Universidad
Rey Juan Carlos

TESIS DOCTORAL

El problema multiobjetivo del comerciante reparador con beneficios

Autor

Rubén Morante González

Directores

Ana Dolores López Sánchez

Jesús Sánchez-Oro Calvo

Programa de doctorado en Tecnologías de la Información y las Comunicaciones

Escuela Internacional de Doctorado

2025

©2025 Rubén Morante González
Algunos derechos reservados
Este documento se distribuye bajo la licencia
“Atribución-CompartirIgual 4.0 Internacional” de Creative Commons,
disponible en https://creativecommons.org/licenses/bysa/4.0/deed.es

Esta Tesis Doctoral ha sido desarrollada bajo la financiación de los fondos asociados al proyecto TEC-2024/COM-404 de la Comunidad Autónoma de Madrid, el proyecto PID2021-125709OA-C22 del Ministerio de Economía y Competitividad, el proyecto TSI-100930-2023-3 del Ministerio para la Transformación Digital y de la Función Pública, y el proyecto PID2022-139543OB-C41 del Ministerio de Ciencia e Innovación de España.

Agradecimientos

Esta tesis doctoral ha sido la culminación de muchos años dedicados al estudio de las nuevas tecnologías. Durante ese tiempo, he tenido no sólo la suerte, sino el privilegio de compartir el tiempo con magnificas personas que me han enriquecido mucho, enseñado y apoyado. Y quiero hacer un recorrido por ellos para dedicarles algunas palabras.

Como toda buena ruta, tiene un principio: un depósito desde el que partir y al que volver. La Universidad Rey Juan Carlos ha sido ese punto de partida en esta aventura, estando presente desde los comienzos de mi carrera, pasando por el nivel de maestría, hasta llegar donde me encuentro ahora.

Pasando por nexos enriquecedores en que he podido intercambiar conocimiento y experiencias tanto con compañeros de clase como de trabajo, así como distintos profesores y académicos. En especial agradecer al grupo de investigación GRAFO su paciencia y apoyo. Jesús Sánchez-Oro ha puesto en varias ocasiones solución a problemas en mi vida personal, comenzando desde que terminé el máster, como tutor de mi Proyecto de Fin de Máster, e impulsando el desarrollo de mi carrera profesional ayudándome a encontrar un nuevo trabajo hasta día de hoy en el que ha sido uno de los mayores apoyos en el desarrollo del doctorado. Gracias a Jesús conocí a Alfredo, mi primer contacto con OGA.

El siguiente nodo de este viaje comenzó con una llamada a Alfredo, miembro de qosIT Consulting, y luego OGA. Con ellos empecé a trabajar en problemas de rutas, poner en práctica conocimientos de metaheurística y zambullirme en el mundo de la inteligencia artificial fuera de los libros. Muchos recuerdos y mucha gente de Sevilla permanecerán por siempre en mi corazón. Cuando digo “un orgullo”, se queda corto, así que probaré a ampliarlo con auténtico-absoluto-privilegioso y rotundo orgullo de haber formado parte de la familia OGA.

Y entre Jesús y Alfredo, conocí a Ana. No hay suficientes palabras en el diccionario que me permitieran describir a Ana. Sólo los alumnos que hayan tenido la suerte de experimentar esta vivencia con ella podrían comprenderme. A parte de ser mi guía en esta andadura, es una amiga, y en otros sentidos, una segunda madre. Ese es el enorme nivel de cariño y de consideración que tengo con ella.

Decir que ha sido un reparto duro, difícil y con muchas grandes piedras en el camino, sería quedarse corto; pero contaba con Jesús y Ana, que han ayudado mucho poniendo parches a las ruedas y trayendo bidones de gasolina cuando hacía falta.

Y tras el reparto, de vuelta al depósito. Profesores de la ETSII, miembros de GRAFO, alumnos de mis clases. ¡Cómo no seguir en la universidad con gente que te aporta tanto cada día!.

Pero para que un vehículo realice el reparto, para que pueda seguir su ruta, hay muchas personas detrás en las sombras. Mecánicos, personal de gasolinera, ITV... Mis amigos y mi familia. Os quiero muchísimo.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

ACO	Ant Colony Optimization
ACS	Ant Colony System
AMP	Adaptive Memory Programming
AKRed	Alternate K-exchange Reduction
AS	Ant System
aTSP	Asymmetric Traveling Salesman Problem
AtTSP	Attractive Traveling Salesman Problem
AVNS	Adaptive Variable Neighborhood Search
BACs	Bacterial Artificial Chromosomes
BOMTSPP	Bi-Objective Multiple Traveling Salesman Problem with Profits
BTSP	Bottleneck Traveling Salesman Problem
BVNS	Basic Variable Neighborhood Search
CLK	Chained Lin-Kernighan
COP	Combinatorial Optimization Problem
CPP	Chinese Postman Problem
CSP	Covering Salesman Problem
CTSP	Capacitated Traveling Salesman Problem
CVaR	Conditional Value at Risk
DDNN	Double Dynamic Nearest Neighbor
DHCP	Directed Hamiltonian Circuit Problem
DNN	Dynamic Nearest Neighbor
DP	Dynamic Programming
DRTSP	Discounted-Reward Traveling Salesman Problem
EC	Ejection Chain
EiLS	Evolutionary Algorithm with Intelligent Local Search

EVNS	Extended Variable Neighborhood Search
FO	Función Objetivo
FPTAS	Full Polynomial Time Approximation Scheme
GA	Genetic Algorithm
GTSP	Generalized Traveling Salesman Problem
GRA	Greedy Randomized Adaptive
GRASP	Greedy Randomized Adaptive Search Procedure
GVNS	Greedy Variable Neighborhood Search
HESA	Hybrid Evolutionary Search Algoritm
HGA	Hybrid Genetic Algoritm
IDLS	Intensification-Driven Local Search
ILP	Integer Linear Programming
ILS	Iterative Local Search
k-dTSP	k-Delivery Traveling Salesman Problem
KP	Knapsack Problem
KTSP	Kinetic Traveling Salesman Problem
MA	Memetic Algorithm
MAX TSP	Maximum Traveling Salesman Problem
MCP	Maximum Collection Problem
MDTSP	Multiple Depot Traveling Salesman Problem
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
MLPP	Minimum Latency Problem with Profits
Mo-TSPP	Multi-objective Traveling Salesman Problem with Profits
Mo-TSRP	Multi-objective Traveling Salesman Repairman Problem
Mo-TSRPP	Multi-objective Traveling Salesman Repairman Problem with Profits
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
MOGA	Multi-Objective Genetic Algorithm
MS-ACS	Multi-Strategic Ant Colony System
MST	Minimum Spanning Tree
MT-TSP	Moving-Target Traveling Salesman Problem
mTSP	Multiple Traveling Salesman Problem

MVP	Multi-objective Vending Problem
NN	Nearest NeighBour
NSGA	Non-dominated Sorting Genetic Algorithm
OP	Orienteering Problem
OPTW	Orienteering Problem with Time Windows
PCT	Prize-Collecting Tours
PCTSP	Prize-Collecting Traveling Salesman Problem
PC-TSP	Precedence-Constrained Traveling Salesman Problem
PGCH	Profit Guided Coordination Heuristic
PNN	Prioritized Nearest Neighbor
PNL	Promising Node List
PR	Path Relinking
PTP	Profitable Tour Problem
PTSP	Probabilistic Traveling Salesman Problem
QTSP	Quota Traveling Salesman Problem
QTSP-PIC	Quota Traveling Salesman Problem with Passengers, Incomplete Ride and Collection Time
RCL	Restricted Candidates List
RDI	Relative Deviation Index
RIP	Route Inspection Problem
RVNS	Reduced Variable Neighborhood Search
SA	Simulated Annealing
SMSA	Strategic Memetic Search Algorithm
SOE	Orienteering Score Event
STSP	Selective Traveling Salesman Problem
sTSP	Symmetric Traveling Salesman Problem
SOE	Orienteering Score Event
SPEA	Strength Pareto Evolutionary Algorithm
Speed TSP	Speed Traveling Salesman Problem
SS	Scatter Search
Steiner TSP	Steiner Traveling Salesman Problem
SVNS	Skewed Variable Neighborhood Search

TDTSP	Time Dependant Traveling Salesman Problem
TPP	Traveling Purchaser Problem
TRP	Traveling Repairman Problem
TRPP	Traveling Repairman Problem with Profits
TS	Tabu Search
TSP	Traveling Salesman Problem
TSPP	Traveling Salesman Problem with Profits
TSPPD	Traveling Salesman Problem with Pickup and Delivery
TSPPSC	Traveling Salesman Problem with Profits and Stochastic Customers
TSPTW	Traveling Salesman Problem with Time Windows
TSPWR	Traveling Salesman Problem with Refueling
TSRP	Traveling Salesman Repairman Problem
TTP	Traveling Thief Problem
UAV	Unmanned Aerial Vehicle
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
WSN	Wireless Sensor Networks
WT-TSP	Waiting Time Traveling Salesman Problem

Resumen

En esta Tesis Doctoral se presenta un problema de optimización combinatoria que puede considerarse una fusión de dos problemas clásicos de la literatura: el problema del vendedor ambulante con beneficios (TSPP) y el problema del reparador ambulante con beneficios (TRPP). La combinación de estos problemas da lugar a una nueva variante denominada problema multiobjetivo del comerciante reparador con beneficios (Mo-TSRPP). El Mo-TSRPP aparece para modelar una situación real en la que un técnico autónomo, especializado en la reparación de electrodomésticos, debe planificar su ruta diaria con el fin de maximizar su rentabilidad y eficiencia operativa. Esta resolución encaja en muchas otras situaciones reales.

El Mo-TSRPP tiene como objetivo optimizar simultáneamente tres criterios fundamentales: el coste total de desplazamiento, la latencia total de servicio y el beneficio total obtenido. La latencia hace referencia al tiempo transcurrido desde el inicio de la ruta hasta la prestación del servicio en cada cliente, lo que tiene un impacto directo en la satisfacción del usuario. Indirectamente, el número de clientes atendidos también se considera en la optimización, aunque no como un objetivo explícito, sino como un efecto emergente del proceso de selección de rutas en el Frente de Pareto. Este frente representa el conjunto de soluciones no dominadas en las que no es posible mejorar un criterio sin empeorar otro.

El problema surge en un contexto donde muchos profesionales independientes deben gestionar de manera eficiente su tiempo y recursos, eligiendo estratégicamente qué clientes visitar y en qué orden. A diferencia de otros problemas de ruteo, en el Mo-TSRPP no es obligatorio atender a todos los clientes disponibles, ya que cada uno ofrece un beneficio frente al coste de desplazamiento y la latencia acumulada. Esto introduce una dimensión de toma de decisiones clave en la que se deben evaluar múltiples alternativas con diferentes compromisos entre los objetivos.

Para abordar la resolución del Mo-TSRPP, se propone un enfoque basado principalmente en la metaheurística Variable Neighborhood Descent (VND), la cual se encarga de explorar sistemáticamente distintas estructuras de vecindad con el objetivo de mejorar iterativamente las soluciones. Como paso inicial, VND se apoya en un procedimiento constructivo basado en Greedy Randomized Adaptive Search Procedure (GRASP), utilizado exclusivamente para generar soluciones iniciales de alta calidad mediante una selección aleatoria guiada. Esta integración permite al algoritmo VND partir de buenos puntos de inicio y potenciar su capacidad de intensificación, logrando así una aproximación eficiente y diversa al Frente de Pareto.

Para evaluar el rendimiento del método propuesto, se realiza una comparación

con tres algoritmos evolutivos ampliamente utilizados en la optimización multiobjetivo: NSGA-II, SPEA-2 y MOEA/D. Se analizan diversos conjuntos de instancias experimentales, considerando variaciones en la cantidad de clientes, la distribución geográfica y los parámetros de beneficio y coste. Los resultados muestran que la estrategia híbrida GRASP-VND es capaz de generar soluciones competitivas, superando en varios casos a los algoritmos evolutivos en términos de calidad del Frente de Pareto y eficiencia computacional.

Finalmente, para ilustrar la aplicabilidad del enfoque desarrollado, se presenta un caso de estudio realista basado en datos obtenidos de un servicio técnico de reparaciones. Se analiza cómo la metodología propuesta permite mejorar la planificación de rutas en escenarios reales, proporcionando herramientas útiles para la toma de decisiones en entornos dinámicos y altamente competitivos. Esta validación práctica destaca la relevancia del Mo-TSRPP como un problema de optimización aplicable a múltiples contextos empresariales donde la gestión eficiente de recursos móviles es un factor clave para la rentabilidad y sostenibilidad del negocio.

Abstract

In this PhD Thesis we present a combinatorial optimisation problem that can be considered a fusion of two classical problems in the literature: the Traveling Salesman Problem with Profits (TSPP) and the Traveling Repairman Problem with Profits (TRPP). The combination of these problems gives rise to a new variant called the multi-objective traveling salesman for profit problem (Mo-TSRPP). The Mo-TSRPP appears to model a real situation in which a self-employed technician, specialised in the repair of household appliances, must plan his daily route in order to maximise his profitability and operational efficiency. This resolution fits in many other real situations.

The Mo-TSRPP aims to simultaneously optimise three fundamental criteria: the total cost of travel, the total service latency and the total profit obtained. Latency refers to the time elapsed from the start of the route to the delivery of the service to each customer, which has a direct impact on customer satisfaction. Indirectly, the number of customers served is also considered in the optimisation, although not as an explicit objective, but as an emergent effect of the route selection process in the Pareto Front. This front represents the set of non-dominated solutions in which it is not possible to improve one criterion without worsening another.

The problem arises in a context where many freelancers must efficiently manage their time and resources, strategically choosing which clients to visit and in what order. Unlike other routing problems, in Mo-TSRPP it is not mandatory to serve all available clients, as each offers a benefit in terms of travel cost and accumulated latency. This introduces a key decision-making dimension where multiple alternatives with different trade-offs between objectives must be evaluated.

To address the resolution of Mo-TSRPP, we propose an approach based mainly on the Variable Neighbourhood Descent (VND) metaheuristic, which systematically explores different neighbourhood structures with the aim of iteratively improving the solutions. As an initial step, VND relies on a constructive procedure based on the Greedy Randomized Adaptive Search Procedure (GRASP), which is used exclusively to generate high-quality initial solutions by means of guided random selection. This integration allows the VND algorithm to start from good starting points and to enhance its intensification capability, thus achieving an efficient and diverse approach to the Pareto Front.

To evaluate the performance of the proposed method, a comparison is made with three evolutionary algorithms widely used in multi-objective optimisation: NSGA-II, SPEA-2 and MOEA/D is performed. Different sets of experimental instances are

analysed, considering variations in the number of customers, geographical distribution, and profit and cost parameters. The results show that the hybrid GRASP-VND strategy is able to generate competitive solutions, outperforming in several cases the evolutionary algorithms in terms of Pareto Front quality and computational efficiency.

Finally, to illustrate the applicability of the developed approach, a realistic case study based on data obtained from a technical repair service is presented. It is analysed how the proposed methodology allows to improve route planning in real scenarios, providing useful tools for decision making in dynamic and highly competitive environments. This practical validation highlights the relevance of Mo-TSRPP as an optimisation problem applicable to multiple business contexts where the efficient management of mobile resources is a key factor for the profitability and sustainability of the business.

Capítulo 1

Introducción

El primer capítulo de este documento establece el contexto en el que se desarrolla esta Tesis Doctoral. Se detalla el problema que se pretende resolver, conocido como "el problema multiobjetivo del comerciante reparador con beneficios", en adelante Mo-TSRPP por sus siglas en inglés "Multi-objective Traveling Salesman Repairman Problem with Profits". Además, en este capítulo se incluyen la motivación, la hipótesis de trabajo y los objetivos. También se brinda un resumen de la propuesta algorítmica para abordar el problema, que será explicada con más detalle en los siguientes capítulos, así como el método de investigación utilizado.

1.1. Problemas de optimización

La optimización se conceptualiza como una disciplina fundamental que subyace en diversas áreas, incluyendo la Inteligencia Artificial, la Ciencia de la Computación y la Investigación Operativa [2]. Su objetivo principal radica en encontrar soluciones viables para problemas con aplicaciones en ingeniería, medicina, economía y numerosos campos científicos [2, 3].

El campo de la investigación operativa ha experimentado un crecimiento notable en los últimos años, en gran medida impulsado por el rápido avance de la tecnología informática. Esta investigación operativa es esencial a la hora de fundamentar la toma de decisiones, siendo éste el proceso que deriva en el planteamiento del problema y el análisis de las diferentes líneas de resultados tras la aplicación de métodos de resolución, buscando la mejora de la eficiencia y la gestión de los recursos.

La resolución de los problemas de optimización consiste en el hallazgo de las soluciones candidatas como resultado de la evaluación de uno o varios objetivos, sujeta a una serie de restricciones que surgen de la aplicación del problema en casos reales. En algunos de estos problemas existen un gran número de variables de decisión, y una gran cantidad de restricciones que deben satisfacerse, y es ahí donde se abre un amplio abanico de algoritmos que nos ayudan a encontrar una solución de muy buena calidad. Usualmente estos algoritmos requieren números

iteraciones para hayar la solución al problema. Debido a este número de iteraciones, adquiere un papel fundamental encontrar una forma óptima de hallar dicha solución. Esto se refleja en la optimización de los tiempos de ejecución y de los recursos computacionales, buscando minimizar estos valores en los métodos de resolución propuestos.

Una amplia gama de problemas de interés en áreas científicas y tecnológicas puede ser formulada como problemas de optimización. Un problema de optimización se caracteriza por tener múltiples soluciones posibles, junto con un método definido para compararlas entre sí [4]. La calidad de cada solución se evalúa mediante una Función Objetivo (FO) designada, que refleja los criterios de optimización específicos del problema en cuestión.

En el contexto matemático, un problema de optimización se formula con el objetivo de minimizar o maximizar el valor de una FO, denotada como $f(x)$, siendo x el conjunto de variables de decisión, sujeta a un conjunto de restricciones establecidas [5]. De manera más precisa y sin pérdida de generalidad, un problema de minimización puede ser definido como [6]:

$$\text{Problema de optimización} = \begin{cases} \text{Minimizar} & f(x) \\ \text{sujeto a} & x \in \Omega \end{cases} \tag{1.1}$$

donde Ω representa el conjunto de soluciones que cumplen con todas las restricciones impuestas por el problema. Cada solución en este conjunto se denomina solución factible. Asimismo, tomando el problema de minimización formulado anteriormente, una solución factible se considera óptima si el valor de la FO asociada a dicha solución es el más bajo en comparación con todas las demás soluciones factibles.

Atendiendo a diferentes criterios, los problemas de optimización se pueden clasificar de diferentes maneras. Estás se detallan en las siguientes secciones.

1.1.1. Problemas de optimización según el número de funciones objetivo

Uno de los enfoques fundamentales de los problemas de optimización es la definición de la cantidad y naturaleza de las funciones a optimizar. Centrando la atención en esta clasificación, se pueden identificar dos tipos de modalidades:

- **Problemas de optimización mono-objetivo**: en los se trata de optimizar una sola FO o criterio.
- **Problemas de optimización multiobjetivo**: en los que se busca optimizar en varias funciones objetivo de manera simultánea. Cada FO se vincula con un criterio diferente. Es muy típico que en este tipo de problemas de optimización, surjan conflictos entre las funciones objetivo, de tal forma que una mejora de un objetivo puede empeorar otro.

Problemas de optimización mono-objetivo

Como se ha mencionado previamente, los problemas de optimización mono-objetivo son aquellos en los que solo existe una FO, es decir, que el objetivo del problema a optimizar se puede modelar en una única expresión matemática que reúna todas las variables de decisión y que atienda a un único criterio. Entre los casos de uso más comunes se encontrarían problemas de minimización de costes o maximización de ganancias.

$$\min_{S \in \Omega} f(S)$$

donde $f(S)$ representa el valor de la función objetivo que se pretende minimizar.

Problemas de optimización multiobjetivo

Los problemas de optimización multiobjetivo buscan el conjunto de soluciones que ofrecen un equilibrio entre varios objetivos o *trade-offs*.

Formalmente y sin pérdida de generalidad, un problema de minimización multiobjetivo puede definirse como sigue:

$$\min_{S \in \Omega} F(S) = \{f_1(S), f_2(S), \ldots, f_m(S)\}$$

donde la imagen del conjunto factible de soluciones, ω, es $F(S) = \{(f_1(S), f_2(S), \ldots, f_m(S) : S \in \Omega\}$ y se denomina el espacio objetivo de soluciones.

Por tanto, dadas dos soluciones factibles $S_1 \in \Omega$ y $S_2 \in \Omega$ se dice que:

- S_1 ***domina débilmente a*** S_2, denotado como $S_1 \preceq S_2$, si y sólo si $f_i(S_1) \leq f_i(S_2)$, $\forall i = 1, \ldots, m$.
- S_1 ***domina a*** S_2, denotado como $S_1 \prec S_2$, si y sólo si $f_i(S_1) \leq f_i(S_2), \forall i = 1, \ldots, m$, y $\exists i_0$ tal que $f_{i_0}(S_1) < f_{i_0}(S_2)$.
- S_1 ***domina estrictamente a*** S_2, denotado como $S_1 \preceq\preceq S_2$, si y sólo si $f_i(S_1) < f_i(S_2)$, $\forall i = 1, \ldots, m$.

Una solución es *eficiente*, *no dominada* u *Pareto óptima* si no hay otra solución que la *domine*. Por lo tanto, el frente de Pareto, también conocido como *la frontera eficiente*, es el conjunto de todas las soluciones eficientes.

1.1.2. Problemas de optimización según el número de soluciones

Los problemas de optimización pueden clasificarse según el número de soluciones en dos grandes categorías: aquellos con una única solución óptima y aquellos

con múltiples soluciones óptimas.

Problemas de optimización con solución única

Los problemas de optimización con solución única son aquellos en los que existe una sola solución que maximiza o minimiza una FO, por lo tanto, no existe ninguna otra combinación de variables para los que se obtenga un mismo valor óptimo de la FO.

Un ejemplo de ello serían aquellas funciones objetivo monótonas en las que la solución óptima se encontraría en su punto de corte con una o varias de las restricciones; o por ejemplo, para un problema de minimización, aquellas funciones objetivo convexas cuyas restricciones conforman una región factible igualmente convexa que contenga a la FO, puesto que el mínimo se encontrará en el vértice de la función.

Problemas de optimización con múltiples soluciones óptimas

Cabe notar que, al evaluar algunos problemas de optimización, puede ser que se obtenga como resultado un conjunto conformado por varias soluciones óptimas, todas ellas obtienen el mismo valor de la FO cumpliendo con las restricciones:

- **Función objetivo y restricciones degeneradas**: cuando teniendo la región factible acotada por las restricciones, la solución óptima se encuentra sobre la arista de una restricción que es paralela a la FO, lo que da lugar a varias soluciones en esa línea o arista.
- **Problemas con múltiples variables dependientes**: cuando hay una relación entre las variables que las hace interdependientes, resultando en que varias combinaciones entre ellas que proporcionan el mismo valor óptimo. Un caso muy típico, es un problema de asignación de flotas en las que varias combinaciones de asignaciones producen un mismo coste o gasto de combustible, ya que los recursos pueden ser intercambiables.
- **Frontera de soluciones no única**: surgen especialmente en algunos problemas de optimización multiobjetivo, donde puede haber varias combinaciones de variables de decisión que proporcionen el mismo valor de los diferentes criterios o donde se obtiene un conjunto de soluciones no dominadas en las que, al reajustar los valores de las variables, ya no se puede obtener la mejora en un objetivo sin empeorar la del otro (frente de Pareto).

1.1.3. Problemas de optimización según el tipo de variable de decisión

A la hora de plantear un problema de optimización, es significativamente importante conocer la naturaleza de las variables en cuanto al tipo de valores que

pueden tomar, ya que puede aumentar significativamente la potencia computacional requerida a la hora de evaluar las diferentes iteraciones. Así pues, atendiendo a este análisis, se pueden identificar las variantes más comunes como:

- **Problemas de optimización continua**: en los que las variables pueden tomar cualquier valor dentro de un intervalo de números reales. Estos problemas son comunes en casos de entrenamiento de redes neuronales o de optimización financiera.
- **Problemas de optimización discreta**: en los que las variables pueden tomar solo valores que pertenecen al espacio de los números enteros, esta restricción cobra gran valor porque conlleva que la solución es indivisible. Un caso particular de ellos son los problemas de optimización combinatoria (*Combinatorial Optimization Problem*, COP), caracterizados precisamente por tener soluciones expresadas como conjuntos de números enteros [5, 7].
- **Problemas de optimización binaria**: en los que las variables de decisión solo pueden tomar valores 0 o 1, concretizándose en decisiones como verdadero/-falso, sí/no, etc. Este tipo de variables son comunes en problemas como diseño de redes o de planificación.

No obstante, es importante tener en cuenta que en la mayoría de los casos reales en las que contribuyen diferentes variables, lo más común es que se traten de problemas mixtos en los que cada variable estará restringida a un conjunto de datos y rangos distintos.

1.1.4. Problemas de optimización según la complejidad computacional

Los problemas de optimización también pueden ser categorizados según su complejidad computacional, es decir, la "dificultad" que supone su resolución para una computadora [8]. Esta clasificación se basa en clases de complejidad, que agrupan conjuntos de problemas que comparten ciertas propiedades relacionadas con su complejidad y que son generalmente representados en un gráfico de dominio como el de la Figura 1.1.

Algunos problemas pueden resolverse en tiempo polinómico, lo que significa que existe una función polinómica con un exponente máximo p que relaciona el tamaño de la entrada del algoritmo n con el tiempo máximo necesario para resolver el problema (medido en iteraciones). La complejidad de esta función se denota como $O(n^P)$. Estos problemas se encuentran en la clase $\mathcal{P}$ de problemas computacionales. Sin embargo, hay una gran cantidad de problemas de interés práctico para los cuales no se conoce un algoritmo que pueda resolverlos en tiempo polinómico de manera exacta. Dentro de estos problemas, hay algunos para los cuales sí se puede determinar, en tiempo polinómico, siempre y cuando se haya proporcionado previamente un valor que forme parte de la solución del problema. Estos problemas se clasifican en la clase $\mathcal{NP}$. Los problemas en la clase $\mathcal{P}$ también están en $\mathcal{NP}$, ya que se puede verificar en tiempo polinómico si una solución dada es válida para el problema.

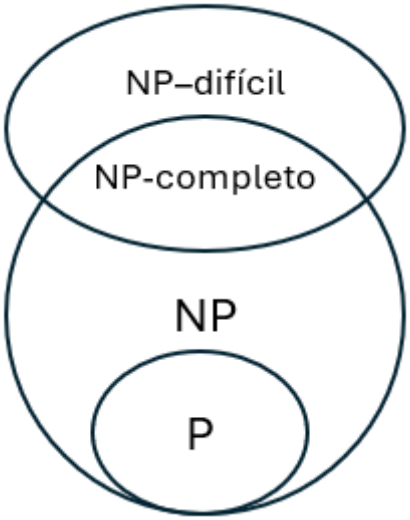


Figura 1.1: Dominio de los problemas $\mathcal{NP}$

Además, dentro de $\mathcal{NP}$, hay problemas para los cuales no se ha encontrado un algoritmo de tiempo polinómico, aunque no se ha podido demostrar que no exista tal algoritmo. Estos problemas se conocen como $\mathcal{NP}$-Completos. Los problemas $\mathcal{NP}$-Completos comparten la característica de ser tan difíciles como cualquier otro problema en $\mathcal{NP}$. Por lo tanto, si se demostrara que uno de ellos se puede resolver en tiempo polinómico, entonces todos los problemas en $\mathcal{NP}$ podrían resolverse en tiempo polinómico [9].

Por último, existe una clase de problemas llamada $\mathcal{NP}$-Difíciles, que no necesariamente pertenecen a $\mathcal{NP}$, pero son al menos tan difíciles como los problemas más difíciles en $\mathcal{NP}$ [8, 9]. Un problema se considera $\mathcal{NP}$-Difícil si se puede reducir polinómicamente a algún problema en $\mathcal{NP}$-Completo.

1.1.5. Técnicas de resolución de problemas de optimización

Conforme a las características de los problemas de optimización que se han mencionado previamente, como el número de funciones objetivo y la naturaleza de las variables de decisión, y al tipo de restricciones que acotan el problema, existen diversas metodologías de resolución que se adecúan en mayor o menor medida a la hora de la búsqueda de resultados.

Métodos exactos

Los métodos exactos son aquellos que se tratan de buscar una solución óptima global, cumpliendo con todas las restricciones y optimizando la FO, por lo tanto, los hace ideales para problemas pequeños, ya que hacen una evaluación exhaustiva de toda la región factible, asegurando que la solución obtenida es la mejor.

En contrapartida, la complejidad de este método de resolución aumenta exponencialmente a medida que aumenta el tamaño del problema debido, precisamente, a esta evaluación exhaustiva del espacio de soluciones factibles.

Dentro de los principales métodos de resolución exactos están el método simplex para los problemas de programación lineal; método de puntos interiores para problemas de programación lineal o no lineal convexa; método de multiplicadores de Lagrange que se emplean para combinar la FO con las restricciones para obtener un problema sin restricciones; o métodos de programación dinámica en la que el problema se divide en varios en las que las soluciones de los primeros se reutilizan para evaluar el resto.

En problemas de mayor escala, esta metodología de resolución se vuelve ineficiente y a menudo se recurre a técnicas heurísticas para su resolución.

Métodos heurísticos

Conforme al Diccionario de la Real Academia Española [10], la etimología de la palabra "heurística", de raíces griegas, denota "hallar" o "inventar". Entre sus definiciones se incluyen "Técnica de la indagación y del descubrimiento" y "En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.". La popularización del término "heurística" en el ámbito de la investigación se atribuye al matemático George Pólya, quien lo introdujo por primera vez en su obra "*How to solve it*" [11].

En el campo de la optimización, el concepto de heurística se refiere a métodos que pueden generar soluciones para problemas de optimización. Estos métodos pueden ser clasificados como genéricos o específicos. Los métodos genéricos son generalmente simples, adaptables y robustos, y no dependen del problema específico que se esté abordando. Por otro lado, los métodos específicos tienden a proporcionar soluciones de mayor calidad que los genéricos, pero requieren ser ajustados para el problema en particular que se esté tratando.

Una heurística puede iniciar su proceso desde una solución previamente obtenida o puede generar una solución desde cero. Aquellas heurísticas que crean soluciones se conocen como heurísticas constructivas, mientras que las que buscan mejorar una solución mediante movimientos a partir de una ya existente se denominan heurísticas de búsqueda o búsquedas locales. Para profundizar:

- Método Constructivo: Este tipo de procedimiento heurístico es capaz de generar una solución para un problema específico a partir de una solución inicial vacía. Dichas heurísticas van agregando elementos a la solución gradualmente hasta que esta se complete. En cada iteración, se selecciona un elemento utilizando un criterio específico. Algunos de los criterios de selección más comunes incluyen:
    - Estrategia voraz: también denominada miope o ávida, en cada etapa selecciona el elemento que parece ser más beneficioso para la solución parcial en ese momento, sin considerar las consecuencias futuras.

- Estrategia de descomposición: implica dividir el problema en subproblemas más pequeños de forma recursiva hasta que el tamaño del problema tratado sea manejable. Luego, se combinan las soluciones de los subproblemas para formar una solución final.
- Estrategia de reducción: implica restringir el espacio de búsqueda mediante la identificación de atributos comunes presentes en soluciones de alta calidad, y luego incorporar estos atributos en la solución en construcción, basándose en la premisa de que la solución óptima también exhibirá dichos atributos.
- Estrategias de manipulación del modelo: Se fundamentan en la simplificación del modelo del problema y la resolución de un modelo simplificado. Una vez obtenida esta solución, se aplica al problema original. Entre los métodos destacados de este tipo de estrategias se encuentran la linealización, la agregación de variables y la introducción de nuevas restricciones.

- Método de búsqueda Local: también conocida como método de mejora, es un procedimiento heurístico que parte de una solución previamente generada y busca una solución de mayor calidad. Se basa en uno de los métodos de optimización más antiguos: el ensayo y error [5]. Para ello, realiza modificaciones o movimientos dentro de la solución inicial y evalúa la calidad de la solución resultante. La heurística concluye cuando se han agotado todas las posibles modificaciones y ninguna de ellas ha mejorado la solución original. Los movimientos pueden seguir diversas estrategias:
  - Estrategia *Best Improvement*: implica evaluar todos los movimientos posibles desde una solución dada y seleccionar aquel que genere la mayor mejora en la FO.
  - Estrategia *First Improvement*: implica evaluar de manera secuencial los movimientos posibles desde una solución dada y seleccionar el primer movimiento que resulte en una mejora de la FO.

Las técnicas heurísticas suelen enfrentarse al desafío de quedar atrapadas en óptimos locales, lo que significa que pueden encontrar soluciones que son las mejores dentro de su vecindad, donde la vecindad de una solución se refiere al conjunto de soluciones que se pueden alcanzar mediante un solo movimiento. Sin embargo, estas soluciones locales no necesariamente son óptimas globalmente.

Métodos Metaheurísticos

Como se ha mencionado previamente, una limitación significativa de las heurísticas en el campo de la optimización es su incapacidad para escapar de óptimos locales. Como respuesta a esta limitación, se han desarrollado un nuevo tipo de técnicas conocidas como metaheurísticas. El prefijo “meta”, según el Diccionario de la Real Academia Española [10], denota la idea de “junto a”, “después de”, “entre”, “con” o “acerca de”, aunque comúnmente se interpreta como “más allá de”. En

inglés, el uso de “meta” también se acepta en compuestos sustantivos con el significado de “por encima de” [12]. El término metaheurística fue acuñado por Fred Glover en 1986 [13] con la intención de describir un procedimiento maestro de alto nivel que guiara y modificara otras heurísticas para explorar soluciones más allá de los óptimos locales.

Las metaheurísticas han evolucionado hasta convertirse en una clase amplia de algoritmos de optimización, caracterizados por su flexibilidad, generalidad y capacidad para abordar problemas complejos donde los métodos exactos son computacionalmente inviables. Estas técnicas combinan procesos de exploración (búsqueda global) y explotación (búsqueda local) del espacio de soluciones, lo que les permite equilibrar la calidad de las soluciones encontradas con la eficiencia computacional. A diferencia de las heurísticas clásicas, las metaheurísticas no están diseñadas específicamente para un solo problema, sino que proporcionan un marco adaptable a múltiples dominios, desde la ingeniería y la economía hasta la bioinformática y la inteligencia artificial.

Existen diversas clasificaciones de metaheurísticas, siendo una de las más comunes la distinción entre métodos basados en trayectorias (como el Recocido Simulado (*Simulated Annealing*) o la Búsqueda Tabú (*Tabu Search*)) y métodos poblacionales, que operan sobre un conjunto de soluciones simultáneamente, como los Algoritmos Genéticos, la Optimización por Colonia de Hormigas o los Algoritmos de Enjambre de Partículas. Además, algunos enfoques hibridan distintas estrategias o incorporan mecanismos de aprendizaje automático para mejorar su rendimiento adaptativo.

La creciente relevancia de las metaheurísticas en investigación y aplicaciones prácticas se debe a su capacidad para abordar problemas multiobjetivo, dinámicos, ruidosos o con restricciones complejas. Esta versatilidad ha motivado el desarrollo de variantes y extensiones cada vez más sofisticadas, consolidando a las metaheurísticas como una herramienta clave en el campo de la optimización computacional moderna.

Esta Tesis Doctoral propone el desarrollo de heurísticas, abordando tanto enfoques constructivos como de búsqueda local, con el objetivo de resolver un problema de optimización combinatoria que además es multiobjetivo y que se conoce como Multi-objective Traveling Salesman Repairman Problem with Profits (Mo-TSRPP).

Cuando se resuelven problemas de optimización de un solo objetivo, la forma de comparar dos soluciones es ver si el valor de la FO de la solución S_1 es mejor que el valor de la FO de S_2. Sin embargo, cuando se resuelven problemas de optimización multiobjetivo, al comparar dos soluciones, S_1 y S_2, determinar que solución es mejor no es fácil, sobretodo cuando hay objetivos enfrentados.

La forma más sencilla de considerar una mejora es mediante la agregación de todas las funciones objetivo para tratar el problema como un problema de optimización con un único objetivo, es decir, $\sum_{i=1}^{m} \beta_i \cdot f_i(S)$, donde $\sum_{i=1}^{m} \beta_i = 1$, y $\beta_i \in [0,1]$ $\forall i = 1, \ldots, m$ que representa el peso o la importancia de cada FO.

Otra posibilidad de definir una mejora fue propuesta por [14], que definió la mejora pura (aleatoria u ordenada) y la mejora combinada (secuencial o ponderada):

- La mejora pura se centra en considerar un único objetivo hasta que no se puedan conseguir más mejoras para dicho objetivo aunque se permita el deterioro de los demás objetivos. El término de aleatorio u ordenado está relacionado con la forma de seleccionar el objetivo a mejorar.
- La mejora combinada considera, en cada paso del procedimiento, una FO diferente, por lo que no permite el deterioro de los demás objetivos, ya que todos ellos pueden aplicarse. El término de secuencial indica que en cada paso se considera una FO diferente y el ponderado combina todos los objetivos en un único objetivo.

Otros autores utilizan la forma más generalizada aunque consume mucho tiempo, véase [15], [16] o [17] que consideran una mejora si el Frente de Pareto cambia, es decir, si se obtiene una nueva solución eficiente durante la búsqueda, aunque eso signifique que otras soluciones en el Frente de Pareto estén ahora dominadas y por tanto deban ser eliminadas del Frente de Pareto.

1.1.6. Clasificación del Mo-TSRPP

Atendiendo a todas las características de los problemas de optimización mencionadas anteriormente, en esta Tesis Doctoral se ha resuelto el problema Mo-TSRPP que tiene las siguientes características:

- Es un problema de optimización combinatoria.
- Es multiobjetivo debido a que se buscan mejorar los siguientes objetivos, minimizar coste de viaje y latencia, y maximizar beneficio e implícitamente el número de nodos que representan clientes visitados.
- El frente de Pareto está formado por múltiples soluciones Pareto óptimas.
- Tiene una complejidad computacional $\mathcal{NP}$-Difícil, puesto que no hay un algoritmo polinómico que lo resuelva.
- Se resuelve utilizando técnicas metaheurísticas, en particular, se utiliza VND cuyas soluciones iniciales se obtienen siguiendo una construcción GRASP.
- Tiene aplicación práctica en un caso real.

Como se viene introduciendo en esta tesis, se aborda un problema de optimización real, que busca maximizar el beneficio visitando el mayor número de clientes posibles, teniendo en cuenta que se debe recorrer la menor distancia posible, y que los clientes esperen el menor tiempo posible para ser visitados.

1.2. Planteamiento y justificación del trabajo

Una vez que se ha delimitado el ámbito de estudio que abarca esta Tesis Doctoral, se procede a ofrecer una descripción más detallada del problema que se busca

abordar: "*Mo-TSRPP*". Para ello, se realizará una introducción sucinta a los problemas clásicos *Traveling Salesman Problem* y *Traveling Repairman Problem*, seguido de la definición precisa del problema en cuestión.

1.2.1. Problema del TSP y del TSPP

El problema del viajante de comercio (TSP) es uno de los problemas de optimización combinatoria más ampliamente estudiados en la literatura. De hecho, se han publicado más de 12000 trabajos según ScienceDirect y hay más de 48000 entradas en google scholar, tanto desde el punto de vista teórico como práctico, estudiando su complejidad, sus propiedades, o resolviéndose mediante algoritmos exactos o aproximados.

El objetivo del TSP es encontrar un ciclo hamiltoniano, que es un camino cerrado que visita cada nodo del grafo exactamente una vez, excepto el primer y el último que coinciden, con un coste total mínimo. En otras palabras, se trata de determinar un orden en el que cada nodo debe ser visitado de forma que cada nodo sea visitado una sola vez, y el coste total incurrido (distancia recorrida o tiempo de viaje) sea mínimo. Para estudiar el TSP y sus variantes, véase [18] y, desde una perspectiva computacional, se remite al lector a [19].

