabajo futuras

Como se ha comentado en el capítulo inicial de este documento, el transporte es una actividad con una gran relevancia tanto económica, como social y medioambiental. Por ese motivo, las técnicas de resolución y optimización de problemas que surgen en este

ámbito, son y serán, fuente de interés científico. Con todo esto, el trabajo presentado en este documento representa el punto inicial de un proceso de investigación; y en ningún caso un punto final.

La primera actividad a realizar en cuanto se termine la redacción de este documento, es la difusión del trabajo realizado y los resultados obtenidos. Esta tarea se enlaza directamente con el objetivo personal que se auto-impuso el autor del presente trabajo de investigación: *Ofrecer libremente a la comunidad los aportes generados durante el trabajo de investigación*. Lamentablemente, este objetivo no ha podido ser cubierto a fecha de cierre del presente documento. El trabajo restante para la consecución de este objetivo consistirá en finalizar la documentación del código de acuerdo a las convenciones que propone Oracle¹² y elaborar un artículo en el que se plasmen tanto los algoritmos propuestos como las experimentación realizada y resultados obtenidos.

Los primeros pasos para dar visibilidad al trabajo realizado ya han sido dados, de hecho, ya existe un pre-acuerdo para publicar el trabajo, incluyendo los resultados completos y el código fuente, en la revista de acceso libre *Algorithms* (<http://www.mdpi.com/journal/algorithms>). Esta revista ha sido seleccionada debido a su carácter abierto y de libre acceso por parte de cualquier lector que así lo desee (sin restricciones de *Copyright*). La motivación fundamental para la elección de esta revista frente a otras alternativas, que tienen un mayor impacto en la comunidad científica, es la consecución del objetivo personal que ha perseguido el autor durante la realización de esta investigación.

Una vez terminado el trabajo de investigación, y a tenor de los resultados obtenidos, parece razonable continuar ahondando en la temática con objeto de mejorar y ampliar tanto los resultados obtenidos como el conocimiento adquirido. En ese sentido, se han definido una serie de líneas de trabajo, que se detallan a continuación:

- Una línea de trabajo, está relacionada con el análisis del rendimiento de los nuevos operadores de reducción del número de rutas en combinación con una metaheurística híbrida de características similares a las que reportan los mejores resultados conocidos para los juegos de ensayo de referencia en el contexto del VRPTW (SINTEF 2015). Asimismo, ese análisis debiera ser validado utilizando el juego de ensayo de Gehring y Homberger (ver sección 5.1.3) puesto que las metaheurísticas más novedosas lo utilizan como referencia en (Gendreau y Tarantilis 2010) o (Vidal, y otros 2014). De esta forma, se podría analizar también el impacto del tamaño de las instancias de los problemas en los tiempos de proceso que requieren los nuevos operadores.

¹² <http://www.oracle.com/technetwork/articles/java/index-137868.html> [consultado en septiembre de 2015]

En este sentido, y a raíz de una colaboración previa, el autor de la presente investigación y sus compañeros de investigación están en contacto con el profesor Xin-She Yang con quien han colaborado recientemente en la adaptación al VRP (aún en revisión) de su exitosa metaheurística conocida como algoritmo del murciélago (*Bat Algorithm*) (Yang 2010a). El profesor Yang, está en proceso de edición de un libro relacionado ingeniería inspirada en el comportamiento de la naturaleza, que recibirá el nombre de: “*Nature-Inspired Computation in Engineering*” y será publicado antes de final de 2015 por la editorial Springer. El autor de la presente tesis doctoral, junto con sus colegas de investigación, elaborará (en octubre de 2015) un artículo en el que se adaptará el algoritmo de las luciérnagas (*Firefly Algorithm*) (Yang 2010b) al problema de tipo VRPTW utilizando los operadores de mejora propuestos en la presente tesis doctoral. El citado artículo pretende poner de manifiesto las bondades de los aportes realizados en la presente tesis doctoral. Para ello, los nuevos operadores definidos, se integrarán en el algoritmo de las luciérnagas, que pese a su amplia difusión, todavía no ha sido adaptado a problemas de tipo VRPTW.

- Por otro lado, la estructura del nuevo tipo de operador propuesto (sección 4.2.2) contempla diferentes pasos en los que tienen cabida multitud de variantes en cuanto a procesos de reordenación, reinserción de clientes y reconstrucción final de rutas. En el trabajo presentado en este documento, se han seleccionado mecanismos sencillos y ampliamente referenciados en la literatura (como el operador Or-opt o la reinserción de clientes en la ruta más próxima geográficamente), pero la prueba con diferentes alternativas podría mejorar el rendimiento de los operadores tanto en calidad de la solución como en tiempo de ejecución.

Junto con la selección de alternativas para los diferentes procesos que contempla el nuevo tipo de operador, también se abordará de una manera más concienzuda, el estudio de la repercusión que tiene cada una de las fases en relación al resultado final del operador.

- La última línea de trabajo identificada, que realmente sería sencilla de llevar a cabo, consistiría en extender la aplicación del operador de mejora propuesto a otras variantes del VRP distintas del VRPTW siguiendo la tendencia de las técnicas de resolución multiatributo (*MAVRP*) que fueron presentadas en la sección 2.4. En este sentido, gracias a las buenas prácticas y patrones de diseño orientado a objetos incorporados en la implementación de los algoritmos realizados, la resolución de variantes del VRP que se correspondan con la clase EVAL podrían resolverse rápidamente. Bastaría con implementar una clase que incorpore las validaciones oportunas que garanticen que las asignaciones de clientes a rutas permiten la

generación de soluciones válidas (haciendo uso del patrón Strategy, referenciado en la nota al pie número 10 de la página 118).

En relación a esta última línea de trabajo se pretende modelar un problema de distribución diaria de periódicos como un caso especial de *MAVRP* que contemplará las siguientes características: existencia de varios almacenes centrales, ventanas de tiempo, una flota de vehículos heterogénea, junto con puntos de recogida y entrega no simultánea. En este caso, tanto la heurísticas de construcción como los nuevos operadores de mejora serán integrados en la metaheurística multipoblacional basada en conceptos futbolísticos denominada *Golden Ball* (Osaba, Díaz y Onieva 2014) y (Osaba, y otros 2014). Una vez finalizado este trabajo, los resultados serán publicados en una revista científica que será seleccionada entre las que aparecen de forma recurrente en la bibliografía que se incorpora en este documento.