En general, el TSP se puede formular como: Hay $|V| = n$ nodos y el coste de viajar por cada arista se conoce con precisión. El objetivo es encontrar el menor coste de viaje, que pase por todos los nodos exactamente una vez y que empiece y termine en el mismo nodo. Sin embargo, obtener una solución óptima al TSP resulta complicado debido a la gran cantidad de soluciones factibles posibles. El TSP puede formularse como

$$\text{mín} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

Sujeto a:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V$$

$$u_i - u_j + |V| \cdot x_{ij} \leq |V| - 1 \qquad \forall i, j \in V, i \neq j$$

$$u_i \in \mathbb{Z}, \qquad 2 \leq u_i \leq |V| \qquad \forall i > 1$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V$$

donde:

- V: es el conjunto de nodos.
- x_{ij}: es una variable binaria. 1 si el vendedor viaja del nodo i al nodo j. 0 en caso contrario.
- c_{ij}: es el coste de viajar entre el nodo i y el nodo j.
- u_i: es una variable de posición que toma valores entre 2 y V (para $i > 1$).

Una solución TSP, denotada por el conjunto K, forma un recorrido que representa la secuencia de como se visita cada nodo y puede representarse como un conjunto de n nodos ordenados:

$$K = \{i_1, i_2, i_3, i_4, \dots \dots, i_{n-1}, i_n, i_1)\}$$

El objetivo principal es encontrar la permutación de nodos que minimice el coste de viaje total. En la Figura 1.2 se muestra una solución factible para el TSP donde la distancia total recorrida es 10. El viajante comienza en el nodo inicial denotado como "Base", se desplaza hasta el nodo "e", después al "f", continua por el "d", "a", "b", "c" y vuelve al nodo inicial "Base". El recorrido entero $K = \{i_{Base}, i_e, i_f, i_d, i_a, i_b, i_c, i_{Base}\}$, volviendo al nodo inicial, la distancia recorrida por el viajante sería de 10 $(c_{Base,e} + c_{e,f} + c_{f,d} + c_{d,a} + c_{a,b} + c_{b,c} + c_{c,Base} = 2 + 2 + 1 + 1 + 2 + 1 + 1 = 10)$.

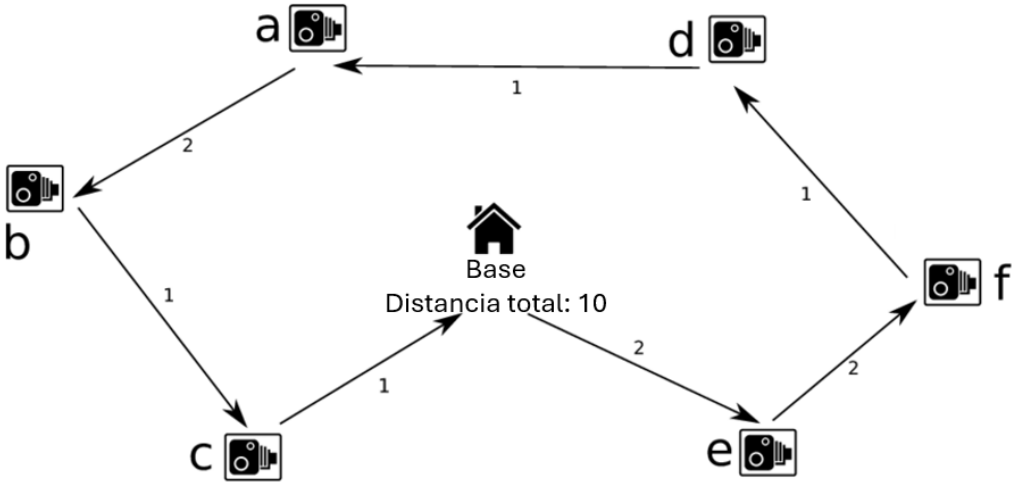


Figura 1.2: Una solución factible para el TSP

Un problema más general que el TSP es el TSP con beneficios (Traveling Salesman Problem with Profits (TSPP)), en el que cada nodo tiene asociado un beneficio y una solución factible no requiere visitar todos los nodos. Es decir, hay que encontrar un ciclo hamiltoniano sobre un subconjunto de nodos tal que el beneficio recogido se maximice mientras que el coste de viaje se minimiza. Como sostienen [20], el objetivo es encontrar un equilibrio adecuado entre el beneficio total recogido y el coste de viaje, ya que la maximización de la recogida de beneficios implica que el vendedor debe visitar un gran número de nodos, mientras que la minimización de costes implica que el vendedor debe visitar un número reducido de nodos. Sin embargo, la mayoría de los artículos resuelven el TSPP considerando sólo un objetivo

en lugar de los dos, que además, están en conflicto. Muchos autores, resuelven este problema biobjetivo combinando ambas funciones objetivo y surgiendo las variantes que se mencionan a continuación. El problema del recorrido rentable, también conocido como Problema del Ciclo Simple e introducido por [21], maximiza la diferencia entre los beneficios totales recogidos y el coste total de viaje del recorrido; el problema de orientación, también conocido como TSP selectivo o problema de recogida máxima, introducido por [22], busca maximizar los beneficios totales recogidos manteniendo un coste de viaje máximo (un valor fijo dado; véase [23]; en el TSP de recogida de premios (Prize-Collecting Traveling Salesman Problem (PCTSP)), el vendedor recoge un beneficio de cada nodo visitado y paga una penalización por cada nodo no visitado, siendo el objetivo la minimización de la suma de los costes de viaje y las penalizaciones, pero recogiendo una cantidad mínima preestablecida de beneficios; y el TSP de cuotas, en el que el objetivo es minimizar el coste total de viaje pero el vendedor necesita cumplir una cuota. Obsérvese que el TSP de cuota es un caso particular del PCTSP en el que todas las penalizaciones son cero (para más detalles, véase [24]).

Hasta donde sabemos, el primer intento de abordar el TSPP desde una perspectiva biobjetiva fue realizado por [25]. Aplican una metaheurística híbrida para resolver el problema utilizando un proceso de expulsión en cadena como búsqueda local combinado con un algoritmo evolutivo multiobjetivo que genera una población de soluciones. En concreto, los autores inicializan la población resolviendo el procedimiento de insertar y agitar, propuesto por [26], y extendiendo el recorrido hasta que no se puedan añadir más nodos sin violar un límite de longitud. A continuación, se aplica GENIUS de [27] para intentar obtener un recorrido más corto en los nodos de la solución. Si GENIUS no consigue un recorrido mejor, el procedimiento termina. Seguidamente, todas las soluciones no dominadas se guardan en la población y el procedimiento se repite durante un número de iteraciones. Se seleccionan dos padres de la población y se aplican operadores genéticos para obtener un descendiente. Tras esto, se aplica el proceso de expulsión en cadena con los descendientes como solución inicial. La peor solución de la población se sustituye por la huella o por una solución mejor encontrada al aplicar el proceso de expulsión en cadena. Un año después, [28] resolvieron el TSPP utilizando un método de ϵ-constraint combinado con dos heurísticas de mejora para acelerar la convergencia del algoritmo. El método ϵ-constraint convierte un problema de optimización multiobjetivo en subproblemas de un solo objetivo transformando todos los objetivos excepto uno en restricciones. La primera heurística de mejora aplica algoritmos de plano de corte para reducir el tamaño del espacio de soluciones relajado añadiendo cortes. La segunda heurística de mejora utiliza las similitudes entre soluciones consecutivas obtenidas con el método de las restricciones, ya que una solución inicial mejor aumenta el límite superior del valor de la solución óptima y poda ramas por adelantado, lo que permite reducir el número de nodos explorados.

$$\text{máx} \sum_{i \in V} p_i z_i - \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

Sujeto a:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V$$

$$u_i - u_j + |V| \cdot x_{ij} \leq |V| - 1 \qquad \forall i, j \in V, i \neq j$$

$$u_i \in \mathbb{Z}, \qquad 2 \leq u_i \leq |V| \qquad \forall i > 1$$

$$z_i = \sum_{j \in V} x_{ij} \qquad \forall i \in V$$

$$\sum_{i \in V} p_i z_i \geq P_{\text{mín}}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V$$

$$z_i \in \{0, 1\} \qquad \forall i \in V$$

donde:

- V: es el conjunto de nodos.
- x_{ij}: es una variable binaria. 1 si el vendedor viaja del nodo i al nodo j. 0 en caso contrario.
- c_{ij}: es el coste de viajar entre el nodo i y el nodo j.
- p_i: es el beneficio obtenido por visitar el nodo i.
- u_i: es una variable de posición que toma valores entre 2 y V (para $i > 1$).
- $P_{\text{mín}}$: es el beneficio mínimo requerido para que la solución sea válida.
- z_i: es una variable binaria. 1 si el vendedor viaja a través del nodo i. 0 en caso contrario.

En la Figura 1.3 se muestra una solución factible para el TSPP, el cual no requiere visitar todos los nodos, donde la distancia total recorrida es 10 y el beneficio obtenido es 53. El viajante comienza en el nodo inicial denotado como "Base", se desplaza hasta el nodo "e", después al "f", continua por el "d", "a", "b", "c" y vuelve al nodo inicial "Base". El recorrido entero $K = \{i_{Base}, i_e, i_f, i_d, i_a, i_b, i_c, i_{Base}\}$, volviendo al nodo inicial, la distancia recorrida por el viajante sería de 10 $(c_{Base,e} + c_{e,f} + c_{f,d} + c_{d,a} + c_{a,b} + c_{b,c} + c_{c,Base} = 2 + 2 + 1 + 1 + 2 + 1 + 1 = 10)$, y el beneficio total sumaría 53 $(p_e + p_f + p_d + p_a + p_b + p_c = 8 + 13 + 7 + 5 + 9 + 11 = 53)$.

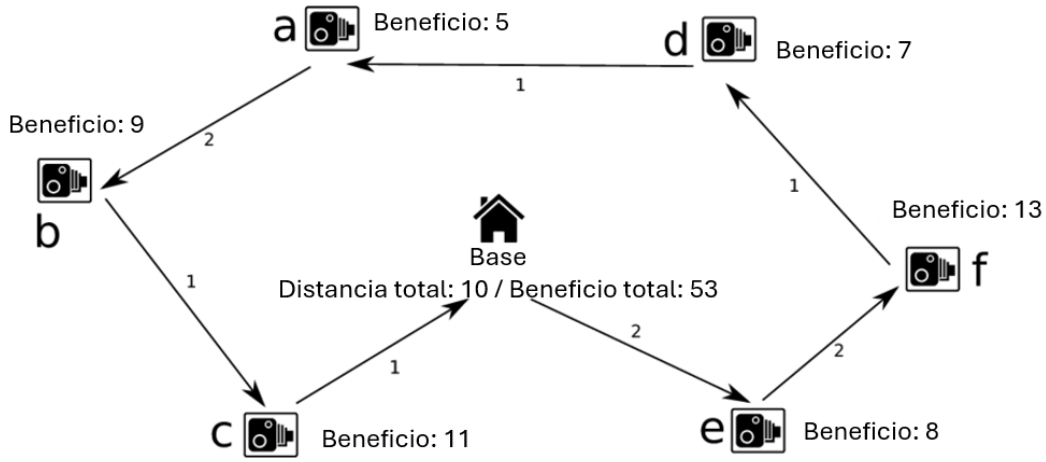


Figura 1.3: Una solución factible para el TSPP

Nodos	Recorrido	Beneficio	Distancia	FO
1	{Base, e, Base}	8	4	4
2	{Base, e, f, Base}	21	5	16
3	{Base, e, f, d, Base}	28	6	22
4	{Base, e, f, d, a, Base}	33	7	26
5	{Base, e, f, d, a, b, Base}	42	9	33
6	{Base, e, f, d, a, b, c, Base}	53	10	43

Tabla 1.1: Comparativa de diversas soluciones factibles de TSPP

En la Figura 1.4 se puede ver otra solución factible para el problema del TSPP donde la distancia total recorrida es 6 y el beneficio obtenido es 28. El viajante comienza en el nodo inicial denotado como "Base", se desplaza hasta el nodo "e", después al "f", continua por el "d" y vuelve al nodo inicial "Base". El recorrido entero $K = \{i_{Base}, i_e, i_f, i_d, i_{Base}\}$, volviendo al nodo inicial, la distancia recorrida por el viajante sería de 6 $(c_{Base,e} + c_{e,f} + c_{f,d} + c_{d,Base} = 2+2+1+1 = 6)$, y el beneficio total sumaría 28 $(p_e + p_f + p_d = 8+13+7 = 28)$. A la hora de comparar está solución con la anterior vista en en la Figura 1.3 surge el conflicto de las funciones objetivo. En la Tabla 1.1 se puede ver como al ir aumentando el número de nodos en la solución, el beneficio de esta va en aumento, en contra punto, la distancia aumenta. Ninguna solución es mejor que la otra y a estas soluciones se las denomina soluciones *eficientes, no dominadas* o *Pareto óptimas*.

1.2.2. Problema del TRP y del TRPP

Otro problema muy estudiado relacionado con el TSP es el problema del reparador viajero (TRP), también llamado problema de latencia mínima. La latencia de un nodo se define como el coste acumulado (longitud o tiempo) del camino desde el primer nodo hasta cada nodo. Entonces, este problema busca encontrar un circuito hamiltoniano abierto partiendo de un nodo inicial, que minimice la suma de los costes del camino a todos los nodos. Nótese que este problema minimiza el tiempo

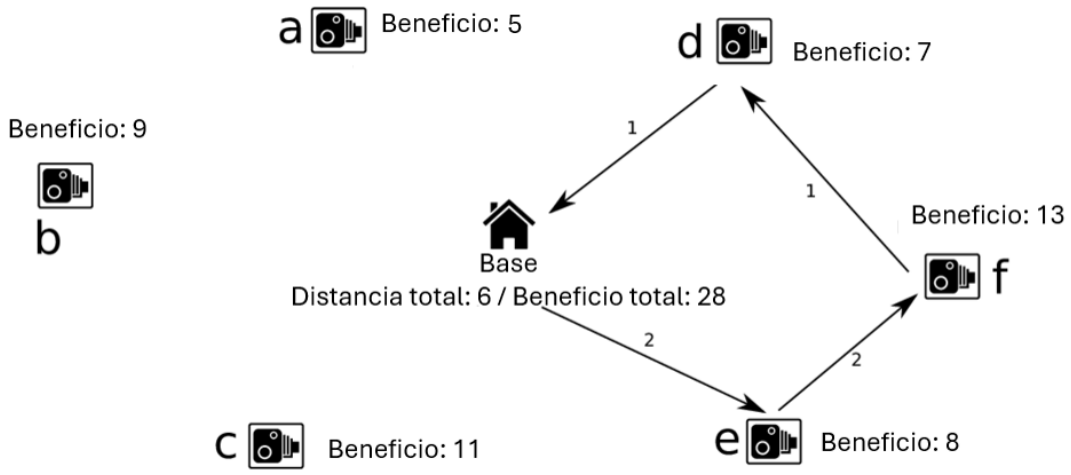


Figura 1.4: Otra solución factible para el TSPP

total de espera de todos los clientes en lugar del tiempo total de viaje del vendedor como se minimiza en el TSP. Véase, por ejemplo, [29], donde se puede encontrar una aproximación inicial en un algoritmo exacto para resolver el TRP; [30] que propusieron una metaheurística simple, basada en construcciones aleatorias voraces junto con un ordenamiento aleatorio de vecindarios para la mejora de la solución para resolver el TRP; o [31] que propusieron una búsqueda de vecindad variable (VNS) que utiliza como paso de búsqueda local cinco vecindarios incluidos en un algoritmo VND para tratar el TRP.

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} y_{ij}$$

Sujeto a:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V$$

$$u_i - u_j + |V| \cdot x_{ij} \leq |V| - 1 \qquad \forall i, j, i \neq j$$

$$u_i \in \mathbb{Z}, \qquad 2 \leq u_i \leq |V| \qquad \forall i > 1$$

$$y_{ij} \in [0, V - k + 1] \qquad \forall i, j \in V$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V$$

donde:

- V: es el conjunto de nodos.
- x_{ij}: es una variable binaria. 1 si el vendedor viaja del nodo i al nodo j. 0 en caso contrario.
- c_{ij}: es el coste de viajar entre el nodo i y el nodo j.
- y_{ij}: es una variable continua correspondiente a la latencia asociada entre el nodo i y el nodo j.
- k: es la posición del nodo en la solución.
- u_i: es una variable de posición que toma valores entre 2 y V (para $i > 1$).

En la Figura 1.5 se muestra una solución factible para el TRP donde la latencia total es 34. El viajante comienza en el nodo inicial denotado como "Base", se desplaza hasta el nodo "e", después al "f", continua por el "d", "a", "b" y finaliza en el nodo "c". El recorrido entero viene representado por $K = \{i_{Base}, i_e, i_f, i_d, i_a, i_b, i_c\}$, la latencia total del recorrido sería de 34 $(y_{Base,e} + y_{Base,f} + y_{Base,d} + y_{Base,a} + y_{Base,b} + y_{Base,c} = 2 + 4 + 5 + 6 + 8 + 9 = 34)$.

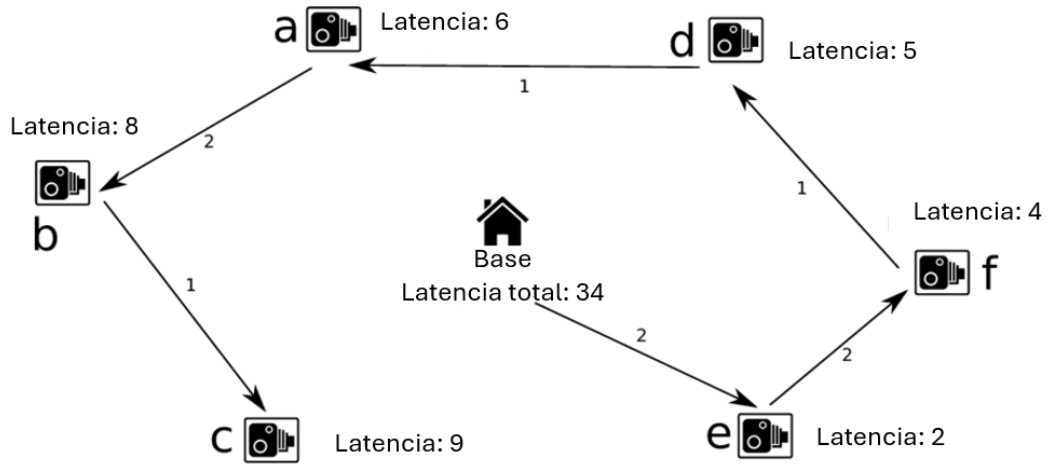


Figura 1.5: Una solución factible para el TRP

Un problema relacionado con el TRP es el TRP con beneficios (*Traveling Repairman Problem with Profits (TRPP)*), cuyo objetivo es encontrar un circuito hamiltoniano abierto tal que se maximice el beneficio recogido mientras se minimiza la latencia. Estos objetivos están en conflicto, ya que cuantos más nodos visite el vendedor, mayor será la latencia de cada nodo. La mayoría de los artículos tratan el TRPP como un problema de optimización de un solo objetivo en lugar de como un problema biobjetivo. En concreto, minimizan los ingresos que se definen como la diferencia entre el beneficio de un nodo y el número de veces que la arista que precede a ese nodo se contabiliza en la latencia total. Véase, por ejemplo, [32] cuyos autores propusieron un modelo matemático aunque resuelven el problema utilizando una metaheurística basada en búsqueda tabú; [33] que implementaron un procedimiento de búsqueda adaptativa aleatoria voraz (GRASP) para construir soluciones iniciales y una búsqueda local iterada con un mecanismo de perturbación adaptativa

para mejorarlas. El único intento de resolver este problema desde un punto de vista biobjetivo sin agregar las funciones ha sido realizado por [34] cuyos autores propusieron una búsqueda local iterada para tal fin. Una variable biobjetivo diferente fue abordada por [35] cuyos objetivos son minimizar el coste total del viaje y la latencia y los autores asumen que el beneficio puede ser visto como opuesto al coste total del viaje. Implementan un algoritmo evolutivo basado en una búsqueda local inteligente basada en las ideas seguidas por [36].

$$\text{máx} \sum_{i \in V} p_i z_i - \sum_{i \in V} \sum_{j \in V} c_{ij} y_{ij}$$

Sujeto a:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V$$

$$u_i - u_j + |V| \cdot x_{ij} \leq |V| - 1 \qquad \forall i, j, i \neq j$$

$$u_i \in \mathbb{Z}, \qquad 2 \leq u_i \leq |V| \qquad \forall i > 1$$

$$z_i = \sum_{j \in V} x_{ij} \qquad \forall i \in V$$

$$\sum_{i \in V} p_i z_i \geq P_{\text{mín}}$$

$$y_{ij} \in [0, V - k + 1] \qquad \forall i, j \in V$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V$$

$$z_i \in \{0, 1\} \qquad \forall i \in V$$

donde:

- V: es el conjunto de nodos.
- x_{ij}: es una variable binaria. 1 si el vendedor viaja del nodo i al nodo j. 0 en caso contrario.
- c_{ij}: es el coste de viajar entre el nodo i y el nodo j.

- y_{ij}: es una variable continua correspondiente a la latencia asociada entre el nodo i y el nodo j.
- k: es la posición del nodo en la solución.
- u_i: es una variable de posición que toma valores entre 2 y V (para $i > 1$).
- p_i: es beneficio obtenido por visitar el nodo i.
- z_i: es una variable binaria. 1 si el vendedor viaja a través del nodo i. 0 en caso contrario.
- $P_{\text{mín}}$: es el beneficio mínimo requerido para que la solución sea válida.

En la Figura 1.6 se muestra una solución factible para el TRPP donde la latencia total es 34 y el beneficio obtenido es 53. El viajante comienza en el nodo inicial denotado como "Base", se desplaza hasta el nodo "e", después al "f", continua por el "d", "a", "b" y finaliza en el nodo "c". El recorrido entero viene representado por $K = \{i_{Base}, i_e, i_f, i_d, i_a, i_b, i_c\}$, la latencia total del recorrido sería de 34 $(y_{Base,e} + y_{Base,f} + y_{Base,d} + y_{Base,a} + y_{Base,b} + y_{Base,c} = 2 + 4 + 5 + 6 + 8 + 9 = 34)$, y el beneficio total sumaría 53 $(p_e + p_f + p_d + p_a + p_b + p_c = 8 + 13 + 7 + 5 + 9 + 11 = 53)$.

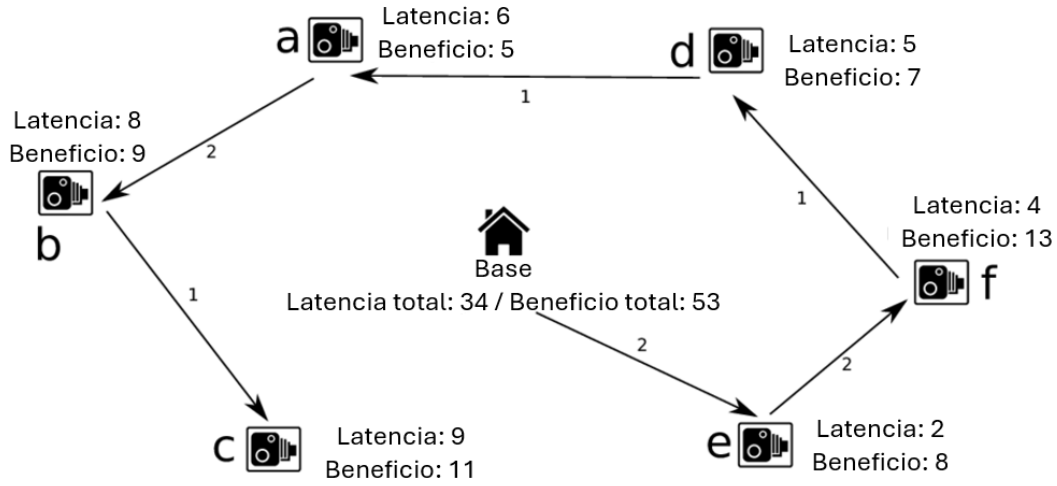


Figura 1.6: Una solución factible para el TRPP

Nodos	Recorrido	Beneficio	Latencia	FO
1	{Base, e}	8	2	6
2	{Base, e, f}	21	6	15
3	{Base, e, f, d}	28	11	17
4	{Base, e, f, d, a}	33	17	16
5	{Base, e, f, d, a, b}	42	25	17
6	{Base, e, f, d, a, b, c}	53	34	19

Tabla 1.2: Comparativa de diversas soluciones factibles de TRPP

En la Figura 1.7 se puede ver otra solución factible para el problema del TRPP donde la latencia total es 40 y el beneficio obtenido es 56. El viajante comienza en

el nodo inicial denotado como "Base", se desplaza hasta el nodo "e", después al "f", continua por el "d" y finaliza en el nodo "a". El recorrido entero $K = \{i_{Base}, i_e, i_f, i_d, i_a\}$, la latencia total del recorrido suma 17 $(y_{Base,e} + y_{Base,f} + y_{Base,d} + y_{Base,a} = 2{+}4{+}5{+}6 = 17)$, y el beneficio total llega a 33 $(p_e{+}p_f{+}p_d{+}p_a = 8{+}13{+}7{+}5 = 33)$. A la hora de comparar está solución con la anterior vista en la Figura 1.6 surge el conflicto de las funciones objetivo. En la Tabla 1.2 se puede ver como al ir aumentando el número de nodos en la solución, el beneficio de esta va en aumento, en contra punto, la distancia aumenta. Ninguna solución es mejor que la otra y a estas soluciones se las denomina soluciones *eficientes, no dominadas* u *Pareto óptimas*.

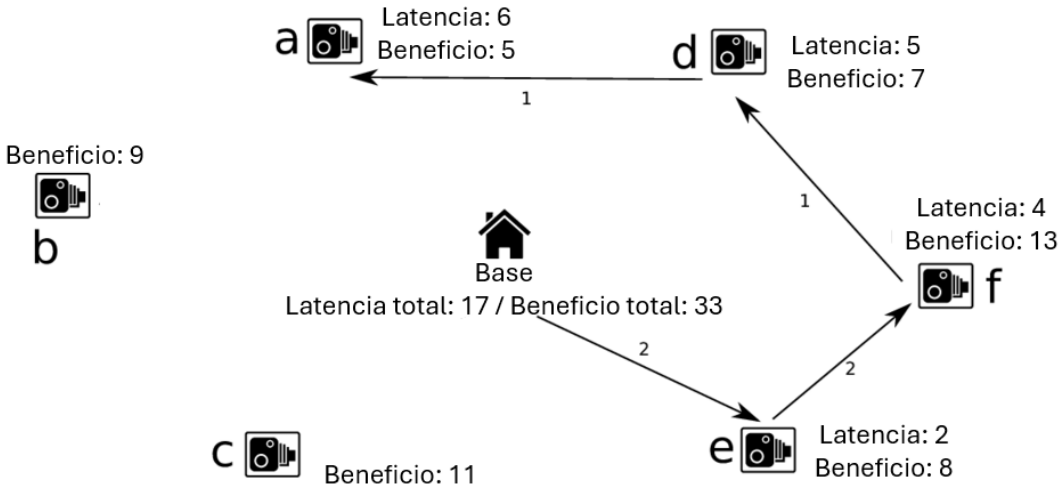


Figura 1.7: Otra solución factible para el TRPP

1.2.3. Problema del Mo-TSRPP

La combinación del TSP y del TRP únicamente es aborado en [37], cada uno con diferentes objetivos: minimizar el coste total del viaje (TSP) y minimizar el tiempo de espera de los clientes (TRP). Se presenta un modelo multiobjetivo que maneja la incertidumbre en los tiempos de viaje utilizando múltiples escenarios para representar distintas condiciones. El estudio también considera variantes con y sin restricciones de plazo para los clientes. El objetivo es explorar la complejidad de los frentes de Pareto y proponer soluciones mediante programación dinámica.

La principal contribución de esta tesis es describir y resolver un problema innovador acuñado como el problema multiobjetivo del comerciante reparador con beneficios (Mo-TSRPP). Este problema puede considerarse una generalización del TSPP y del TRPP. El Mo-TSRPP determina el orden en que debe visitarse un subconjunto del conjunto de nodos para minimizar el coste total del viaje (distancia recorrida o tiempo de viaje necesario), minimizar la latencia total y maximizar el beneficio total. Dado que este problema está motivado por una aplicación real, es interesante considerar rutas con todas las combinaciones del número de nodos visitados, por lo que hemos optado por no incluir el tamaño de la solución como objetivo, sino considerar soluciones con cada número posible de nodos visitados en el conjunto de soluciones no dominadas.

Para resolver el problema se implementa un algoritmo Variable Neighborhood Descent (VND), ya que ha demostrado ser muy eficaz a la hora de resolver problemas similares; véase [38, 39]. Desde un punto de vista multiobjetivo, VND se ha aplicado con éxito para resolver otros problemas difíciles de optimización combinatoria; véanse [16, 40, 41], y dada la versatilidad de esta metodología la hemos encontrado apropiada para resolver el Mo-TSRPP. Para generar una aproximación inicial del frente de Pareto, hemos optado por implementar un procedimiento Greedy Randomized Adaptive (GRA) que construye soluciones desde cero utilizando una función greedy e incluyendo aleatorización durante la construcción. Para ser más exactos, el procedimiento GRA es la fase de construcción del algoritmo GRASP pero sin aplicar la fase de mejora; véanse [42, 43]. De hecho, para reforzar la búsqueda y enriquecer el conjunto de soluciones no dominadas, cada vez que se incluye un nodo en la solución parcial, se envía al conjunto no dominado de soluciones intentando explorar todos los tipos de solución. A continuación, todas las soluciones se mejorarán utilizando tres vecindarios, incluidos en un marco VND, manteniendo el tamaño de la solución, para encontrar una buena aproximación de frente de Pareto.

El Mo-TSRPP puede enunciarse formalmente de la siguiente manera: Sea $G = (V, A)$ un grafo completo no dirigido, donde $V = \{1, \ldots, n\}$ es el conjunto de vértices o nodos y $A = \{(i, j) : i, j \in V, i \neq j\}$ es el conjunto de aristas. El vértice 1 representa el vértice donde el vendedor comienza y termina la ruta y $N = \{2, \ldots, n\}$ es el conjunto de clientes que requieren una visita/servicio. Cada arista $(i, j) \in A$ tiene asociado un coste finito $c_{ij} \geq 0$ (este coste podría medirse como una longitud o incluso como un tiempo). Además, cada cliente $i \in N$ paga un precio $p_i > 0$ al recibir el servicio, o lo que es lo mismo, se cobra un beneficio cada vez que se visita/sirve un vértice (nótese que el primer vértice tiene asociado un beneficio nulo, $p_1 = 0$). El Mo-TSRPP busca planificar una ruta visitando un subconjunto de clientes, $S \subset N$, para optimizar los siguientes objetivos: el coste total del viaje, la latencia total y el beneficio total, aunque indirectamente el número de clientes servidos también se tiene en cuenta.

El Mo-TSRPP es un problema de optimización multiobjetivo, ya que implica optimizar más de una FO simultáneamente y, además, están en conflicto, lo que significa que no es posible mejorar un objetivo sin deteriorar, al menos, uno de los objetivos restantes. A continuación, se combinan todas las funciones objetivo por pares para ver un posible conflicto entre ellas; véase la Tabla 1.3. Obsérvese que, a primera vista, el coste total de desplazamiento del vendedor (o tiempo total de desplazamiento o distancia total de desplazamiento) no está en conflicto con la latencia total que los clientes esperan para ser atendidos, pero podría ser un conflicto dependiendo del caso considerado, por lo que en la Tabla 1.3 se escribe “A veces”. Sin embargo, estos objetivos están en conflicto con el número de clientes servidos y con el beneficio. Por supuesto, el número de clientes que recibieron el servicio y el beneficio total obtenido por el vendedor no están en conflicto.

Para ilustrar gráficamente el conflicto entre los objetivos considerados, en las Figuras 1.8 y 1.9 se incluye un ejemplo sencillo en el que una solución factible con dos y tres clientes atendidos son representados, respectivamente. En estos ejemplos, el punto de partida está situado en $(37, 52)$ y cuatro clientes requieren el servicio $N = \{(49, 49), (52, 64), (20, 26), (40, 30)\}$ pagando $p = (23, 35, 56, 40)$.

¿Conflicto?	Coste del viaje	Latencia	Clientes servidos	Beneficio
Coste del viaje	-	A veces	Si, siempre	Si, siempre
Latencia	-	-	Si, siempre	Si, siempre
Clientes servidos	-	-	-	No, nunca
Beneficio	-	-	-	-

Tabla 1.3: Conflicto de objetivos

Ambas figuras representan dos soluciones que se compararán, una se representa en línea continua y la otra en línea discontinua.

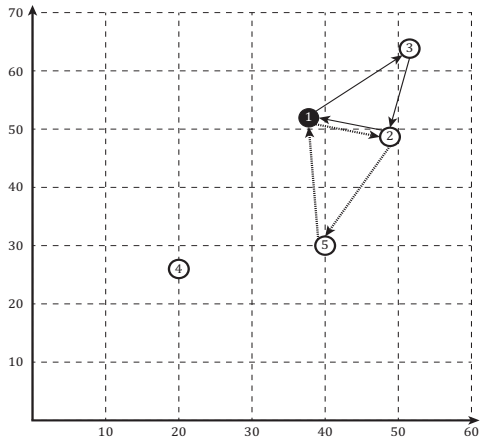


Figura 1.8: Dos soluciones factibles con 2 clientes servidos

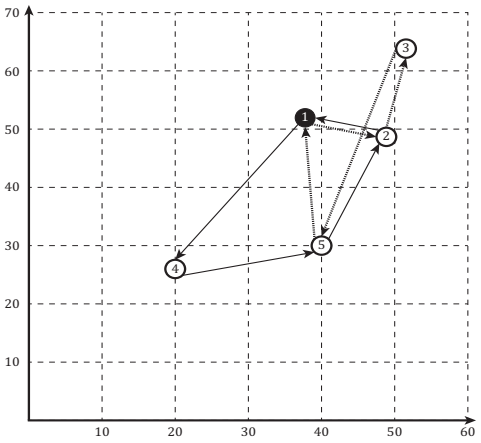


Figura 1.9: Dos soluciones factibles con 3 clientes servidos

Las soluciones de la Figura 1.8 dan servicio a dos clientes. En la primera solución, la ruta recorrida por el viajero visita primero al cliente 2 y luego al cliente 3, y los valores de las funciones objetivo son $46{,}88$ (coste total del viaje), $53{,}72$ (latencia total), 58 (beneficio total) y 2 (clientes servidos), mientras que la segunda solución realizada por el viajero visita primero al cliente 2 y luego al cliente 5 con $55{,}60$ (coste total del viaje), $45{,}76$ (latencia total), 63 (beneficio total) y 2 (clientes servidos) como valores de las funciones objetivo. Comparando ambas soluciones, se observa que la primera obtiene un mejor resultado para el coste total del viaje, pero la segunda es capaz de alcanzar mejores resultados para la latencia total y el beneficio cuando se sirve al mismo número de clientes.

Las soluciones de la Figura 1.9 sirven a tres clientes. La primera solución es la ruta que atiende primero al cliente 4, luego al cliente 5 y, por último, al cliente 2, y los valores de las funciones objetivo son $84{,}85$ (coste total del viaje), $115{,}01$ (latencia total), 119 (beneficio total) y 3 (clientes servidos), mientras que la segunda solución es la ruta que primero visita al cliente 2, luego al cliente 3 y finalmente al cliente 5 con $85{,}93$ (coste total del viaje), $103{,}76$ (latencia total), 98 (beneficio total) y 3 (clientes servidos) como valores de las funciones objetivo. Comparando ambas soluciones, se puede observar que la primera obtiene un mejor resultado para el coste total del viaje, pero la segunda solución es capaz de obtener mejores resultados para la latencia total y el beneficio cuando sirve al mismo número de clientes.

Ahora bien, si nos centramos en ambas cifras, las soluciones que visitan a dos

clientes son mejores que las soluciones que visitan a tres clientes en cuanto al coste total del viaje y la latencia total, sin embargo, cuando se visitan tres clientes en lugar de dos, el beneficio es mejor y también la satisfacción de los clientes, ya que se sirve a más clientes. En consecuencia, se dice que las cuatro soluciones son soluciones *eficientes*, *no dominadas* u *Pareto óptimas*.

Para finalizar esta Sección, se vuelve a destacar la motivación que lleva a abordar el Mo-TSRPP. El problema aparece para intentar dar solución a una situación real en la que un autónomo tiene interés en satisfacer no sólo su propio interés minimizando el coste total del viaje y maximizando el beneficio total, sino también la satisfacción de los clientes minimizando la latencia total e, indirectamente, visitando al máximo número de clientes posibles para aumentar el número de clientes satisfechos con el servicio prestado. Al autónomo también le gustaría obtener el máximo beneficio, por supuesto, al mínimo coste (normalmente medido como duración o tiempo) sin embargo, ambos objetivos están en conflicto ya que no es posible mejorar un objetivo sin deteriorar otro. Simultáneamente, a los clientes les gustaría ser atendidos esperando el mínimo tiempo posible.

1.2.4. Aplicaciones del problema

Aunque el Mo-TSRPP surge para encontrar la solución de una situación real en la que un autónomo, que repara electrodomésticos, desea realizar la planificación del servicio a los clientes, existen otras aplicaciones del mundo real que pueden modelizarse a través del Mo-TSRPP. A continuación se mencionan brevemente otras situaciones reales que pueden modelarse utilizando el Mo-TSRPP.

El primer ejemplo, desconocido para muchos, es el funcionamiento de algunas empresas en países emergentes. La escasez de muchos productos básicos hace que un vendedor ambulante intente venderlos maximizando el beneficio total, minimizando el coste del viaje, maximizando el número de ciudadanos que reciben el producto para ser justos con los ciudadanos, y minimizando la latencia ya que algunos productos son perecederos. Además, la latencia afecta a la duración del turno de los trabajadores, que es un aspecto relevante en un escenario real como el evaluado en esta investigación.

Otra interesante aplicación en el mundo real que puede modelarse utilizando el Mo-TSRPP está relacionada con el diseño de rutas para enfermeras en centros de salud rurales. En algunos centros de salud ubicados en zonas pequeñas se tiene asignada una enfermera que cada día debe visitar un conjunto de pacientes de una zona. La enfermera debe visitar al máximo número de pacientes en el mínimo tiempo posible. Además, los pacientes están esperando recibir el servicio y prefieren esperar el mínimo tiempo posible. Por supuesto, la enfermera obtendrá una hipotética medida de beneficio en función de la gravedad de los pacientes. En la situación ideal, todos los pacientes deberían ser visitados, pero esto no suele ocurrir y los pacientes no visitados serán visitados al día siguiente.

Para incluir una tercera situación real, supongamos que hay varias poblaciones (representadas como nodos en el problema) afectadas por un corte de agua y sólo

hay disponible un camión cisterna en una zona de suministrar agua potable. Cada población requiere una cantidad de agua proporcional al número de habitantes (que representan el beneficio en el modelo). El objetivo de la empresa de reparto es minimizar la distancia total recorrida y maximizar el beneficio, sin embargo el objetivo de los municipios es obtener el suministro lo antes posible, lo que se optimiza minimizando la latencia total, y por supuesto, sería importante servir al mayor número de municipios posible para evitar la insatisfacción de los habitantes.

Pero no se reduce sólo a los ejemplos expuestos, el problema del viajante de comercio tiene diversas aplicaciones prácticas en varios campos. A continuación, se enumeran algunas de las aplicaciones más comunes:

1. Logística y Distribución:
    - Optimización de rutas de entrega para minimizar costos y tiempos de transporte.
    - Planificación de rutas para servicios de mensajería y entrega de paquetes.
2. Planificación de Rutas de Vehículos:
    - Diseño de rutas eficientes para vehículos de reparto, autobuses y servicios de taxis.
3. Manufactura y Producción:
    - Optimización de rutas en procesos de manufactura y ensamblaje para minimizar el tiempo de movimiento de herramientas y piezas entre estaciones de trabajo.
4. Robótica y Drones:
    - Planificación de rutas para robots autónomos y drones en tareas de inspección, recolección y entrega.
5. Telecomunicaciones:
    - Optimización de rutas en redes de telecomunicaciones para minimizar la latencia y mejorar la eficiencia de la transmisión de datos.
6. Gestión de Proyectos:
    - Secuenciación de tareas en proyectos complejos para minimizar el tiempo total de ejecución.
7. Turismo y Viajes:
    - Planificación de itinerarios turísticos para visitar múltiples destinos de manera eficiente.
8. Biología Computacional:
    - Análisis de secuencias genómicas para encontrar patrones y relaciones entre genes.
9. Mantenimiento y Reparación:

- Optimización de rutas para técnicos de mantenimiento y reparación que deben visitar múltiples sitios.

10. Servicios Públicos:
    - Planificación de rutas para la recolección de residuos y servicios de emergencia, como bomberos y ambulancias.

Estas aplicaciones demuestran la versatilidad del TSP y su relevancia en una amplia gama de industrias y sectores, abriendo un abanico más amplio de aplicaciones para el Mo-TSRPP.

1.3. Hipótesis y objetivos

Seguidamente, se propone la hipótesis que sirve como fundamento de la investigación en esta Tesis Doctoral, junto con los diversos objetivos que se aspiran lograr durante su ejecución.

1.3.1. Hipótesis

Tras la identificación del problema a resolver, el proceso investigativo procede con la elaboración de una hipótesis inicial. Esta hipótesis constituye una proposición tentativa destinada a ofrecer una solución al problema planteado. Su importancia radica en que actúa como un punto de referencia crucial durante el desarrollo de la investigación, proporcionando orientación en todo el proceso.

La hipótesis propuesta para esta Tesis Doctoral se resume de la siguiente manera: el *Mo-TSRPP* es un problema $\mathcal{NP}$-Difícil con aplicaciones prácticas en diversas áreas de la ciencia y la ingeniería. Por esta razón, resulta importante desarrollar algoritmos capaces de resolverlo de manera eficiente, logrando soluciones óptimas o, al menos, de alta calidad en el menor tiempo posible. Los algoritmos heurísticos son clave para abordar este tipo de problemas, ya que permiten obtener soluciones de gran calidad en un tiempo aceptable, ofreciendo además enfoques multiobjetivo, dado que el problema que os concierne involucra tanto la minimización como la maximización.

Los algoritmos propuestos emplearán técnicas metaheurísticas, que han demostrado ser eficaces en la resolución de problemas de optimización. Dentro de estas, las metaheurísticas trayectoriales son una subcategoría que se caracteriza por comenzar con una solución inicial y generar una trayectoria a través del espacio de soluciones. Un ejemplo de este tipo de metaheurística es VNS [44]. Por otro lado, existe otra subfamilia llamada metaheurísticas multiarranque, las cuales se basan en la construcción y mejora de soluciones durante un número fijo de iteraciones. Un ejemplo es GRASP [45], en el que, en cada iteración, el proceso de construcción y mejora de soluciones puede asemejarse al funcionamiento de una metaheurística trayectorial.

Objetivos

El objetivo principal de esta Tesis Doctoral se desprende directamente de la hipótesis previamente enunciada, y se centra en la elaboración de algoritmos de solución para el *Mo-TSRPP*, abordándolo desde una perspectiva heurística.

El algoritmo heurístico planteado será enriquecido con la aplicación de técnicas metaheurísticas más apropiadas para el problema, específicamente, se contemplarán GRASP y VND.

Para lograr el objetivo principal descrito, es imperativo abordar los siguientes objetivos parciales:

- **Adquirir el conocimiento del estado actual del problema**: se requiere llevar a cabo una revisión bibliográfica exhaustiva para recopilar las diversas técnicas empleadas en su resolución. Asimismo, se recopilarán los conjuntos de instancias sobre los cuales se han evaluado dichos algoritmos.

- **Elaborar e implementar un algoritmo heurístico destinado a resolver el problema**: se emplearán técnicas metaheurísticas para este propósito. Específicamente, se concentrará en la creación de un algoritmo fundamentado en GRASP y VND, que abarcará tanto algoritmos constructivos como de mejora.

- **Validación del algoritmo heurístico**: Es necesario verificar la factibilidad de las soluciones generadas por el algoritmo propuesto. Asimismo, se busca confirmar que cada componente del algoritmo, refleje los conceptos planteados y contribuya al esquema general de manera significativa.

- **Realizar una comparación experimental entre el algoritmo propuesto y los algoritmos de vanguardia**: es necesario identificar los algoritmos líderes en la resolución del problema desde una perspectiva heurística y llevar a cabo una comparación rigurosa entre las distintas propuestas utilizando un conjunto de instancias de referencia.

- **Presentar los resultados parciales de la investigación a procesos de revisión por parte de instituciones independientes**: los hallazgos de la investigación serán remitidos a conferencias y revistas de renombre tanto a nivel nacional como internacional dentro del ámbito pertinente, con el fin de considerar su publicación potencial.

- **Elaborar un documento de investigación**: como paso final, se redactará el informe de la Tesis Doctoral, el cual incluirá la descripción del problema a abordar, un resumen del estado actual de la cuestión, una exposición detallada de los algoritmos propuestos, así como la presentación de las contribuciones y los resultados alcanzados.

1.4. Propuesta algorítmica

Esta sección proporciona un análisis más exhaustivo del contenido de la propuesta que se desarrollará en esta Tesis Doctoral. Específicamente, se propone abordar la resolución del *Mo-TSRPP* utilizando técnicas de resolución heurística.

1.4.1. Técnicas de resolución heurística

Se propone emplear técnicas heurísticas para generar soluciones aproximadas al Mo-TSRPP. Específicamente, se utilizarán técnicas metaheurísticas con el objetivo de obtener resultados de alta calidad en intervalos de tiempo computacionales aceptables. Sin embargo, es importante señalar que estas técnicas carecen de la capacidad de garantizar la optimalidad de las soluciones encontradas. A continuación, se presenta una breve descripción de las heurísticas y metaheurísticas GRASP y VND, propuestas para la resolución del Mo-TSRPP, las cuales serán expuestas con mayor detalle en el Capítulo 3.

Greedy Randomied Adaptive Search Procedure

GRASP es una metaheurística de propósito general, cuyo acrónimo proviene de *Greedy Randomized Adaptative Search Procedure*, que podría traducirse como "Procedi- miento de Búsqueda Voraz, Aleatoria y Adaptativa" [46]. Fue presentada por Tom Feo y Mauricio Resende en [47], aunque no fue hasta [45] cuando adquirió su terminología y forma definitiva actual.

Esta metaheurística puede clasificarse como un procedimiento de múltiples arranques, en el cual cada arranque corresponde a una iteración del algoritmo que, a su vez, se compone de dos fases principales: construcción y mejora.

Las construcciones se llevan a cabo mediante un procedimiento heurístico constructivo que simultáneamente cumple con tres de las características que definen el método: voracidad, aleatoriedad y adaptabilidad. Durante la construcción de una solución, los elementos se agregan de forma iterativa. Normalmente, en cada paso, se selecciona el elemento más prometedor de manera voraz. Sin embargo, en lugar de seleccionar siempre el elemento más prometedor, GRASP introduce cierto grado de aleatoriedad en el proceso, eligiendo uno de los mejores elementos de manera aleatoria. La introducción de aleatoriedad en la fase de construcción diversifica la búsqueda cuando se realizan múltiples construcciones, permitiendo explorar diferentes regiones del espacio de soluciones. Además, el procedimiento es adaptativo, ya que al seleccionar un nuevo elemento, se reevalúa el subconjunto de elementos prometedores, que pueden ser diferentes de la iteración anterior debido a cambios en la solución.

La fase de mejora se aplica directamente a la solución obtenida después de cada construcción. Esta mejora puede ser conducida por una búsqueda local o por

otra metaheurística, lo que hace que GRASP sea un procedimiento altamente atractivo. Después de completar el número especificado de construcciones y mejoras, el algoritmo devuelve la mejor solución encontrada.

En la Figura 1.10 se presentan las principales etapas de un diseño básico de GRASP. Como se ilustra en la figura, el procedimiento empieza inicializando la mejor solución encontrada, la cual será actualizada durante el proceso de búsqueda. En cada iteración se lleva a cabo una construcción GRASP y se mejora la solución construida. Si la nueva solución obtenida es superior a la mejor solución hasta el momento, esta se actualiza. El proceso se repite el número de iteraciones predefinido.

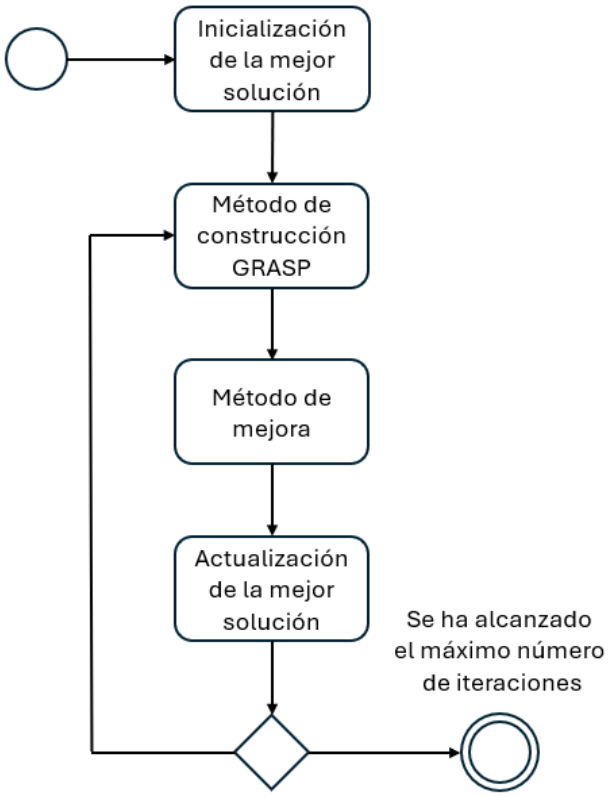


Figura 1.10: Representación esquemática de un diseño básico de GRASP

A lo largo del tiempo, GRASP ha evolucionado de diversas maneras, algunas de las variantes más extendidas son:

- Biased GRASP: es una variante en el que se introduce un sesgo o preferencia para guiar la selección de soluciones en cada iteración. Mientras que GRASP clásico construye soluciones mediante la elección aleatoria y adaptativa de elementos basados en un criterio de factibilidad y calidad, el Biased GRASP dirige esta aleatoriedad hacia opciones específicas que se consideran más prometedoras. Esto se logra utilizando una función de probabilidad o un mecanismo de sesgo que incrementa la probabilidad de seleccionar ciertas opciones en función de características particulares, como el coste o la distancia en el caso del TSP.

- Reactive GRASP: Ajusta dinámicamente el parámetro de aleatoriedad, siendo este usualmente el tamaño de la lista restringida de candidatos (RCL), en función del rendimiento observado de las soluciones anteriores. Así, el algoritmo puede adaptarse y concentrarse en configuraciones más efectivas de manera automática.
- Hybrid GRASP: Combina GRASP con otros métodos metaheurísticos, como algoritmos genéticos, recocido simulado o búsqueda tabu. Esta hibridación permite aprovechar las fortalezas de cada método, como la intensificación de GRASP y la exploración global de otras metaheurísticas.
- VNS-GRASP: Combina GRASP con el VNS al aplicar múltiples vecindades durante la búsqueda local. Esta combinación ayuda al algoritmo a escapar de óptimos locales y explorar diferentes regiones del espacio de soluciones, como en [48].
- Path-Relinking GRASP: Integra el método de Path Relinking como una fase posterior a la búsqueda local de GRASP. En esta variante, se explora el "camino" entre una solución inicial y una solución objetivo (generalmente una de las mejores soluciones encontradas) en un intento de generar soluciones intermedias de alta calidad. Un ejemplo se encuentra en [49].

Variable Neighborhood Search

La búsqueda de vencidad variable (Variable Neighborhood Search (VNS)), es una metaheurísitica desarrollada por P. Hansen y N. Mladenović en 1997 [44], cuyo propósito es evitar la convergencia en óptimos locales. VNS explora sistemáticamente diferentes vecindarios, combinando un movimiento de perturbación. Un ejemplo de su aplicación en problemas de enrutamiento se encuentra en [50], donde los autores presentan el algoritmo Alternate K-exchange Reduction (AKRed), el cual, en su fase de mejora, emplea la metaheurísitica VNS con las búsquedas locales k-opt, intercambio e inserción.

A continuación se mencionan algunas variantes del VNS:

- Centradas en diversificar: La Búsqueda de Vecindad Variable Aleatoria, o Reduced Variable Neighborhood Search (RVNS), es un ejemplo de este tipo. A diferencia de VNS, que combina fases de búsqueda local y cambios sistemáticos de vecindad, RVNS omite la fase de búsqueda local y se centra en explorar diferentes vecindades de manera aleatoria. Este enfoque reduce la complejidad computacional al evitar la búsqueda exhaustiva dentro de cada vecindad, lo que puede ser beneficioso en problemas de gran escala o cuando el tiempo de cómputo es limitado.
- Centradas en intensificar: como VND. Esta variante omite la fase de perturbación, lo que la hace determinista. Se centra en la búsqueda local. Explora secuencialmente varias vecindades, pero, a diferencia de VNS, solo cambia de vecindad cuando se ha encontrado una mejora en la solución. Se queda en la vecindad actual hasta que no encuentra más mejoras, y luego pasa a una vecindad diferente. En resumen, VND sigue una estrategia de descenso,

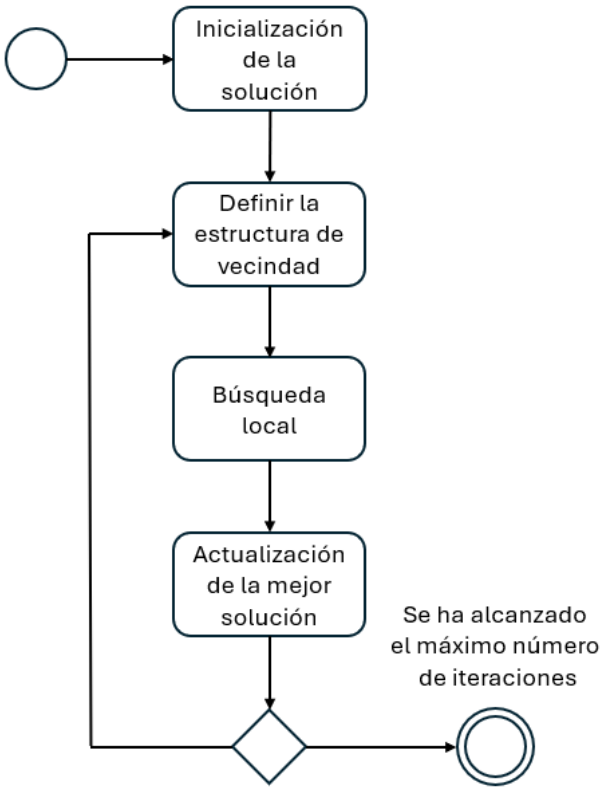


Figura 1.11: Representación esquemática de un diseño básico de VNS

donde el algoritmo se queda en una vecindad hasta encontrar una mejora y luego cambia a una nueva vecindad solo si ya no puede mejorar en la actual. En [38] se estudia el VND y su aplicación en el TSP.

- Combinación de Diversificar e intensificar: VNS Básico, o Basic Variable Neighborhood Search (BVNS), mantiene la esencia de la exploración de vecindades pero sin algunos de los elementos avanzados y adaptativos que pueden encontrarse en otras variantes de VNS. BVNS se centra en un esquema más sencillo de cambio de vecindad, intensificación y diversificación, y resulta útil para problemas donde un enfoque básico puede proporcionar soluciones de buena calidad con menor complejidad computacional.
- Adaptativos: En lugar de utilizar un conjunto predefinido de vecindarios, en esta variante, el tamaño y la estructura de los vecindarios se ajustan dinámicamente durante la ejecución del algoritmo. Por ejemplo, Adaptive Variable Neighborhood Search (AVNS) ajusta dinámicamente los parámetros de las vecindades en función del progreso de la búsqueda y Skewed Variable Neighborhood Search (SVNS) introduce un sesgo en el proceso de perturbación para orientar la búsqueda hacia regiones menos exploradas del espacio de soluciones.
- Guiar la solución. Se incorpora una estrategia de guía para dirigir la búsqueda hacia soluciones prometedoras. Mientras que el VNS tradicional explora diferentes vecindades de manera sistemática o aleatoria, el Greedy Variable

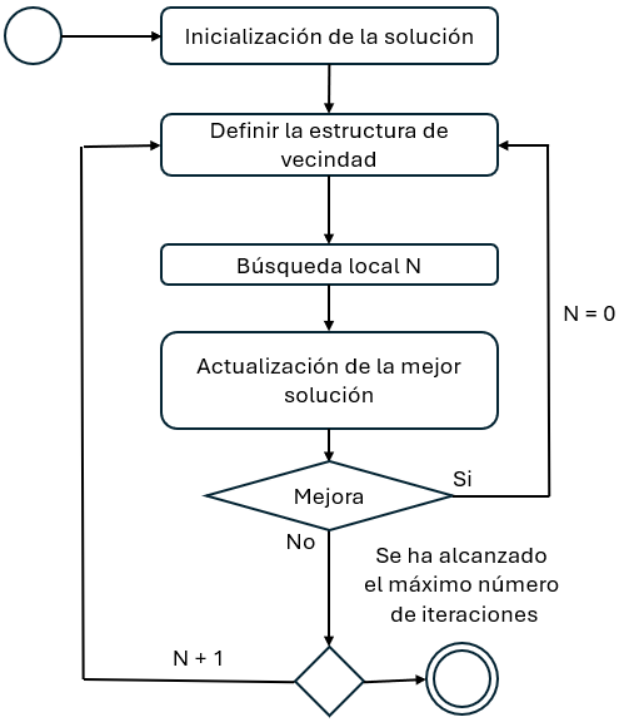


Figura 1.12: Representación esquemática de un diseño básico de VND

Neighborhood Search (GVNS) utiliza una heurística adicional o criterios de aprendizaje para seleccionar vecindades que tienen más probabilidades de mejorar la solución. En [51] se aplica satisfactoriamente el GVNS para resolver un TSP asimétrico.

1.5. Método de investigación

El proceso de investigación en optimización condensa los pasos fundamentales a seguir en el desarrollo de una investigación científica en este campo. Este proceso comparte similitudes con otras áreas, pero también posee características específicas que son únicas en la optimización y requieren una atención especial.

Una vez que se ha identificado el problema de optimización a resolver, el proceso se inicia con una revisión exhaustiva del estado del arte del mismo. Esta revisión implica la identificación de los algoritmos previamente desarrollados para abordar el problema en cuestión, así como el conjunto de instancias utilizadas en su evaluación. Basándose en esta revisión, se formula una hipótesis para la resolución del problema, que luego se traduce en la implementación de uno o varios algoritmos específicos. La validación de estos algoritmos se lleva a cabo en varias etapas. En primer lugar, se verifica la integridad del código para garantizar que no contiene errores. Luego, se evalúa la contribución de los componentes individuales del algoritmo y, por lo tanto, la efectividad de las estrategias propuestas. Si los resultados de estas validaciones preliminares son satisfactorios, se realiza una validación final de la propuesta, comparando sus resultados con los de otros algoritmos presentes

en el estado del arte. Para esto, se utilizan conjuntos de instancias de referencia que permiten la comparación entre los algoritmos. Este proceso es iterativo, ya que es común que, después de completar las validaciones, sea necesario regresar a la etapa de desarrollo o incluso ajustar la hipótesis inicial.

El proceso de investigación descrito se representa visualmente en el diagrama de actividad mostrado en la Figura 1.13. Como se evidencia en dicha figura, si los resultados finales corroboran la hipótesis inicial, el proceso finaliza con la publicación de la misma junto con los resultados experimentales.

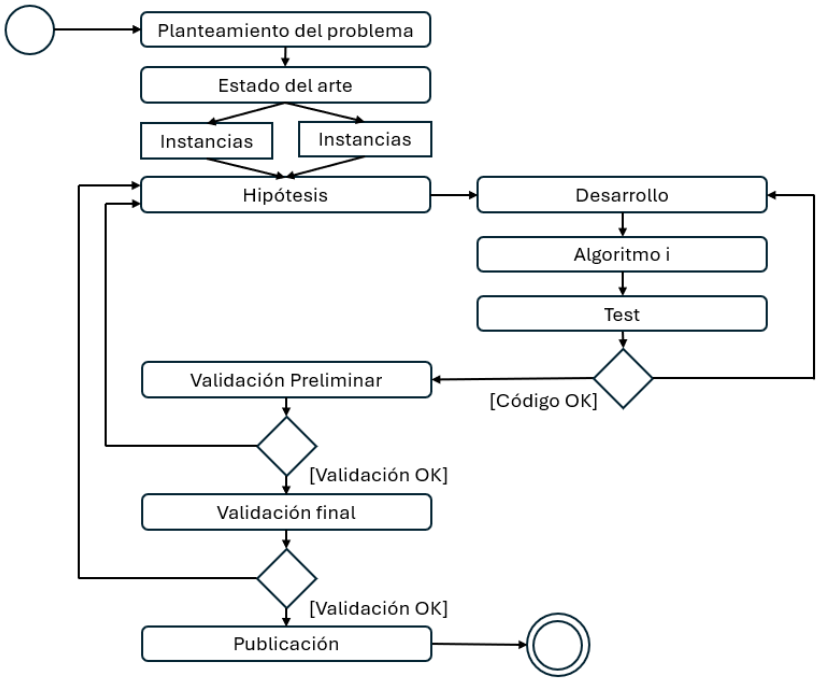


Figura 1.13: Diagrama de flujo de actividades del proceso de investigación en el ámbito de optimización.

Algunas demandas surgidas de las etapas más críticas del proceso de investigación son:

- Estado del arte: El resultado de esta fase implica la identificación de un conjunto relevante de algoritmos, artículos e instancias asociados al problema objetivo. Por lo tanto, es crucial establecer un método eficiente para almacenar y acceder a esta información.
- Desarrollo: La fase de implementación requiere una considerable cantidad de tiempo en el proceso de investigación. Se puede optimizar mediante el uso de herramientas como entornos de desarrollo y repositorios de código. Además, dado que muchos esquemas algorítmicos se utilizan repetidamente, tener plantillas con las partes comunes ya implementadas puede simplificar la tarea del desarrollador.
- Validación: En esta etapa se requieren mecanismos para ejecutar algoritmos tanto localmente como de forma remota, junto con herramientas para analizar los resultados obtenidos. El diseño experimental es una tarea compleja que se

beneficia de la definición de procesos sistemáticos para llevar a cabo experimentos. Además, los análisis realizados suelen estar dirigidos por estadísticos bien establecidos, por lo que el uso de librerías que incluyan los análisis más comunes puede resultar muy útil.

- Difusión o publicación de resultados: Durante esta fase, es crucial tener acceso nuevamente a la información recopilada en la etapa de revisión del estado del arte. Además, se requieren soportes auxiliares, como una página web, que faciliten el acceso a instancias y a los mejores resultados encontrados.

1.6. Medios utilizados

La ejecución de este trabajo de investigación se ha llevado a cabo dentro del grupo de investigación GRAFO, integrado por profesores adscritos al Departamento de Informática y Estadística de la Escuela Técnica Superior de Ingeniería Informática de la Universidad Rey Juan Carlos. Además, durante el transcurso de esta investigación, se ha mantenido una estrecha colaboración con el Departamento de Economía, Métodos Cuantitativos e Historia Económica de la Universidad Pablo de Olavide. Para llevar a cabo esta Tesis Doctoral, se han utilizado diversos recursos, entre los cuales se destacan:

- Laboratorio de investigación del grupo GRAFO (*Group for Research in Algorithms For Optimization*. Cuyos cabezas de grupo son los reconocidos investigadores Abraham Duarte y Eduardo García Pardo.
- Biblioteca de la Universidad Rey Juan Carlos, con acceso al préstamo interbibliotecario, Consorcio Madroño, bases de datos especializadas y acceso a las bibliotecas digitales de IEEE y ACM, entre otros recursos.
- $Hardware$ provisto por el grupo de investigación: AMD Ryzen 9 5950x 16 cores ($3{,}4$ GHz) con 128GB RAM.

1.7. Estructura de la memoria

Para concluir el capítulo introductorio de esta Tesis Doctoral, se esboza de manera concisa la estructura del resto del documento, proporcionando una breve descripción de cada capítulo que lo compone:

- Capítulo 2: En este capítulo se presenta un estado del arte del "problema del viajante reparador de comercio con beneficios". Se recopilan y clasifican las publicaciones más relevantes relacionadas con el Mo-TSRPP, poniendo especial énfasis en los trabajos que abordan su resolución desde una perspectiva tanto exacta como heurística. Se describen los algoritmos heurísticos desarrollados para su resolución.
- Capítulo 3: Este capítulo aborda los algoritmos propuestos para resolver el problema desde una perspectiva heurística. Comienza con una revisión de las

diversas técnicas disponibles para la resolución aproximada de problemas de optimización. Luego, se detallan las heurísticas propuestas para abordar el problema, centrándose en un algoritmo basado en la combinación de GRASP y VND. Se diseñan varias heurísticas constructivas, de mejora y métodos de combinación de soluciones dentro de este enfoque.

- Capítulo 4: En este capítulo se realiza una exhaustiva evaluación experimental de los algoritmos propuestos en el Capítulo 3. La validación de estas propuestas se lleva a cabo mediante la comparación de los resultados obtenidos por dichos algoritmos con los resultados obtenidos por los algoritmos del estado del arte.
- Capítulo 5: Finalmente, en el Capítulo Quinto se expondrán las conclusiones de este trabajo de investigación. Se resumirán las principales contribuciones derivadas del mismo, incluyendo las publicaciones resultantes, así como las áreas potenciales para investigaciones futuras.

Capítulo 2

Estado del arte

En este capítulo se repasan los trabajos relacionados con "el problema multiobjetivo del comerciante reparador con beneficios". En él, se recogen los algoritmos heurísticos considerados más importantes presentes en el estado del arte. No obstante, de acuerdo con el enfoque de esta Tesis Doctoral, sólo se describirán en detalle los algoritmos que abordan el problema desde un punto de vista heurístico.

2.1. Introducción

Revisando la literatura se llega a la conclusión de que hasta la fecha no se ha tratado el problema del Mo-TSRPP. Sin embargo, si se han estudiando problemas similares como el TSP y el TRP (ambos con un único objetivo), el TSPP y el TRPP (con dos objetivos aunque usualmente se agregan en un único objetivo), e incluso el Multi-objective Traveling Salesman Repairman Problem (Mo-TSRP) sin beneficio. Existen estudios que investigan su complejidad computacional, otros que proponen relajaciones del problema, aplicaciones prácticas, algoritmos aproximados, algoritmos polinómicos para clases específicas de grafos, e incluso una formulación completa del problema.

En este capítulo se realiza un recorrido por los trabajos mencionados. Además, se presenta un histograma que resume la aparición cronológica de las publicaciones relacionadas con los problemas relacionados con el Mo-TSRPP.

Los problemas de optimización a menudo reciben diferentes denominaciones en las publicaciones del estado del arte. Asimismo, es común encontrar problemas con nombres similares que no corresponden exactamente al mismo problema. Cuando se pretende trabajar en un problema específico, es crucial identificar los distintos nombres que los autores han utilizado para referirse a él en sus publicaciones. Esto, a su vez, permite identificar los métodos aplicados para abordar dicho problema. En particular, se han encontrado referencias a problemas relacionados con el "problema del viajante de comercio reparador con beneficio" con las siguientes denominaciones:

- Traveling Salesman Repairman Problem (TSRP)

- Traveling Salesman Problem with Profits (TSPP)
- Multi-objective Traveling Salesman Problem with Profits (Mo-TSPP)
- Traveling Repairman Problem with Profits (TRPP)
- Minimum Latency Problem with Profits (MLPP)

2.1.1. Complejidad computacional

Como se ha mencionado en la Sección 1.1, existe una clase de problemas conocida como $\mathcal{NP}$-Completos, para la cual no se ha encontrado aún un algoritmo que los resuelva en tiempo polinómico, aunque tampoco se ha demostrado que no pueda existir tal algoritmo. Lo distintivo de esta clase es que, si se lograra resolver uno de estos problemas en tiempo polinómico, es decir, si se descubriera un algoritmo eficiente para uno de ellos, entonces todos los problemas de esta familia también podrían resolverse en tiempo polinómico.

La base de la teoría de los problemas $\mathcal{NP}$-Completos fue establecida por Stephen A. Cook en 1971, quien en su artículo *The Complexity of Theorem-Proving Procedures* [52], destacó el concepto de "*polynomial time reducibility*" o "reducibilidad en tiempo polinómico". Este término se refiere a aquellas transformaciones entre problemas que pueden realizarse en tiempo polinómico. En otras palabras, si un problema puede ser reducido en tiempo polinómico a otro, cualquier algoritmo polinómico para el segundo problema se puede adaptar para resolver el primero. Además, Cook se enfocó en la "clase de problemas de decisión $\mathcal{NP}$", que incluye problemas cuya respuesta es "sí" o "no". De manera crucial, demostró que un problema dentro de $\mathcal{NP}$, conocido como "*Satisfiability*" (SAT), posee la propiedad de que cualquier otro problema $\mathcal{NP}$ puede ser reducido a él en tiempo polinómico. Si este problema se resolviera en tiempo polinómico, todos los problemas $\mathcal{NP}$ podrían resolverse de manera eficiente. Sin embargo, si alguno de los problemas $\mathcal{NP}$ resultara ser intratable (es decir, no resoluble en tiempo polinómico), entonces SAT también sería intratable.

Poco después de la publicación de Cook, Richard Karp presentó una serie de resultados en los que demostró que las versiones de decisión de muchos problemas combinatorios ampliamente conocidos eran, al menos, tan complejas como el problema SAT [53]. La Figura 2.1 ilustra cómo se relacionan los problemas estudiados por Karp en términos de la reducción de unos a otros. Entre ellos se encuentra el denominado problema del ciclo hamiltoniano (en inglés *Directed Hamilton Circuit*), a partir del cual se demostró que el problema del viajante de comercio (equivalente al problema del ciclo hamiltoniano pero fijando la distancia entre dos nodos en uno si son adyacentes y en dos en caso contrario, y comprobando que la distancia total recorrida es igual a n) era $\mathcal{NP}$-Difícil. Esto tiene su explicación matemática por la evidente dificultad computacional para encontrar recorridos óptimos.

En el problema se generan $n!$ rutas posibles, pero se puede simplificar el cálculo al considerar que el punto de partida de una ruta no importa, lo que reduce el número de rutas a examinar a $(n-1)!$. Además, dado que la dirección del viaje no

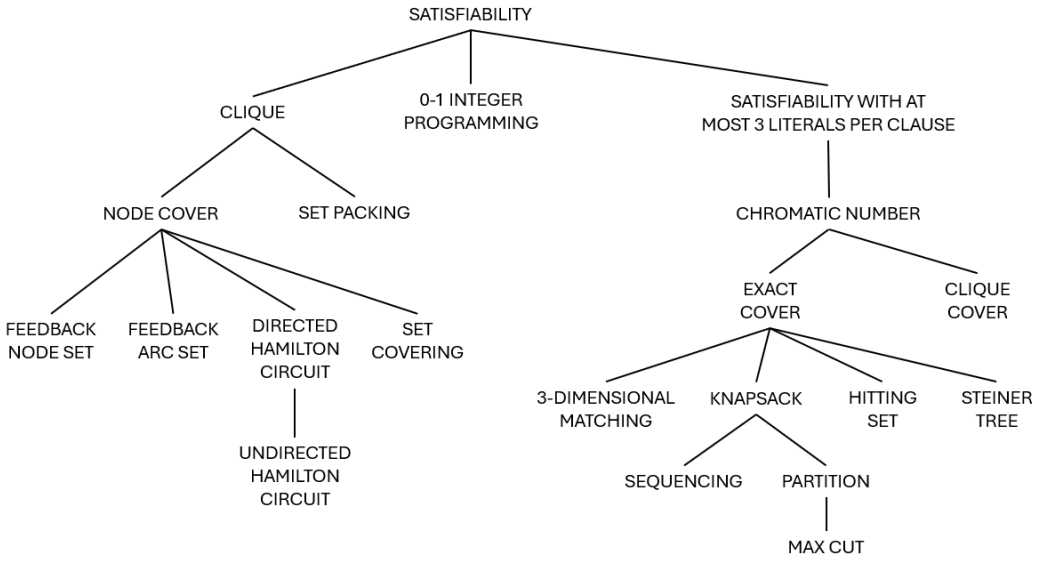


Figura 2.1: Representación esquemática de los 21 problemas $\mathcal{NP}$-Completos enunciados por Karp

afecta la solución, el número de rutas se reduce aún más en un factor de 2. Por lo tanto, el total de rutas a considerar es $(n-1)!/2$.

En la práctica, para un problema del viajante con 5 nodos, hay $(5-1)!/2 = 12$ rutas diferentes. Sin embargo, a medida que aumenta el número de nodos, el número de posibilidades crece de manera factorial:

- Para 10 nodos, hay $(10-1)!/2 = 181440$ rutas diferentes.
- Para 30 nodos, hay más de 4×10^{30} rutas posibles. Un ordenador que calcule un millón de rutas por segundo necesitaría 10^{17} años para resolver el problema. En otras palabras, si se hubiera empezado a calcular al inicio del universo (hace aproximadamente 13400 millones de años), aún no se habría completado.

Se puede observar que cada nuevo nodo incrementa el número de rutas en un factor de n, creciendo factorialmente.

2.1.2. Aplicaciones del problema Mo-TSRPP

Como se menciona en el Capítulo 1, el Mo-TSRPP es un problema con diversas apicaciones prácticas. Los trabajos publicados al respecto se clasifican según la aplicación práctica del problema. A continuación se revisan dichas aplicaciones:

1. **Logística y Distribución**

   En el campo de Logística y Distribución, el TSP se aplica frecuentemente para optimizar la planificación de rutas en situaciones donde se deben visitar múltiples ubicaciones con un vehículo, minimizando la distancia total recorri-

da o los costes asociados. Este problema es fundamental para las empresas de logística que buscan entregar productos o servicios de manera eficiente y rápida.

En la logística y distribución, una aplicación clásica del TSP es la planificación de rutas de distribución de mercancías. Se puede imaginar una empresa que debe entregar productos a una serie de puntos de entrega (clientes) ubicados en diferentes ubicaciones geográficas. El desafío es encontrar la ruta más corta o más rápida, según se mida en distancia o en tiempo, que permita visitar cada punto de entrega una sola vez y regresar al punto de partida.

En [54], se aborda una de las variantes más importantes del clásico TSP, el Multiple Traveling Salesman Problem. Este problema surge en situaciones donde hay que planificar en lugar de una única ruta, varias rutas para visitar a todos los nodos una única vez partiendo y regresando al punto de partida, de forma que se minimicen los costes o la distancia recorrida. El interés en el Multiple Traveling Salesman Problem (mTSP) proviene principalmente de su aplicación en la logística y distribución, donde las empresas buscan mejorar la eficiencia de rutas de entrega y la asignación de recursos.

El mTSP aparece para afrontar una situación más realista que el TSP en la cual surge de la necesidad de encontrar soluciones optimizadas para problemas de logística con múltiples vehículos. En el mundo real, las empresas a menudo tienen flotas de camiones o vehículos que deben repartir productos en varias ubicaciones.

Bektas analiza que el mTSP es una generalización del TSP clásico, donde en lugar de tener una sola ruta que debe recorrer todas los nodos, ahora se tiene más de una ruta, y cada uno debe cubrir una parte del total de nodos, partiendo y regresando al mismo punto de partida. Esto refleja situaciones más realistas, como la asignación de múltiples camiones a diferentes rutas en la logística de distribución.

Dado que el mTSP es un problema $\mathcal{NP}$-Completo, resolverlo de manera exacta para grandes instancias es extremadamente difícil, por lo que en el artículo se proponen y analizan diversas heurísticas y metaheurísticas para obtener soluciones aproximadas en tiempos razonables. Bektas clasifica las soluciones en dos enfoques principales: algoritmos exactos y algoritmos heurísticos/metaheurísticos.

Algoritmos Exactos:

- Programación Lineal Entera Mixta (*Mixed Integer Linear Programming*, MILP): En el enfoque exacto, el problema se formula como un problema de programación lineal entera mixta. Esta formulación incluye restricciones para asegurar que cada nodo sea visitado una vez y que cada ruta regrese al punto de partida. Sin embargo, debido a su complejidad, este método solo puede resolver problemas pequeños, de hasta 100 nodos en problemas simétricos y un número de rutas limitado y, de 500 para problemas asimétricos dependiendo de las características del problema y del número de rutas.

- Descomposición de Benders: Esta técnica divide el problema en subproblemas más pequeños, resolviendo primero la asignación de nodos a rutas y luego optimizando las rutas dentro de esas asignaciones, es decir, determinando el orden en el cual deben visitarse los nodos que han sido asignados a esa ruta. Esta es una aproximación eficiente para instancias más grandes, llegando a resolver problemas de miles de nodos, pero aún es limitada por el tamaño del problema.

Algoritmos Heurísticos y Metaheurísticos: Debido a las limitaciones de los algoritmos exactos, el enfoque más común para resolver instancias más grandes del mTSP es el uso de heurísticas y metaheurísticas, que pueden encontrar soluciones cercanas a la óptima en tiempos razonables.

- Algoritmo de Búsqueda Tabú (*Tabu Search*): Se basa en realizar una búsqueda local mejorando las mejores soluciones dadas, mientras se evita regresar a soluciones previamente visitadas. Para el mTSP, se adapta el algoritmo para manejar múltiples rutas, permitiendo una búsqueda más amplia del espacio de soluciones.
- Algoritmo de recocido simulado (*Simulated Annealing*): Este algoritmo se inspira en el proceso de enfriamiento de los metales. Comienza explorando aleatoriamente el espacio de soluciones y, gradualmente, se enfoca en mejorar las soluciones al reducir la temperatura de búsqueda. En el mTSP, este enfoque es útil para escapar de óptimos locales y mejorar las soluciones globales.
- Algoritmos Genéticos (*Genetic Algorithm*): El mTSP se puede abordar utilizando algoritmos genéticos, que simulan el proceso de evolución natural. Las soluciones iniciales (representadas como cromosomas) se seleccionan, cruzan y mutan, generando nuevas soluciones. Este proceso iterativo permite una optimización gradual y efectiva.
- Algoritmos de Colonias de Hormigas (*Ant Colony*): Inspirado por el comportamiento de las hormigas en la naturaleza, este enfoque simula cómo las hormigas encuentran rutas cortas entre el nido y la comida. Para el mTSP, cada "hormiga" simula una posible ruta, y el algoritmo mejora las soluciones a medida que más hormigas refuerzan las mejores rutas encontradas.

Bektas señala que, aunque los métodos exactos son efectivos para problemas pequeños, los métodos heurísticos y metaheurísticos son los más utilizados en la práctica debido a su capacidad para ofrecer soluciones de alta calidad en tiempos razonables. Los resultados obtenidos a través de estos métodos en aplicaciones de logística real muestran importantes mejoras en la eficiencia de las rutas de distribución, reducción de costes y optimización del uso de vehículos.

Bektas concluye su artículo revisando las aplicaciones reales del mTSP en logística y distribución. Destaca cómo la planificación de rutas y la asignación de vehículos en grandes redes logísticas pueden optimizarse utilizando estos enfoques. También menciona la importancia de tener en cuenta restricciones

reales, como ventanas de tiempo o limitaciones de capacidad de los vehículos, pero son restricciones que no se contemplan en el mTSP.