Bibliografía

- Aarts, E., & Lenstra, J. (2003). *Local Search in Combinatorial Optimization* (Paperback Ed. ed.). Princeton University Press.
- Alba, E. (2005). *Parallel metaheuristics: a new class of algorithms*. Willey.
- Altinkemer, K., & Gavish, B. (1991). Parallel savings based heuristic for the delivery problem. *Operations Research*, 39(3), 456-469.
- Applegate, D., Bixby, R., Chvatal, V., & Cook, W. (2007). *The Traveling Salesman Problem: A Computational Study*. New Jersey: Princeton University Press.
- Atkinson, J. (1994). A Greedy Look-Ahead Heuristic for Combinatorial Optimization: An Application to Vehicle Scheduling with Time Windows. *The Journal of the Operational Research Society*, 45(6), 673-684.
- Babin, G., Deneault, S., & Laporte, G. (2007). Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem. *Journal of the Operational Research Society*, 58, 402-407.
- Badeau, P., Guertin, F., Gendreau, M., Potvin, J.-Y., & Taillard, E. (1997). A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 5(2), 109-122.
- Baker, E., & Schaffer, J. (1986). Solution improvement heuristics for the vehicle routing and scheduling problem with time-window constraints. *American Journal of Mathematical and Management Sciences*, 6(3-4), 261-300.
- Balakrishnan, N. (1993). Simple heuristics for the vehicle routing problem with soft time-windows. *Journal of the Operational Research Society*, 44(3), 279-287.
- Baldacci, R., Battarra, M., & Vigo, D. (2008). Routing a heterogeneous set of vehicles. En B. Golden, S. Raghavan, & W. Wasil (Edits.), *The Vehicle Routing Problem: Latest Advances and New Challenges* (págs. 3-27). Springer.
- Baldacci, R., Christofides, N., & Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2), 351-385.

- Baldacci, R., Toth, P., & Vigo, D. (2007). Recent advances in vehicle routing exact algorithms. *4OR*, 5(4), 269-298.
- Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G., & Stewart, W. R. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*(1), 9-32.
- Beasley, J. (1983). Route first - cluster second methods for vehicle routing. *Omega*, 4(11), 403-408.
- Bell, J., & McMullen, P. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1), 41-48.
- Bellmore, M., & Nemhauser, G. (1 de Junio de 1968). The Traveling Salesman Problem: A Survey. *Operations Research*, 16(3), 538-558.
- Berbeglia, G., Cordeau, J.-F., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1), 8-15.
- Blocho, M., & Czech, Z. (2011). An Improved Route Minimization Algorithm for the Vehicle Routing Problem With Time Windows. *Studia Informatica*, 32(3B), 5-19.
- Blum, C., & Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys (CSUR)*, 35(3), 268-308.
- Blum, C., Puchinger, J., Raidl, G., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6), 4135-4151.
- Bodin, L. (1975). A taxonomic structure for vehicle routing and scheduling problems. *Computers & Urban Society*, 1(1), 11-29.
- Bodin, L. (1983). Solving large vehicle routing and scheduling problems in small core. *Proceedings of the ACM '83 Conference*, 27-37.
- Bodin, L., & Berman, L. (1979). Routing and scheduling of school buses by computer. *Transportation Science*, 2(13), 113.
- Bodin, L., Golden, B., Assad, A., & Ball, M. (1983). Routing and scheduling of vehicles and crews: the state of the art. *Computers and Operations Research*, 10(2).
- Bräysy, O. (2003). A Reactive Variable Neighborhood Search for the Vehicle Routing Problem with Time Windows. *Journal on Computing*, 15, 347-368.

- Bräysy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *INFORMS Transportation Science*(39), 104-118.
- Bräysy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows, Part II : Metaheuristics. *Transportation Science*, 39(1), 119-139.
- Bräysy, O., Berger, J., Barkaoui, M., & Dullaert, W. (2003). A Threshold Accepting Metaheuristic for the Vehicle Routing Problem with Time Windows. *Central European Journal of Operations Research*, 11(4), 369-387.
- Bullnheimer, B., Hartl, R., & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89, 319-328.
- Campbell, A., & Savelsbergh, M. (8 de 2004). Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems. *Transportation Science*, 38(3).
- Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Optimization Theory and Applications*, 45, 41-51.
- Chen, S., Golden, B., & Wasil, E. (2007). The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, 49(4), 318-329.
- Christofides, N. (1976). The Vehicle Routing Problem. *Operations Research*, 10, 55-70.
- Christofides, N. (1985). Vehicle routing. En E. Lawler, J. Lenstra, A. Rinnooy Kan, & D. Shmoys, *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization* (págs. 431-448). Chicester: Wiley.
- Christofides, N., & Eilon, S. (1972). Algorithms for Large-Scale Travelling Salesman Problems. *Operational Research Quarterly (1970-1977)*, 23(4), 511-518.
- Christofides, N., Mingozzi, A., Toth, P., & Sandi, C. (1979). *Combinatorial Optimization*. Chicester: Wiley.
- Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *INFORMS Operations Research*, 4(12), 568-581.
- Coleho, L., Renaud, J., & Laporte, G. (2015). *Road-Based Goods Transportantion: A survey of Real-World Applications from 2000 to 2015*. CIRRENT.
- Cordeau, J.-F., & Maischberger, M. (2012). A Parallel Iterated Tabu Search Heuristic for Vehicle Routing Problems. *Computers & Operations Research*, 39(9), 2033-2050.