2. **Planificación de Rutas de Vehículos**

   En el campo de la Planificación de Rutas de Vehículos, el TSP es fundamental para resolver problemas relacionados con la optimización de rutas, como la gestión eficiente de flotas de vehículos que deben visitar múltiples ubicaciones en un área geográfica. El objetivo es minimizar la distancia total recorrida o el tiempo de viaje, considerando diversas restricciones como las ventanas de tiempo en que los clientes pueden ser visitados, las capacidades de los vehículos y las rutas de tráfico.

   El trabajo de Laporte [55], es un trabajo fundamental en la evolución de los estudios sobre el TSP y su aplicación en problemas de planificación de rutas, como el *Vehicle Routing Problem* (VRP).

   La idea principal de Laporte es proporcionar una visión general de las metodologías existentes para resolver el TSP, clasificándolas en dos grandes grupos: los algoritmos exactos y los algoritmos aproximados. En su análisis, Laporte pone especial énfasis en los desafíos computacionales que enfrentan estos algoritmos, dado que el TSP es un problema $\mathcal{NP}$-Difícil. La importancia del estudio radica en que muchas aplicaciones industriales reales, como la planificación de rutas de vehículos, necesitan soluciones rápidas y de alta calidad, lo cual exige tanto métodos exactos como heurísticos.

   Laporte revisa los algoritmos exactos que garantizan encontrar la solución óptima del problema, pero que son computacionalmente muy costosos y solo aplicables a instancias pequeñas del TSP. Los principales algoritmos exactos discutidos en el artículo son:

   - Ramificación y Poda (*Branch and Bound*): Este algoritmo explora el espacio de soluciones de manera sistemática, pero podando ramas del árbol de búsqueda cuando se determina que no pueden mejorar la mejor solución encontrada hasta el momento. Aunque asegura la solución óptima, su tiempo de ejecución crece exponencialmente con el número de nodos.
   - Programación Dinámica (*Dynamic Programming*): Utiliza un enfoque de "divide y vencerás", descomponiendo el problema en subproblemas más pequeños que se resuelven de manera óptima. Este método también sufre de limitaciones en problemas de gran tamaño debido a su complejidad exponencial.
   - Ramificación y Corte (*Branch and Cut*): Una variación del Branch and Bound que combina cortes lineales para reducir el espacio de búsqueda. Se aplica especialmente en instancias grandes de TSP donde se requieren soluciones exactas.

   Para superar las limitaciones de los algoritmos exactos, Laporte también revisa varios algoritmos aproximados y heurísticos que generan soluciones subópti-

mas en tiempos razonables. Algunos de los más relevantes son:

- Algoritmos de Búsqueda Local: Se parte de una solución inicial y se mejora iterativamente. Ejemplos incluyen Recocido Simulado y Búsqueda Tabú, los cuales permiten escapar de óptimos locales explorando soluciones cercanas. Aunque no garantizan la solución óptima, logran buenos resultados en problemas reales.
- Algoritmos Genéticos.
- Algoritmos de Colonia de Hormigas.

Laporte explora cómo las variantes del TSP, como el VRP, se aplican en la optimización de rutas de vehículos. Este problema extiende el TSP para incluir múltiples vehículos y restricciones adicionales, como las ventanas de tiempo y las capacidades de carga. Los resultados obtenidos en la resolución del VRP con algoritmos heurísticos demostraron que las aproximaciones basadas en el TSP pueden ser adaptadas con éxito para problemas más complejos en la industria logística.

El estudio concluye que, aunque los algoritmos exactos ofrecen soluciones óptimas, su aplicabilidad se limita a instancias pequeñas debido a su alta complejidad computacional. Los algoritmos heurísticos y metaheurísticos, en cambio, son más flexibles y se desempeñan mejor en instancias grandes, proporcionando soluciones cercanas al óptimo en tiempos razonables.

En el caso de la planificación de rutas de vehículos, los métodos heurísticos como el Recocido Simulado y la Búsqueda Tabú mostraron ser especialmente efectivos, con mejoras notables en la eficiencia de las rutas y reducciones de costes en contextos reales de transporte y logística.

El trabajo de Laporte proporciona una base sólida para la comprensión de los diferentes enfoques en la resolución del TSP y sus variantes, como el VRP. Su análisis de algoritmos exactos y heurísticos ha sido de gran valor en la planificación de rutas de vehículos, un área con aplicaciones directas en industrias que requieren optimización logística. El artículo ha influido significativamente en la investigación futura, proporcionando una referencia esencial para quienes buscan resolver problemas complejos de planificación de rutas.

3. **Manufactura y Producción**

En el campo de Manufactura y Producción, el TSP se utiliza para optimizar procesos industriales, como la secuenciación de tareas en líneas de producción o la planificación de operaciones de máquinas. El objetivo es minimizar el tiempo o los costes asociados con el movimiento entre diferentes estaciones de trabajo, tareas o máquinas, mientras se completa una lista de operaciones.

El artículo de Bérubé, Gendreau y Potvin [56] se centra en la optimización biobjetivo y, en particular, en el TSP con beneficios (TSPP), del que versa esta tesis. En esta variante, no es necesario visitar todos los nodos, sino que

se busca seleccionar un subconjunto de ellos, de forma que se maximicen las ganancias y a su vez se minimicen los costes asociados al recorrido realizado.

La idea para desarrollar el TSPP proviene de la necesidad en el campo de la manufactura y producción de resolver problemas en los que hay dos objetivos en conflicto, maximizar los beneficios obtenidos al visitar los nodos y minimizar los costes que se incurren al realizar el recorrido para visitar a los nodos visitados. En muchas aplicaciones reales de manufactura, no es necesario realizar todas las tareas o visitar todas las ubicaciones, sino que algunas tareas pueden generar mayores beneficios y otras pueden ser menos prioritarias. El TSPP surgió como una respuesta a este tipo de problemas, donde no es solo importante encontrar una ruta óptima, sino también balancear entre distancia, tiempo o coste del recorrido (según las unidades) y las ganancias obtenidas al visitar ciertos nodos.

Para abordar este problema biobjetivo, los autores proponen una metodología exacta basada en el método ϵ-constraint, que se centra en encontrar un conjunto de soluciones eficientes (o de Pareto) para los dos objetivos conflictivos: maximizar los beneficios obtenidos al visitar los nodos y minimizar los costes que se incurren al realizar el recorrido para visitar a los nodos visitados. El método ϵ-constraint es una técnica muy utilizada en la optimización multiobjetivo. Su principio consiste en optimizar uno de los objetivos mientras se impone una cota (ϵ) en el segundo objetivo. Luego, se repite el proceso variando el valor de ϵ para obtener una frontera eficiente de soluciones.

Los pasos del algoritmo propuesto incluyen:

a) Definir el problema biobjetivo: En este caso, se trata de maximizar el beneficio total (al visitar un subconjunto de nodos) y minimizar los costes que se incurren al realizar el recorrido para visitar a los nodos visitados.

b) Aplicar el método ϵ-constraint:

   - Se optimiza uno de los objetivos (en este caso, la minimización de los costes de recorrido).
   - Se impone una cota en el segundo objetivo (el beneficio) y se genera un conjunto de soluciones donde el beneficio no es menor que un valor ϵ dado.
   - Se ajusta ϵ en varias iteraciones para obtener diferentes soluciones en la frontera eficiente, lo que permite obtener un conjunto de soluciones entre las cuales el decisor puede elegir según sus prioridades.

c) Búsqueda de la solución óptima: Se usa un enfoque exacto (en lugar de heurístico) para garantizar que las soluciones obtenidas sean las óptimas dentro de las restricciones impuestas.

Los resultados obtenidos por los autores fueron prometedores. La metodología propuesta fue capaz de generar soluciones de alta calidad para el TSPP, incluso en instancias grandes del problema. Los autores lograron encontrar soluciones eficientes que maximizaban las ganancias de visitar un subcon-

junto de nodos mientras mantenían los costes de recorrido dentro de límites razonables.

En términos de rendimiento, el método ϵ-constraint permitió explorar un gran número de posibles soluciones y obtener una frontera de Pareto que proporcionaba múltiples opciones para la toma de decisiones. Esto es especialmente útil en contextos de manufactura y producción, donde los gerentes pueden necesitar soluciones flexibles para adaptarse a diferentes limitaciones de tiempo, recursos o costes.

Por ejemplo, en una línea de producción donde diferentes estaciones generan distintos niveles de valor o rentabilidad, la resolución del TSPP utilizando el método ϵ-constraint permitió determinar qué estaciones o tareas eran más rentables de completar en función de los costes de transición entre estaciones.

El trabajo de Bérubé, Gendreau y Potvin muestra cómo el TSPP puede aplicarse eficazmente en el ámbito de la manufactura y producción, donde se necesita un equilibrio entre la maximización de beneficios y la minimización de costes. El uso del método ϵ-constraint fue clave para generar soluciones exactas y eficientes, proporcionando una herramienta valiosa para la optimización multiobjetivo en contextos industriales.

Además, este estudio demuestra que, aunque los métodos exactos pueden ser costosos en términos computacionales, ofrecen un alto valor cuando se trata de encontrar soluciones de calidad en problemas complejos, como la secuenciación de operaciones en fábricas o la optimización de rutas en procesos de producción.

4. **Robótica y Drones**

En el campo de Robótica y Drones, el TSP es altamente relevante para optimizar las rutas que deben seguir robots móviles o drones autónomos. Estos vehículos necesitan moverse a través de múltiples puntos de interés o destinos para realizar tareas como inspección, recolección de datos o entrega de paquetes. El objetivo es minimizar el tiempo o la distancia total recorrida mientras se cumple con todas las tareas requeridas.

El trabajo de Savla, Frazzoli y Bullo [57] aborda una variación del TSP, aplicando el concepto a vehículos con restricciones de movimiento, conocidos como vehículos de Dubins. Esta investigación surgió en el contexto de la necesidad de planificar rutas óptimas para vehículos autónomos como drones y robots, donde las limitaciones de movimiento, como la incapacidad de realizar giros bruscos, juegan un papel crucial.

La idea del TSP para vehículos de Dubins proviene de la creciente necesidad de planificar rutas para vehículos autónomos en aplicaciones como la vigilancia, la inspección de infraestructuras o la entrega de paquetes. A diferencia del TSP clásico, donde las conexiones entre puntos se pueden realizar en línea recta, los vehículos de Dubins tienen restricciones de curvatura, lo que significa

que no pueden cambiar de dirección instantáneamente ni seguir trayectorias rectas en la mayoría de los casos.

La planificación de rutas para estos vehículos requiere que se tomen en cuenta las restricciones geométricas asociadas a sus trayectorias, lo que añade complejidad al problema de optimización. De esta necesidad surge el problema de adaptar el TSP a las características específicas de estos vehículos.

Para abordar el problema del TSP en vehículos de Dubins, los autores exploran diferentes enfoques y algoritmos:

- Algoritmos basados en la Teoría de Control Óptimo: Los autores hacen uso de la teoría de control para modelar las trayectorias de los vehículos de Dubins, empleando ecuaciones de movimiento que restringen el giro y la curvatura. Estos modelos aseguran que las trayectorias propuestas sean físicamente viables para los vehículos.
- Algoritmos de Aproximación: Dado que el TSP con restricciones de curvatura es un problema $\mathcal{NP}$-Difícil, los autores proponen algoritmos de aproximación que buscan soluciones cercanas al óptimo sin requerir tiempo de cálculo excesivo. Estos algoritmos no siempre encuentran la mejor solución, pero proporcionan rutas de alta calidad en tiempos computacionalmente manejables.
- Heurísticas basadas en el TSP clásico: Se utilizan heurísticas tradicionales del TSP clásico, como el Algoritmo del Vecino Más Cercano (*Nearest NeighBour*, NN) y el Algoritmo de Búsqueda Local, que son adaptadas para respetar las restricciones de curvatura. Estas heurísticas mejoran soluciones tentativas al ajustar las rutas para minimizar el tiempo o la distancia recorrida.
- Descomposición del Problema: Otro enfoque utilizado es dividir el problema en subproblemas más manejables. Por ejemplo, se puede dividir la ruta global en segmentos más pequeños, resolviendo cada segmento de forma óptima y luego combinándolos. Este enfoque de descomposición es útil cuando el número de puntos a visitar es elevado.

Los resultados del trabajo de Savla y sus colegas demostraron que es posible generar rutas óptimas o cercanas al óptimo para vehículos de Dubins en escenarios prácticos. Los algoritmos desarrollados lograron reducir significativamente el tiempo y la distancia recorrida en comparación con enfoques más simples que no tomaban en cuenta las restricciones de curvatura.

Algunos resultados clave incluyen:

- Reducción del tiempo de cálculo: Los algoritmos de aproximación propuestos encontraron soluciones en tiempos mucho menores que los algoritmos exactos, lo que es crucial para aplicaciones en tiempo real, como el control de drones y robots autónomos.
- Mejora en la eficiencia de la ruta: Aunque los algoritmos no siempre alcanzaron la solución óptima, las rutas obtenidas fueron muy cercanas al

óptimo, lo que demostró que la aproximación es adecuada para su uso en la práctica.

- Aplicabilidad a diferentes escenarios: Los algoritmos desarrollados fueron flexibles y aplicables a una variedad de escenarios de robótica y drones, desde la vigilancia aérea hasta la inspección de infraestructuras.

El artículo de Savla, Frazzoli y Bullo [57] proporciona una contribución importante al campo de la robótica y la optimización, adaptando el TSP para vehículos con restricciones geométricas como los vehículos de Dubins. Los algoritmos desarrollados permiten encontrar soluciones eficientes para problemas de planificación de rutas, lo que es crucial para la robótica autónoma en aplicaciones del mundo real.

5. **Telecomunicaciones**

El Problema del Viajante tiene varias aplicaciones en el campo de las telecomunicaciones. Un ejemplo notable es la optimización de rutas en redes de telecomunicaciones para minimizar el coste y el tiempo de transmisión de datos.

Otro enfoque es el de Alemayehu y Kim [58] hacen. La idea surgió de la necesidad de mejorar la eficiencia en la recolección de datos en redes de sensores inalámbricos (*Wireless Sensor Networks*, WSN) utilizando vehículos aéreos no tripulados (*Unmanned Aerial Vehicle*, UAV). El uso de UAV como relés en WSN puede reducir significativamente el consumo de energía de los nodos sensores, ya que los UAV reemplazan la comunicación multi-salto entre nodos. Sin embargo, esto conlleva un aumento en la latencia de entrega de datos. La latencia en este contexto se refiere al tiempo total que toma un dato desde que se origina hasta que llega a su destino. Para abordar este problema, los autores propusieron modificar el algoritmo heurístico del NN para reducir la latencia de entrega de datos sin un cambio significativo en el tiempo computacional.

El NN es una técnica simple y eficiente para aproximar soluciones al TSP. El algoritmo NN sigue estos pasos:

a) Inicio: Selecciona un nodo inicial arbitrariamente.

b) Selección del vecino más cercano: Desde el nodo actual, selecciona el nodo más cercano que aún no haya sido visitado.

c) Visita del nodo: Mueve al nodo seleccionado y lo marca como visitado.

d) Repetición: Repite los pasos 2 y 3 hasta que todos los nodos hayan sido visitados.

e) Retorno al nodo inicial: Una vez que todos los nodos han sido visitados, regresa al nodo inicial para completar el ciclo.

Volviendo al trabajo de Alemayehu y Kim, ellos propusieron tres algoritmos eficientes que modifican el algoritmo NN existente:

a) *Prioritized Nearest Neighbor*: Este algoritmo PNN modifica el NN tradicional al priorizar ciertos nodos en función de criterios específicos, como la importancia de los datos que contienen o su ubicación estratégica en la red. La idea es reducir la latencia de adquisición de datos al visitar primero los nodos más críticos.

b) *Dynamic Nearest Neighbor*: El algoritmo DNN introduce una dinámica en la selección del siguiente nodo a visitar. En lugar de seguir una ruta predefinida, el UAV ajusta su trayectoria en tiempo real en función de la disponibilidad de datos y las condiciones de la red. Esto permite una mayor flexibilidad y puede reducir la latencia en escenarios donde la red es altamente variable.

c) *Double Dynamic Nearest Neighbor*: El DDNN es una extensión del DNN que incorpora una doble dinámica. No solo ajusta la trayectoria en tiempo real, sino que también considera la posibilidad de reprogramar visitas a nodos previamente visitados si se detecta que hay nuevos datos disponibles o si las condiciones de la red han cambiado significativamente. Esto maximiza la eficiencia en la recolección de datos y minimiza la latencia.

Los resultados mostraron que los algoritmos propuestos (PNN, DNN y DDNN) superaron al algoritmo NN original en la reducción de la latencia de entrega de datos, especialmente en nodos densamente desplegados y con rangos de transmisión relativamente grandes. Los algoritmos lograron una reducción significativa en la longitud del recorrido, lo que permitió una entrega de datos más rápida sin un aumento considerable en el tiempo computacional.

6. **Gestión de Proyectos**

En el campo de la Gestión de Proyectos, el TSP puede utilizarse para optimizar la planificación y programación de tareas que requieren la visita a varios sitios o la realización de múltiples actividades en secuencia. Un ejemplo es la optimización del recorrido de un UAV que debe visitar varios puntos de un proyecto, como obras de construcción, para realizar inspecciones o entregas, con el objetivo de minimizar el tiempo total de operación o el coste asociado al combustible y mano de obra.

Sundar y Rathinam [59] presentan un enfoque para optimizar el enrutamiento de UAVs en escenarios donde es necesario recargar combustible en depósitos intermedios. La idea principal del estudio surgió a partir de la necesidad de mejorar la eficiencia en la operación de UAVs en tareas de vigilancia, inspección o entrega, donde las limitaciones de combustible son un factor clave que afecta el rendimiento de las misiones.

Con la expansión del uso de UAVs en aplicaciones civiles y militares, una de las limitaciones más importantes es la capacidad de los vehículos de completar largas misiones sin interrumpir la operación para recargar combustible. En este contexto, el problema de que el UAV visite todos los puntos de interés en el menor tiempo posible, considerando paradas en depósitos de recarga, puede

modelarse como una extensión del TSP. En lugar de simplemente optimizar el recorrido entre puntos, el problema también involucra decisiones sobre cuándo y dónde recargar, lo que lo convierte en un problema más complejo.

El artículo utiliza y propone varios algoritmos para abordar este problema de enrutamiento, combinando enfoques heurísticos y metaheurísticos para encontrar soluciones eficientes:

- Programación dinámica: Esta técnica se usa para modelar el proceso de toma de decisiones sobre cuándo y dónde realizar las paradas de recarga de manera óptima, minimizando el coste total de la misión.
- Descomposición del problema.
- Algoritmos genéticos.

Los experimentos realizados mostraron que los algoritmos propuestos fueron capaces de generar buenas soluciones en escenarios realistas, con mejoras significativas en el tiempo de misión y el coste asociado al combustible en comparación con enfoques más tradicionales. En particular, los UAVs pudieron visitar todos los puntos de interés minimizando las paradas innecesarias en los depósitos de recarga, lo que se tradujo en operaciones más rápidas y económicas.

El estudio concluye que este enfoque es especialmente útil en aplicaciones donde los UAVs necesitan realizar tareas prolongadas con restricciones de combustible, y que estos algoritmos pueden ser adaptados a una variedad de escenarios logísticos, tanto en la gestión de proyectos como en la planificación de rutas de vehículos autónomos.

7. **Turismo y Viajes**

En el campo de turismo y viajes, el TSP se puede aplicar para optimizar itinerarios turísticos o planificar rutas de viaje que minimicen la distancia recorrida o el tiempo de desplazamiento entre múltiples destinos. Esto es especialmente útil en la planificación de tours donde el objetivo es visitar varias ciudades o atracciones turísticas de manera eficiente.

Un ejemplo destacado es el estudio de Vansteenwegen et al. [60]. Se enfoca en una variante del TSP llamada Orienteering Problem (OP). La idea principal de este trabajo aparece dada la necesidad de encontrar rutas para turistas o viajeros con limitaciones de tiempo o recursos. Mientras que el TSP clásico se enfoca en encontrar la ruta más corta para visitar una serie de destinos, el OP añade restricciones de tiempo o presupuesto, lo que refleja de manera más realista los desafíos en la planificación de viajes turísticos.

El OP se inspiró en el deporte de orientación, donde los participantes deben encontrar la mejor ruta para visitar una serie de puntos de control dentro de un tiempo limitado. En el contexto del turismo, esta problemática se traduce en la optimización de itinerarios que permitan a los turistas visitar tantos lugares

de interés como sea posible en el tiempo disponible, maximizando el valor o satisfacción obtenida de cada visita.

El artículo de Vansteenwegen y su equipo surge como una respuesta a la creciente demanda por algoritmos que puedan resolver problemas de optimización en tiempo real, aplicados no solo al turismo, sino también a otros campos como la logística y el diseño de rutas comerciales.

El estudio revisa y analiza varios tipos de algoritmos que se han utilizado para resolver el OP. Entre los principales enfoques mencionados se encuentran:

- Algoritmos exactos: Aunque estos algoritmos garantizan encontrar la solución óptima, su complejidad hace que no sean prácticos para instancias grandes del problema. Entre los métodos exactos revisados se encuentran la programación lineal entera mixta y la programación dinámica.
- Algoritmos heurísticos: Estos algoritmos ofrecen soluciones aproximadas de buena calidad en un tiempo razonable. Vansteenwegen et al. destacaron la búsqueda local y la búsqueda de vecindad variable (VNS) como heurísticas eficaces para resolver el OP en instancias más grandes y complejas.
- Metaheurísticas: Algoritmos como el Recocido Simulado y los algoritmos genéticos también fueron explorados en el artículo. Estas técnicas utilizan principios inspirados en procesos naturales o físicos para mejorar iteratívamente las soluciones, explorando el espacio de búsqueda de manera más eficiente.

El análisis mostró que los algoritmos heurísticos y metaheurísticos ofrecieron un buen equilibrio entre calidad de la solución y tiempo de cómputo, lo que los hace ideales para aplicaciones en tiempo real en la industria del turismo. En particular, el uso de la búsqueda de vecindad variable y el recocido simulado proporcionaron soluciones que estaban cerca del óptimo, con tiempos de cálculo considerablemente menores que los algoritmos exactos.

El artículo concluyó que la investigación futura debe enfocarse en desarrollar algoritmos que no solo optimicen las rutas, sino que también sean flexibles para adaptarse a cambios en las preferencias de los viajeros o restricciones dinámicas, como el tráfico o la disponibilidad de recursos.

8. **Biología Computacional**

En el campo de la Biología Computacional, el TSP se utiliza para resolver problemas de alineación y comparación de secuencias genéticas. Un ejemplo concreto es el problema de ensamblaje de fragmentos de ADN, donde se necesita reconstruir una secuencia genética a partir de pequeños fragmentos. Este problema puede modelarse como una instancia del TSP, donde cada fragmento de ADN es un “nodo” y la “distancia” entre dos fragmentos es la medida de superposición o similitud entre ellos. El objetivo es encontrar la secuencia que minimiza las “distancias”, es decir, que maximiza las coincidencias entre

fragmentos.

El artículo de Agarwala et al. [61] aborda un problema crítico en biología computacional: la comparación y ensamblaje de grandes secuencias genéticas. La motivación detrás de este trabajo surgió de la creciente necesidad de procesar grandes cantidades de datos genómicos, particularmente en el contexto del Proyecto Genoma Humano, donde se estaban secuenciando enormes fragmentos de ADN conocidos como BACs (*Bacterial Artificial Chromosomes*).

El desafío principal en la comparación de secuencias genéticas radica en la magnitud y complejidad de los datos. A medida que las tecnologías de secuenciación avanzaban, los métodos tradicionales de comparación y alineación de secuencias se volvían insuficientes debido a su ineficiencia y escalabilidad. El equipo de investigadores identificó que el proceso de ensamblaje de fragmentos de ADN podría ser abordado de manera similar al TSP, donde las secuencias se tratan como "nodos" y las "distancias" entre ellas representan el grado de superposición o similitud entre fragmentos.

La idea central fue desarrollar un algoritmo que permitiera realizar estas comparaciones de manera rápida y eficiente, incluso cuando se manejaban secuencias genéticas extremadamente grandes.

El enfoque desarrollado por Agarwala et al. es una combinación de algoritmos de búsqueda local y heurísticas basadas en grafos, lo que les permitió reducir drásticamente el tiempo de cálculo necesario para comparar y ensamblar secuencias genéticas. La base de su algoritmo consiste en los siguientes pasos clave:

- División de Secuencias en Fragmentos: Las secuencias de ADN se dividen en fragmentos más pequeños, lo que facilita la comparación de secuencias entre ellas.
- Construcción de Grafos de Similaridad: Se genera un grafo donde cada nodo representa un fragmento de secuencia y cada borde mide la "distancia" o el grado de superposición entre dos fragmentos. El objetivo es encontrar un camino a través de este grafo que minimice las distancias entre fragmentos, lo que se asemeja a la solución del TSP.
- Heurísticas del TSP: Aunque el problema del ensamblaje de secuencias no es idéntico al TSP, se pueden aplicar técnicas heurísticas del TSP para encontrar soluciones aproximadas al problema de ensamblaje. Utilizaron un enfoque basado en la búsqueda local mejorada con una serie de restricciones específicas del problema genético, que les permitió evitar la explosión combinatoria que usualmente ocurre en problemas $\mathcal{NP}$-Completos.

Los resultados fueron altamente prometedores. El método propuesto permitió comparar secuencias genéticas de gran tamaño de manera rápida y con un alto grado de precisión. En particular, el equipo aplicó su metodología a BACs solapados, secuencias utilizadas en el ensamblaje de grandes genomas, como el del Proyecto Genoma Humano. Los algoritmos demostraron ser capaces de

manejar secuencias mucho más grandes que los enfoques anteriores, y los tiempos de cálculo se redujeron considerablemente.

Los autores informaron que, con su método, se podían comparar secuencias de cientos de miles de pares de bases en cuestión de minutos, algo que antes habría llevado horas o incluso días con enfoques tradicionales.

La investigación de Agarwala et al. [61] proporcionó un enfoque revolucionario para la comparación de grandes secuencias genéticas, aprovechando las técnicas inspiradas en el TSP para abordar uno de los problemas clave en la biología computacional. Los algoritmos desarrollados permitieron ensamblar fragmentos de ADN con una eficiencia sin precedentes, contribuyendo al avance de la genómica y sentando las bases para futuros trabajos en el ensamblaje de secuencias.

9. **Mantenimiento y Reparación**

Un ejemplo del uso del TSP en el campo de mantenimiento y reparación se encuentra en la planificación de rutas para vehículos de servicio, como los que realizan reparaciones y mantenimiento de infraestructuras críticas o instalaciones industriales. Uno de los enfoques se aplica a la optimización de rutas para la recolección de residuos sólidos y servicios de mantenimiento en industrias como la alimentación, bebidas o la distribución de productos lácteos.

Golden, Assad y Wasil [62] abordan el uso del TSP y sus variantes en aplicaciones del mundo real, enfocadas principalmente en la logística de rutas de vehículos de recolección y distribución, así como en el mantenimiento de infraestructuras críticas.

La idea de aplicar el TSP a problemas de mantenimiento y distribución surgió debido a la necesidad de optimizar el uso de recursos en industrias donde los vehículos deben realizar múltiples paradas para recoger o entregar bienes, o realizar tareas de mantenimiento y reparación. Las rutas de estos vehículos implican costes operativos altos en términos de combustible, tiempo y personal, por lo que es fundamental minimizar las distancias recorridas y cumplir con las restricciones de tiempo y capacidad. El TSP y sus variantes proporcionan una forma de modelar y optimizar estos problemas.

En el trabajo de Golden, Assad y Wasil se emplean una variedad de enfoques para resolver las distintas versiones del TSP. Entre los algoritmos utilizados destacan:

- Heurísticas: Para obtener soluciones rápidas y cercanas a las óptimas, los autores utilizan heurísticas como el NN y el 2-opt. 2-opt es un algoritmo heurístico utilizado para mejorar soluciones iniciales del TSP basado en un procedimiento de optimización local iterativo. Estas técnicas proporcionan soluciones viables al reducir la longitud total de las rutas de manera incremental.
- Metaheurísticas: Se aplican metaheurísticas más avanzadas, como el

Recocido Simulado y algoritmos genéticos. Estas técnicas son útiles cuando las instancias del problema son grandes y complejas, lo que hace impracticable el uso de métodos exactos.

- Programación Entera Mixta (*Mixed Integer Programming*, MIP): En algunos casos, los autores utilizan MIP para formular el problema y obtener soluciones exactas. Sin embargo, debido a la complejidad computacional de estas técnicas en problemas de gran escala, los autores optan por métodos híbridos que combinan MIP con heurísticas para mejorar la eficiencia.

Los resultados mostraron que el uso de algoritmos basados en el TSP permitía reducir significativamente los costes de operación en las industrias analizadas. En el caso de la recolección de residuos sólidos, las rutas optimizadas lograron reducir las distancias recorridas en un 15 % en promedio, lo que representaba una reducción significativa en el consumo de combustible y los tiempos de operación. De manera similar, en la industria alimentaria y de bebidas, las rutas optimizadas permitieron a los vehículos cumplir con horarios de entrega más precisos, mejorando así la satisfacción del cliente.

10. **Servicios Públicos**

Un ejemplo de la aplicación del TSP en el campo de servicios públicos es la optimización de rutas para vehículos en tareas como el suministro de agua, electricidad, o la recolección de residuos sólidos. Este problema es relevante porque implica la visita a múltiples ubicaciones, minimizando el tiempo de recorrido y los costes asociados.

El artículo de Repoussis y Tarantilis [63] trata de resolver uno de los problemas más complejos en el ámbito de la logística y los servicios públicos: optimizar el tamaño y la composición de una flota de vehículos que debe operar bajo restricciones de tiempo y capacidad.

La idea surge de la necesidad de optimizar las operaciones de flotas de vehículos en aplicaciones del mundo real donde hay restricciones tanto de tiempo como de capacidad. En muchos contextos de servicios públicos, como la recolección de residuos o la distribución de recursos energéticos, las rutas no solo deben minimizarse en términos de distancia y costes, sino también cumplir con horarios estrictos para maximizar la eficiencia y reducir los tiempos de espera o interrupción del servicio. Este tipo de problema se conoce como el problema de rutas de vehículos con ventanas de tiempo (VRP-TW), una extensión del TSP.

El enfoque utilizado en este estudio es el de la programación de memoria adaptativa (*Adaptive Memory Programming*, AMP). Este es un método heurístico que combina técnicas de búsqueda local con memoria para almacenar soluciones parciales y mejorar iteratívamente las soluciones a lo largo del proceso. Algunas de las características clave de este algoritmo son:

- Memoria adaptativa: La memoria permite que el sistema recuerde solu-

ciones de alta calidad encontradas durante el proceso de búsqueda y las reutilice en iteraciones posteriores.

- Enfoque iterativo: El algoritmo opera de manera iterativa, donde las soluciones parciales se generan, mejoran y almacenan. Luego, estas soluciones se combinan para formar rutas más optimizadas.
- Búsqueda local: Se emplean algoritmos de búsqueda local para explorar las soluciones vecinas a las actuales y realizar intercambios de rutas entre vehículos con el objetivo de mejorar las soluciones de manera incremental.

El algoritmo AMP se combina con técnicas tradicionales del TSP, como la heurís- tica del NN, para obtener soluciones iniciales que luego se refinan. También se utilizan metaheurísticas avanzadas como el Recocido Simulado y Búsqueda Tabú para explorar de manera más eficiente el espacio de soluciones y evitar caer en óptimos locales.

Los resultados obtenidos en el estudio muestran que el algoritmo AMP es muy efectivo para resolver problemas de gran escala en el contexto de servicios públicos. En términos de reducción de costes, el uso de AMP permitió reducir significativamente tanto el tamaño de la flota de vehículos como el tiempo total de operación. Para problemas que involucran cientos de ubicaciones y múltiples vehículos, AMP demostró ser capaz de encontrar soluciones de alta calidad en tiempos de cómputo razonables.

En resumen, la aplicación del TSP en la planificación de rutas de vehículos bajo restricciones de tiempo y capacidad, en combinación con técnicas avanzadas como AMP, ha mostrado un alto potencial para mejorar la eficiencia operativa en los servicios públicos, reduciendo costes y mejorando el cumplimiento de ventanas de tiempo críticas.

2.1.3. Relación del Mo-TSRPP con otros problemas de optimización

El Mo-TSRPP está relacionado con otros problemas de optimización, de tal forma que las restricciones de unos afectan a las de otros. A continuación, se examinan las relaciones más importantes, mencionando los distintos problemas por su nombre en inglés, que es como se les conoce comúnmente.

Directed Hamiltonian Circuit Problem

El problema del Circuito Hamiltoniano Dirigido (*Directed Hamiltonian Circuit Problem*, DHCP) es un ciclo en un grafo dirigido que visita cada nodo exactamente una vez y regresa al nodo inicial [64]. A diferencia del TSP, el objetivo sólo es recorrer todos los nodos, y no se minimiza la distancia.

El problema del circuito hamiltoniano dirigido tiene aplicaciones en diversas áreas, como la planificación de rutas, la optimización de redes y la bioinformática. Por ejemplo, en la optimización de redes, se puede utilizar para encontrar rutas eficientes en redes de comunicación o transporte.

Chinese Postman Problem

El problema del Cartero Chino (*Chinese Postman Problem*, CPP) o problema de inspección de rutas (*Route Inspection Problem*, RIP), es un problema de optimización en teoría de grafos. El objetivo es encontrar el camino más corto que visite cada arista de un grafo al menos una vez y regrese al punto de partida [65], a diferencia del TSP, que hay que pasar por cada nodo sólo una vez.

El CPP se puede aplicar tanto a grafos dirigidos como no dirigidos. Si el grafo tiene un circuito euleriano (un ciclo que visita cada arista exactamente una vez), ese circuito es la solución óptima. Si no tiene un circuito euleriano, el problema se convierte en encontrar el menor número de aristas adicionales que se deben duplicar para que el grafo resultante tenga un circuito euleriano.

El CPP tiene aplicaciones prácticas en la planificación de rutas para servicios de entrega, mantenimiento de redes, y cualquier situación donde se necesite recorrer todas las conexiones de una red de la manera más eficiente posible.

Vehicle Routing Problem

El problema de enrutamiento de vehículos (*Vehicle Routing Problem*, VRP) encuentra las rutas óptimas para una flota de vehículos que deben entregar paquetes a un conjunto de clientes. [66]

El VRP generaliza el TSP. En lugar de un solo vendedor, se tienen múltiples vehículos que deben visitar un conjunto de nodos. El objetivo es minimizar el coste total de las rutas, que puede incluir la distancia total recorrida, el tiempo de viaje, o el coste monetario. Los vehículos tienen restricción de capacidad, lo que aumenta la complejidad del problema.

El VRP tiene aplicaciones prácticas en la logística y la planificación de rutas de entrega, donde se necesita optimizar las rutas para reducir costes y mejorar la eficiencia. Es utilizado por empresas de transporte, servicios de entrega, y en la gestión de flotas.

2.1.4. Variantes del problema TSP

Ateniendo a [18], el modelo tradicional del TSP ha sido modificado por varios investigadores para aplicarlo eficazmente a muchos problemas de la vida real. Existen dos variantes básicas del TSP: el TSP simétrico (sTSP) y el TSP asimétrico (aTSP). Estas variantes básicas de TSP representan a los tipos de grafo no dirigido y dirigido. Se pueden describir como sigue:

- sTSP: El TSP simétrico (*Symmetric* TSP, sTSP) o (*Euclidean* TSP), c_{ij} es la distancia euclídea entre los nodos i y j. Si $c_{ij} = c_{ji}$, el recorrido puede evaluarse en ambas direcciones con el mismo coste [67]. El sTSP es el problema de encontrar un recorrido cerrado de longitud mínima que visite cada nodo una vez.
- aTSP: Si $c_{ij} \neq c_{ji}$ para al menos una arista(i, j), entonces el TSP se convierte en un aTSP (*Asymmetric* TSP) [67].

Del mismo modo, no siempre es más rápido viajar por el enlace directo del nodo i al nodo k, a veces puede ser más rápido viajar a través del nodo j. Por lo tanto, debemos permitir que no se aplique la desigualdad triangular.

Aparte de las variantes básicas de TSP, se han segregado otras categorías de variantes de TSP más específicas de las restricciones de aplicación. Se han identificado 6 categorías: centradas en los nodos, en las aristas, con dependencias dinámicas o temporales, con múltiples rutas o agentes, con incertidumbre o riesgo e híbridas o con elementos adicionales. Las diferentes variantes del TSP pertenecientes a cada una de estas categorías se discuten en las siguientes secciones.

Variantes centradas en los nodos

Estos problemas afectan principalmente a la selección, el valor o las características de los nodos.

- **TSP con beneficios**

  Dominique et al. [68] propusieron los TSP con Beneficios por sus siglas en inglés TSPP (*Traveling Salesman Problem with Profit*), son una simplificación del TSP tradicional, donde no es necesario visitar todos los nodos. A cada nodo se le asocia un beneficio predefinido. El objetivo del problema es encontrar una ruta con un beneficio recogido satisfactorio (maximizado) y un coste de viaje (minimizado). [69, 70]

  Las restricciones serían visitar cada nodo como máximo una vez, empezar y terminar en el mismo punto, eliminación de subtours y coste máximo.

  El TSPP es relevante en situaciones donde los recursos son limitados y se debe maximizar el beneficio, como en la logística y la planificación de rutas de ventas. Este problema ayuda a optimizar las rutas no solo en términos de distancia, sino también en términos de beneficios recogidos.

- **TSP selectivo**

  En el TSP Selectivo, por sus siglas en inglés STSP (*Selective Traveling Salesman Problem*), también conocido como problema de la orientación, (en inglés OP, *Orienteering Problem*), o problema de recogida máxima (en inglés MCP, *Maximum Collection Problem*), el objetivo es encontrar una ruta que maximice

el beneficio obtenido y que los costes de viaje no superen un valor predeterminado. Por lo tanto, el objetivo del STSP es encontrar una ruta óptima que incluya el depósito y que maximice el beneficio recogido, con unos costes de viaje dentro de un límite máximo dado.

Las restricciones serían visitar cada nodo como máximo una vez, empezar y terminar en el mismo punto, eliminación de subtours y coste máximo.

El problema se ha aplicado para un problema de rutas de inventario [71] o en un contexto de competición de orientación [72].

- **TSP con Cuotas**

  El TSP con Cuotas, por sus siglas en inglés QTSP (*Quota Traveling Salesman Problem*), trata de encontrar un camino que minimice los costes totales de viaje y cuyo beneficio recogido no sea mínimo que un valor preestablecido. El TSP con Cuotas fue introducido por Awerbuch et al. [73] utilizando un grafo no dirigido y se ha introducido un valor preestablecido mínimo. La intención de la TSP de cuota puede ejemplificarse como la identificación de un recorrido óptimo que incluya el depósito que minimice los costes de viaje y cuyo beneficio recogido nunca sea menor que el mínimo.

  Las restricciones serían visitar cada nodo como máximo una vez, empezar y terminar en el mismo punto, eliminación de subtours, coste máximo y beneficio por encima de un mínimo establecido.

  Awerbuch et al. [73] introdujeron el problema TSP con Cuotas sin relación con una aplicación práctica, pero propusieron el siguiente escenario. Un vendedor debe vender una cuota p_{min} de cepillos. El vendedor posee un mapa con las localizaciones de las ciudades, las distancias entre ellas y sabe cuántos cepillos podría vender en cada ciudad. Y su objetivo es viajar a lo largo de un recorrido minimizando el coste del viaje y vendiendo la cuota requerida de cepillos.

- **Problema del Recorrido Rentable**

  El Problema del Recorrido Rentable, por sus siglas en inglés PTP (*Profitable Tour Problem*), o Problema del Ciclo Simple (en inglés SCP, *Simple Cycle Problem*), combina las funciones objetivo del STSP y del *Quota* TSP. Para el PTP, puede considerarse que consiste en minimizar los costes del viaje menos el beneficio. Su función objetivo podría escribirse de forma más natural como un beneficio global maximizado menos los costes totales de viaje.

  El PTP puede expresarse para el caso no dirigido de forma que los valores de beneficio recogidos en los nodos corresponden característicamente a los valores duales asociados a las restricciones de cobertura de los nodos. En la mayoría de los casos, puede resolverse como un problema del camino más corto óptimo entre dos copias del depósito [74]. Volgenant et al. [75] explicaron el método para resolver un PTP, bajo la identidad del TSP Generalizado, en inglés *Generalized* TSP, a costa de resolver un TSP Asimétrico sobre un grafo

transformado que tiene $2n+1$ nodos. Este esquema de transformación permite resolver el PTP con todos los procedimientos de solución dedicados al TSP Asimétrico. Por otra parte, el caso dirigido se asocia una variable binaria a cada arista dirigida, igual a 1 si y sólo si la arista correspondiente se utiliza en el recorrido óptimo, y otra variable binaria a cada nodo, igual a 1 si y sólo si se visita el nodo relacionado.

Por lo tanto, el objetivo del PTP es encontrar un recorrido simple que incluya el depósito en el grafo y minimice los costes de viaje menos el beneficio.

- **Problema del Comprador Itinerante**

  El Problema del Comprador Itinerante, por sus siglas en inglés TPP (*Traveling Purchaser Problem*), puede describirse como sigue. Consideremos un conjunto de productos y un conjunto de mercados, y cada mercado recibe una cantidad limitada de cada producto a un precio determinado. El TPP consiste en seleccionar un subconjunto de mercados de forma que se pueda comprar una demanda dada de cada producto, minimizando el coste de desplazamiento y el coste de compra. Un vecino de una solución TPP dada es otra solución TPP obtenida eliminando una ruta de mercados consecutivos, e insertando otros mercados para restablecer la viabilidad. [76, 77]

  El comprador parte de cualquier ciudad, y viaja a un subconjunto de las ciudades y compra cada uno de los productos en las ciudades visitadas, y regresa a la ciudad de partida. Cada producto está asociado a una demanda y está disponible (es decir, con una oferta positiva) en un subconjunto de mercados. No obstante, en cada mercado sólo pueden adquirirse una cantidad de unidades del producto.

  El objetivo es encontrar un recorrido para el comprador que minimice las sumas de los costes de viaje y de compra. Este problema encaja en varias aplicaciones, principalmente en contextos de enrutamiento y programación, y es $\mathcal{NP}$-Difícil en sentido fuerte.

- **TSP con descuentos**

  En el TSP con descuentos, o con recompensas descontadas, por sus siglas en inglés DRTSP (*Discounted-Reward Traveling Salesman Problem*), además de la distancia, se considera un factor de descuento aplicado a las recompensas obtenidas en cada nodo visitado. Esto significa que las recompensas obtenidas en etapas posteriores del recorrido tienen menos valor que las obtenidas al principio. El objetivo es maximizar la suma total de las recompensas descontadas a lo largo de la ruta. [78]

  Este problema es relevante en situaciones donde el valor de las recompensas disminuye con el tiempo o con la distancia recorrida, y se busca optimizar no solo la distancia, sino también el valor acumulado de las recompensas.

- **TSP con recogida de premios**

  El TSP con recogida de premios, por sus siglas en inglés PCTSP (*Prize-Collecting Traveling Salesman Problem*), consiste en minimizar los costes de viaje y las penalizaciones por los nodos que no se visitan, y visitar suficientes nodos para recoger una cantidad de dinero en premios previamente descrita. [79]

- **TSP de atracción**

  El TSP de atracción, por sus siglas en inglés AtTSP (*Attractive Traveling Salesman Problem*), es una variante TSP que incorpora un modelo de atracción para maximizar el beneficio total de una ruta. En el AtTSP, el conjunto de nodos se divide en nodos de instalaciones y nodos de clientes. El objetivo es construir una ruta de máximo beneficio sobre un subconjunto de los nodos de instalaciones. [80]

  En el AtTSP, cada nodo de instalación atrae una porción del beneficio de los nodos de clientes en función de la distancia entre ellos y la "atracción" de la instalación. El beneficio se calcula utilizando una función de atracción, que puede basarse en modelos de gravedad o de decaimiento exponencial.

  El AtTSP tiene aplicaciones en la planificación de rutas para instalaciones móviles, como los circos o unidades de salud móviles, donde se busca maximizar el beneficio al atraer a la mayor cantidad de clientes posible. También se aplica en la planificación de rutas de vehículos de reconocimiento militar, donde se busca maximizar la información recopilada.

- **TSP con ventanas de tiempo**

  El TSP con ventanas de tiempo, por sus siglas en inglés TSPTW (*Traveling Salesman Problem with Time Windows*), es una variante del TSP que añade restricciones de tiempo a cada nodo que debe ser visitada. En el TSPTW, cada nodo tiene una ventana de tiempo específica durante la cual debe ser visitada, lo que añade una capa adicional de complejidad al problema. [81]

  En el TSPTW, el objetivo es encontrar la ruta de coste mínimo que visite todos los nodos exactamente una vez y regrese al punto de partida, respetando las ventanas de tiempo asignadas a cada nodo. Si el vendedor llega a un nodo antes de que comience su ventana de tiempo, debe esperar hasta que la ventana se abra.

  El TSPTW tiene aplicaciones prácticas en la logística y la planificación de rutas, donde las entregas deben realizarse dentro de ventanas de tiempo específicas. Es relevante en la distribución de bienes, la planificación de rutas de vehículos y la programación de tareas en sistemas de manufactura.

- **TSP con costes de espera**

El TSP con costes de espera, por sus siglas en inglés WT-TSP (*Waiting Time Traveling Salesman Problem*), es una variante del TSP en la que se consideran los costes asociados al tiempo de espera en cada parada. En este problema, además de encontrar la ruta más corta, se debe minimizar el tiempo de espera en cada punto de la ruta, lo que añade una capa adicional de complejidad. [82]

El WT-TSP es relevante en situaciones donde las entregas deben realizarse dentro de ventanas de tiempo específicas, como en la logística y la planificación de rutas de vehículos de reparto. Este problema es común en la industria de la distribución y el transporte, donde la eficiencia y la puntualidad son cruciales.

- **TSP con Restricciones de Precedencia**

  El TSP con Restricciones de Precedencia, por sus siglas en inglés PC-TSP (*Precedence-Constrained Traveling Salesman Problem*), es una variante del TSP en la que se deben cumplir ciertas restricciones de precedencia al visitar los puntos. Esto significa que algunos puntos deben ser visitados antes que otros, lo que añade una capa adicional de complejidad al problema, ya que no solo se debe encontrar la ruta más corta, sino también respetar el orden de visita de ciertos puntos. [83]

  El PC-TSP es relevante en situaciones donde ciertas tareas deben realizarse en un orden específico, como en la planificación de rutas de mantenimiento, la programación de tareas en la manufactura, y la logística de entregas.

- **TSP con k entregas**

  El TSP con k entregas, por sus siglas en inglés k-dTSP (*k-Delivery Traveling Salesman Problem*), es una variante del TSP que se centra en la entrega de artículos desde puntos de recogida a puntos de entrega, utilizando un vehículo con capacidad limitada. En el k-dTSP, se tiene un grafo ponderado donde cada nodo representa un punto que puede ser un punto de recogida (donde se recoge un artículo) o un punto de entrega (donde se entrega un artículo). El vehículo tiene una capacidad de k, lo que significa que puede transportar como máximo k artículos a la vez. El objetivo es encontrar la ruta más corta que permita recoger y entregar todos los artículos, respetando la capacidad del vehículo. [84]

  El k-dTSP es relevante en la logística y la planificación de rutas de vehículos de reparto, especialmente en situaciones donde la capacidad del vehículo es limitada y se deben realizar múltiples entregas y recogidas. Este problema ayuda a optimizar las rutas para minimizar los costes de transporte y mejorar la eficiencia.

- **TSP con recogida y entrega**

El TSP con recogida y entrega, por sus siglas en inglés TSPPD (*Traveling Salesman Problem with Pickup and Delivery*), además de visitar nodos, el viajante debe recoger y entregar objetos en nodos específicos. El objetivo es minimizar el coste del viaje total, teniendo en cuenta restricciones de capacidad y secuencia de recolección y entrega. [85]

Este problema es relevante en la logística y distribución, donde un vehículo debe recoger mercancías de varios puntos y entregarlas en otros puntos, asegurando que las recogidas se realicen antes que las entregas correspondientes.

Variantes centradas en las aristas

Estos problemas se enfocan en la estructura, costes o restricciones de las aristas que conectan los nodos.

- **TSP generalizado simétrico**

  El TSP generalizado simétrico, por sus siglas en inglés GTSP (*Generalized Traveling Salesman Problem*) es una variante común del TSP simétrico clásico en el que los nodos se dividen en clusters y el vendedor tiene que visitar al menos un nodo por cada cluster. El problema implica las dos decisiones, elegir un subconjunto de nodos, y encontrar un recorrido hamiltoniano de coste mínimo en el subgrafo.

  Se puede crear una versión diferente del GTSP llamada E-GTSP cuando se impone la restricción adicional de que se debe visitar exactamente un nodo de cada cluster. GTSP y E-GTSP son modelos útiles para algunos problemas críticos que implican decisiones simultáneas de selección y secuenciación. Tiene una valiosa contribución sobre las aplicaciones prácticas en el diseño de redes en anillo, secuenciación de archivos informáticos, enrutamiento de clientes de asistencia social a través de agencias gubernamentales, selección y enrutamiento de aeropuertos para aviones de mensajería, programación flexible de fabricación y enrutamiento postal [86, 87].

- **TSP máximo**

  El TSP máximo, por sus siglas en inglés MAX TSP (*Maximum Traveling Salesman Problem*), es una variante del TSP en la que el objetivo es encontrar un ciclo hamiltoniano de peso máximo en un grafo ponderado. A diferencia del TSP clásico, donde se busca minimizar la distancia total recorrida, en el MAX TSP se busca maximizar el peso total de las aristas incluidas en el ciclo. [88]

  En el Max TSP, se tiene un grafo completo no dirigido con pesos no negativos en las aristas. El objetivo es encontrar un ciclo que visite cada nodo exactamente una vez y regrese al punto de partida, maximizando la suma de los pesos de las aristas en el ciclo.

El MAX TSP tiene aplicaciones en áreas donde se busca maximizar el beneficio o la utilidad de recorrer ciertos caminos, como en la planificación de rutas turísticas, la optimización de redes de comunicación y la logística de transporte. La situación de ejemplo que necesita de un *Maximum* TSP es el problema del "robo de taxis" (*taxi ripoff*), que es un famoso subconjunto de los problemas de enrutamiento de vehículos. Imagina que necesitas ir del punto A al punto B. El taxista podría tomar una ruta directa (A $\rightarrow$ B) o una ruta más larga (A $\rightarrow$ C $\rightarrow$ D $\rightarrow$ B) para aumentar la tarifa.

- **TSP con cuello de botella**

  El TSP con cuello de botella, por sus siglas en inglés BTSP (*Bottleneck Traveling Salesman Problem*), consiste en encontrar un ciclo hamiltoniano (un recorrido que visita cada nodo exactamente una vez) en un grafo ponderado que minimice el peso de la arista más pesada del ciclo. En otras palabras, el objetivo es minimizar el coste de la arista más cara en el recorrido. [89]

  El BTSP es relevante en situaciones donde es crucial minimizar el riesgo asociado con el tramo más costoso del recorrido, como en la planificación de rutas de emergencia o en la logística de transporte de mercancías valiosas.

- **TSP con restricciones de capacidad**

  El TSP con restricciones de capacidad, por sus siglas en inglés CTSP (*Capacitated Traveling Salesman Problem*), es una variante del TSP en la que se consideran limitaciones de capacidad para los vehículos o recursos utilizados. En este problema, cada viajante tiene una capacidad limitada para transportar bienes o servicios, y el objetivo es minimizar la distancia total recorrida mientras se respetan estas restricciones de capacidad. Cada nodo tiene una demanda, y el viajante tiene una capacidad máxima. [90]

  El CTSP es relevante en la planificación de rutas de vehículos de reparto, donde es crucial optimizar las rutas para minimizar los costes y cumplir con las restricciones de capacidad.

- **TSP de Steiner**

  El TSP de Steiner, por sus siglas en inglés STSP (*Steiner Traveling Salesman Problem*), es una variante del TSP en la que el viajante debe visitar ciertos puntos específicos (paradas obligatorias) antes de completar su recorrido. Esto añade una capa adicional de complejidad al problema, ya que no solo se debe encontrar la ruta más corta, sino también asegurarse de que se visiten todas las paradas obligatorias en el orden requerido. [91]

  El STSP es útil en situaciones donde ciertos nodos deben ser visitados obligatoriamente, como en la planificación de rutas de entrega con puntos de entrega

prioritarios o en la optimización de rutas de mantenimiento donde ciertos sitios requieren visitas regulares.

- **TSP con repostar**

  El TSP con repostar, por sus siglas en inglés TSPWR (*Traveling Salesman Problem with Refueling*), es una variante del TSP que incorpora la necesidad de repostar combustible durante el recorrido. En el TSPWR, el objetivo es encontrar la ruta más corta que visite todos los nodos exactamente una vez y regrese al punto de partida, teniendo en cuenta que el vehículo debe parar en estaciones de servicio para repostar cuando sea necesario. Esto añade una capa de complejidad, ya que el vendedor debe planificar paradas estratégicas para evitar quedarse sin combustible. [92]

  Este problema es relevante para la planificación de rutas de vehículos con capacidad limitada de combustible, como camiones de reparto y vehículos eléctricos.

- **Problema del vendedor encubridor**

  El Problema del vendedor encubridor, por sus siglas en inglés CSP (*Covering Salesman Problem*), es una generalización del TSP. En el CSP, el objetivo es encontrar un ciclo hamiltoniano de longitud mínima que cubra un grafo no dirigido, pero no es necesario visitar todos los nodos. En su lugar, cada nodo debe ser visitado o estar dentro de un radio de cobertura de un nodo visitado. [93]

  En el CSP, se tiene un conjunto de nodos y un radio de cobertura asociado a cada nodo. El objetivo es determinar una ruta que minimice la longitud total del ciclo, asegurando que cada nodo sea visitado o esté dentro del radio de cobertura de al menos un nodo visitado.

  El CSP tiene aplicaciones en diversas áreas, como la logística, la planificación de rutas de servicios de salud en áreas rurales, y la gestión de redes de telecomunicaciones. Por ejemplo, en la logística de última milla, se puede utilizar para optimizar las rutas de entrega asegurando que todos los clientes estén cubiertos por una estación de entrega cercana.

Variantes con dependencias dinámicas o temporales

Estos problemas incorporan cambios dinámicos, restricciones temporales o interacciones con el entorno.

- **Problema del Viajante dependiente del tiempo**

  El Problema del Viajante dependiente del tiempo, por sus siglas en inglés TD-TSP (*Time Dependant Traveling Salesman Problem*), es una variación del TSP

en el que el coste de viajar entre nodos depende del momento en que se realiza el viaje. Este problema es particularmente relevante en contextos donde las duraciones de los viajes varían a lo largo del día debido a factores como el tráfico [94]. También se puede enunciar de forma que el coste del viaje depende del orden en que se recorren los nodos. El objetivo es minimizar el coste total del recorrido, teniendo en cuenta que los costes varían con el tiempo.

- **TSP cinético**

  El TSP cinético, por sus siglas en inglés KTSP, (*Kinetic Traveling Salesman Problem*), es una versión del TSP que considera el movimiento de los nodos a lo largo del tiempo. En estas variantes, cada punto puede moverse con una velocidad constante en una dirección fija, lo que añade una capa de complejidad adicional al problema. El objetivo de la variante es visitar los puntos en movimiento con una distancia de recorrido mínima.[95, 96, 97]

  El KTSP es relevante en la planificación de rutas para vehículos autónomos en entornos dinámicos, donde los destinos pueden moverse, como en la logística de entrega de drones o en la navegación de robots móviles en áreas con objetivos móviles.

- **TSP de objetivo móvil**

  El TSP de objetivo móvil, por sus siglas en inglés MT-TSP (*Moving-Target Traveling Salesman Problem*), consiste en que un perseguidor tiene que atrapar en un tiempo mínimo un conjunto de puntos que se mueven en el plano con velocidad constante. Dado un conjunto de objetivos, cada objetivo moviéndose a velocidad constante desde una posición inicial, y dado un perseguidor que empieza en el origen y con una velocidad máxima, el objetivo es obtener la ruta más rápida empezando (y terminando) en el origen, que intercepte todos los objetivos. [96]

  El TSP del objetivo en movimiento puede generalizarse con el reabastecimiento para permitir más de un perseguidor que se mueva con la misma velocidad máxima. Este problema puede aplicarse directamente a la programación multiprocesador, donde el tiempo de procesamiento depende de la hora de inicio del procesamiento de un trabajo. El problema en cuestión es $\mathcal{NP}$-Difícil y la formulación del problema tiene un objetivo de tiempo total que es diferente de los objetivos típicos de "*makespan*" y mínima latencia. El makespan se refiere al período que transcurre desde que comienza el primer trabajo (momento de referencia 0) hasta que finaliza el último trabajo, abarcando así todo el tiempo necesario para completar todos los trabajos. Se consideran dos casos especiales de TSP de objetivo móvil, cuando los objetivos se limitan a una sola línea y cuando el número de objetivos móviles es pequeño.

  Considérese el MT-TSP en el que el perseguidor y todos los puntos se limitan a una sola línea. Para resolver la variante del objetivo móvil, primero se calcula el coste del recorrido que intercepta todos los puntos a la izquierda del punto

de partida y luego intercepta todos los puntos a la derecha del punto de partida. Posteriormente, calcule el coste del otro recorrido "natural" que intercepta todos los puntos a la derecha del lugar de partida y luego intercepta todos los puntos a la izquierda del lugar de partida. Finalmente, de este par de rutas posibles, se puede elegir la solución con el menor coste.

Cuando el número de objetivos móviles es pequeño, sólo se consideran algunos de los puntos que están en movimiento (mientras que la mayoría están estacionarios). Este tipo de variante tiene una aplicación crítica cuando un buque de aprovisionamiento reabastece a lanchas patrulleras, o cuando un avión debe interceptar una serie de unidades terrestres móviles.

- **TSP con Restricciones en la Velocidad**

  El TSP con Restricciones en la Velocidad, por sus siglas en inglés STSP (*Speed Traveling Salesman Problem*), añade la restricción de que el viajante (por ejemplo, un dron) debe cumplir con ciertos límites de velocidad en diferentes tramos de su ruta. Esto complica la planificación de la ruta óptima, ya que no solo se debe encontrar la ruta más corta que permita visitar todos los nodos una sola vez y regresar al punto de partida, sino que también se deben respetar las restricciones de velocidad en cada tramo. [98]

  El STSP es relevante en la planificación de rutas en áreas donde las restricciones de velocidad son significativas, como en zonas urbanas con límites de velocidad variables o en rutas de transporte de mercancías peligrosas. Este problema ayuda a optimizar las rutas no solo en términos de distancia, sino también en términos de tiempo y cumplimiento de regulaciones de tráfico.

Variantes con múltiples rutas o agentes

Estos problemas introducen múltiples vendedores, depósitos o interacción entre rutas.

- **Multiple Traveling Salesman Problem**

  En el Problema del Viajante Múltiple, por sus siglas en inglés mTSP (*Multiple Traveling Salesman Problem*), en lugar de un solo viajante, hay múltiples viajantes que deben cubrir los nodos. El objetivo sigue siendo minimizar la distancia total recorrida, pero ahora se deben considerar las rutas de varios viajantes. Este problema es más complejo que el TSP tradicional debido a la necesidad de coordinar las rutas de múltiples viajantes para asegurar que todos los nodos sean visitados sin solapamientos y de la manera más eficiente posible. [54]

  El mTSP es relevante en la planificación de rutas de vehículos, logística y distribución, donde múltiples vehículos deben visitar un conjunto de ubicaciones de manera eficiente.

- **TSP con Múltiples Depósitos**

  El TSP con Múltiples Depósitos, por sus siglas en inglés MDTSP (*Multiple Depot Traveling Salesman Problem*), consiste en que se tienen varios depósitos y un conjunto de clientes que deben ser visitados por varios vendedores. Cada vendedor comienza y termina su ruta en uno de los depósitos disponibles. El objetivo es minimizar el coste total de las rutas, asegurando que cada cliente sea visitado exactamente una vez por un solo vendedor. [99]

  El MDTSP tiene aplicaciones en diversas áreas, como la logística y la planificación de rutas para vehículos de reparto. Es especialmente útil en situaciones donde hay múltiples centros de distribución y se necesita optimizar las rutas de entrega para reducir costes y tiempos.

Variantes con incertidumbre o riesgo

Estos problemas incorporan incertidumbre o condiciones estocásticas en los parámetros del problema.

- **TSP probabilístico**

  En el TSP probabilístico, por sus siglas en inglés PTSP (*Probabilistic Traveling Salesman Problem*), consiste en que solo un subconjunto de nodos potenciales necesita ser visitado en cualquier instancia dada del problema. La cantidad de nodos a visitar cada vez es una variable aleatoria. El objetivo es encontrar un recorrido a priori que minimice la longitud esperada, con la estrategia de visitar los nodos presentes en una instancia particular en el mismo orden en que aparecen en el recorrido a priori. [100, 101]

  Las características del PTSP radican en que solo un subconjunto de los nodos necesita ser visitado, la cantidad de nodos a visitar es una variable aleatoria con una distribución de probabilidad conocida y en que se busca un recorrido que minimice la longitud esperada del recorrido total, considerando las probabilidades de visita de cada nodo.

  Este problema es crucial en aplicaciones prácticas como la logística y la planificación de rutas en situaciones inciertas.

Variantes híbridas o con elementos adicionales

Estos problemas combinan el TSP con otros problemas o incluyen restricciones adicionales específicas.

- **Problema del ladrón viajero**

El Problema del ladrón viajero, por sus siglas en inglés TTP (*Traveling Thief Problem*), es una combinación de dos problemas clásicos de optimización: el Problema del Viajante (TSP) y el Problema de la Mochila (KP). En el TTP, un ladrón debe planificar una ruta para visitar un conjunto de nodos (como en el TSP) y decidir qué objetos robar de cada nodo para maximizar el beneficio sin exceder la capacidad de su mochila (como en el KP). La complejidad del TTP radica en la interdependencia entre la ruta de viaje y la selección de objetos, ya que el peso de los objetos afecta la velocidad de viaje del ladrón. [102]

El TTP tiene varias aplicaciones prácticas en el mundo real, especialmente en problemas de logística y gestión de recursos, tales como planificación de rutas de entrega, el cual consiste en optimizar las rutas para vehículos de reparto que también deben recoger y entregar paquetes, la gestión de inventarios móviles en el que se toman decisiones sobre qué productos llevar en vehículos de ventas móviles para maximizar las ganancias, y la recolección de residuos, cuyo propósito es la optimización de rutas para camiones de basura que deben recoger residuos de diferentes ubicaciones.

2.1.5. Grafos específicos para algoritmos polinómicos

Aunque el TSP es generalmente $\mathcal{NP}$-Difícil, hay ciertas clases específicas de grafos para las cuales existen algoritmos eficientes que pueden encontrar soluciones óptimas en tiempo polinómico.

- **Grafos de sólidos platónicos**

  Los grafos de sólidos platónicos (*Platonic Solids Graphs*), subyacen un grafo hamiltoniano completo con un grado máximo de 2 (es decir, es un ciclo simple), el problema se puede resolver en tiempo polinómico. Esto se debe a que solo hay una manera de recorrer todos los nodos en un ciclo sin repetir ninguno. [5]

  Todos los grafos de sólidos platónicos son hamiltonianos. Los sólidos platónicos o regulares son poliedros convexos tal que todas sus caras son polígonos regulares iguales entre sí, y en que todos los ángulos sólidos son iguales.

  Los sólidos platónicos son cinco: tetraedro, cubo, octaedro, dodecaedro y icosaedro. Para resolverlo en tiempo polinómico, un algoritmo podría crear una lista de adyacencia, esto se hace en O(1) porque los sólidos platónicos tienen un número fijo de vértices y aristas. Luego se aplica Backtracking con poda heurística, pero dado que los sólidos platónicos tienen estructura altamente simétrica, se puede encontrar recorridos hamiltonianos con reglas fijas predefinidas en O(n). [105]

- **Grafos Euleriano**

Variante	Características	Referencia
TSPP	Beneficio	[68, 69, 70]
STSP	Beneficio	[71, 72]
QTSP	Cuota	[73]
PTP	Beneficio	[74, 75, 103]
TPP	Beneficio	[76]
DRTSP	Recompensa descontada	[78]
PCTSP	Beneficio	[79]
AtTSP	Beneficio	[80]
TSPTW	Ventanas de tiempo	[81]
WTTSP	Tiempos de espera	[82]
PC-TSP	Orden de paradas	[83]
k-dTSP	Recoger y entregar	[84]
TSPPD	Recoger y entregar	[85]
GTSP	Beneficio	[86, 87]
MAX TSP	Reducción de máximos	[88, 104]
BTSP	Cuello de botella	[89]
CTSP	Capacidad	[90]
Steiner TSP	Paradas obligatorias	[91]
TSPWR	Repostar combustible	[92]
CSP	cobertura	[93]
TDTSP	Beneficio	[94]
KTSP	Puntos en movimiento	[95, 96, 97]
MT-TSP	Puntos en movimiento	[96]
STSP	Restriccopnes de velocidad	[98]
mTSP	múltiples viajantes	[54]
MDTSP	múltiples depósitos	[99]
PTSP	Probabilístico	[100, 101]
TTP	Beneficio	[102]

Tabla 2.1: Variantes del TSP

Cuando los nodos son puntos en un plano euclidiano y las distancias cumplen con las propiedades métricas (como la desigualdad triangular), se pueden desarrollar algoritmos polinómicos que encuentren aproximaciones cercanas a la óptima. Aunque no es exacto, el algoritmo de Christofides encuentra una solución que es a lo sumo $1{,}5$ veces la longitud de la ruta óptima en tiempo polinómico. [106]

Para resolverlo usa un Árbol de Expansión Mínima (*Minimum Spanning Tree*), luego se encuentran los vértices de grado impar y se emparejan en pares de costo mínimo. Se combinan las aristas del MST y el emparejamiento mínimo para formar un multigrafo euleriano y luego se convierte en un ciclo hamiltoniano eliminando repeticiones. [106, 107]

- **Grafos de Tour Constante y de Rueda**

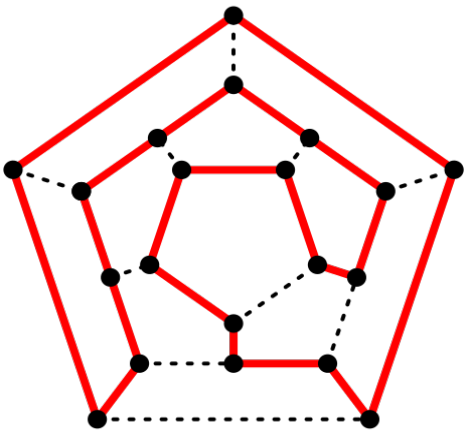


Figura 2.2: Grafo de sólido platónico, Hamiltoniano y camino Hamiltoniano

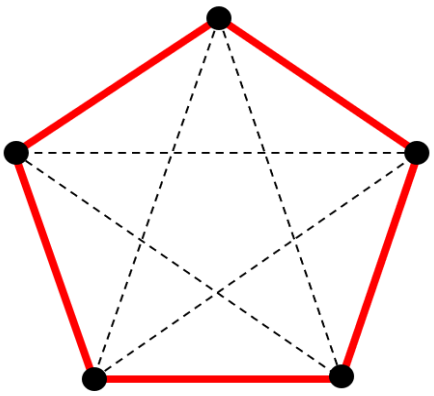


Figura 2.3: Grafo Euleriano

Los grafos de Tour Constante (*CT-Graphs*) son grafos en los cuales todos los recorridos tienen el mismo coste. Un grafo es un CT-graph si, para cualquier matriz de costes compatible con dicho grafo, siempre se obtiene un coste constante para cualquier tour posible. [108]

En un grafo de rueda (*Wheel Graph*), hay un nodo central conectado a todos los demás nodos que forman un ciclo. Debido a la simplicidad y la estructura centralizada del grafo, es posible resolver el TSP en tiempo polinómico, ya que la solución óptima puede derivarse directamente a partir de esta estructura. [109]

Para resolverlo se puede escoger como nodo inicial el punto central, recorrer todos los nodos exteriores en orden y terminar en el punto central. Esto resulta en una complejidad de O(n).

- **Grafos de Halin**

  En casos donde los nodos se organizan en una estructura de árbol o en un camino lineal (es decir, cada nodo está conectada solo con sus vecinos inmediatos de manera ordenada), existen algoritmos que pueden resolver el problema en tiempo polinómico. [110]

  En [111], los autores resuelven el grafo usando distintos enfoques para obtener puntos no dominados.

Estos grafos demuestran que, bajo ciertas restricciones o estructuras específicas, es posible desarrollar algoritmos polinómicos para el TSP. La clave está en la explotación de las propiedades estructurales del grafo para simplificar el problema.

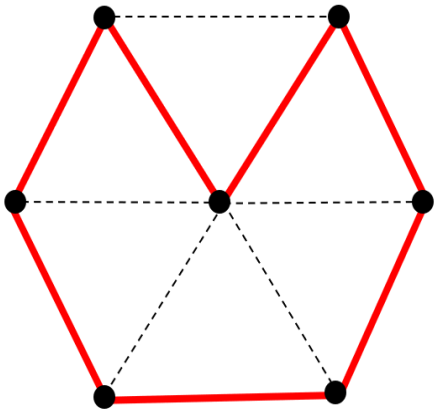


Figura 2.4: Grafo de Tour Constante y de Rueda

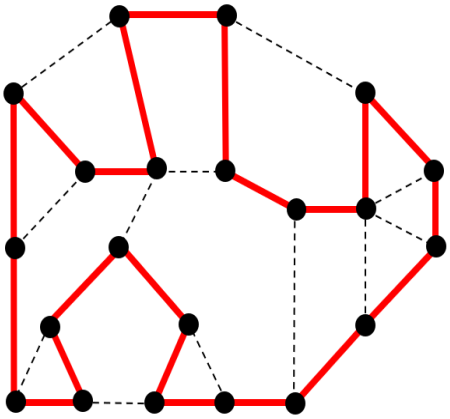


Figura 2.5: Grafo de Halin

2.2. Revisión de propuestas relacionadas

La resolución heurística del Mo-TSRPP para grafos generales es una de las bases fundamentales de esta Tesis Doctoral. Aunque los algoritmos heurísticos han demostrado ser efectivos para encontrar soluciones de alta calidad en problemas complejos de optimización combinatoria, el problema del Mo-TSRPP no ha recibido mucha atención por parte de los investigadores. Por esta razón, se realiza un estudio separado de los problemas Multi-objective Traveling Salesman Repairman Problem (Mo-TSRP), Traveling Salesman Problem with Profits (TSPP) y Traveling Repairman Problem with Profits (TRPP). Esta revisión no es exhaustiva, sino que se ha hecho una selección de los artículos más similares al trabajo realizado en esta Tesis Doctoral.

2.2.1. Publicaciones relacionadas con Mo-TSRP

Hasta donde se sabe, el único trabajo publicado relacionado con el Mo-TSRP fue publicado por S. Bock y K. Klamroth en el año 2019. El artículo, que llevaba por título "Combining Traveling Salesman and Traveling Repairman Problems: A multi-objective approach based on multiple scenarios" [37], aborda la combinación del TSP y el TRP, cada uno con diferentes objetivos: minimizar el coste total del viaje (TSP) y minimizar el tiempo de espera de los clientes (TRP). Se presenta un modelo multiobjetivo que maneja la incertidumbre en los tiempos de viaje utilizando múltiples escenarios para representar distintas condiciones. El estudio también considera variantes con y sin restricciones de plazo para los clientes. El objetivo es explorar la complejidad de los frentes de Pareto y proponer soluciones mediante programación dinámica.

La restricción de tiempos de viaje asegura que el tiempo de inicio en un nodo sea mayor o igual a la suma del tiempo de llegada al nodo anterior y el tiempo de viaje entre ambos, manteniendo una secuencia lógica en la ruta. Además, si se consideran plazos límite, los tiempos de inicio en cada nodo deben ser inferiores o

iguales a dichos plazos, asegurando que todos los clientes sean atendidos dentro de los tiempos estipulados.

El artículo utiliza programación dinámica para resolver el problema en redes de líneas y modelos multiobjetivo. También se analiza la complejidad de las fronteras de Pareto generadas. Los autores proponen algoritmos de programación dinámica que generan las soluciones óptimas y exploran la eficiencia en términos de complejidad computacional, dependiendo del número de estados y la cantidad de escenarios.

El problema Mo-TSRP (combinación de TSP y TRP) es intratable en redes generales, incluso en el caso más simple. En cambio, en redes lineales, se puede resolver el problema utilizando programación dinámica, aunque el tamaño de la frontera de Pareto puede crecer exponencialmente, incluso en configuraciones simples, pero con un número controlado de escenarios y restricciones de tiempo es efectiva.

2.2.2. Publicaciones relacionadas con TSPP

En 1984, T. Tsiligirides examina la aplicación de métodos heurísticos en el deporte de la orientación en el artículo "Heuristic methods applied to orienteering" [72], específicamente en la variante conocida como Orienteering Score Event (SOE). Este problema es análogo a una forma generalizada del TSP, donde se deben maximizar las puntuaciones al visitar puntos de control dentro de un tiempo o distancia máxima permitida.

El estudio presenta dos algoritmos heurísticos para resolver este problema: uno basado en métodos estocásticos y otro determinista. Los algoritmos se comparan en términos de eficiencia y calidad de las soluciones obtenidas.

- Algoritmo Estocástico: utiliza un método Monte Carlo para generar múltiples rutas, evaluando cada ruta en función de una métrica que combina la distancia y la puntuación. Este enfoque permite seleccionar rutas probables basadas en una muestra representativa.
- Algoritmo Determinista: construye rutas dividiendo el área en sectores y formando rutas en estructura de pétalo, optimizando la cobertura dentro de cada sector sin exceder el tiempo máximo. Este algoritmo no utiliza aleatoriedad en la construcción de las rutas.

Los resultados experimentales mostraron que el algoritmo estocástico produce soluciones de mayor calidad en comparación con el algoritmo determinista. Sin embargo, ambos algoritmos tienen un tiempo de ejecución similar. Además, los autores aplicaron un algoritmo de mejora de rutas para optimizar las rutas iniciales generadas, logrando mejoras adicionales en la eficiencia de las soluciones.

Cuatro años después, en 1988, C.P. Keller y M.F. Goodchild, introducen el problema del vendedor multiobjetivo (*Multi-objective Vending Problem*, MVP), una generalización del TSP en el artículo "The multiobjective vending problem: A generalization of the traveling salesman problem. Environment and Planning" [112]. En el MVP, cada nodo tiene una recompensa asociada y se incurre en una penalización al viajar entre nodos. El objetivo es encontrar un circuito que maximice la recompensa

total mientras minimiza la penalización de viaje, considerando que las dos funciones objetivo son inconmensurables y deben tratarse de manera separada. Cada nodo seleccionado debe visitarse exactamente una vez, además la ruta debe formar un circuito cerrado, comenzando y terminando en el depósito. Finalmente, se imponen restricciones para evitar la formación de subtours.

Los autores utilizan el método de restricción para transformar el problema multiobjetivo en una serie de problemas de un solo objetivo. Optimizan una de las funciones objetivo mientras limitan la otra a un valor específico. El algoritmo propuesto utiliza una heurística que combina elementos de enfoques previos para el TSP, como movimientos de nodos en el recorrido y reordenamientos de la secuencia de nodos para optimizar el balance entre la recompensa y la penalización. La heurística también incluye la capacidad de ajustar interactivamente las soluciones para adaptar las operaciones al conjunto de datos específico.

El algoritmo se probó en un conjunto de veinticinco ciudades en Alemania, utilizando distancias entre ciudades como penalizaciones y poblaciones como recompensas. La heurística fue capaz de generar un conjunto de soluciones no inferiores con diferentes valores de penalización máxima. Los resultados mostraron que el algoritmo puede equilibrar efectivamente las dos funciones objetivo y generar rutas que maximizan la recompensa dentro de las restricciones de penalización impuestas.

Dos años después, en 1990, se publica el artículo “The selective traveling salesman problem” [23], escrito por Gilbert Laporte y Silvano Martello. En él se explora el problema del vendedor viajero selectivo (STSP), también conocido como problema de orientación (OP). En este problema, se busca determinar un circuito simple que maximice las ganancias asociadas a los nodos visitados. Se impone la restricción de que la longitud del recorrido no exceda un valor predefinido, que el recorrido debe comenzar y terminar en un nodo específico y, restricciones de conectividad para evitar la formación de subtours.

Los autores presentan formulaciones de programación lineal entera (ILP) para el STSP y desarrollan algoritmos exactos de enumeración que incorporan cotas superiores e inferiores:

- Algoritmo ILP basado en enumeración: Utiliza cotas superiores e inferiores derivadas de formulaciones de mochila (knapsack) y un proceso de enumeración ramificado para buscar la solución óptima.
- Algoritmos heurísticos: Implementan versiones modificadas del algoritmo NN y del algoritmo de inserción más barata para generar soluciones iniciales que se mejoran en el proceso de enumeración.

Las pruebas computacionales realizadas mostraron que el algoritmo basado en ILP es efectivo para resolver instancias pequeñas y medianas del STSP, alcanzando soluciones óptimas en tiempos razonables para grafos con hasta 90 nodos. Las heurísticas desarrolladas proporcionaron soluciones cercanas al óptimo en tiempos significativamente más rápidos, aunque su efectividad disminuyó en instancias con mayores restricciones de longitud.

En 1995, M. Fischetti, J.J. Salazar González y P. Toth publican el artículo "On prize-collecting tours and the asymmetric traveling salesman problem" [74], en el cual exploran el problema de tours recolectores de premios (PCT) y su relación con el problema asimétrico del vendedor viajero (ATSP). En el PCT, el objetivo es encontrar un recorrido que maximice el premio recolectado mientras minimiza los costes de viaje. Cada nodo puede ser visitado solo una vez o no ser visitado en absoluto, el recorrido debe formar un ciclo continuo y conectado que incluya al menos el depósito y algunos de los nodos seleccionados, y se debe evitar la formación de subtours.

Los autores desarrollan formulaciones de programación lineal entera (ILP) y presentan algoritmos exactos para resolver el problema:

- Formulaciones de ILP: Utilizan una formulación extendida que incluye variables para modelar la recolección de premios y la elección de arcos, complementada con restricciones para asegurar la conectividad y evitar subciclos.
- Algoritmos de ramificación y corte: Se emplea este método para explorar sistemáticamente las posibles soluciones del PCT, implementando cortes específicos que ayudan a reducir el espacio de búsqueda y mejorar la eficiencia del proceso.

El enfoque propuesto se probó en instancias asimétricas derivadas del ATSP y mostró una mejora significativa en los tiempos de cómputo comparado con métodos previos. Las pruebas indicaron que, en instancias de tamaño medio, el algoritmo fue capaz de encontrar soluciones óptimas en tiempos razonables, demostrando la efectividad de la combinación de formulaciones ILP y técnicas de ramificación y corte.

Tres años después, en 1998, se publica el artículo "A Tabu Search Heuristic for the Undirected Selective Traveling Salesman Problem" [113], escrito por Michel Gendreau, Gilbert Laporte y Frédéric Semet. El artículo se centra en el problema del vendedor viajero selectivo no dirigido (STSP). El STSP busca encontrar un ciclo hamiltoniano que maximice las ganancias asociadas a los nodos seleccionados, manteniendo la longitud del recorrido por debajo de un límite preestablecido.