- Cordeau, J.-F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic multi-vehicle dial-a-ride problem. *Networks*, 30(2), 105-119.
- Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J. (2005). New heuristics for the vehicle routing problem. En A. Langevin, & D. Riopel, *Logistics systems: design and optimization* (págs. 279-297). Springer.
- Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*(53), 512-522.
- Cordeau, J.-F., Laporte, G., & Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of Operational Research Society*, 52(8), 928-936.
- Cordeau, J.-F., Laporte, G., & Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. En B. Golden, S. Raghavan, & E. Wasil (Edits.), *The Vehicle Routing Problem: Latest Advances and New Challenges* (Vol. 43, págs. 327-357). USA: Springer US.
- Cordeau, J.-F., Laporte, G., Savelsbergh, M., & Vigo, D. (2007). Vehicle Routing. *Handbooks in Operations Research and Management Science: Transportation*, 14, 367-428.
- Corne, D., Dorigo, M., & Glover, F. (1999). *New ideas in optimisation*. Maidenhead, England: MacGraw-Hill.
- Crainic, T., & Toulouse, M. (2010). Parallel meta-heuristics. En M. Gendreau, & J.-Y. Potvin, *Handbook of Metaheuristics* (págs. 497-541). Boston, MA, USA: Springer US.
- Creput, J.-C., & Koukam, A. (2008). Self-organization in evolution for the solving of distributed terrestrial transportation problems. En B. Prasad, *Soft Computing Applications in Industry, Studies in Fuzziness and Soft Computing* (págs. 189-205). Heidelberg: Springer Berlin.
- Croes, G. (1958). A method for solving large scale symmetric traveling salesman problems to optimality. *Operations Research*, 6, 791-812.
- Dantzig, G. B., & Ramser, R. H. (1959). The Truck Dispatching Problem. *Management Science*(6), 80-91.
- Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America*, 2(4), 393-310.

- Dassault Systèmes. (2015). *VRPTW World Records*. Recuperado el 03 de 09 de 2015, de <http://www.quintiq.com/optimization/vrptw-world-records.html>
- Davendra, D. (2010). *Traveling Salesman Problem, Theory and Applications*. (D. Davendra, Ed.) Rijeka, Croatia: InTech.
- De Jaegere, N., Defraeye, M., & Van Nieuwenhuyse, I. (2014). *The vehicle routing problem: state of the art classification and review*. LEUVEN Research Center for Operations Management, Faculty of Economics and Business.
- Derigs, U., & Kaiser, R. (2007). Applying the attribute based hill climber heuristic to vehicle routing problem. *European Journal of Operational Research*, 177(2), 719-732.
- Derrac, J., Garcia, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1), 3-18.
- Desrochers, M., & Verhoog, T. (1989). *A matching-based savings algorithm for the vehicle routing problem*. Technical report.
- Dongarra, J. (1998). *Performance of various computers using standard linear equation software. Report CS-89-85*. Tennessee, U.S.A.: University of Tennessee, Department of Computer Science.
- Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.
- Eksioğlu, B., Vural, A., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4), 1472-1483.
- Figliozzi, M. (2010). An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. *Transportation Research Part C: Emerging Technologies*, 18(5), 668-679.
- Fisher, M., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 2(39), 109-124.
- Francis, P., Smilowitz, K., & Tzur, M. (2008). The period vehicle routing problem and its extensions. En B. Golden, S. Raghavan, & E. Wasil (Edits.), *The Vehicle Routing Problem: Latest Advances and New Challenges* (págs. 73-102). Springer.
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32, 674-701.

- Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11, 86-92.
- Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M., Reis, M., Uchoa, E., & Werneck, R. (2006). Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 106(3), 491-511.
- Gaskell, T. (1967). Bases for vehicle fleet scheduling. *Operational Research Quarterly*, 3(18), 281-295.
- Gendreau, M., & Potvin, J.-Y. (2005). Metaheuristics in Combinatorial Optimization. *Annals of Operations Research*, 140(1).
- Gendreau, M., & Potvin, J.-Y. (2010). *Handbook of Metaheuristics* (I ed.). Springer.
- Gendreau, M., & Tarantilis, C. (2010). Solving large-scale vehicle routing problems with time windows: The state-of-the-art.
- Gendreau, M., Bräysy, O., Potvin, J.-Y., Hasle, G., & Lokketangen, A. (2008). Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. En B. Golden, S. Raghavan, & E. Wasil, *The Vehicle Routing Problems: Latest Advances and New Challenges* (págs. 143-169). New York, USA: Springer.
- Gendreau, M., Hertz, A., & Laporte, G. (1992). A new insertion and postoptimization procedures for the traveling salesman problem. *INFORMS Operations Research*(40), 1086-1093.
- Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10), 1276-1290.
- Gendreau, M., Laporte, G., & Potvin, J.-Y. (2002). Metaheuristics for the capacitated VRP. En P. Toth, & D. Vigo, *The vehicle routing problems. Monographs on Discrete Mathematics and Applications* (págs. 129-154). Philadelphia, USA: SIAM.
- Ghaziri, H. (1996). Supervision in the self-organizing feature map: Application to the vehicle routing problem. En I. Osman, & J. Kelly, *Meta-heuristics: Theory & applications* (págs. 651-660). Boston, MA, USA: Kluwer.
- Gillett, B., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 2(22), 340-349.
- Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8, 533-549.

- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 15(5), 533-549.
- Glover, F. (1996). Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65, 223-253.
- Glover, F., & Kochenberger, G. (2003). *Handbook of Metaheuristics*. Norwell, Massachusetts, USA: Kluwer Academic Publisher.
- Goel, A. (2010). Truck driver scheduling in the European Union. *Transportation Science*, 44(4), 429-441.
- Goel, A., & Kok, L. (2011). Truck driver scheduling in the united states. *Transportation Science*, 46(3), 317-326.
- Golden, B. (1977). Evaluating a sequential vehicle routing algorithm. *AIIE Transactions*, 9, 204-208.
- Golden, B., & Assad, A. (1988). *Vehicle routing: methods and studies* (Vol. 16). (North-Holland, Ed.) Amsterdam: Elsevier.
- Golden, B., Raghavan, S., & Wasil, E. (2008). *The vehicle Routing Problem: Latest Advances and New Challenges*. USA: Springer-US.
- Groër, C., & Golden, B. (2011). A Parallel Algorithm for the Vehicle Routing Problem. *Journal on Computing*, 23(2), 315-330.
- Gulczynski, D., Golden, B., & Wasil, E. (2008). Recent Developments in Modeling and Solving the Split Delivery Vehicle Routing Problem. *Tutorials in Operations Research: State-of-the-Art Decision-Making Tools in the Information-Intensive Age*, 170-180.
- Gulczynski, D., Golden, B., & Wasil, E. (2011). The period vehicle routing problem: New heuristics and real-world variants. *Transportation Research Part E: Logistics and Transportation Review*, 47(5), 648-668.
- Hashimoto, H., & Yagiura, M. (2008). A Path Relinking Approach with an Adaptive Mechanism to Control Parameters for the Vehicle Routing Problem with Time Windows. *EvoCOP'o8 Proceedings of the 8th European conference on Evolutionary computation in combinatorial optimization*. 4972, págs. 254-265. Napoles: Springer.
- Holland, J. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence* (Nueva edición (1 de julio de 1992) ed.). Mit University Press Group Ltd.

- Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6, 65-70.
- Hooker, J. N. (1995). Testing heuristics: we have it all wrong. *Journal of Heuristics*(1), 33-42.
- Ioannou, G., Kritikos, M., & Prastacos, G. (2001). A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society*(52), 523-537.
- Irnich, S. (2008a). A Unified Modeling and Solution Framework for Vehicle Routing and Local Search-Based Metaheuristics. *Journal on Computing*, 20(2), 270-287.
- Irnich, S. (2008b). Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*, 30(1), 113-148.
- Irnich, S., Funke, B., & Grünert, T. (2006). Sequential search and its application to vehicle-routing problems. *Computers & Operations Research*, 33(8), 2405-2429.
- Jin, J., Crainic, G., & Lokketagen, A. (2012). A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European Journal of Operational Research*, 222(2).
- Joubert, J., & Claasen, S. (2006). A sequential insertion heuristic for the initial solution to a constrained vehicle routing problem. *ORiON: The Journal of The Operations Research Society of South Africa*, 22(1), 105-116.
- Kindervater, G., & Savelsbergh, M. W. (1997). Vehicle Routing: Handling edge exchanges. En E. H. Aarts, & J. K. Lenstra (Edits.), *Local Search in Combinatorial Optimization* (págs. 337-360). Princeton University Press.
- Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simulated annealing: Quantitative studies. *Science*, 220(4598), 671-680.
- Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science*, 43(4), 408-416.
- Lawler, E., Lenstra, J., Rinnooy Kan, A., & Shmoys, D. (1985). *The Traveling Salesman Problem : A guided tour of combinatorial optimization*. Hoboken, USA: Wiley.
- Lenstra, J. K., & Kan, A. H. (1981). Complexity of vehicle routing and scheduling problems. *INFORMS Networks*(11), 221-227.
- Li, F., Golden, B., & Wasil, E. (2007). The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & Operations Research*, 34(10), 2918-2930.

- Lim, A., & Zhang, X. (2007). A Two-Stage Heuristic with Ejection Pools and Generalized Ejection Chains for the Vehicle Routing Problem with Time Windows. *Journal on Computing*, 19(3), 443-457.
- Lin, S. (1965). Computer solutions to the traveling salesman problem. *Bell System Technical Journal*(44), 2245-2269.
- Lin, S., & Kernighan, B. (1973). An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research*, 21(2), 498-516.
- Lourenço, H., Martin, O., & Stützle, T. (2010). Iterated Local Search: Framework and Applications. En M. Gendreau, & J.-Y. Potvin, *Handbooks of Metaheuristics, International Series in Operations Research & Management Science* (Vol. 146, págs. 363-397). Boston, MA, USA: Springer.
- Marinakis, Y., & Marinaki, M. (2010). A Hybrid Genetic - Particle Swarm Optimization Algorithm for the Vehicle Routing Problem. *Expert Systems with Applications*, 32(7), 1446-1455.
- Marinakis, Y., & Marinaki, M. (2011). Bumble bees mating optimization algorithm for the vehicle routing problem. En B. Panigrahi, Y. Shi, M.-H. Lim, L. Hiot, & Y. Ong, *Handbook of Swarm Intelligence, Adaptation, Learning, and Optimization* (Vol. 8, págs. 347-369). Heidelberg: Springer Berlin.
- Masutti, T., & De Castro, L. (2008). A neuro-immune algorithm to solve the capacitated vehicle routing problem. En P. Bentley, D. Lee, & S. Jung, *Artificial Immune Systems, Lecture Notes in Computer Science* (Vol. 5132, págs. 210-219). Heidelberg: Springer Berlin.
- Mole, R., & Jameson, S. (1976). A sequential route-building algorithm employing a generalised savings criterion. *Operational Research Quarterly*, 27, 503-511.
- Moscato, P., & Cotta, C. (2010). A modern introduction to memetic algorithms. En M. Gendreau, & J.-Y. Potvin, *Handbook of Metaheuristics* (págs. 141-183). Springer.
- Mühlenbein, H., Gorges-Schleuter, M., & Krämer, O. (1988). Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7(1), 65-85.
- Nagata, Y., & Bräysy, O. (2009a). A Powerful Route Minimization Heuristic for the Vehicle Routing Problem with Time Windows. *Operations Research Letters*, 37, 333-338.
- Nagata, Y., & Bräysy, O. (2009b). Edge Assembly based Memetic Algorithm for the Capacitated Vehicle Routing Problem. *Networks*, 54(4), 205-215.

- Nagata, Y., Bräysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4), 724-737.
- Nelson, M., Nygard, K., Griffin, J., & Shreve, W. (1988). Implementation techniques for the vehicle routing problem. *Computers and Operations Research*, 12(2), 273-283.
- Newton, R., & Thomas, W. (1974). Bus routing in a multi-school system. *Computers & Operations Research*, 2(1), 213-222.
- Northeastern University. (2005). *Solomon's VRPTW Bechmark Problems*. Recuperado el 26 de 7 de 2015, de <http://web.cba.neu.edu/~msolomon/problems.htm>
- Nygard, K., Greenberg, P., Bolkan, W., & Swenson, E. (1988). Generalized assignment methods for the deadline vehicle routing problem. En B. Golden, & A. Assad (Edits.), *Vehicle Routing: Methods and Studies* (págs. 107-126). Amsterdam: North-Holland.
- Ombuki-Berman, B., & Hanshar, T. (2009). Using genetic algorithms for multi-depot vehicle routing. En F. Pereira, & J. Tavares (Edits.), *Bio-inspired Algorithms for the Vehicle Routing Problem* (págs. 77-99). Springer.
- Or, I. (1976). *Travelling salesman-type combinatorial problems and their relation to the logistics of blood banking*. Ph.D. Thesis. Evanston, Illinois: Northwestern University, Department of Industrial Engineering and Management Sciences.
- Osaba, E., & Carballedo, R. (2012). A Methodological Proposal to Eliminate Ambiguities in the Comparison of Vehicle Routing Problem Solving Techniques. *International Conference on Evolutionary Computation Theory and Applications* (págs. 310-313). Barcelona: Springer.
- Osaba, E., Carballedo, R., Diaz, F., Onieva, E., & Perallos, A. (2014). A proposal of good practice in the formulation and comparison of meta-heuristics for solving routing problems. *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14* (págs. 31-40). Bilbao: Springer.
- Osaba, E., Diaz, F., & Onieva, E. (2014). Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts. *Applied Intelligence*, 41(1), 145-166.
- Osaba, E., Onieva, E., Diaz, F., Carballedo, R., & Perallos, A. (2014). Focusing on the Golden Ball Metaheuristic: An Extended Study on a Wider Set of Problems. *The Scientific World Journal*, 41.