Se presenta un enfoque heurístico basado en búsqueda tabú (TS) que inserta o elimina conjuntos de nodos en la ruta actual para mejorar la solución:

- Heurística de inserción y sacudida (Insert and Shake): se utiliza como punto de partida, extendiendo gradualmente un tour hasta que no se puedan introducir más nodos sin violar el límite de distancia.
- Algoritmo GENIUS: se emplea para optimizar localmente las rutas obtenidas mediante la inserción y eliminación de nodos. El TS opera sobre clústeres de nodos, permitiendo la inserción o eliminación simultánea de múltiples nodos durante el proceso de búsqueda.
- Evaluación de movimientos: se basa en la relación entre las ganancias adicionales obtenidas y las distancias incrementadas para seleccionar las mejores modificaciones en cada iteración.

El algoritmo fue probado en instancias generadas aleatoriamente de hasta 300 nodos. Los resultados mostraron que el TS es capaz de producir soluciones óptimas o cercanas al óptimo en la mayoría de las instancias. La desviación de las soluciones respecto al óptimo fue, en promedio, inferior al 1 %. Además, se comparó la eficiencia del TS con otras heurísticas previas, demostrando que el enfoque propuesto ofrece mejores resultados en términos de calidad y estabilidad.

En 2005, Dominique Feillet, Pierre Dejax y Michel Gendreau publican "Traveling Salesman Problems with Profits" [70], en el presentan una revisión y clasificación de las variantes del TSP en las que se considera la recolección de beneficios (TSPP). Estas variantes del TSP permiten al vendedor elegir qué nodos visitar para maximizar las ganancias obtenidas mientras se minimizan los costes de viaje, sin la obligación de visitar todos los nodos. El artículo explora diferentes formulaciones, aplicaciones y métodos exactos y heurísticos utilizados para resolver estas variantes.

El TSPP puede formularse a través de tres variantes principales según la forma en que se gestionan las ganancias y los costes:

1. **Problema de Tour Rentable (PTP)**

   La función objetivo de este modelo busca minimizar el coste neto del recorrido, definido como la diferencia entre los costes totales de viaje y las ganancias recolectadas al visitar nodos específicos. Cada nodo puede ser visitado una sola vez o no ser visitado en absoluto. El recorrido debe formar un ciclo conectado que incluya el nodo de inicio y, opcionalmente, un subconjunto de nodos seleccionados, evitando la formación de subtours.

2. **Problema de Orientación (OP)**

   La función objetivo de este modelo busca maximizar las ganancias totales obtenidas al visitar un subconjunto de nodos seleccionados. Se limita el coste total del recorrido a un valor máximo permitido, asegurando que la solución sea factible en términos de los recursos disponibles. Adicionalmente, las restricciones implícitas de conectividad garantizan que los nodos seleccionados formen un ciclo continuo desde el nodo inicial.

3. **Problema de Recolección de Premios (PCTSP)**

   La función objetivo de este modelo tiene como propósito minimizar los costes totales del recorrido. Este enfoque se centra en encontrar una solución económicamente eficiente que cumpla con el requisito de ganancias acumuladas. La restricción principal asegura que se recolecte al menos una ganancia mínima predefinida. Esto garantiza que el recorrido incluya un subconjunto de nodos seleccionados que cumplan con el umbral de ganancias. Las restricciones adicionales implícitas incluyen la conectividad del recorrido.

   El artículo describe tanto métodos exactos como heurísticos:

- Métodos exactos: Incluyen enfoques de ramificación y corte adaptados del TSP clásico, aprovechando estructuras de relajación de asignación y árboles de 1-arborescencia para calcular límites inferiores efectivos.

- Heurísticas: Se utilizan estrategias de inserción, eliminación, y búsqueda local para construir y mejorar rutas. También se emplean metaheurísticas como búsqueda tabú, algoritmos genéticos y enfoques basados en redes neuronales para mejorar la exploración del espacio de soluciones.

Los resultados muestran que los enfoques heurísticos y metaheurísticos logran soluciones cercanas al óptimo en tiempos de cómputo razonables, especialmente en instancias grandes. Los métodos exactos, aunque efectivos para instancias pequeñas y medianas, tienen limitaciones en términos de escalabilidad para problemas de mayor tamaño debido a la complejidad computacional.

Transcurridos dos años, en 2007, se publica el artículo "The biobjective traveling salesman problem with profit" [114], presentado por Ömür Şimşek. En él se aborda el TSPP desde una perspectiva multiobjetivo. Tradicionalmente, este problema se estudia como uno de objetivo único, ya sea maximizando las ganancias o minimizando los costes de viaje. Sin embargo, en esta investigación se utiliza un enfoque multiobjetivo para identificar el conjunto eficiente de Pareto, permitiendo analizar el equilibrio entre las ganancias recolectadas y los costes de viaje incurridos. Cada nodo puede ser visitado solo una vez o no ser visitado. Además, el recorrido debe formar un ciclo continuo que conecte el nodo de inicio con los nodos seleccionados, evitando subtours.

El artículo propone un enfoque basado en el método de ϵ-constraint, que convierte el problema multiobjetivo en una serie de problemas de un solo objetivo al limitar uno de los objetivos mientras se optimiza el otro. Este método se combina con una heurística de inserción y optimización local (CGW) para encontrar soluciones eficientes y generar el frente de Pareto. La heurística CGW incluye técnicas como el intercambio de puntos y el movimiento de nodos para mejorar las rutas generadas.

Los experimentos computacionales realizados con el algoritmo en instancias de problemas clásicos y generados aleatoriamente demostraron que el método propuesto es capaz de generar un conjunto eficiente de soluciones de Pareto. Los resultados indican que el enfoque multiobjetivo permite analizar de manera efectiva las compensaciones entre maximizar las ganancias y minimizar los costes de viaje, mostrando mejoras en comparación con enfoques de objetivo único tradicionales.

Al año siguiente, en 2008, Serdar Karademir explora el TSPP en su artículo "A genetic algorithm for the biobjective traveling salesman problem with profits" [115]. La función objetivo de este modelo multiobjetivo busca minimizar el coste total de la ruta y maximiza la ganancia total recolectada al visitar nodos. Las restricciones del modelo corresponden a las condiciones comunes del TSP.

Utiliza un algoritmo genético multiobjetivo (MOGA), específicamente el NSGA-II, modificado para incluir:

- La heurística Lin-Kernighan para calcular eficientemente el coste del recorrido.
- Un mecanismo de preservación de la diversidad que mantiene un conjunto de soluciones no dominadas (frente de Pareto) durante las generaciones del algoritmo.

- Mejoras en el NSGA-II para optimizar la eficiencia en instancias grandes, lo que incluye un archivo externo de soluciones elitistas que se integran de forma dinámica en el proceso evolutivo.

Las pruebas computacionales realizadas en instancias de problemas de hasta 400 nodos demostraron que las mejoras al NSGA-II permiten aproximar de manera efectiva el frente de Pareto. Los resultados indicaron que el algoritmo modificado (mNSGA-II) supera al NSGA-II estándar en términos de convergencia y diversidad de las soluciones generadas. Los valores obtenidos para el volumen hipercúbico y otras métricas de rendimiento muestran que mNSGA-II proporciona soluciones más cercanas al óptimo en menos tiempo computacional.

También en 2008 aparece el artículo "Multi-objective Meta-heuristics for the Traveling Salesman Problem with Profits" [116], de N. Jozefowiez, F. Glover y M. Laguna donde se aborda el TSPP. El enfoque de este estudio es encontrar soluciones eficientes utilizando la noción de optimalidad de Pareto, generando un conjunto de soluciones no dominadas para construir una frontera eficiente. Busca minimizar costes y maximizar beneficios. Establece una longitud máxima del recorrido, que es el límite permitido para los costes de viaje. Otra restricción impone un requisito de ganancia mínima asegurando que las soluciones sean rentables.

Se implementó una heurística híbrida que combina un algoritmo evolutivo multiobjetivo (MOEA) y un proceso de búsqueda local basado en cadenas de eyección (EC). El MOEA genera soluciones iniciales diversas en el espacio objetivo, mientras que el EC mejora estas soluciones a través de movimientos en el recorrido para acercarlas a la frontera de Pareto. La combinación de ambos métodos permite tanto la exploración como la intensificación del espacio de soluciones.

Los resultados experimentales demostraron que la heurística híbrida propuesta (HM) es eficiente para encontrar aproximaciones de alta calidad a la frontera de Pareto en comparación con métodos existentes, como el enfoque de ϵ-constraint aplicado a la búsqueda tabú para el TSP selectivo. La HM mostró tiempos de cómputo significativamente menores y produjo más soluciones potencialmente óptimas en el conjunto de Pareto.

Un año más tarde, en 2009, Jean-François Bérubé, Michel Gendreau y Jean-Yves Potvin publican "An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits" [56]. En él se presenta un método exacto basado en la técnica ϵ-constraint para resolver problemas de optimización combinatoria biobjetivo. El objetivo es maximizar la ganancia total recogida y minimizar los costes de viaje simultáneamente. Para abordar la naturaleza multiobjetivo del problema, se utiliza el método ϵ-constraint, que transforma uno de los objetivos en una restricción parametrizada. Esto permite resolver iterativamente problemas mono-objetivo, generando un conjunto de soluciones que forman el frente de Pareto exacto.

El artículo emplea un método exacto de corte y ramificación para resolver cada subproblema generado por la técnica ϵ-constraint. Además, se aplican heurísticas para acelerar el proceso, aprovechando información de subproblemas previos. El método incluye mejoras como el uso de desigualdades activas y soluciones iniciales

optimizadas que permiten reducir significativamente los tiempos de cómputo.

Los resultados muestran que el método propuesto es capaz de encontrar el frente de Pareto exacto para instancias del TSPP derivadas de problemas clásicos con hasta 150 nodos. Las pruebas computacionales indican que, a pesar de la complejidad del problema, el enfoque es efectivo y eficiente, logrando tiempos de cómputo razonables y una alta precisión en la generación del conjunto de soluciones óptimas no dominadas.

En 2010, se publica "Genetic Algorithm Finds Routes in Traveling Salesman Problem with Profits" [117] de Anna Piwońska, donde se aborda el TSPP y el objetivo es maximizar las ganancias recogidas sin exceder una longitud de recorrido preestablecida. En este estudio, se propusieron dos supuestos adicionales: el grafo no necesita ser completo y se permite visitar un nodo más de una vez, aunque las ganancias solo se contabilizan la primera vez que se visita. Estos supuestos hacen el problema más realista, con aplicaciones en logística.

El estudio emplea un algoritmo genético (GA) que utiliza operadores específicos como selección, cruce y mutación para optimizar las soluciones. El algoritmo comienza generando una población inicial de recorridos y evalúa la "aptitud" de cada individuo en función de las ganancias recolectadas. Posteriormente, se aplican operadores de selección por torneo, cruce (en el que se combinan partes de recorridos de diferentes individuos) y mutación para introducir variabilidad y mejorar la población a lo largo de varias generaciones.

Los experimentos se llevaron a cabo en redes de ciudades en tres regiones de Polonia. Para cada región, se probaron diferentes valores de longitud máxima, y se obtuvieron soluciones que maximizaban las ganancias dentro de los límites de recorrido permitidos. Los resultados mostraron que el algoritmo propuesto fue eficaz en encontrar recorridos optimizados.

Dos años después, en 2012, se publica el artículo "A two-phase method for bi-objective combinatorial optimization and its application to the TSP with profits" [118] escrito por Carlo Filippi y Elisa Stevanato, quienes presentan un método de dos fases para la optimización combinatoria biobjetivo, aplicándolo específicamente al TSPP. Este problema busca equilibrar dos objetivos en conflicto: minimizar la distancia total recorrida y maximizar las ganancias obtenidas. Se aplican las restricciones comunes del TSP (cada nodo se visita una vez, ciclo continuo que conecte los nodos seleccionados y evitar subtours).

El enfoque se basa en un algoritmo enumerativo que se incorpora en un marco de dos fases. En la primera fase, se calculan los puntos eficientes soportados utilizando un método de suma ponderada. En la segunda fase, se identifican los puntos no soportados mediante un procedimiento específico que explora subespacios del conjunto de soluciones. Esta estrategia permite optimizar el tiempo de computación al dividir el problema en etapas manejables y utilizar técnicas adecuadas para cada tipo de punto eficiente.

Los autores probaron su método en instancias generadas aleatoriamente y en instancias de la biblioteca TSPLIB, comparándolo con la técnica ϵ-constraint. Los resultados mostraron que el enfoque de dos fases propuesto fue más eficiente en

términos de tiempo de cómputo y estabilidad en la generación de puntos eficientes, destacando especialmente en problemas de tamaño medio a grande donde otras metodologías tuvieron dificultades para obtener resultados óptimos en tiempos razonables.

El siguiente año, 2013, N. Labadie, J. Melechovsky y C. Prins presentan el artículo "An Evolutionary Algorithm with Path Relinking for a Bi-objective Multiple Traveling Salesman Problem with Profits" [119]. Los autores abordan el problema del vendedor viajero múltiple con beneficios y dos objetivos (BOMTSPP), una generalización del TSP clásico con beneficios (TSPP). El estudio se centra en maximizar el beneficio total recolectado y minimizar la distancia total recorrida, considerando múltiples vehículos. Las restricciones consisten en un número de vehículos fijo, cada vehículo debe seguir un ciclo cerrado que comience y termine en el depósito, unicidad de los nodos visitados por cada vehículo y la conectividad de los recorridos.

El enfoque propuesto utiliza un algoritmo evolutivo multiobjetivo basado en el NSGA-II, mejorado con una búsqueda local para optimizar las soluciones intermedias. Además, se aplica un esquema de *Path Relinking* (PR) como proceso de post-optimización para explorar trayectorias entre soluciones de alta calidad y mejorar el conjunto de soluciones no dominadas. El algoritmo evolutivo se combina con procedimientos de cruzamiento y mutación para generar nuevas soluciones que son evaluadas y mejoradas mediante PR.

Los experimentos computacionales se realizaron en un conjunto de instancias de problemas bien conocidos, mostrando que el uso del esquema de PR mejora significativamente el conjunto de soluciones no dominadas generadas por NSGA-II. En instancias con un número mayor de nodos (clientes), PR fue particularmente efectivo, incrementando en un 50 % la calidad de las soluciones y mejorando el indicador de volumen hipercúbico en un promedio de 1,5 % en comparación con el NSGA-II estándar.

En 2014, Enrico Angelelli, Cristina Bazgan, M. Grazia Speranza y Zsolt Tuza publica su trabajo "Complexity and approximation for Traveling Salesman Problems with profits" [120], que se enfoca en el análisis de la complejidad computacional y en el desarrollo de algoritmos de aproximación para tres variantes del TSPP: el Profitable Tour Problem (PTP), el OP y el PCTSP. Estos problemas buscan un equilibrio entre maximizar las ganancias y minimizar el coste del recorrido, pero son $\mathcal{NP}$-Difícil en general. Los autores estudian estos problemas en diferentes topologías de red, como caminos, ciclos, estrellas y árboles, con y sin tiempos de servicio, presentando algoritmos polinomiales y esquemas de aproximación en tiempo completamente polinomial (FPTAS) para los casos $\mathcal{NP}$-Difícil.

Las variantes del problema TSPP exploran diferentes enfoques para equilibrar beneficios y costes en recorridos optimizados:

- PTP (Profitable Tour Problem): La función objetivo busca maximizar el beneficio neto del recorrido, definido como la diferencia entre las ganancias totales recolectadas y el tiempo total empleado en el recorrido.
- OP (Orienteering Problem): Este modelo maximiza las ganancias totales reco-

lectadas, sujeto a la restricción de que el tiempo total del recorrido no exceda un límite predefinido.

- PCTSP (Prize-Collecting Traveling Salesman Problem): Aquí, la función objetivo minimiza el tiempo total del recorrido, asegurando que se alcance una ganancia mínima requerida.

Las restricciones del problema garantizan la factibilidad de los recorridos:

- En OP y PCTSP, las restricciones clave incluyen el límite máximo de tiempo total del recorrido o la condición de ganancia mínima. Estas condiciones controlan el equilibrio entre tiempo y beneficios, ajustando el enfoque según la variante.
- Todas las variantes comparten restricciones básicas del TSP.

Los autores presentan algoritmos polinomiales para casos específicos en estructuras de grafo simples, como caminos y ciclos sin tiempos de servicio. Además, desarrollan esquemas de tiempo polinomial (FPTAS) para resolver instancias $\mathcal{NP}$-Difícil de OP y PCTSP en árboles y otras topologías. Por ejemplo, se emplea un algoritmo pseudopolinomial que se adapta a la topología del grafo, permitiendo una solución aproximada eficiente en tiempo de ejecución dependiendo de la complejidad del grafo.

Los resultados muestran que las variantes OP y PCTSP se vuelven $\mathcal{NP}$-Difícil incluso en topologías simples como estrellas y árboles cuando se introducen tiempos de servicio. Sin embargo, se desarrollaron algoritmos polinomiales para estas variantes en estructuras sin tiempos de servicio, demostrando la solvibilidad en tiempo lineal en ciertos casos. Además, los FPTAS propuestos permiten aproximaciones eficientes para instancias más complejas, adaptándose a la topología y características del grafo.

Ese mismo año, en 2014, Ms. K. Ilavarasi y Dr. K. Suresh Joseph publican "Variants of traveling salesman problem: A survey" [18], el cual proporciona un estudio exhaustivo de las variantes del TSP. El artículo clasifica las variantes en cuatro categorías principales: variantes basadas en beneficios, ventanas de tiempo, maximización y cinética, analizando su formulación matemática y aplicaciones prácticas.

Las variantes del TSP se formulan para maximizar o minimizar diferentes criterios, dependiendo de la categoría. En las basadas en beneficios, se busca maximizar la ganancia recolectada mientras se minimizan los costes de viaje, sin ser obligatorio visitar todos los nodos. Las restricciones varían según la variante. En las basadas en beneficios: No se requiere visitar todos los nodos, pero se debe alcanzar un umbral mínimo de ganancia o mantener el coste del recorrido por debajo de un límite.

El artículo revisa una variedad de algoritmos utilizados para resolver las variantes del TSP, incluidos métodos exactos como la programación dinámica y heurísticas como algoritmos genéticos y búsqueda local. Dependiendo de la variante, los algoritmos se adaptan para manejar las restricciones específicas y optimizar los objetivos, como maximizar ganancias o respetar ventanas de tiempo. Además, se mencionan enfoques de aproximación y algoritmos pseudopolinomiales para instancias $\mathcal{NP}$-Difícil.

El estudio proporciona una comparación de las técnicas existentes y cómo se aplican a las distintas variantes del TSP. Se destaca que, aunque muchos de estos problemas siguen siendo $\mathcal{NP}$-Difícil, ciertos enfoques pueden ofrecer soluciones aproximadas eficientes o exactas bajo condiciones específicas. En particular, se muestra que los algoritmos heurísticos y las técnicas de optimización multiobjetivo son efectivos para tratar con las complejidades añadidas por las restricciones y los objetivos múltiples de las variantes.

Dos años después, en 2016, Mengying Zhang, Jin Qin, Yugang Yu y Liang Liang, introducen una clase de problemas de selección y enrutamiento, denominada ”problema del vendedor viajero con ganancias y clientes estocásticos”(TSPPSC) en el trabajo “Traveling salesman problems with profits and stochastic customers” [121]. Este problema es una extensión del TSPP y considera la presencia estocástica de los clientes. Se formula en un grafo completo donde se asocian ganancias a los nodos y costes de viaje a las aristas. El objetivo es optimizar simultáneamente las ganancias y los costes de viaje. Como restricciones, cada cliente puede ser visitado una sola vez o no ser visitado en absoluto, restricción de ciclo cerrado, evitar subtours y, la inclusión de un cliente en la ruta debe estar respaldada por su probabilidad de requerir servicio.

El artículo presenta un Genetic Algorithm (GA) para resolver el OP con clientes estocásticos. El GA emplea un cruce basado en el orden, mutaciones locales (añadir, omitir, intercambiar, etc.), y un algoritmo de recocido simulado para mejorar la mejor solución encontrada. El enfoque se evalúa en función de su capacidad para maximizar las ganancias esperadas mientras se minimizan los costes de viaje, respetando las restricciones del problema.

Los resultados experimentales mostraron que el GA propuesto supera a métodos anteriores en términos de velocidad y calidad de las soluciones. Se aplicaron pruebas con instancias de 50 y 100 nodos, y el GA demostró ser más eficiente que las soluciones exactas y aproximadas previas, como las resueltas con LocalSolver y CPO, especialmente en problemas grandes. El GA proporcionó mejoras significativas en soluciones comparadas con otros métodos en instancias de clientes homogéneos y heterogéneos.

Un año después, en 2017, el artículo “A unified matheuristic for solving multi-constrained traveling salesman problems with profits” [122], escrito por Rahma Lahyani, Mahdi Khemakhem y Frédéric Semet, aborda variantes complejas del TSPP.

La función objetivo del problema tiene como propósito minimizar un coste total que integra tres componentes principales: los costes asociados al transporte, las penalizaciones por tiempos de espera en los nodos y las ganancias derivadas de los nodos visitados.

El modelo está sujeto a un conjunto de restricciones. La restricción de capacidad máxima del vehículo. La restricción de incompatibilidad prohíbe el transporte conjunto de productos conflictivos en el mismo compartimiento. Restricción de ventanas temporales para cada cliente y el depósito. Y penalización con tiempos de espera cuando los vehículos llegan antes del inicio de la ventana temporal.

Los autores desarrollan un enfoque basado en VNS que se combina con técni-

cas de optimización exacta para gestionar las vecindades de carga y enrutamiento. Los algoritmos incluyen:

- Vecindades de enrutamiento: se utilizan para modificar la secuencia y los clientes visitados.
- Vecindades de carga: se optimizan mediante programas matemáticos que resuelven problemas de asignación de compartimientos de manera óptima.
- Fase de mejora local: se aplica una búsqueda local para mejorar la calidad de las soluciones, utilizando un algoritmo de 2-opt para reducir el tiempo de viaje y un algoritmo de optimización del tiempo de espera para minimizar las penalizaciones.

La matheurística propuesta se probó en un conjunto de datos amplio que incluye instancias del OP y Orienteering Problem with Time Windows (OPTW). Los resultados mostraron que la matheurística es competitiva frente a métodos avanzados existentes, logrando soluciones de alta calidad en tiempos computacionales razonables. Las pruebas indicaron que las configuraciones específicas del VNS son clave para el rendimiento del enfoque, especialmente en instancias con restricciones complejas.

En 2019, transcurridos dos años, se publica "Cooperation of customers in traveling salesman problems with profits" [123], escrito por Ondrej Osicka, Mario Guajardo y Kurt Jörnsten, donde se explora una variante cooperativa del problema del vendedor viajero con beneficios, denominado "problema del tour rentable" (*Profitable Tour Problem*, PTP). La función objetivo del PTP busca maximizar la diferencia entre los premios recolectados y los costes incurridos durante el recorrido. Las restricciones del PTP garantizan que los premios recolectados sean suficientes para cubrir los costes adicionales generados por la incorporación de clientes específicos.

El estudio utiliza un enfoque de teoría de juegos cooperativos para definir el PTP. Este enfoque se basa en la cooperación de los clientes, quienes deben ofrecer premios para garantizar que el vendedor los visite. Se desarrolla un marco de teoría de juegos para analizar cómo los clientes pueden formar coaliciones y determinar las cantidades óptimas que deben ofrecer para ser visitados. Se desarrolla un modelo de programación lineal entera que maximiza la diferencia entre las ganancias y los costes de viaje, considerando coaliciones de clientes que determinan colectivamente los premios a ofrecer. Además, se explora el uso de métodos como el núcleo y la nucleolus para asignar los premios de manera equitativa y garantizar que las coaliciones sean estables y justas.

Los resultados del análisis muestran que, si el núcleo del juego del vendedor viajero es no vacío, las asignaciones óptimas de premios coinciden con este núcleo. Esto significa que los clientes pueden cooperar para garantizar que todos sean visitados al asignar las recompensas de manera equitativa. En casos donde el núcleo es vacío, se requiere una modificación para garantizar la viabilidad de las asignaciones, utilizando un parámetro adicional (ϵ) para ajustar los premios necesarios.

En 2020, Bruno C.H. Silva, Islame F.C. Fernandes, Marco C. Goldbarg y Elizabeth F.G. Goldbarg escriben "Quota traveling salesman problem with passengers,

incomplete ride and collection time optimization by ant-based algorithms" [24], donde abordan una variante del TSP con cuota (QTSP), denominada "Quota Traveling Salesman Problem with Passengers, Incomplete Ride and Collection Time"(QTSP-PIC). Este problema considera la integración de un sistema de movilidad compartida donde el vendedor viajero transporta pasajeros para reducir costes de transporte mientras cumple con una cuota mínima en los puntos visitados.

La función objetivo del QTSP-PIC combina la minimización de los costes de transporte y las penalizaciones asociadas a incumplimientos en los destinos de los pasajeros. El modelo está sujeto a complejas restricciones, como que el vendedor viajero debe garantizar que la cuota mínima de beneficios sea alcanzada. Además, se deben respetar las limitaciones de capacidad del vehículo, y las restricciones de tiempo asociadas a las demandas de los pasajeros. Finalmente, se aplican penalizaciones si un pasajero es dejado en un lugar diferente al destino solicitado. Estas restricciones hacen del QTSP-PIC un problema altamente complejo y no lineal.

El estudio implementa varios algoritmos basados en la optimización por colonia de hormigas (ACO), incluyendo:

- Algoritmo de sistema de hormigas (AS): una versión básica para construir rutas minimizando los costes y maximizando el cumplimiento de solicitudes de transporte.
- Sistema de colonia de hormigas (ACS): una variante del AS con un sesgo voraz introducido para mejorar la selección de rutas.
- Sistema de colonia de hormigas multi-estrategia (MS-ACS): introduce diferentes fuentes de información heurística (orientada al coste, tiempo, cuota y pasajeros) para diversificar la búsqueda y mejorar la eficiencia

Los resultados experimentales mostraron que el MS-ACS superó a las otras versiones de ACO y a las heurísticas básicas implementadas. Este algoritmo encontró las mejores soluciones en la mayoría de las instancias, especialmente en aquellas con mayor complejidad. Además, se demostró que el enfoque basado en MS-ACS es efectivo para manejar el equilibrio entre el coste del recorrido y las penalizaciones asociadas.

Un año después, en 2021, aparece el artículo "A Profit Guided Coordination Heuristic for Traveling Thief Problems" [124], escrito por Majid Namazi, M. A. Hakim Newton, Abdul Sattar y Conrad Sanderson, donde se aborda el problema del ladrón viajero (*Traveling Thief Problems*, TTP), que combina dos componentes $\mathcal{NP}$-Difícil: el TSP y el problema de la mochila (KP). El TTP plantea la complejidad de resolver ambos problemas de forma coordinada, ya que la solución de uno afecta al otro.

El objetivo del Problema del Ladrón y el Viajero (TTP) es maximizar las ganancias netas obtenidas por un ladrón que selecciona objetos de diferentes ubicaciones mientras recorre un circuito similar al del problema del vendedor viajero.

El modelo está sujeto principalmente a dos restricciones: La selección de objetos está limitada por el peso máximo que la mochila puede transportar y, la restricción de velocidad dependiente del peso total transportado en la mochila.

En este trabajo, los autores introducen la heurística "*Profit Guided Coordination Heuristic*" (PGCH) que extiende la clásica 2-opt, una técnica de inversión de segmentos del TSP, para coordinar la selección de objetos en el KP. La PGCH revisa los segmentos de la ruta y ajusta la selección de objetos para maximizar el beneficio en cada segmento modificado, reemplazando objetos menos rentables por otros más rentables cuando es posible. Este enfoque se aplica en un solver denominado CTTP, que utiliza la heurística Chained Lin-Kernighan (CLK) para inicializar el tour y optimiza iterativamente las soluciones mediante PGCH y ajustes en el plan de selección de objetos.

El solver CTTP fue evaluado en diversas instancias del TTP y se comparó con solvers como MATLS, S5 y CS2SA*. Los resultados indicaron que CTTP, empleando la heurística PGCH, supera consistentemente a los otros métodos en términos de índice de desviación relativa (RDI). En todas las categorías de instancias evaluadas (diferentes configuraciones de objetos y capacidades de mochila), el solver mostró mejoras significativas en el valor del objetivo, demostrando la eficacia de la coordinación explícita en la solución de los componentes TSP y KP.

Dos años después, en 2023, Pengfei He, Jin-Kao Hao y Qinghua Wu publican "Hybrid genetic algorithm for undirected traveling salesman problems with profits" [125], donde se introduce un algoritmo genético híbrido (HGA) para resolver dos variantes del TSPP: el problema de orientación (OP) y el problema del vendedor viajero recolector de premios (PCTSP). Estos problemas optimizan simultáneamente el beneficio recolectado y los costes de viaje, ambos $\mathcal{NP}$-Difícil. El algoritmo combina un operador de cruce extendido con búsqueda local y estrategias de diversificación para mejorar la calidad de las soluciones y gestionar la diversidad de la población.

Las variantes del OP y Prize-Collecting Traveling Salesman Problem (PCTSP) tienen objetivos distintos que optimizan ganancias y costes bajo restricciones específicas. En el caso del OP, el objetivo es maximizar la ganancia total recolectada, mientras se mantiene el coste del recorrido por debajo de un límite máximo permitido. En el PCTSP, el objetivo es minimizar el coste total del recorrido, asegurando que la ganancia total alcance al menos un valor mínimo.

El HGA fue probado en instancias de referencia para OP y PCTSP, mostrando resultados competitivos en comparación con otros métodos de vanguardia. El algoritmo superó los mejores límites conocidos en varias instancias y mejoró la calidad de las soluciones en menos tiempo que los algoritmos exactos, especialmente en instancias de gran tamaño.

Ese mismo año, en 2023, se publica "Maximizing Total Net Profit for Traveling Salesman Problem with Profits Using Metaheuristic Algorithms" [126], escrito por Eyüp Ensar Işık y Mısra Şimşir. El artículo se enfoca en maximizar el beneficio neto obtenido al realizar un recorrido en el TSPP. Esto se logra mediante la maximización de la suma de los beneficios asociados a los nodos visitados, menos la suma de los costes incurridos al recorrer las aristas seleccionadas.

Las restricciones del modelo son: En primer lugar cada nodo puede ser visitado solo una vez, a menos que sea un nodo opcional. En segundo lugar, la longitud total del recorrido no debe exceder un valor máximo predefinido.

Los autores implementan dos algoritmos metaheurísticos para resolver el problema:

- Recocido Simulado (SA): Utiliza múltiples estructuras de vecindad para explorar soluciones. La temperatura inicial se establece en un valor alto para permitir una exploración amplia, y se reduce de manera geométrica para concentrarse en soluciones óptimas.
- Búsqueda de Vecindad Variable (VNS): Emplea diferentes operadores de vecindad (2-Swap, 3-Opt, Inserción, Eliminación) para diversificar y mejorar las soluciones. Este enfoque permite explorar eficazmente el espacio de soluciones para evitar estancarse en óptimos locales.

Los experimentos se realizaron en instancias de diferentes tamaños con valores de beneficios bajos y altos. Los resultados indicaron que el VNS produjo valores de objetivo superiores y visitó más nodos en comparación con SA en la mayoría de las instancias. Las diferencias de rendimiento fueron notables especialmente en casos de alta ganancia, donde VNS aprovechó mejor la diversificación proporcionada por sus múltiples estructuras de vecindad.

2.2.3. Publicaciones relacionadas con TRPP

Uno de los artículos más relevantes sobre el problema de la latencia con beneficios aparece en 2007, "A latency problem with profits" [127] de Sofie Coene y Frits C. R. Spieksma. En el TRPP, un servidor debe decidir qué clientes visitar, considerando que cada cliente tiene un beneficio asociado que disminuye con el tiempo debido a la latencia del viaje. El objetivo es maximizar el beneficio total, sustrayendo el tiempo de llegada a cada cliente del beneficio obtenido. El servidor comienza en el origen y viaja a una velocidad constante, y no todos los clientes necesitan ser visitados.

El modelo está sujeto a restricciones prácticas que garantizan la viabilidad de las soluciones. En primer lugar, el tiempo de viaje está condicionado por una velocidad constante del servidor. En segundo lugar, no es obligatorio visitar a todos los clientes. Por último, el beneficio asociado a cada cliente disminuye linealmente con el tiempo debido a la latencia en el servicio.

El artículo presenta un algoritmo de programación dinámica para resolver el TRPP en un entorno lineal. Este algoritmo es una extensión de soluciones previas para el problema de latencia mínima sin beneficios (TRP clásico) y resuelve el TRPP en tiempo polinómico. En particular, el algoritmo sigue un enfoque similar al de Afrati et al. [128] (para el TRP), pero generaliza su solución para incorporar la consideración de beneficios que disminuyen con el tiempo. El algoritmo resuelve el problema del TRPP en tiempo polinómico.

Un año más tarde, en 2008, se publica "Profit-based latency problems on the line" [129], escrito por Sofie Coene y Frits C.R. Spieksma. En este problema, se considera una línea como espacio métrico y se asocian beneficios a cada cliente

que es servido. El objetivo es maximizar la diferencia entre los beneficios obtenidos y la latencia incurrida al servir a los clientes.

El modelo presenta dos restricciones principales. La elección de clientes debe priorizar aquellos que maximizan la ganancia neta, considerando tanto el beneficio directo de servirlos como el tiempo necesario para alcanzarlos. Y la segunda restricción consiste en que todos los servidores deben comenzar en un punto de origen predefinido y no se requiere que atiendan a todos los clientes. Solo se seleccionarán aquellos que contribuyan positivamente al beneficio total.

El artículo propone un algoritmo de programación dinámica (DP) para resolver el TRPP en tiempo polinomial. El DP se basa en identificar estados que representan las posiciones actuales del servidor y la cantidad de clientes restantes por servir, maximizando así la ganancia acumulada en cada estado. Además, se demuestra que el TRPP con múltiples servidores idénticos se puede resolver en tiempo polinomial mediante una extensión del DP.

El algoritmo de programación dinámica presentado resuelve el TRPP en tiempo $O(n^3)$, y se extiende a variantes con servidores múltiples idénticos en $O(n^4)$. Los autores prueban que cuando los servidores son no idénticos y se introducen fechas de liberación, el problema se vuelve fuertemente $\mathcal{NP}$-Difícil. Estos resultados establecen límites claros sobre la solvencia del problema bajo distintas configuraciones.

Cinco años más tarde, en 2013, Thijs Dewilde, Dirk Cattrysse, Sofie Coene, Frits C.R. Spieksma y Pieter Vansteenwegen escriben “Heuristics for the traveling repairman problem with profits” [130]. En este artículo, el problema se centra en maximizar la ganancia total obtenida al visitar nodos en un grafo, donde cada nodo tiene un beneficio asociado que disminuye con el tiempo de llegada del reparador.

El problema está sujeto a un conjunto de restricciones clave. No es obligatorio visitar todos los nodos del grafo, cada nodo puede ser visitado como máximo una vez, no se requiere que el reparador regrese al depósito después de completar el recorrido, la distancia entre nodos debe cumplir la desigualdad triangular, y se asume que el tiempo dedicado a servir en cada nodo es despreciable en comparación con el tiempo de viaje.

El artículo emplea un algoritmo de búsqueda tabú con múltiples vecindades para resolver el TRPP. El algoritmo comienza con una fase de construcción basada en inserciones que genera una solución inicial, y luego se mejora utilizando movimientos como inserción, eliminación y reemplazo de nodos, así como el ajuste de la secuencia de visita mediante técnicas como 2-opt y Or-opt. El enfoque incluye listas tabú para evitar ciclos y se ajusta dinámicamente según el progreso de la solución.

Los experimentos mostraron que el algoritmo de búsqueda tabú es capaz de encontrar soluciones óptimas para instancias pequeñas (hasta 20 nodos) en tiempos razonables. Para instancias más grandes, el algoritmo proporcionó soluciones cercanas al óptimo, con un margen de error del 3 % al 9 % en comparación con un límite superior calculado. Además, el algoritmo demostró ser competitivo en términos de tiempo de cómputo y calidad de las soluciones frente a otros enfoques exactos y heurísticos.

Transcurridos cuatro años, en 2017, aparece "A GRASP with iterated local search for the traveling repairman problem with profits" [131], escrito por Mustafa Avci y Mualla Gonca Avci, donde se presenta un enfoque metaheurístico para resolver el TRPP. La función objetivo de este problema busca maximizar la ganancia neta del recorrido considerando un subconjunto de nodos respecto al tiempo que se invierte en alcanzarlos.

El problema está sujeto a tres restricciones principales. En primer lugar, cada nodo puede ser visitado únicamente una vez, pero no es obligatorio incluir todos los nodos en el recorrido. En segundo lugar, la secuencia de visitas debe estar diseñada para minimizar la latencia acumulada.. Por último, la ruta debe comenzar en un nodo específico, pero no es necesario que forme un ciclo cerrado.

El enfoque propuesto es el GRASP-ILS, que consiste en dos fases principales:

- Construcción de solución inicial GRASP: Utiliza un procedimiento greedy con aleatorización controlada para generar soluciones iniciales diversificadas.
- Mejora de la solución Iterative Local Search (ILS): Aplica una búsqueda local iterada, integrada con un mecanismo de perturbación adaptativo y una búsqueda de vecindad variable descendente mejorada con tabú, para evitar ciclos y mejorar las soluciones encontradas. Este enfoque híbrido permite explorar y mejorar efectivamente el espacio de soluciones.

El GRASP-ILS fue probado en diversas instancias del TRPP con hasta 500 nodos. Los resultados mostraron que este enfoque produjo soluciones de alta calidad en tiempos razonables, mejorando los mejores resultados conocidos en 46 instancias y alcanzando las mejores soluciones conocidas en las restantes. El algoritmo demostró ser eficiente para manejar instancias de gran tamaño, destacándose por su adaptabilidad y simplicidad.

En 2018, "The risk-averse traveling repairman problem with profits" [132], escrito por P. Beraldi, M. E. Bruni, D. Laganà y R. Musmanno, presenta una variante estocástica del TRPP en la que los tiempos de viaje son inciertos. Esta versión del problema incorpora una perspectiva adversa al riesgo, utilizando restricciones de probabilidad (*chance constraints*) para modelar la incertidumbre en los tiempos de viaje y maximizar las ganancias con un nivel de confiabilidad predeterminado.

El problema incluye una restricción de probabilidad que vincula la ganancia total de la ruta con un nivel de confiabilidad predefinido. En particular, esta restricción se basa en la modelación de la incertidumbre de los tiempos de viaje mediante distribuciones elípticas, lo que permite capturar tanto la correlación como la variabilidad inherente a estas incertidumbres. Esta restricción asegura que las decisiones tomadas sean robustas frente a las fluctuaciones en los tiempos de viaje, garantizando que el rendimiento del sistema cumpla con los requisitos de confiabilidad definidos.

Los autores desarrollan un modelo no lineal entero y una heurística de búsqueda con haz (*beam search*) para resolver el problema. Esta técnica construye un árbol de búsqueda en el que en cada nivel solo se exploran los nodos más prometedores, limitados por un ancho de haz predefinido. La evaluación de nodos se realiza utilizando funciones heurísticas basadas en evaluaciones un paso adelante

(*one-step look-ahead*) y relajaciones del modelo no lineal.

Los resultados experimentales, realizados en instancias de 10 y 20 nodos, mostraron que el enfoque de búsqueda con haz es efectivo, especialmente en configuraciones de menor tamaño, logrando una mejora significativa en los tiempos de cómputo comparado con soluciones exactas. Para niveles de riesgo bajos, se observaron soluciones más conservadoras pero con menores variaciones en los resultados, mientras que, para niveles más altos de riesgo, las soluciones ofrecieron mayores ganancias esperadas, a costa de mayor variabilidad.

Al cabo de un año, en 2019, Yongliang Lu, Jin-Kao Hao y Qinghua Wu presentan "Hybrid evolutionary search for the traveling repairman problem with profits" [133]. En este artículo se expone un algoritmo de búsqueda evolutiva híbrida (HESA) para resolver el TRPP. Sujeto a tres restricciones principales. Cada nodo puede ser visitado como máximo una vez, y no es obligatorio incluir todos los nodos en el recorrido. El recorrido debe comenzar en un nodo inicial, aunque no es necesario que el recorrido forme un ciclo cerrado. Finalmente, la latencia acumulada, determinada por el orden de visita de los nodos.