- Osman, I., & Laporte, G. (1996). Metaheuristics: A Bibliography. *Annals of Operations Research*, 63, 513-623.
- Parragh, S., Doerner, K., & Hartl, R. (2008a). A survey on pickup and delivery problems Part I: Transportation between customers and depot. *Journal f ur Betriebswirtschaft*, 58(1), 21-51.
- Parragh, S., Doerner, K., & Hartl, R. (2008b). A survey on pickup and delivery problems Part II: Transportation between pickup and delivery locations. *Journal f ur Betriebswirtschaft*, 58(2), 81-117.
- Perboli, G., Pezzella, F., & Tadei, R. (2008). EVE-OPT: a hybrid algorithm for the capacitated vehicle routing problem. *Mathematical Methods of Operations Research*, 68(2), 361-382.
- Pintea, C.-M. (2014). *Advances in Bio-inspired Computing for Combinatorial Optimization Problems* (Intelligent Systems Reference Library ed.). London, UK: Springer.
- Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8), 2403-2435.
- Potvin, J.-Y. (2009). State-of-the art Review: Evolutionary Algorithms for Vehicle Routing. *INFORMS Journal on Computing*, 21(4), 518-548.
- Potvin, J.-Y., & Rousseau, J.-M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66, 331-340.
- Potvin, J.-Y., & Rousseau, J.-M. (1995). An exchange heuristic for routing problems with time windows. *Journal of Operational Research Society*(46), 1433-1446.
- Prescott-Gagnon, E., Desaulniers, G., & Rousseau, L. (2009). A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4), 190-204.
- Prins, C. (2004). A Simple and E ffective Evolutionary Algorithm for the Vehicle Routing Problem. *Computers & Operations Research*, 31(12), 1985-2002.
- Prins, C. (2009). GRASP - evolutionary local search hybrid for the vehicle ruoting problem. En F. Pereira, & J. Tavares, *Bio-Inspired Algorithms for the Vehicle Routing Problem* (págs. 35-53). Heidelberg, Alemania: Springer-Verlag.
- Raidl, R., Puchinger, J., & Blum, C. (2010). Metaheuristic hybrids. En M. Gendreau, & J.-Y. Potvin, *Handbook of Metaheuristics* (págs. 469-496). Boton, MA, USA: Springer US.

- Rego, C. (2001). Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms. *Parallel Computing*, 27(3), 201-222.
- Rego, C. (2001). Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms. *Parallel Computing*, 27(3), 201-222.
- Rego, C., & Roucairol, C. (1996). A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem. En I. Osman, & J. Kelly (Edits.), *Meta-Heuristics: Theory and Applications* (págs. 661-675). Norwell, Massachusetts, USA: Kluwer Academic Publishers.
- Reinelt, G. (1994). *The Traveling Salesman Computational Solutions for TSP Applications*. Berlin: Springer-Verlag.
- Repoussis, P., Tarantilis, C., & Ioannou, G. (2009). Arc-Guided Evolutionary Algorithm for the Vehicle Routing Problem With Time Windows. *IEEE Transactions on Evolutionary Computation*, 13(3), 624-647.
- Resende, M., Ribeiro, C., Glover, F., & Marti, R. (2010). Scatter search and path-relinking: Fundamentals, advances, and applications. En M. Gendreau, & J.-Y. Potvin, *Handbook of Metaheuristics. International Series in Operations Research & Management Science* (págs. 87-107). Boston, MA, USA: Springer US.
- Rochat, Y., & Taillard, E. (1995). Probabilistic Diversification and Intesification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1(1), 147-167.
- Russell, A. (1995). Hybrid Heuristics for the Vehicle Routing Problem With Time Windows. *Transportation Science*, 29(2), 156-166.
- Russell, S., & Norving, P. (2013). *Artificial Intelligence: A Modern Approach* (3.^a Edición (Intenational Ed.) ed.). New York, New Jersey, USA: Pearson Education Inc.
- Savelsbergh, M. (1985). Local Search in Routing Problems with Time Windows. *Annals of Operations Research*, 4(1), 285-305.
- Savelsbergh, M. (1990). A parallel insertion heuristic for vehicle routing with side-constraints. *Statistica Neerlandica*, 44(3), 139-148.
- Savelsbergh, M. (1992). The vehicle routing problem with time windows: Minimizing route duration. *Journal of Computing*, 4(1), 146-154.
- Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. En M. Maher, & J.-F. Puget, *Principle and Practice of Constraint Programming - CP98, Lecture Notes in Computer Science* (Vol. 1520, págs. 417-431). 1998: Springer Berlin.

- SINTEF. (2015). *VRPTW - Benchmarks and best known solutions*. Recuperado el 6 de 8 de 2015, de SINTEF Applied Mathematics, Department of Optimization. Norway: <https://www.sintef.no/vrptw>
- Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35, 254-265.
- Solomon, M., Baker, E., & Schaffer, J. (1988). Vehicle Routing and Scheduling Problems with Time Windows: Efficient Implementations of Solution Improvement Procedures. En B. Golden, & A. Assad, *Vehicle Routing: Methods and Studies* (Vol. 16, págs. 85-105). New York, USA: North-Holland.
- Taillard, E. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8), 661-673.
- Taillard, E., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J.-Y. (1997). A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science*, 31, 170-186.
- Taillard, E., Laporte, G., & Gendreau, M. (1995). Vehicle Routing With Multiple Use Of Vehicles. *The Journal of the Operational Research Society*, 47(8), 1065-1070.
- Tarantilis, C. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research*, 32(9), 2309-2327.
- Tarantilis, C., Zachariadis, E., & Kiranoudis, C. (2007). A guided tabu search for the heterogeneous vehicle routing problem. *Journal of the Operational Research Society*, 59(12), 1659-1673.
- The European Commission. (2015). *Transport sector economic analysis*. Recuperado el 03 de 09 de 2015, de The European Commission's JOINT RESEARCH CENTRE: <https://ec.europa.eu/jrc/en/research-topic/transport-sector-economic-analysis>
- Toth, P., & Vigo, D. (2002a). *The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications* (1.^a Ed. ed.). Philadelphia, USA: SIAM.
- Toth, P., & Vigo, D. (2002b). VRP with backhauls. En P. Toth, & D. Vigo (Edits.), *The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications* (págs. 195-224). Philadelphia: SIAM.
- Toth, P., & Vigo, D. (2015). *The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications* (2.^a Ed. ed.). New Jersey, USA: SIAM.

- Toulouse, M., Crainic, T., & Gendreau, M. (1996). Communication Issues in Designing Cooperative Multi Thread Parallel Searches. En I. Osman, & J. Kelly, *Meta-Heuristics: Theory & Applications* (págs. 501-522). Norwell, MA, USA: Kluwer Academic Publishers.
- Tyagi, M. (1968). A practical method for the truck dispatching problem. *Journal of the Operations Research Society of Japan*, 10, 76-92.
- Van Breedam, A. (1994). *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selection of Problems with Vehicle-Related, Customer-Related, and Time-Related Constraints*. Antwerp, Belgium: University of Antwerp.
- Van Landeghem, H. (1988). A bi-criteria heuristic for the vehicle routing problem with time-windows. *European Journal of Operational Research*, 36, 217-226.
- Vidal, T., Crainic, T., Gendreau, M., & Prins, C. (2013a). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with timewindows. *Computers & Operations Research*, 40(1), 475-489.
- Vidal, T., Crainic, T., Gendreau, M., & Prins, C. (2013b). Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1-21).
- Vidal, T., Crainic, T., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3), 658-673.
- Vidal, T., Crainic, T., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A Hybrid Genetic Algorithm for Multi-Depot and Periodic Vehicle Routing Problems. *Operations Research*, 60(2), 611 - 624.
- Voudouris, C., & Tsang, E. (1999). Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2), 469-499.
- Webb, M. (1972). Relative performance of some sequential methods of planning multiple delivery journeys. *Operational Research Quarterly*, 23(2), 361-372.
- Whittle, I., & Smith, G. (2004). The attribute based hill climber. *Journal of Mathematical Modelling and Algorithms*, 21(2), 281-283.
- Yang, X.-S. (2010a). A New Metaheuristic Bat-Inspired Algorithm. En J. Gonzalez, D. Pelta, C. Cruz, G. Terrazas, & N. Krasnogor, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)* (págs. 65-74). Berlín, Alemania: Springer Berlin Heidelberg.

- Yang, X.-S. (2010b). Firefly Algorithm. En *Engineering Optimization* (págs. 221-230). Hoboken, NJ, USA: John Wiley & Sons, Ltd.
- Yellow, P. (1970). A Computational Modification to the Savings Method of Vehicle Scheduling. *Operational Research Quarterly*, 2(21), 281-283.
- Yu, B., Yang, Z., & Yao, B. (2009). An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, 196(1), 171-176.
- Zachariadis, E., & Kiranoudis, C. (2010). A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computer & Operations Research*, 37(12), 2089-2105.

Apéndice

A

Parámetros de configuración de la experimentación

En este apéndice se recogen todos los aspectos relativos a la configuración de la experimentación y validación de los resultados, que bien por su volumen o relevancia no se han incorporado en las secciones correspondientes del capítulo 5, pero que se incorporan en el documento de tesis para aportar claridad al lector y permitir replicar los resultados obtenidos.

A.1 Parámetros de configuración de las heurísticas de construcción

A continuación se incorporan los parámetros de configuración de cada una de las heurísticas de construcción utilizadas en la experimentación: I1 (Solomon 1987), IMPACT (Ioannou, Kritikos y Prastacos 2001) e IRCI (Figliozzi 2010).

Criterio de elección del cliente inicial	μ	λ	α_1	α_2
CLIENTE CON FIN MÁS TEMPRANO	1,00	1,00	0,00	1,00
CLIENTE CON FIN MÁS TEMPRANO	1,00	1,00	0,60	0,40
CLIENTE CON FIN MÁS TEMPRANO	1,00	1,25	0,00	1,00
CLIENTE CON FIN MÁS TEMPRANO	1,00	1,25	0,90	0,10
CLIENTE CON FIN MÁS TEMPRANO	1,00	1,50	0,20	0,80
CLIENTE CON FIN MÁS TEMPRANO	1,00	1,50	0,30	0,70
CLIENTE CON FIN MÁS TEMPRANO	1,00	1,50	1,00	0,00
CLIENTE CON FIN MÁS TEMPRANO	1,00	2,00	1,00	0,00
CLIENTE MÁS ALEJADO	1,00	1,00	0,00	1,00
CLIENTE MÁS ALEJADO	1,00	1,00	0,40	0,60
CLIENTE MÁS ALEJADO	1,00	2,00	0,70	0,30
CLIENTE MÁS ALEJADO	1,00	2,00	1,00	0,00

Tabla A-1: Parámetros de la heurística de construcción *li*

Criterio de elección del cliente inicial	b_s	b_e	b_r	b_1	b_2	b_3
CLIENTE MÁS ALEJADO	0,00	0,00	1,00	0,30	0,70	0,00
CLIENTE MÁS ALEJADO	0,00	0,00	1,00	0,50	0,50	0,00
CLIENTE MÁS ALEJADO	0,00	0,00	1,00	1,00	0,00	0,00
CLIENTE MÁS ALEJADO	0,00	0,10	0,90	0,00	0,90	0,10
CLIENTE MÁS ALEJADO	0,00	0,10	0,90	0,10	0,90	0,00
CLIENTE MÁS ALEJADO	0,00	0,10	0,90	1,00	0,00	0,00
CLIENTE MÁS ALEJADO	0,10	0,00	0,90	0,30	0,70	0,00
CLIENTE MÁS ALEJADO	0,10	0,00	0,90	1,00	0,00	0,00
CLIENTE MÁS ALEJADO	0,10	0,10	0,80	0,60	0,40	0,00
CLIENTE MÁS ALEJADO	0,10	0,50	0,40	0,10	0,70	0,20
CLIENTE MÁS ALEJADO	0,10	0,50	0,40	0,30	0,10	0,60
CLIENTE MÁS ALEJADO	0,10	0,50	0,40	0,90	0,00	0,10
CLIENTE MÁS ALEJADO	0,00	0,00	1,00	0,30	0,70	0,00