El HESA combina un método de construcción voraz aleatorizado para generar la población inicial de soluciones, con una búsqueda de vecindad variable (VNS) con varias vecindades (inserción, intercambio y operadores 2-opt) y, dos tipos de operadores de cruce (de un punto y de dos puntos) se aplican de manera adaptativa para generar soluciones nuevas a partir de la combinación de soluciones padres.

El HESA fue evaluado en 120 instancias de referencia del TRPP, demostrando un rendimiento competitivo frente a otros algoritmos de vanguardia. Logró mejorar los mejores resultados conocidos en 39 instancias y igualó los mejores resultados en las restantes. El algoritmo demostró ser particularmente efectivo en instancias grandes (de 200 y 500 nodos), mostrando una mejora significativa en la calidad de las soluciones.

El mismo año, 2019, "A bi-objective study of the minimum latency problem" [134], escrito por N. A. Arellano-Arriaga, J. Molina, S. E. Schaeffer, A. M. Álvarez-Socarrás e I. A. Martínez-Salazar, aborda el problema de latencia mínima desde una perspectiva biobjetivo. Se analiza un problema de logística en el que un solo vehículo debe servir a un conjunto de clientes minimizando simultáneamente el tiempo total de viaje y la latencia total (tiempo de espera) de los clientes. El modelo incluye las siguientes restricciones. En primer lugar, cada cliente debe ser visitado exactamente una vez. En segundo lugar, la ruta debe comenzar y terminar en el depósito. Finalmente, se incluye una restricción de no subtours.

El estudio presenta dos algoritmos heurísticos:

- SMSA (Strategic Memetic Search Algorithm): un algoritmo memético que combina la búsqueda local con un enfoque de selección y cruce para generar y mejorar soluciones. Utiliza operadores de cruce y mutación basados en el método NSGA-II para promover la diversidad y calidad de las soluciones.
- EiLS (Evolutionary Algorithm with Intelligent Local Search): un algoritmo evolutivo que incorpora una búsqueda local inteligente para explorar diferentes

vecindades y mejorar las soluciones. Este enfoque adapta dinámicamente las exploraciones según el rendimiento de cada vecindad, optimizando las soluciones según una función de compromiso entre los objetivos.

Las pruebas computacionales mostraron que ambos algoritmos son efectivos para aproximar el frente de Pareto en instancias de hasta 256 clientes. EiLS superó a SMSA en términos de volumen hipercúbico y número de puntos en el frente de Pareto. Además, EiLS demostró ser más eficiente en tiempo de cómputo, proporcionando soluciones de alta calidad de manera más rápida en comparación con SMSA.

Al año siguiente, en 2020, Jun Pei, Nenad Mladenović, Dragan Urošević, Jack Brimberg y Xinbao Liu publican "Solving the traveling repairman problem with profits: A Novel variable neighborhood search approach" [31], donde presentan un enfoque novedoso para resolver el TRPP utilizando una búsqueda de vecindad variable generalizada GVNS. El problema está sujeto a dos restricciones fundamentales. En primer lugar, no es obligatorio visitar todos los nodos. En segundo lugar, el orden de los nodos visitados debe minimizar la latencia acumulada.

El enfoque propuesto es la GVNS. Este algoritmo emplea varias vecindades para optimizar la solución:

- Se utilizan tres estructuras de vecindad principales: inserción hacia adelante, inserción hacia atrás y el clásico 2-opt.
- Se aplican técnicas de intercambio y eliminación de nodos para modificar el conjunto de clientes visitados y su secuencia.
- El algoritmo realiza búsquedas iterativas cambiando de vecindad en función de las mejoras logradas en cada iteración.

Las pruebas se realizaron en instancias de referencia con hasta 500 nodos. La GVNS superó a las heurísticas previas, como la búsqueda tabú y GRASP-ILS, logrando encontrar nuevas soluciones óptimas en 40 de las 60 instancias evaluadas. Además, el algoritmo demostró una eficiencia computacional superior, con tiempos de ejecución significativamente más cortos y mejor calidad de soluciones en comparación con métodos existentes.

A su vez, en 2020, se publica "The bi-objective minimum latency problem with profit collection and uncertain travel times" [135], escrito por Maria Elena Bruni, Sara Khodaparasti y Samuel Nucamendi-Guillén, en el que se presenta un modelo biobjetivo para el problema de latencia mínima con recolección de beneficios y tiempos de viaje inciertos. Este problema combina la minimización de la latencia total en las rutas y la maximización de las ganancias, considerando clientes obligatorios visitados una sola vez y opcionales que pueden no ser visitados. Se tiene una flota fija de vehículos y el número de rutas debe ser igual al número de vehículos.

El enfoque se basa en un algoritmo de búsqueda local iterada para encontrar soluciones eficientes en un contexto de incertidumbre en los tiempos de viaje, utilizando medidas de riesgo coherentes como el Valor en Riesgo Condicional (CVaR):

- Fase constructiva: construye rutas iniciales considerando los clientes obliga-

torios y opcionales.

- Fase de mejora: aplica movimientos locales como intercambio de nodos, inserción y eliminación, y el operador 2-opt para optimizar la solución.
- Fase de perturbación: introduce cambios aleatorios en la solución actual para escapar de óptimos locales y mejorar la diversidad del conjunto de soluciones.

El algoritmo fue probado en instancias de problemas modificados de conjuntos de datos conocidos. Los resultados mostraron que el enfoque propuesto es efectivo para encontrar soluciones de alta calidad en tiempos computacionales razonables, particularmente en instancias pequeñas y medianas. El uso de múltiples niveles de aversión al riesgo permitió generar diferentes frentes de Pareto, ofreciendo opciones según las preferencias del tomador de decisiones.

Dos años después, en 2022, se publica "Intensification-driven local search for the traveling repairman problem with profits" [136], escrito por Jintong Ren, Jin-Kao Hao, Feng Wu y Zhang-Hua Fu. En este artículo se presenta un algoritmo de búsqueda local intensificada (IDLS-TRPP) para resolver el TRPP. El algoritmo propuesto se distingue por su mecanismo de intensificación, que explora exhaustivamente áreas cercanas a óptimos locales de alta calidad. Este enfoque se compara con otros métodos de búsqueda local, destacando su rendimiento superior en instancias grandes y complejas del problema.

El problema está sujeto a tres restricciones principales. En primer lugar, el recorrido debe comenzar en el depósito (nodo inicial). En segundo lugar, no es obligatorio visitar todos los nodos. Finalmente, el orden de los nodos visitados debe respetar la estructura del grafo.

El algoritmo propuesto, IDLS-TRPP, combina:

- Búsqueda de vecindad variable extendida (EVNS): emplea varias vecindades (Swap, Insert, 2-opt, Or-opt) y una nueva vecindad basada en muestreo de intercambios (K-exchange) para mejorar las soluciones locales.
- Fase de perturbación: introduce ligeros cambios en las soluciones para evitar estancamiento en óptimos locales, permitiendo una exploración más amplia del espacio de soluciones. El IDLS-TRPP se basa en una estrategia de intensificación que reexplora de manera sistemática las áreas alrededor de las soluciones óptimas locales ya encontradas, mejorando continuamente el valor objetivo.

El algoritmo IDLS-TRPP fue evaluado en 140 instancias de referencia del TRPP, mostrando un rendimiento destacado. Logró mejorar los límites inferiores conocidos en 36 instancias y empatar con los mejores resultados en 95 instancias adicionales. En comparación con otros enfoques de búsqueda, como el GVNS-TRPP, el IDLS-TRPP demostró una mayor eficiencia en instancias de gran tamaño, optimizando tanto el tiempo de cómputo como la calidad de las soluciones.

Capítulo 3

Algoritmos heurísticos para la resolución del Mo-TSRPP

La resolución heurística de un problema de optimización tiene como objetivo obtener soluciones de alta calidad en un tiempo de cómputo reducido. Este tipo de técnicas, junto con las metaheurísticas, se clasifican dentro de las denominadas técnicas aproximadas. En este capítulo se propone la combinación de las metaheurísticas GRASP (Greedy Randomized Adaptive Search Procedure) y VND (Variable Neighborhood Descent) para la resolución del "problema multiobjetivo del comerciante reparador con beneficios". Para ello, se desarrolla una heurística constructiva y diferentes heurísticas de mejoras, que son integradas en el esquema de la metaheurística anteriormente mencionada.

3.1. Introducción a las técnicas de resolución aproximada

Las técnicas de resolución aproximada constituyen una alternativa a las técnicas de resolución exacta para la obtención de soluciones a problemas de optimización. Esta alternativa se basa en encontrar soluciones de alta calidad, que en ocasiones pueden ser incluso soluciones óptimas pero que no se pueden garantizar, dentro de un tiempo de cómputo limitado. Los algoritmos diseñados para generar soluciones cercanas al óptimo se denominan métodos de aproximación.

El principal inconveniente de las técnicas exactas es que, aunque son capaces de proporcionar soluciones óptimas, generalmente requieren un tiempo de cómputo elevado cuando el tamaño del problema es considerable. Cuando se resuelven problemas de optimización, el tiempo de resolución que requiere un algoritmo aproximado es menor al que utiliza un algoritmo exacto.

Una posible clasificación de los distintos métodos de aproximación fue propuesta en [1], donde se dividen en dos grandes grupos: algoritmos aproximados y algoritmos heurísticos. Los algoritmos aproximados permiten obtener soluciones cuya calidad es mensurable, ya que proporcionan una cota respecto al valor de la

solución óptima, es decir, ofrecen una garantía teórica de la proximidad al óptimo [137]. Por otro lado, los algoritmos heurísticos permiten obtener soluciones de alta calidad en tiempos de cómputo razonables, pero no permiten certificar si dichas soluciones son óptimas, ni la calidad precisa de estas. Los algoritmos heurísticos pueden subdividirse en "heurísticas específicas" y "metaheurísticas" [1]. En la Figura 3.1 se muestra, de manera esquemática, la clasificación de los distintos algoritmos de aproximación anteriormente mencionados.

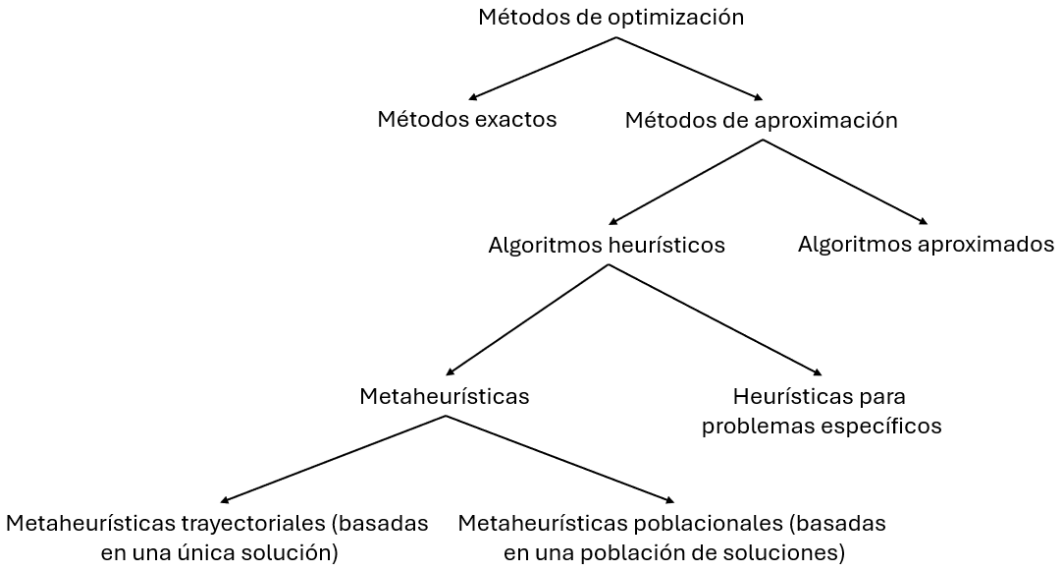


Figura 3.1: Clasificación de los distintos algoritmos de aproximación (Adaptado de [1])

3.1.1. Algoritmos heurísticos

Como se ha mencionado anteriormente, los algoritmos heurísticos son métodos aproximados que permiten encontrar soluciones de alta calidad para problemas de optimización en tiempos de cómputo reducidos. Sin embargo, tienen la desventaja de no poder garantizar que la solución encontrada sea óptima ni determinar cuán cerca está de la solución óptima. En otras palabras, sacrifican la certeza de obtener soluciones óptimas a cambio de una notable reducción en el tiempo de cómputo [138]. Los algoritmos heurísticos se pueden dividir en heurísticas específicas y metaheurísticas:

- Heurísticas específicas: Las heurísticas específicas son métodos creados y ajustados para abordar problemas particulares o situaciones específicas. Generalmente, son conceptos simples que se enfocan en la creación y mejora de soluciones. Sin embargo, su principal limitación es que tienden a quedarse atrapadas en óptimos locales. Dos ejemplos bien conocidos de heurísticas específicas son las heurísticas constructivas y las heurísticas de búsqueda local:
    - Constructivas: Su función es generar soluciones, comenzando generalmente desde una solución vacía a la que se le van añadiendo elementos

hasta encontrar una solución factible.

- Búsqueda local: Parte de una solución factible que la mejora progresivamente. Para ello examina su vecindad y selecciona el primer movimiento que produce una mejora en la solución actual (first improvement) o todos los posibles movimientos seleccionando el mejor movimiento de todos los posibles, es decir aquel que produzca un incremento (en el caso de maximización) más elevado en la función objetivo (best improvement). [4]

En la sección 1.1.5 se revisan algunas de las estrategias constructivas y de optimización más reconocidas.

- Metaheurísticas: Las metaheurísticas son algoritmos versátiles que pueden aplicarse a cualquier problema de optimización combinatoria. Funcionan como procedimientos de alto nivel que guían a otras heurísticas en la resolución de estos problemas. Actualmente, existe una gran variedad de técnicas metaheurísticas, y clasificarlas de manera efectiva sin solapamientos es un desafío. Sin embargo, es posible agruparlas según ciertos criterios específicos, como si parten de una o varias soluciones, si utilizan búsqueda local, si están inspiradas en procesos biológicos, si emplean múltiples estructuras de vecindad, si usan memoria, entre otros. Diversas clasificaciones pueden encontrarse en [4, 139, 140, 141, 142].

  Una posible clasificación, basada en la cantidad de soluciones iniciales que se manejan, podría dividir las metaheurísticas más destacadas en dos categorías: trayectoriales y poblacionales, como se ilustra en la Figura 3.2. A continuación, se detalla cada una de estas categorías:

  - Trayectoriales: Son un tipo de algoritmos de optimización que exploran el espacio de soluciones moviéndose de una solución a otra individualmente, siguiendo una trayectoria determinada por reglas específicas de búsqueda. Muchas de estas metaheurísticas son ampliaciones de métodos de búsqueda local, que por sí solos no son efectivos. Entre las metaheurísticas trayectoriales más reconocidas se encuentran: Búsqueda Voraz Aleatorizada Adaptativa (*Greedy Randomized Adaptive Search Procedure*, GRASP), Búsqueda Tabú (*Tabu Search*, TS) [13, 143], Búsqueda de Vecindad Variable (*Variable Neighbourhood Search*, VNS) [44], Recocido Simulado (*Simulated Annealing*, SA) [144], entre otras.

  - Poblacionales: Las "metaheurísticas poblacionales" reciben su nombre porque, en cada iteración del proceso de búsqueda, utilizan un conjunto de soluciones (población) en lugar de una única solución. El éxito final depende en gran medida de cómo se gestione esta población. Entre las metaheurísticas poblacionales más conocidas se encuentran: Algoritmos Genéticos (*Genetic Algorithm*, GA) [145], Algoritmos Meméticos (*Memetic Algorithm*, MA) [146], Búsqueda Dispersa (*Scatter Search*, SS) [143, 147], y Reencadenamiento de Trayectorias (*Path Relinking*, PR) [148, 149], entre otras.

  Las diferencias técnicas entre las metaheurísticas trayectoriales y poblacionales mencionadas anteriormente son solo una parte de las metaheurísticas

disponibles, las cuales se pueden ver de forma esquemática en la Figura 3.2. Se puede encontrar una descripción más detallada de estas y otras metaheurísticas en [140, 150, 151]. Para una explicación de las metaheurísticas VND y GRASP, utilizadas en esta Tesis Doctoral para resolver el Mo-TSRPP, consulte la Sección 1.4.1.

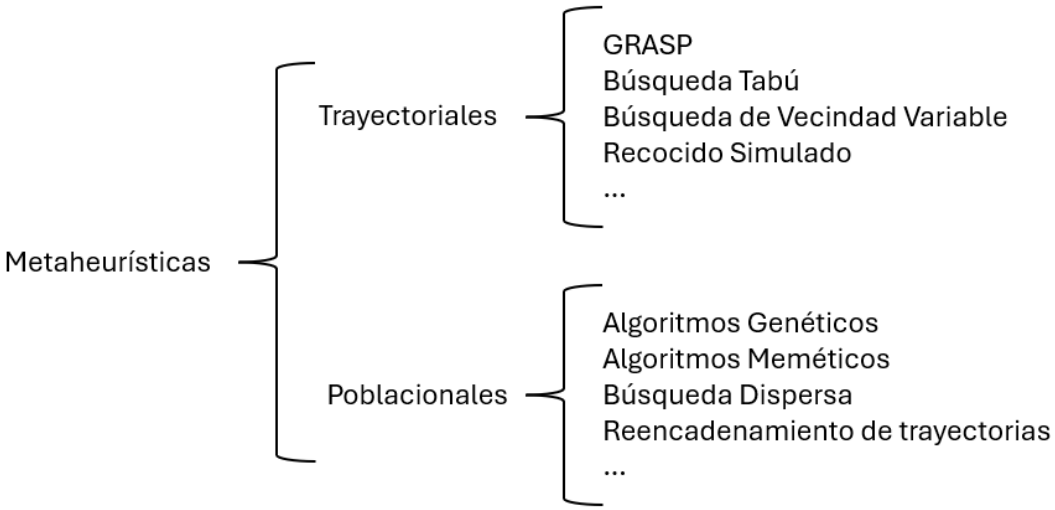


Figura 3.2: Clasificación de distintas metaheurísticas

3.2. Propuesta para la resolución del Mo-TSRPP

En esta Tesis Doctoral se hace uso de una metaheurística trayectorial que combina GRASP dentro de VND para la parte constructiva, y se complementa con tres vecindades, 2-intercambio, inserción y 2-opt. A continucación, se explica detalladamente cada parte.

3.2.1. Algoritmo constructivo

En esta sección se propone una heurística constructiva para la obtención de soluciones al Mo-TSRPP. Dicha heurística están basadas en la metodología GRASP [45, 46, 47].

El objetivo inicial del algoritmo constructivo propuesto (denominado GRA) es proporcionar soluciones iniciales factibles al Mo-TSRPP, de modo que éstas tengan la mejor calidad posible. Sin embargo, como se discutirá más adelante, también es importante la diversidad de las soluciones generadas, ya que el objetivo final del algoritmo constructivo es que pueda ser incorporado en un esquema de VND.

Al diseñar un algoritmo constructivo, es crucial considerar el tipo de solución del problema en cuestión. En este contexto, es importante señalar que, debido a la estructura de una solución para el Mo-TSRPP, cualquier ordenación de los nodos

del grafo de entrada y cualquier tamaño constituye una solución factible al problema. A continuación, se revisa la heurística constructiva propuesta.

En este trabajo, se propone un método constructivo basado en la fase constructiva del GRASP que incluye una fase de diversificación, a saber, la construcción Adaptativa Aleatoria Voraz (*Greedy Randomized Adaptive*, GRA), útil en un contexto multiobjetivo como el de esta investigación [42, 43]. En la Sección 1.4.1 se profundiza en la metaheurística GRASP.

A modo recordatorio, el Mo-TSRPP puede enunciarse formalmente de la siguiente manera: Sea $G = (V, A)$ un grafo completo no dirigido, donde $V = \{1, \ldots, n\}$ es el conjunto de vértices o nodos y $A = \{(i, j) : i, j \in V, i \neq j\}$ es el conjunto de aristas. El vértice 1 representa el vértice donde el vendedor comienza y termina la ruta y $N = \{2, \ldots, n\}$ es el conjunto de clientes que requieren una visita/servicio. Cada arista $(i, j) \in A$ tiene asociado un coste finito $c_{ij} \geq 0$ (este coste podría medirse como una longitud o incluso como un tiempo). Además, cada cliente $i \in N$ paga un precio $p_i > 0$ al recibir el servicio, o lo que es lo mismo, se cobra un beneficio cada vez que se visita/sirve un vértice (nótese que el primer vértice tiene asociado un beneficio nulo, $p_1 = 0$). El Mo-TSRPP busca planificar una ruta visitando un subconjunto de clientes, $S \subset N$, para optimizar los siguientes objetivos: el coste total del viaje, la latencia total y el beneficio total, aunque indirectamente el número de clientes servidos también se tiene en cuenta.

La elección de los nodos en el algoritmo GRA se basa en un criterio voraz que guía la construcción de la solución paso a paso, seleccionando en cada iteración los elementos que parecen más prometedores según una función voraz específica. Este criterio voraz no es fijo, sino que debe adaptarse al problema concreto que se está resolviendo, ya que de su correcta definición depende en gran medida la calidad de las soluciones generadas. Un criterio mal diseñado puede conducir a soluciones pobres o altamente subóptimas, mientras que un buen criterio permite construir soluciones iniciales de alta calidad que pueden ser mejoradas eficazmente en la fase de búsqueda local. Por tanto, el diseño de la función voraz es una etapa crítica en la implementación de GRA y requiere de conocimiento experto del dominio del problema.

El criterio voraz seguido en esta propuesta se basa en una función voraz que combina tanto el coste como el beneficio de los nodos candidatos, representada $g(S, v) = \min_{u \in CL} c_{vu}/p_u$. La función evalúa la conveniencia de añadir un nodo u a la solución parcial S. Para un nodo v que ya está en la solución (el último añadido), se evalúan todos los nodos candidatos restantes $u \in CL$ es decir, aquellos que aún no han sido visitados, y se calcula, para cada uno de ellos, el cociente entre el coste de ir desde v hasta u (c_{vu}) y el beneficio asociado a visitar el nodo u (p_u). El valor resultante de $g(S, v)$ es el mínimo de estos cocientes, es decir $g(S, v) = \min_{u \in CL} c_{vu}/p_u$. Esta expresión mide la eficiencia relativa del mejor nodo candidato en términos de coste por unidad de beneficio. Cuanto menor es el valor de c_{vu}/p_u, más atractivo es el nodo u: implica que se obtiene un alto beneficio por un bajo coste adicional. Por tanto, minimizar esta relación permite construir rutas que logran un buen equilibrio entre recorrer poca distancia (o incurrir en poco coste) y obtener el mayor beneficio posible.

La propuesta parte de una solución vacía, en la que sólo se incluye el nodo inicial, $S = \{1\}$, con lo que la lista de nodos candidatos se inicializa como $CL = V \setminus S$. El método voraz selecciona el nodo $v \in CL$ con el mayor valor de c_{1v}/p_v para todos los $v \in CL$ que se añadirá a la solución. Entonces, una vez que el nodo v ha sido añadido a la solución, $S = \{1, v\}$ y eliminado de CL, el método itera hasta que todos los nodos han sido asignados a la solución en construcción. En cada iteración de la fase de construcción, todos los nodos $u \in CL$ se evalúan considerando la función voraz $g(S, v) = \min_{u \in CL} c_{vu}/p_u$. Como ya se ha explicado antes, esta función voraz evalúa el valor de la función objetivo si el nodo u se añadiera a la solución en construcción, intentando alcanzar un compromiso entre más de una función objetivo: el coste (distancia o tiempo) y el beneficio. De esta forma, no sólo se consideran buenos valores para un objetivo sino para más de uno simultáneamente, permitiendo un compromiso entre ellos.

Si se considera la aplicación directa de la función voraz descrita en el párrafo anterior, el algoritmo resultante sería completamente voraz, y no tendría ningún tipo de diversidad, aspecto crucial al desarrollar algoritmos heurísticos. Con el fin de incluir diversidad en la construcción se utiliza un esquema similar a la fase de construcción de la metaheurística GRASP, que se expone a continuación, llamada GRA. Se tiene que tener en cuenta que la aplicación de una construcción GRA es similar con la única particularidad de que se necesita para mantener el mejor y el peor valor de la función voraz, g_{min} y g_{max}, respectivamente. Estos valores se consideran, ya que es necesario para calcular un umbral, $\mu \leftarrow g_{min} + \alpha(g_{max} - g_{min})$, para crear una lista de candidatos restringidos, *RCL*. La *RCL* contiene los nodos candidatos más prometedores, es decir, los que tienen una mayor probabilidad de contribuir positivamente a la calidad de la solución final, que se añadirán a la solución en construcción y no sólo el más prometedor como se haría en las construcciones voraces o todos los nodos como se haría en las construcciones aleatorias. Un nodo se considera prometedor y entra en la *RCL* si su valor de $g(S, u) \leq \mu$, es decir, si está lo suficientemente cerca del mejor valor observado. Finalmente, el método selecciona aleatoriamente un nodo u de la *RCL*, y lo añade a la solución S, luego, actualiza el *CL* eliminando el nodo seleccionado. El método finaliza devolviendo la solución construida cuando no se pueden añadir más nodos en la solución en construcción. El procedimiento constructivo se implementa siguiendo los detalles dados en [152] para reducir el esfuerzo computacional. Para reducir el esfuerzo computacional, la *RCL* no se crea explicitamente, así que la selección del siguiente elemento se lleva a cabo en $O(1)$.

Es importante destacar que en cada iteración de la fase de construcción, se está generando una solución eficiente ya que se está añadiendo un nuevo nodo a la solución. Por tanto, cada vez que se añada un nodo a la solución en construcción, se obtendrá una nueva solución eficiente y, por tanto, se incluirá en la aproximación inicial del Frente de Pareto, $\hat{PF}$ teniendo un total de $|\hat{PF}| = n - 1$ soluciones eficientes ya que este es el número total de clientes.

El Algoritmo 1 muestra el pseudocódigo del constructivo GRA utilizado en este trabajo, que recibe como parámetros de entrada el grafo $G = (V, E)$, y el parámetro α que controla la voracidad del método, lo que repercute en el tamaño de la lista

restringida de candidatos. El algoritmo GRA basa su rendimiento en un delicado equilibrio entre explotación (voracidad) y exploración (aleatoriedad). Este equilibrio se regula mediante el parámetro $\alpha \in [0, 1]$. $\alpha = 0$ y $\alpha = 1$ hacen que la construcción sea completamente voraz y aleatoria, respectivamente. Esto es debido a que $\mu \leftarrow g_{min} + \alpha(g_{max} - g_{min})$, si $\alpha = 0$, $\mu = g_{min}$ y sólo aquellos elementos cuyo valor de la función voraz sea igual a g_{min} serán considerados en la *RCL*, resultando en un algoritmo totalmente voraz. Por el contrario, si $\alpha = 1$, μ toma el valor de g_{max}, y todos los elementos son incluidos en la *RCL*, volviéndose un proceso completamente aleatorio. Cuanto más grande sea el valor de α, más aleatorio será el algoritmo, y más grande será la *RCL*.

Algoritmo 1 Constructivo GRA ($G = (V, E), \alpha$)

1: $\hat{PF} \leftarrow \emptyset$
2: $S \leftarrow u$
3: $CL \leftarrow N \setminus \{u\}$
4: **while** $|S| \leq n$ **do**
5: $\quad g_{min} \leftarrow min_{v \in CL}\, g(S, v)$
6: $\quad g_{max} \leftarrow max_{v \in CL}\, g(S, v)$
7: $\quad \mu \leftarrow g_{min} + \alpha(g_{max} - g_{min})$
8: $\quad RCL \leftarrow \{v \in CL : g(S, v) \leq \mu\}$
9: $\quad v' \leftarrow SelectRandom(RCL)$
10: $\quad S \leftarrow S \cup \{v'\}$
11: $\quad CL \leftarrow CL \setminus \{v'\}$
12: $\quad \hat{PF} \leftarrow S$
13: **end while**
14: **return** $\hat{PF}$

Cabe mencionar que al tratarse de un problema de optimización multiobjetivo, la salida del procedimiento constructivo es un conjunto de soluciones eficientes que serán la aproximación inicial de $\hat{PF}$ (paso 14). El algoritmo comienza con una aproximación inicial vacía del Frente de Pareto donde no se incluyen soluciones eficientes, $\hat{PF}$ (paso 1). A continuación, se fuerza a que el nodo inicial (nodo u) esté en la solución, S, ya que el viajero iniciará la ruta en ese nodo (paso 2). El método seleccionará nuevos nodos hasta que todos los nodos estén incluidos (pasos 4-13). En cada iteracción se calculan los valores de la función voraz g_{min} y de g_{max} para guiar la construcción (pasos 5-6) y así obtener el valor del umbral μ (paso 7). El nodo v' se añade a la solución S tras ser seleccionado aleatoriamente del *RCL* (paso 9) previamente calculado (paso 8). A continuación, el nodo seleccionado se incluye en la solución S (paso 10), se elimina de la lista de candidatos (paso 11) y también se incluye la solución en $\hat{PF}$.

3.2.2. Algoritmos de búsqueda local

Un método de mejora es un proceso que, a partir de una solución inicial a un problema de optimización, busca encontrar soluciones de mayor calidad en su vecindad. La búsqueda local es un tipo de procedimiento heurístico de mejora en el cual,

en cada iteración, se reemplaza la solución inicial por una solución de mejor calidad dentro de su vecindad, convirtiéndose esta última en la nueva solución inicial para la siguiente iteración del algoritmo. Cuando la búsqueda local no puede encontrar una solución de mejor calidad en la vecindad definida, se considera que ha encontrado un óptimo local y el proceso termina. El principal problema de los procedimientos de búsqueda local es que no pueden escapar de los óptimos locales.

Es habitual utilizar búsquedas locales junto con algoritmos constructivos, de modo que la búsqueda local funciona como una fase posterior al procedimiento constructivo. Primero, el algoritmo constructivo genera soluciones que servirán como punto de partida para la búsqueda local. Las soluciones obtenidas mediante búsquedas locales suelen ser de mayor calidad que las generadas por heurísticas constructivas, aunque también suelen requerir más tiempo de cómputo para lograrlas.

Recuérdese que se está resolviendo un problema multiobjetivo y no se es capaz de encontrar una mejor solución en todos los objetivos. Por ello, se ha considerado que se alcanza una mejora si la nueva solución domina estrictamente a la solución eficiente del mismo tamaño perteneciente al Frente de Pareto (sección 1.1.1 para más información sobre dominancia). Se ha optado por esta estrategia debido a que evita un incremento innecesario del número de soluciones eficientes en el Frente de Pareto, quedando una solución por cada tamaño de nodos en la solución. Además, evita el problema de comparar la solución actual con todas las soluciones que pertenecen al Frente de Pareto, ya que ésta es la razón principal por la que las propuestas anteriores son más lentas.

En esta Tesis Doctoral se proponen tres estructuras de vecindad para el Mo-TSRPP, dos basados en movimientos de intercambio de posiciones dentro de la solución, y el tercero en intercambio de nodos entre los nodos de la solución y nodos fuera de la solución.

2-intercambio

Esta búsqueda local se compone del operador *2-intercambio* y de la vecindad $\mathcal{N}_1$. El operador *2-intercambio* (*2-exchange* o *swap*) selecciona dos nodos pertenecientes a la ruta y los intercambia, lo que genera una nueva solución al reordenar los nodos de la ruta. Es uno de los operadores más sencillos y ampliamente utilizados en problemas de optimización combinatoria como el TSP, debido a su simplicidad y bajo costo computacional. El operador usa first improvement.

La vecindad $\mathcal{N}_1$ está conformada por todas las soluciones que se pueden alcanzar aplicando el operador *2-intercambio*. La vecindad generada mediante el *2-intercambio* suele explorar una gran cantidad de soluciones y permite escapes iniciales de óptimos locales.

$\mathcal{N}_1$ está compuesta por todas las posibles soluciones que se obtienen al intercambiar cualquier par de nodos en la ruta. La exploración de esta vecindad permite encontrar soluciones cercanas que pueden ser mejores al mover dos nodos.

Complejidad:

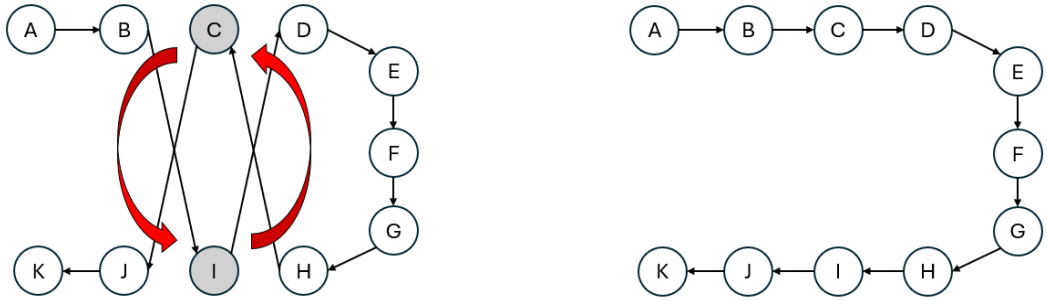


Figura 3.3: Ejemplo del operador 2-intercambio

- La búsqueda local tiene una complejidad de $O(n^2)$, ya que para cada par de nodos en la solución se realiza un intercambio y se evalúa la calidad de la solución resultante.
- El operador tiene una complejidad de $O(1)$, puesto que sólo realiza un intercambio de posición de nodos.
- La función de evaluación tiene una complejidad de $O(1)$, ya que no se requiere revaluar toda la ruta, sino sólo las posiciones intercambiadas, y los nodos anterior y siguiente.

Algoritmo 2 2-intercambio (S)

```
1: for i ∈ S do
2:     for j ∈ S do
3:         S' ← swap_nodes(S, i, j)
4:         if S' is better than S then
5:             S ← S'
6:             i = 0
7:         end if
8:     end for
9: end for return S
```

El algoritmo 2-intercambio parte de una solución inicial S y busca mejorarla iterativamente mediante el intercambio de pares de nodos. Para ello, recorre todos los pares de nodos posibles en S (bucles anidados, pasos 1-9) y genera una nueva solución S' mediante el intercambio de las posiciones de los nodos i y j (paso 3). A continuación, se evalúa si la nueva solución S' mejora respecto a la actual S (paso 4), tal y como se explica en la Sección 1.1.1. Si es así, la solución se actualiza (paso 5) y se reinicia el índice i a cero (paso 6) para reiniciar el proceso de búsqueda con la nueva configuración, permitiendo así una exploración más intensiva de mejoras locales. El algoritmo continúa este procedimiento hasta haber evaluado todas las combinaciones posibles de nodos, y finalmente devuelve la mejor solución encontrada (paso 9).

Tras realizar un movimiento, el método de evaluación no necesita revaluar la solución entera, tan sólo tiene que tener en cuenta los nodos intercambiados. Cada FO tiene su fórmula para revaluarse. En esta búsqueda local, tan sólo hay que re-

valuar el coste de recorrido y la latencia, puesto que el número de nodos no varía y al ser un movimiento intraruta los nodos no cambian, sólo su posición, por lo que el beneficio es el mismo.

Coste de recorrido:

$$\begin{aligned}\text{Coste} \mathrel{+}= & \; c_{i-1j} + c_{ji+1} + c_{j-1i} + c_{ij+1} \\ & - c_{i-1i} - c_{ii+1} - c_{j-1j} - c_{jj+1}\end{aligned}$$

donde:

- c_{ij}: es el coste de viajar entre el nodo i y el nodo j.

Latencia:

$$\begin{aligned}\text{Latencia} \mathrel{+}= & \; y_{i-1j} + y_{j-1i} - y_{i-1i} - y_{j-1j} \\ & + (y_{i-1j} + y_{ji+1} - y_{i-1i} - y_{ii+1}) \cdot (|S| - pos_i) \\ & + (y_{j-1i} + y_{ij+1} - y_{j-1j} - y_{jj+1}) \cdot (|S| - pos_j)\end{aligned}$$

donde:

- $|S|$: es el tamaño de la solución incluyendo el nodo inicial.
- y_{ij}: es la latencia asociada entre el nodo i y el nodo j.
- pos_i: es la posición del nodo en la solución.

Para reevaluar la distancia total de la solución tan sólo hay que restar a la distancia total la distancia del nodo que se va a intercambiar con el nodo anterior y la distancia del nodo que se va a intercambiar con el nodo posterior. Y sumar a la distancia total la distancia entre el nodo anterior y el nuevo nodo intercambiado, y la distancia del nodo intercambiado al nodo posterior. Esto se realiza con los dos nodos intercambiados. En la Figura 3.4 se muestra un ejemplo de como se reevalúa la distancia total de la solución.

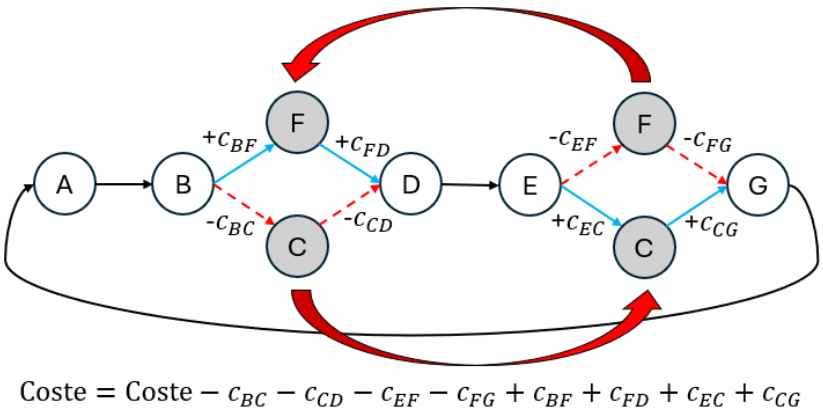


Figura 3.4: Reevaluar distancia total de la solución en 2-intercambio

Para reevaluar la latencia total tan sólo hay que restar a la latencia total la latencia del nodo que se va a intercambiar con el nodo anterior y la latencia del nodo que se va a intercambiar con el nodo posterior. Después se resta el impacto en la latencia de los nodos posteriores. A continuación, se suma a la latencia total la latencia entre el nodo anterior y el nuevo nodo intercambiado, y la latencia del nodo intercambiado al nodo posterior, a lo que se añade el impacto del nuevo nodo en la latencia de los posteriores. Esto se realiza con los dos nodos intercambiados. En la Figura 3.5 se muestra un ejemplo de como se reevalúa la latencia total de la solución.

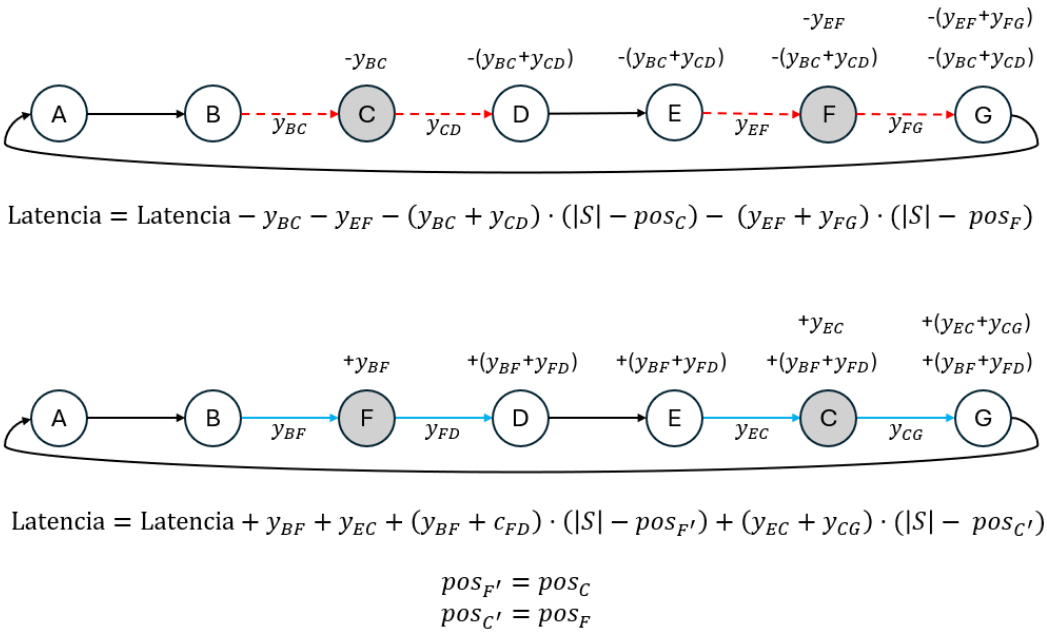


Figura 3.5: Reevaluar latencia total de la solución en 2-intercambio

La primera aplicación se atribuye a George Croes en 1958 [153], quien aplicó técnicas de optimización heurística al problema del TSP. Otros usos en la literatura se pueden encontrar por mano de Lin y Kernighan en [154] y, Clarke y Wright en [155].