Tabla A-2: Parámetros de la heurística de construcción *IMPACT*

δ_0	δ_1	δ_2	δ_3	δ_4
100,00	0,00	0,90	0,10	0,00
100,00	0,00	1,00	0,00	0,00
100,00	0,30	0,50	0,20	0,00
100,00	0,70	0,20	0,10	0,00
100,00	0,00	0,90	0,10	0,00
100,00	0,10	0,70	0,20	0,00
200,00	0,80	0,10	0,10	0,00
200,00	0,20	0,70	0,10	0,00
200,00	0,80	0,10	0,10	0,00
100,00	0,50	0,30	0,20	0,00
200,00	0,10	0,70	0,20	0,00
100,00	0,50	0,50	0,00	0,00
100,00	0,00	0,90	0,10	0,00

Tabla A-3: Parámetros de la heurística de construcción IRCI

B

Resultados ampliados de la experimentación

Los resultados mostrados en la sección 5.3 recogían únicamente los valores acumulados para cada una de las clases de problemas del juego de ensayo de Solomon (SINTEF 2015) puesto que esa es la forma habitual en la que se presentan los resultados en la literatura de referencia, como puede analizarse en (Bräysy y Gendreau 2005a), (Bräysy y Gendreau 2005b), o más recientemente en (Vidal, y otros 2013). Esta manera de presentar los resultados es adecuada siempre que se comparen técnicas de características similares, pero al agrupar los resultados por clase de problema, se pierde la información detallada de los resultados para cada una de las instancias del problema.

En esta sección, se incorporan los resultados obtenidos para cada una de las 56 instancias de juego de ensayo de Solomon para añadir el máximo detalle de cara a la reproducción de la experimentación realizada, siempre con el objetivo de facilitar la labor de retomar el trabajo aquí propuesto con el fin de utilizarlo o mejorarlo. En concreto se muestran dos bloques de resultados: por un lado, los resultados obtenidos por la nueva heurística de construcción, que se ha denominado $I1^{RC}$; y por otro lado, los mejores resultados obtenidos por la búsqueda paralela de vecindario variable utilizada en la experimentación asociada a los nuevos operadores de mejora basados en la reducción del número de rutas.

B.1 Resultados ampliados de la heurística de construcción para el VRPTW

A continuación, se incluyen seis tablas, cada una de ellas asociada a una de las clases de problemas del juego de ensayo de Solomon. En cada tabla se muestran los resultados obtenidos por las versiones de las tres heurísticas de construcción analizadas en la experimentación que utilizan el proceso de mejora propuesto como aporte de la presente tesis doctoral ($I1_{I1^{RC}}$, $IMPACT_{I1^{RC}}$, e $IRCI_{I1^{RC}}$).

Para cada instancia y heurística, los datos incluidos se refieren a número de vehículos y distancia total recorrida; resaltando en fondo verde los mejores resultados.

Instancia	$I1_{I1^{RC}}$		$IMPACT_{I1^{RC}}$		$IRCI_{I1^{RC}}$	
	Vehículos	Distancia	Vehículos	Distancia	Vehículos	Distancia
C101	10	828,937	10	852,645	10	828,937
C102	10	962,164	10	1081,787	10	1034,022
C103	10	1047,876	10	1087,476	10	1081,578
C104	10	1253,169	10	1153,193	10	1214,532
C105	10	828,937	10	871,801	10	828,937
C106	10	836,757	10	1132,049	10	841,443
C107	10	828,937	10	920,507	10	828,937
C108	10	828,937	11	994,348	10	858,136
C109	10	957,680	10	1086,877	10	960,81
C1	10,000	930,377	10,111	1020,076	10,000	941,926

Tabla B-1: Resultados obtenidos por las heurísticas de construcción para los problemas de la clase C1.

Instancia	$I1_{I1^{RC}}$		$IMPACT_{I1^{RC}}$		$IRCI_{I1^{RC}}$	
	Vehículos	Distancia	Vehículos	Distancia	Vehículos	Distancia
C201	3	591,557	3	591,557	4	633,224
C202	3	738,439	3	849,598	4	863,984
C203	3	759,491	3	978,560	4	913,301
C204	4	961,487	4	1124,182	4	934,188
C205	3	604,112	3	634,608	4	658,673
C206	3	617,430	3	867,237	4	820,165
C207	3	663,929	4	720,311	4	668,471
C208	3	649,006	3	727,936	3	763,58
C2	3,125	698,181	3,250	811,749	3,875	781,948

Tabla B-2: Resultados obtenidos por las heurísticas de construcción para los problemas de la clase C2.

Instancia	$I1_{I1^{RC}}$		$IMPACT_{I1^{RC}}$		$IRCI_{I1^{RC}}$	
	Vehículos	Distancia	Vehículos	Distancia	Vehículos	Distancia
R101	20	1859,562	19	1916,139	19	1939,893
R102	19	1748,402	18	1826,814	19	1693,355
R103	15	1538,029	13	1900,034	14	1473,817
R104	11	1375,705	11	1299,743	11	1216,108
R105	16	1623,479	15	1778,372	15	1657,464
R106	14	1639,262	13	1779,564	14	1565,716
R107	13	1454,275	12	1579,739	12	1357,743
R108	11	1252,072	11	1213,696	11	1205,344
R109	13	1479,264	14	1537,634	13	1447,212
R110	12	1390,886	13	1453,988	12	1380,753
R111	13	1429,639	12	1423,059	13	1389,964
R112	11	1238,800	11	1237,345	11	1173,13
R1	14	1502,448	13,500	1578,844	13,667	1458,375

Tabla B-3: Resultados obtenidos por las heurísticas de construcción para los problemas de la clase R1.

Instancia	$I1_{I1^{RC}}$		$IMPACT_{I1^{RC}}$		$IRCI_{I1^{RC}}$	
	Vehículos	Distancia	Vehículos	Distancia	Vehículos	Distancia
R201	4	1636,192	4	1804,018	4	1620,346
R202	4	1620,712	4	1682,112	4	1407,287
R203	4	1404,522	3	1710,549	3	1381,135
R204	3	1154,680	3	1339,582	3	966,891
R205	3	1491,353	3	1697,262	3	1372,403
R206	3	1422,244	3	1394,563	3	1299,391
R207	3	1232,984	3	1413,447	3	1243,575
R208	3	1038,078	3	1138,536	3	971,934
R209	3	1436,878	3	1601,365	3	1424,999
R210	3	1500,280	3	1546,131	3	1467,817
R211	3	1151,929	3	1335,743	3	1039,818
R2	3,273	1371,805	3,182	1514,846	3,182	1290,509

Tabla B-4: Resultados obtenidos por las heurísticas de construcción para los problemas de la clase R2.

Instancia	$I1_{I1^{RC}}$		$IMPACT_{I1^{RC}}$		$IRCI_{I1^{RC}}$	
	Vehículos	Distancia	Vehículos	Distancia	Vehículos	Distancia
RC101	15	1951,47	15	1876,999	15	1828,838
RC102	16	1853,991	14	2042,081	14	1747,025
RC103	13	1747,999	13	1713,515	13	1532,807
RC104	11	1321,91	12	1478,526	11	1300,664
RC105	18	1952,106	15	2165,752	17	2035,675
RC106	14	1675,419	14	1717,72	13	1687,526
RC107	13	1632,287	13	1731,475	12	1521,564
RC108	12	1488,609	12	1495,928	12	1346,195
RC1	14	1702,974	13,500	1777,75	13,375	1625,037

Tabla B-5: Resultados obtenidos por las heurísticas de construcción para los problemas de la clase RC1.

Instancia	$I1_{I1^{RC}}$		$IMPACT_{I1^{RC}}$		$IRCI_{I1^{RC}}$	
	Vehículos	Distancia	Vehículos	Distancia	Vehículos	Distancia
RC201	4	1966,693	4	2241,795	4	1977,898
RC202	4	1875,251	4	1975,706	4	1574,723
RC203	4	1479,646	4	1729,979	3	1526,945
RC204	3	1207,714	3	1233,726	3	1125,445
RC205	4	1971,317	4	1967,898	4	1835,097
RC206	4	1610,845	4	1708,174	4	1494,669
RC207	3	1610,812	4	1793,257	4	1523,518
RC208	3	1226,772	3	1473,434	3	1062,906
RC2	3,625	1618,631	3,75	1765,496	3,625	1515,15

Tabla B-6: Resultados obtenidos por las heurísticas de construcción para los problemas de la clase RC2.

C

Formato de entrada del entorno de visualización y simulación para problemas de tipo VRP

A modo de ejemplo, en este apéndice se incluye el extracto de un documento XML que sirve como entrada al entorno de visualización y simulación de problemas de asignación de rutas a vehículos. En concreto, el fragmento se corresponde con una solución a la instancia C101 del juego de ensayo de Solomon (SINTEF 2015).

Como se puede apreciar, la etiqueta raíz representa al problema (`<problem>`) en su conjunto, a continuación se incluye una etiqueta que con la información del almacén central (`<depot>`) y seguidamente las rutas (`<routes>`), cada una de ellas con la información de ubicación y planificación de cada cliente (`<customer>`).

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<problem class="C1" customers="100" id="C101" maxDistance="1236" maxTime="1236" maxVehicles="25" routes="10">
  <depot close="1236" maxCapacity="200" maxDistance="1236" maxTime="1236" maxVehicles="25" open="0" x="40" y="50"/>
</routes>
  <route begin="0" customers="13" demand="160" distance="64.8" duration="19" end="1144.8" service="1170" travel="64.84" waiting="0">
    <vehicle capacity="200" id="0" speed="60"/>
  </customers>
    <customer arrival="16.5" demand="10" earliest="16" early="16.5" id="43" late="16.7" latest="80" service="90" waiting="0" x="33" y="35"/>
    <customer arrival="109.5" demand="20" earliest="68" early="109.5" id="42" late="109.7" latest="149" service="90" waiting="0" x="33" y="32"/>
    <customer arrival="201.5" demand="10" earliest="166" early="201.5" id="41" late="201.7" latest="235" service="90" waiting="0" x="35" y="32"/>
    <customer arrival="293.5" demand="10" earliest="264" early="293.5" id="40" late="293.7" latest="321" service="90" waiting="0" x="35" y="30"/>
    <customer arrival="386.5" demand="10" earliest="359" early="386.5" id="44" late="386.7" latest="412" service="90" waiting="0" x="32" y="30"/>
    <customer arrival="479.3" demand="30" earliest="448" early="479.3" id="46" late="479.6" latest="509" service="90" waiting="0" x="30" y="32"/>
    <customer arrival="571.3" demand="10" earliest="541" early="571.3" id="45" late="571.6" latest="600" service="90" waiting="0" x="30" y="30"/>
    <customer arrival="663.3" demand="10" earliest="632" early="663.3" id="48" late="663.6" latest="693" service="90" waiting="0" x="28" y="30"/>
    <customer arrival="756.3" demand="10" earliest="725" early="756.3" id="51" late="756.6" latest="786" service="90" waiting="0" x="25" y="30"/>
    <customer arrival="848.6" demand="10" earliest="815" early="848.6" id="50" late="848.8" latest="880" service="90" waiting="0" x="26" y="32"/>
    <customer arrival="941.7" demand="10" earliest="912" early="941.7" id="52" late="942.0" latest="969" service="90" waiting="0" x="25" y="35"/>
    <customer arrival="1034.7" demand="10" earliest="1001" early="1034.7" id="49" late="1035" latest="1066" service="90" waiting="0" x="28" y="35"/>
    <customer arrival="1126.7" demand="10" earliest="1054" early="1126.7" id="47" late="1127" latest="1127" service="90" waiting="0" x="30" y="35"/>
  </customers>
</route>
</routes>
</problem>
```

Figura B-1: Fragmento de un documento XML de inicialización del entorno de visualización y simulación de problemas de VRP.