Inserción

Esta búsqueda local se compone del operador *Inserción* y de la vecindad $\mathcal{N}_2$. El operador de *Inserción* (*Insertion* o *in-out*) elige un nodo de la solución y lo elimina, para luego insertar otro nodo que no esté en la solución en su lugar. Esto modifica la estructura de la ruta y permite una exploración que altera no solo el orden, sino también el contenido de la solución. El operador usa first improvement.

La vecindad $\mathcal{N}_2$ está formada por todas las soluciones que se pueden obtener al reemplazar cada nodo de la ruta actual con otros nodos que no estén presentes. Estas soluciones se obtienen tras aplicar el operador *Inserción*. Esto ofrece una vecindad que abarca soluciones más alejadas de la inicial, explorando diferentes configuraciones de nodos.

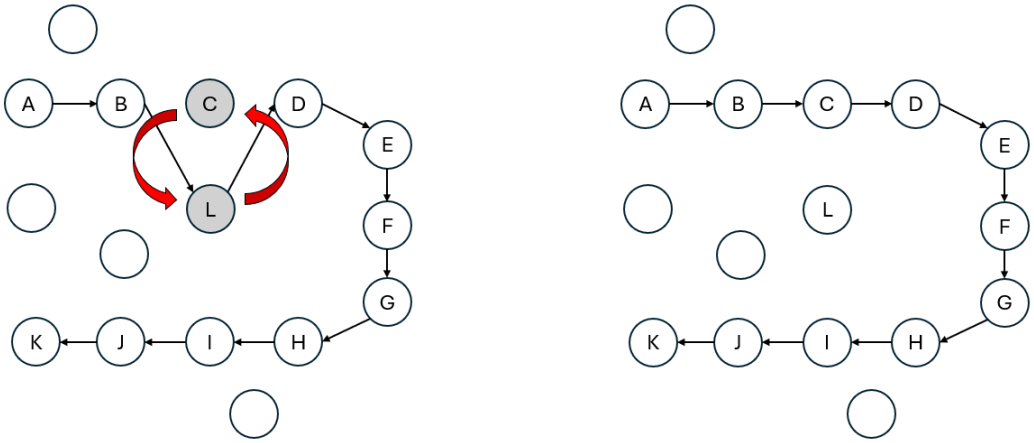


Figura 3.6: Ejemplo del operador Inserción

Complejidad:

- La búsqueda local tiene una complejidad de $O(n^2)$, ya que cada nodo en la solución puede ser intercambiado por cualquier nodo fuera de la solución.
- El operador como tal tiene una complejidad de $O(1)$ al sólo intercambiar un nodo de la solución por un nodo fuera de la solución.
- La función de evaluación tiene una complejidad de $O(1)$, ya que no se necesita reevaluar toda la ruta, sino sólo la posición insertada, y los nodos anterior y siguiente.

Algoritmo 3 Inserción (S, CL)

```
1: for i ∈ S do
2:     for j ∈ V\S do
3:         S' ← remove_node(S, i)
4:         S' ← insert_node(S', j)
5:         if S' is better than S then
6:             S ← S'
7:             V\S ← V\S \ {j}
8:             V\S ← V\S ∪ {i}
9:             i = 0
10:        end if
11:    end for
12: end for return S
```

El algoritmo de Inserción parte de una solución inicial S y el complementario de la solución inicial $V\backslash S$, que representa los nodos no incluidos en la solución. El objetivo es mejorar la solución actual sustituyendo nodos de S por nodos en $V\backslash S$. El procedimiento comienza recorriendo todos los nodos i en la solución actual (paso 1) y, para cada uno, se considera cada nodo j en la lista de candidatos $V\backslash S$ (paso 2). En cada iteración, el nodo i es eliminado de la solución generando una solución temporal S' (paso 3), y en su lugar se inserta el nodo j (paso 4). Una vez generada la nueva solución S', se evalúa si es mejor que la solución original S (paso 5), tal

y como se explica en la Sección 1.1.1. En caso afirmativo, se actualiza la solución actual con la nueva (paso 6), se actualiza la lista de candidatos eliminando el nodo insertado j (paso 7) y añadiendo el nodo extraído i (paso 8), y finalmente se reinicia el índice i (paso 9) para reiniciar la búsqueda de mejoras desde el principio con la nueva configuración. Este proceso se repite hasta haber evaluado todas las combinaciones posibles de intercambio entre la solución actual y los candidatos externos. El algoritmo devuelve finalmente la mejor solución encontrada (paso 12).

Tras realizar un movimiento, el método de evaluación no necesita reevaluar la solución entera, tan sólo tiene que tener en cuenta los nodos intercambiados. Cada FO tiene su fórmula para reevaluarse. En esta búsqueda local hay que reevaluar el coste de recorrido, la latencia y el beneficio, puesto que el número de nodos no varía.

Coste de recorrido:

$$\text{Coste} \mathrel{+}= c_{i-1j} + c_{ji+1} - c_{i-1i} - c_{ii+1}$$

donde:

- c_{ij}: es el coste de viajar entre el nodo i y el nodo j.

Latencia:

$$\text{Latencia} \mathrel{+}= y_{i-1j} - y_{i-1i} + (y_{i-1j} + y_{ji+1} - y_{i-1i} - y_{ii+1}) \cdot (|S| - pos_i)$$

donde:

- $|S|$: es el tamaño de la solución incluyendo el nodo inicial.
- y_{ij}: es la latencia asociada entre el nodo i y el nodo j.
- pos_i: es la posición del nodo en la solución.

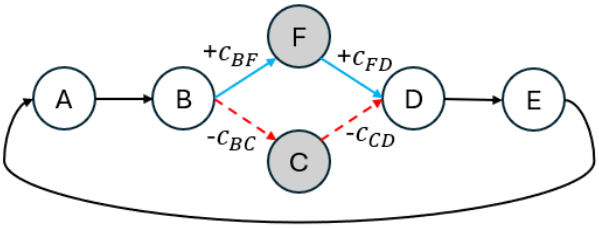
Beneficio:

$$\text{Beneficio} \mathrel{+}= p_j - p_i$$

donde:

- p_i: es el beneficio obtenido por visitar el nodo i.

Para reevaluar la distancia total de la solución tan sólo hay que restar a la distancia total la distancia del nodo que se va a intercambiar con el nodo anterior y la distancia del nodo que se va a intercambiar con el nodo posterior. Y sumar a la distancia total la distancia entre el nodo anterior y el nuevo nodo intercambiado, y la distancia del nodo intercambiado al nodo posterior. En la Figura 3.7 se muestra un ejemplo de como se reevalúa la distancia total de la solución.

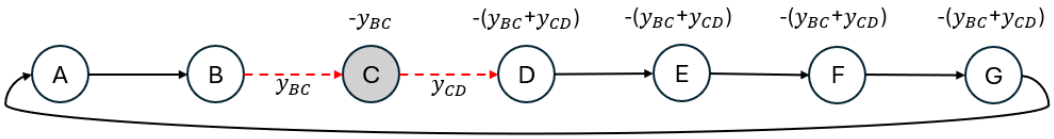


$$\text{Coste} = \text{Coste} - c_{BC} - c_{CD} + c_{BF} + c_{FD}$$

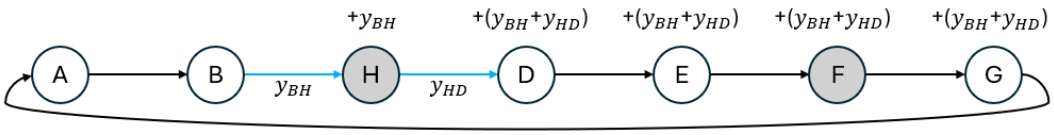
Figura 3.7: Reevaluar distancia total de la solución en Inserción

Para reevaluar la latencia total tan sólo hay que restar a la latencia total la latencia del nodo que se va a intercambiar con el nodo anterior y la latencia del nodo que se va a intercambiar con el nodo posterior. Después se resta el impacto en la latencia de los nodos posteriores. A continuación, se suma a la latencia total la latencia entre el nodo anterior y el nuevo nodo intercambiado, y la latencia del nodo intercambiado al nodo posterior, a lo que se añade el impacto del nuevo nodo en la latencia de los posteriores. En la Figura 3.8 se muestra un ejemplo de como se reevalúa la latencia total de la solución.

El beneficio se actualiza sustrayendo el beneficio del nodo retirado y sumando el beneficio del nuevo nodo al beneficio total de la solución.



$$\text{Latencia} = \text{Latencia} - y_{BC} - (y_{BC} + y_{CD}) \cdot (|S| - pos_C)$$



$$\text{Latencia} = \text{Latencia} + y_{BH} + (y_{BH} + c_{HD}) \cdot (|S| - pos_H)$$

$$pos_H = pos_C$$

Figura 3.8: Reevaluar latencia total de la solución en Inserción

Este operador permite una exploración más amplia y tiene la capacidad de alterar la configuración de nodos, a diferencia de los operadores de solo intercambio.

Este operador ha sido utilizado ampliamente en problemas de enrutamiento de vehículos y TSP desde la década de los 80. Algunos trabajos, como por ejemplo el de Gendreau, Hertz, y Laporte en [156] y, Laporte y Osman en [157] utilizan este operador.

2-opt

Esta búsqueda local se compone del operador *2-opt* y de la vecindad $\mathcal{N}_3$. El operador *2-opt* elimina dos aristas (es decir, dos segmentos de ruta) de la solución y conecta los extremos resultantes de manera diferente para formar una nueva ruta. De esta manera, los nodos de dicho segmento se recorren en orden inverso. Es un operador muy popular en TSP porque puede cambiar la estructura de la ruta sin afectar el conjunto de nodos. Para reducir el esfuerzo computacional, sólo se consideran los nodos más prometedores, que son aquellos situados a una distancia inferior a un determinado umbral γ que se analizará en los resultados computacionales. Estos nodos prometedores conforman la lista de nodos prometedores PNL. El operador usa first improvement.

La vecindad $\mathcal{N}_3$ incluye todas las soluciones obtenidas del operador *2-opt*. Las soluciones se obtienen al reemplazar dos aristas en la ruta con otras dos aristas que conecten los extremos resultantes de manera diferente. Esta vecindad explora cambios en la estructura de la ruta que no alteran el conjunto de nodos, pero sí la secuencia.

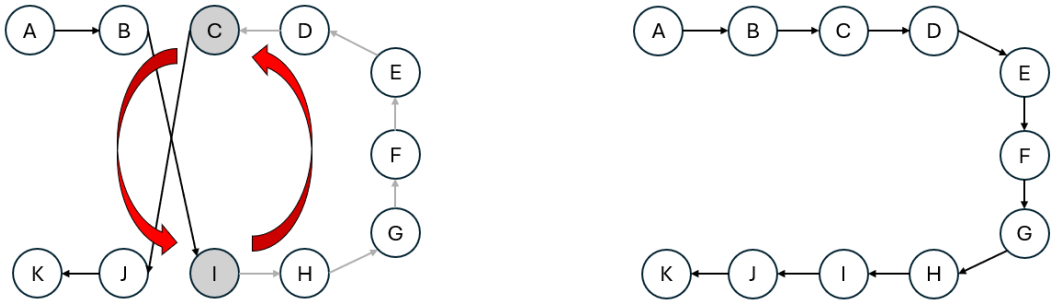


Figura 3.9: Ejemplo del operador 2-opt

Complejidad:

- La búsqueda local tiene una complejidad de $O(n^2)$, ya que para cada par de aristas en la ruta se prueban todas las reconexiones posibles.
- El operador tiene una complejidad de $O(n)$, ya que al intercambiar dos nodos en la solución, hay que reconectar todos los nodos que se encuentran entre los dos nodos.
- La función de evaluación tiene una complejidad de $O(n)$, debido a que se debe reevaluar todos los nodos desplazados.

Este método parte de una solución inicial y realiza intercambios de segmentos de la ruta para reducir su coste. El procedimiento comienza iterando sobre cada nodo i de la solución S (paso 1). Para cada nodo i, se identifica un subconjunto de nodos prometedores PNL (paso 2), limitado mediante el parámetro γ que restringe la búsqueda local y reduce el coste computacional. Este conjunto contiene los nodos candidatos con los que se podrían formar nuevas conexiones beneficiosas. A continuación, para cada nodo j del conjunto PNL (paso 3), se genera una nueva solución S' mediante la operación *reconnect_edges*, que simula el intercambio

Algoritmo 4 2-opt (S, γ)

```
1: for i ∈ S do
2:     PNL ← promising_nodes(S, i, γ)
3:     for j ∈ PNL do
4:         S' ← reconnect_edges(S, i, j)
5:         if S' is better than S then
6:             S ← S'
7:             i = 0
8:         end if
9:     end for
10: end for return S
```

2-opt entre los nodos i y j (paso 4). Si la solución generada S' mejora con respecto a la actual S (paso 5), tal y como se explica en la Sección 1.1.1, se acepta como nueva solución (paso 6) y se reinicia el índice i (paso 7) para permitir nuevas oportunidades de mejora a partir de la nueva configuración. Este proceso se repite hasta explorar todas las combinaciones prometedoras dentro de los límites marcados por γ, devolviendo finalmente la mejor solución encontrada (paso 10).

A contrario que en las otras búsquedas locales, en el *2-opt* no es tan sencillo reevaluar de forma parcial las FO de la Solución. En esta búsqueda local, tan sólo hay que reevaluar el coste de recorrido y la latencia, puesto que el número de nodos no varía y al ser un movimiento intraruta los nodos no cambian, sólo su posición, por lo que el beneficio es el mismo. En el caso de la latencia, no se ha logrado una fórmula para reevaluar el valor que sea más eficiente que la evaluación total de la solución.

A diferencia de las otras búsquedas locales vistas, en la evaluación del *2-opt* es relevante saber si es un TSP simétrico o asimétrico. Se recuerda que en el TSP simétrico, c_{ij} es la distancia entre los nodos i y j tiene el mismo coste en ambas direcciones $c_{ij} = c_{ji}$, por lo que el recorrido puede evaluarse en ambas direcciones con el mismo coste. En cambio, para el caso asimétrico se tiene que reevaluar la solución entera.

Coste de recorrido en el caso simétrico:

$$\text{Coste} \mathrel{+}= c_{i-1j} + c_{ij+1} - c_{i-1i} - c_{jj+1}$$

donde:

- c_{ij}: es el coste de viajar entre el nodo i y el nodo j.

El operador 2-opt permite mejorar la solución eliminando "cruces" en la ruta, lo que suele reducir la distancia total de la ruta. También suele ser usado en combinación con otros operadores en el TSP.

La técnica 2-opt fue introducida por George Croes en 1958 [153], como una mejora local para el TSP. Shen Lin, por su parte, amplió esta idea en su influyente trabajo de 1965 [158], y luego junto con Brian W. Kernighan en 1973 [154] desarrollaron el algoritmo Lin-Kernighan, que generaliza 2-opt y 3-opt usando un enfoque

más flexible y poderoso. Otros usos en la literatura se pueden encontrar por mano de Lin y Kernighan en [154] y, Johnson y McGeoch en [159].

Operador	Vecindad	Complejidad (BL, Op., Ev.)	Característica principal
2-intercambio	$\mathcal{N}_1$	$O(n^2)$ | $O(1)$ | $O(1)$	Permite intercambiar dos nodos en la ruta.
Inserción	$\mathcal{N}_2$	$O(n^2)$ | $O(1)$ | $O(1)$	Intercambia un nodo de la ruta con otro de fuera.
2-opt	$\mathcal{N}_3$	$O(n^2)$ | $O(n)$ | $O(n)$	Cambia la estructura de la ruta sin alterar el conjunto de nodos.

Tabla 3.1: Resumen de operadores y vecindades.

3.2.3. GRASP con VND

En esta tesis doctoral se propone desarrollar un algoritmo metaheurístico fundamentado en la Búsqueda de Vecindad Variable Descendente (*Variable Neighborhood Descent*, VND) con el objetivo de encontrar soluciones para el Mo-TSRPP. Esta es una versión determinista de la Búsqueda de Vecindad Variable (*Variable Neighborhood Search*, VNS) [160, 161], el cual está explicado en la Sección 1.4.1.

El VND comienza por la generación de una solución, ha dicha solución de partida se le intenta mejorar mediante la exploración de varias vecindades iterativamente. Para aplicar esta metodología es necesario definir un conjunto de estructuras de vecindad. Cada vez que se explora una vecindad, se analiza si se ha encontrado una mejora o no. En caso afirmativo, la búsqueda comienza de nuevo con la primera vecindad. En caso contrario, la búsqueda continúa con la siguiente vecindad. VND se ha aplicado con éxito para resolver otros problemas de optimización combinatoria, véase por ejemplo [162, 163].

Para la construcción de soluciones, se aplica la heurística GRA descrita anteriormente en el apartado 3.2.1. Con ello obtenemos un frente de Pareto de soluciones diversas.

Para aplicar correctamente el algoritmo VND hay que tomar dos decisiones importantes: la selección de las vecindades y el orden de aplicación. Consideramos tres vecindades diferentes para el Mo-TSRPP: 2-intercambio ($\mathcal{N}_1$), inserción ($\mathcal{N}_2$) y 2-opt ($\mathcal{N}_3$). En la Figura 3.10 se puede ver el esquema propuesto para VND.

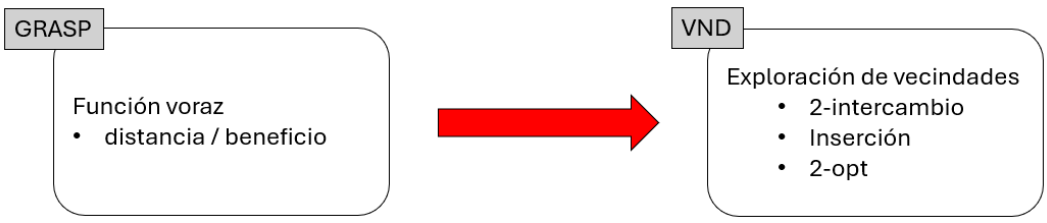


Figura 3.10: Esquema propuesto de VND

En el contexto de VND, el orden en el que se exploran las vecindades suele estar relacionado con su complejidad. En concreto, las vecindades más pequeñas y rápidas de evaluar se exploran en primer lugar, mientras que las más grandes y lentas son las últimas en explorarse. Con el objetivo de confirmar o refutar esta hipótesis, se ha realizado un experimento para evaluar el efecto de este orden en la calidad final del conjunto de soluciones no dominadas (ver Sección 4 para más detalles).

El algoritmo 5 muestra el pseudocódigo del algoritmo VND propuesto en esta tesis, que recibe como parámetros de entrada una aproximación inicial del Frente de Pareto, $\hat{PF}$, y las vecindades, $\mathcal{N}_k$ para $k = 1, 2, 3$.

Algoritmo 5 Mo-VND ($\hat{PF}, \mathcal{N}_k$ for $k = 1, ..., k_{max}$)

1: **for** $S \in \hat{PF}$ **do**
2: **while** $k \leq k_{max}$ **do**
3: $S' \leftarrow \mathcal{N}_k(S)$
4: **if** S' is better than S **then**
5: $S \leftarrow S'$
6: k = 1
7: **else**
8: k = k + 1
9: **end if**
10: **end while**
11: **end for**
12: **return** $\hat{PF}$

La metaheurística VND se aplica a cada solución eficiente S perteneciente al Frente de Pareto inicial aproximado, $\hat{PF}$, que se ha obtenido en la fase de construcción previamente explicada, (paso 1-11). A continuación, cada solución S se mejora con la vecindad k (paso 3). Si la solución resultante S' es *mejor* que S, tal y como se explica en la Sección 1.1.1, entonces el algoritmo reinicia la búsqueda desde la primera vecindad (pasos 5 y 6). Dado que el concepto de mejora es siempre controvertido en un contexto multiobjetivo, se quiere destacar que, en este punto, se encuentra una mejora cuando una solución es mejor en uno o más objetivos que la original sin deteriorar el resto de objetivos. En caso contrario, el algoritmo pasa al siguiente vecindario (paso 8). Teniendo en cuenta que cada solución generada se evalúa para entrar en el conjunto de soluciones eficientes, pero, en aras de la simplicidad, no se ha incluido que los pasos del algoritmo.

Una vez definido el VND propuesto, es necesario describir cómo generar la aproximación inicial del Frente de Pareto, $\hat{PF}$, ya que ésta será una de las entradas del VND. La hipótesis original de la metodología VNS [160] sugiere que considerar soluciones iniciales aleatorias no afecta al rendimiento del algoritmo, pero recientemente se ha demostrado que VNS tiene un mejor rendimiento (en el contexto de la optimización multiobjetivo, se considera que una solución es mejor que otra si la nueva solución domina a la original) cuando se parte de regiones prometedoras del espacio de búsqueda, como por ejemplo para el problema de colocación de monitores [164] y para el problema multiobjetivo de detección de comunidades [165]. En

este trabajo, se propone un método constructivo basado en la parte constructiva del GRASP que incluye una fase de diversificación (GRA).

3.3. Algoritmos multiobjetivos

Dado que no hay trabajos previos en la literatura que resuelvan el MO-TSRPP se ha comparado nuestra propuesta algoritmica con los algoritmos multiobjetivos más extendidos en el estado del arte. De entre ellos se han escogido tres algoritmos evolutivos: Non-dominated Sorting Genetic Algorithm (NSGA-II), Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D), y Strength Pareto Evolutionary Algorithm (SPEA2).

Antes de empezar a hablar de estos algoritmos, se va a realizar un breve repaso a los conceptos de poblacional, evolutivo y genético:

- Poblacional: Trabajan con una población de soluciones en cada generación. Los tres algoritmos previamente mencionados (NSGA-II, MOEA/D y SPEA2) son algoritmos poblacionales.
- Evolutivo: Se basa en los principios generales de evolución biológica (selección, mutación, reproducción, elitismo). Los tres algoritmos previamente mencionados (NSGA-II, MOEA/D y SPEA2) son algoritmos evolutivos.
- Genético: Usa operadores genéticos clásicos como cruce y mutación, inspirados directamente en algoritmos genéticos. NSGA-II y SPEA2 claramente lo son. MOEA/D puede usar operadores genéticos, pero no depende exclusivamente de ellos, así que se dice que es una metaheurística evolutiva por descomposición, no estrictamente genética.

3.3.1. NSGA-II

El NSGA-II (*Non-dominated Sorting Genetic Algorithm*) traducido como Algoritmo Genético de Clasificación No Dominada, propuesto por [166], es un algoritmo evolutivo ampliamente utilizado para la optimización multiobjetivo. Comienza generando una población inicial de soluciones, que es evaluada y ordenada en función del principio de no dominación, asignando a cada solución un valor de rango según su pertenencia a distintos frentes de Pareto.

Una vez completada la clasificación, se calcula un valor de distancia de apiñamiento para cada solución, que mide la densidad de soluciones vecinas en el espacio objetivo. Este valor es crucial para fomentar la diversidad: se favorecen aquellas soluciones que se encuentran en regiones menos densas.

La selección de soluciones que participarán en la reproducción se realiza mediante un torneo binario, donde dos soluciones se comparan utilizando como criterios primero el rango de no dominación y luego la distancia de apiñamiento (prefiriendo soluciones con menor rango y mayor distancia).

A partir de la población seleccionada de padres (soluciones actuales), se aplican operadores genéticos de cruce binario simulado y mutación polinómica para generar una nueva población de descendientes (también llamada población hija). Estos descendientes no se mezclan directamente con la población anterior, sino que forman, junto con los padres, una población combinada de tamaño $2N$.

Esta población combinada es reordenada mediante una nueva clasificación por frentes de no dominación. A continuación, se seleccionan las mejores N soluciones obtenidas (en este contexto reciben el nombre de individuos) para conformar la siguiente generación, priorizando las soluciones de menor rango y, en caso de empate, aquellas con mayor distancia de apiñamiento dentro del último frente considerado.

Este proceso iterativo permite que NSGA-II mantenga un buen equilibrio entre convergencia hacia el óptimo de Pareto y diversidad en las soluciones.

3.3.2. MOEA/D

El MOEA/D (*Multi-Objec- tive Evolutionary Algorithm based on Decomposition*) traducido como Algoritmo Evolutivo Multiobjetivo basado en la Descomposición, fue propuesto por [167], como una alternativa a los enfoques basados en dominancia, como NSGA-II. A diferencia de estos últimos, que evalúan las soluciones en términos de relaciones de no dominación globales, MOEA/D aborda la optimización multiobjetivo descomponiendo el problema en una serie de subproblemas escalares, cada uno asociado a una dirección o región específica del frente de Pareto.

El algoritmo comienza generando un conjunto de vectores de pesos uniformemente distribuidos, que representan distintas combinaciones de importancia relativa entre los objetivos. Cada vector de pesos define un subproblema escalar, cuya optimización permite aproximar una región distinta del frente eficiente. Entre las funciones de agregación comúnmente utilizadas para la descomposición se encuentran: el método de Tchebycheff, la suma ponderada y el método de frontera normal. Esto permite convertir el problema multiobjetivo en un conjunto de problemas de optimización monobjetivo, más fáciles de manejar y resolver de forma paralela.

Cada subproblema está asociado a un conjunto de vecinos, definidos como aquellos subproblemas cuyos vectores de pesos están más cercanos en el espacio. La evolución de las soluciones se realiza de forma cooperativa, ya que al generar descendencia para un subproblema, se utilizan principalmente soluciones dentro de su vecindario. Esta estrategia fomenta la convergencia local y mejora la eficiencia al reducir la aleatoriedad de los operadores evolutivos.

El supuesto es que las soluciones óptimas de subproblemas vecinos son similares, por lo que la información compartida entre ellos puede acelerar la convergencia hacia el frente de Pareto. Además, esta estructura promueve una exploración más controlada del espacio de búsqueda y mantiene una buena diversidad poblacional a través del cubrimiento de distintas regiones del frente.

En cada iteración, se seleccionan padres (generalmente dentro del vecinda-

rio), se aplican operadores genéticos como cruce y mutación, y la descendencia se evalúa. Si una solución hija mejora la función escalar correspondiente a su subproblema (y potencialmente a otros vecinos), entonces se actualizan las soluciones actuales.

Este proceso se repite hasta que se alcanza un criterio de parada, como un número máximo de generaciones o evaluaciones. A lo largo del proceso, se mantiene una población de soluciones, donde cada solución está vinculada a un subproblema.

3.3.3. SPEA2

El SPEA2 (*Strength Pareto Evolutionary Algorithm*) traducido como Algoritmo Evolutivo de Pareto Fortalecido, fue propuesto por [168] como una mejora sustancial del algoritmo original SPEA. Al igual que otros algoritmos evolutivos multiobjetivo, SPEA2 se basa en el principio de dominancia de Pareto para evaluar y seleccionar soluciones. Sin embargo, introduce innovaciones clave en la asignación de aptitud, la gestión de archivos elitistas y el mantenimiento de la diversidad que lo hacen más robusto y eficiente que su predecesor.

SPEA2 mantiene dos conjuntos de soluciones durante el proceso evolutivo: Una población activa que evoluciona generación tras generación y un archivo externo elitista que almacena las mejores soluciones no dominadas encontradas hasta el momento. El algoritmo comienza generando una población inicial aleatoria y un archivo externo vacío.

Una de las principales mejoras de SPEA2 respecto a su versión anterior es la forma en que calcula la aptitud de cada individuo. A diferencia de otros algoritmos que utilizan sólo la dominancia binaria, SPEA2 introduce una combinación de fuerza (cada solución tiene una medida de cuántas otras soluciones domina) y aptitud (se calcula como la suma de las fuerzas de las soluciones que la dominan. Cuanto menor sea el valor de aptitud, mejor es la solución ya que es dominada por menos individuos). Este enfoque permite una clasificación más gradual y diferenciada de las soluciones, especialmente útil cuando muchas soluciones no son dominadas entre sí (como suele ocurrir en problemas con muchos objetivos).

SPEA2 emplea un archivo externo con capacidad limitada que actúa como mecanismo elitista, preservando las mejores soluciones encontradas. En cada generación se copian las soluciones no dominadas de la población al archivo, se eliminan duplicados o soluciones dominadas si el tamaño del archivo excede el límite, y en caso de sobrepoblación, se aplica un mecanismo de poda basado en distancia, eliminando las soluciones más cercanas entre sí para mantener la diversidad. Este enfoque asegura que el archivo evoluciona de forma controlada y que las soluciones elitistas representen diferentes regiones del frente de Pareto.

Tras la evaluación y actualización del archivo, se combinan las soluciones del archivo y de la población para formar un conjunto sobre el cual se aplica una selección mediante torneo binario. La selección considera la aptitud asignada, favoreciendo soluciones no dominadas y diversas. Los individuos seleccionados se reproducen

utilizando operadores genéticos convencionales, como el cruce y la mutación, generando así una nueva población de descendientes. Esta nueva población se somete al mismo proceso en la siguiente generación.

En la Tabla 3.2 se muestra una comparativa de los tres algoritmos y sus principales características:

Característica	NSGA-II	SPEA2	MOEA/D
Enfoque	Ordenamiento no dominado	Dominancia y densidad	Descomposición
Diversidad	Distancia de apiñamiento	k-vecinos más cercanos	Vecindario local
Elitismo	Implícito en la selección	Archivo externo	Mediante reemplazo local
Selección	Torneo basado en ranking y distancia	Aptitud dominancia y densidad	Basada en vecinos y agregación
Escalabilidad	Limitada para muchos objetivos	Mejor que NSGA-II	Alta escalabilidad
Complejidad por generación	$O(MN^2)$ (M: objetivos, N: tamaño)	Algo mayor que NSGA-II	Más eficiente en grandes dimensiones

Tabla 3.2: Comparativa entre los algoritmos NSGA-II, SPEA2 y MOEA/D.

Capítulo 4

Resultados experimentales

El proceso experimental permite verificar la eficacia de los algoritmos propuestos al comparar sus resultados con los obtenidos por los métodos más avanzados. En este capítulo, se presentan los resultados experimentales obtenidos en esta Tesis Doctoral para resolver "el problema multiobjetivo del comerciante reparador con beneficios" (Mo-TSRPP). Se realiza una amplia experimentación del algoritmo metaheurístico propuesto en el Capítulo 3 utilizando diversos conjuntos de instancias de referencia.

4.1. Introducción

La experimentación es un proceso habitual en ciencia y tecnología que se utiliza para estudiar un fenómeno específico y extraer conclusiones sobre él. En el campo de la optimización, permite comparar diferentes algoritmos para resolver un mismo problema. Además, es útil para validar y depurar el código fuente, así como para ajustar los parámetros de los algoritmos.

Algunos factores clave en el proceso de experimentación incluyen el conjunto de instancias utilizadas para llevar a cabo los experimentos, las métricas de calidad empleadas para comparar las diferentes propuestas y las características del ordenador en el que se realizan. En las secciones 4.1.1, 4.1.2 y 4.1.3 se revisan algunos de estos aspectos.

4.1.1. Conjuntos de instancias de referencia

Para evaluar el rendimiento de los algoritmos propuestos, se han utilizado tres tipos diferentes de instancias. Para la generación de instancias se ha considerado la librería TSPLIB con 77 instancias debidamente transformadas tal y como propone [28]. Para dichas instancias TSP se calcula la distancia euclídea entre cada dos puntos (nótese que el valor se redondea hacia arriba) y además se generan los beneficios p_i para todos los $i \in V$ de tres formas distintas para eliminar el sesgo que puede producir la distribución de beneficios en los resultados. De cada uno de

estos tipos de instancias, se han creado dos conjuntos diferentes que se emplean en la experimentación de los algoritmos heurísticos. A continuación, se describen cada uno de estos conjuntos:

- **Conjunto de instancias con el mismo beneficio para cada nodo:** $p_i = 1$, para todos los $i \in V$. En estos casos todos los beneficios son iguales y fijos a 1. Obsérvese que en este conjunto la maximización del beneficio y la maximización del número total de clientes son equivalentes. Este conjunto de datos está formado por un total de 77 grafos.

- **Conjunto de instancias con beneficio aleatorio para cada nodo:** $p_i = 1 + (7141i + 73) \bmod 100$, para todo $i \in V$. El conjunto genera instancias con beneficios pseudoaleatorios entre 1 y 100. Este conjunto de datos está formado por un total de 77 grafos.

- **Conjunto de instancias con beneficio ponderado en función de la distancia a la que se encuentra cada nodo del depósito:** $p_i = 1 + \left\lfloor 99\frac{c_{1i}}{\theta} \right\rfloor$, para todo $i \in V$ donde $\theta = \max_{j \in V} c_{1j}$. El conjunto produce casos más complejos en los que los mayores beneficios se asocian a nodos más alejados del depósito. Este conjunto de datos está formado por un total de 77 grafos.

Se han utilizado un total de 231 instancias para evaluar nuestra propuesta metaheurística y compararlos con los tres algoritmos evolutivos previamente mencionados, divididas en tres conjuntos con un tamaño de 77 instancias cada uno. El número de nodos de las instancias en estos conjuntos varía entre 51 y 18512. La lista completa de instancias que componen cada subconjunto se puede consultar en la Tabla 4.1. El nombre de cada instancia incluye el número de nodos que contiene el grafo, y los grafos son completos.

Instancias						
a280	berlin52	bier127	brd14051	ch130	ch150	d198
d493	d657	d1291	d1655	d2103	d15112	d18512
eil51	eil76	eil101	fl417	fl1400	fl1577	fl3795
fnl4461	gil262	kroA100	kroA150	kroA200	kroB100	kroB150
kroB200	kroC100	kroD100	kroE100	lin105	lin318	nrw1379
p654	pcb442	pcb1173	pcb3038	pr76	pr107	pr124
pr136	pr144	pr152	pr226	pr264	pr299	pr439
pr1002	pr2392	rat99	rat195	rat575	rat783	rd100
rd400	rl1304	rl1323	rl1889	rl5915	rl5934	rl11849
st70	ts225	tsp225	u159	u574	u724	u1060
u1432	u1817	u2152	u2319	usa13509	vm1084	vm1748

Tabla 4.1: Conjunto de instancias

También se usa otra instancia con coordenadas reales del municipio de Móstoles, perteneciente a la provincia de Madrid (España), para la aplicación del algoritmo en un caso real. Esta instancia cuenta con un grafo de 104 nodos y 5356 aristas.

4.1.2. Medidas de calidad

Una vez establecidos los conjuntos de instancias para la experimentación, es crucial definir una serie de métricas de calidad que permitan diferenciar entre los distintos algoritmos. Aunque existen métricas genéricas aplicables a diversas disciplinas, generalmente dependen del campo específico de trabajo. Las métricas de calidad empleadas en esta Tesis Doctoral y a lo largo de este capítulo son ampliamente reconocidas en el ámbito de la optimización. En particular, se han considerado las siguientes métricas:

La calidad de los resultados obtenidos por los distintos algoritmos se mide comparándolos con el frente de Pareto exacto, pero al no ser conocido, se compara con el mejor frente de Pareto. Como no suele haber uno mejor que los demás, se calcula el frente de Referencia, que se denomina Conjunto de Referencia (denotado en adelante por RS por sus siglas en inglés de *Reference Set*) ya que el frente de Pareto óptimo del Mo-TSRPP es desconocido. El Conjunto de Referencia se estima como el mejor frente de Pareto que contiene todas las soluciones no dominadas resultantes de fusionar las soluciones encontradas por todos los algoritmos objeto de comparación. Se calcula combinando todos los frentes de Pareto generados por los diferentes algoritmos que se quieren comparar en un único conjunto conjunto global. Se elimina cualquier solución que esté dominada por otra dentro del conjunto global fusionado. El conjunto resultante contiene solo soluciones no dominadas entre sí. Este conjunto es el mejor frente de Pareto aproximado disponible con la información obtenida. Se utiliza como base común para evaluar y comparar todos los algoritmos aplicando métricas. Por lo tanto, a continuación se describen brevemente las métricas utilizadas para medir la calidad de los diferentes algoritmos de optimización multiobjetivo.

Normalmente, la cardinalidad del frente de Pareto, la proximidad de las soluciones obtenidas al frente de Pareto óptimo y la diversidad de las soluciones deben incluirse para establecer la calidad del frente de Pareto obtenido. En este trabajo, hemos considerado varias de las métricas multiobjetivo más extendidas (ver [169] o [170]): el número de soluciones eficientes, la cobertura, la dispersión, el hipervolumen, el indicador ϵ, la distancia generacional y la distancia generacional invertida.

- La *cardinalidad* o el *número de soluciones eficientes en el frente de Pareto* (*cardinality*, $|\hat{PF}|$) cuenta el número de soluciones eficientes encontradas por el algoritmo considerado. El responsable de la toma de decisiones suele preferir un mayor número de soluciones eficientes. Sin embargo, esta métrica no tiene en cuenta la calidad de las soluciones.

- La *métrica de cobertura* (*coverage metric*, CV), evalúa la proporción de soluciones del conjunto de referencia RS que domina a las soluciones del frente de Pareto obtenidas por el algoritmo A. Sea k el número de funciones objetivo, entonces la cobertura se calcula como:

$$C(A, RS) = \frac{|\forall b \in RS | \exists a \in A : f_k(a) \leq f_k(b) \forall k|}{|RS|}.$$

$C(A, RS) \in [0, 1]$ y $C(A, RS) = 0$ significa que todas las soluciones del conjunto de Referencia dominan estrictamente a las soluciones obtenidas por el algoritmo A y $C(A, RS) = 1$ significa que todas las soluciones del conjunto de Referencia están dominadas por las soluciones obtenidas por el algoritmo A, por lo que, cuanto menor sea el valor de $C(A, RS)$, mejor es el algoritmo A.

- La *dispersión* (*spread*, Δ) mide la extensión de la dispersión por el conjunto de soluciones computadas obtenidas por el frente de Pareto del algoritmo A.

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{|A|-1} |d_i - \bar{d}|}{d_f + d_l + (|A| - 1)\bar{d}}$$

donde d_i es la distancia euclídea entre soluciones consecutivas en el frente de Pareto obtenido por el algoritmo A, $\hat{d}$ es la media de estas distancias, y d_f y d_l son las distancias euclídeas a las soluciones extremas del conjunto de Referencia en el espacio objetivo. El mejor valor posible es $\Delta = 0$, que indica una dispersión perfecta de las soluciones en el frente de Pareto obtenido por el algoritmo considerado.

- El *hipervolumen* (*hypervolume*, HV) evalúa el volumen en el espacio objetivo que cubre el frente de Pareto obtenido por el algoritmo A. Para calcular el HV, para cada solución $a \in A$, se construye un hipercubo v_i con un punto de referencia W (un vector de los peores valores de la función objetivo) y la solución a como las esquinas diagonales del hipercubo. Por lo que, el hipervolumen es la unión de todos los hipercubos que encuentra.

$$HV = volume(\bigcup_{a=1}^{|A|} v_a).$$

Una buena aproximación algorítmica tendrá un valor HV grande.

- El *indicador* ϵ (ϵ-*indicator*, ϵ) mide la menor distancia (el menor valor ϵ) necesaria para transformar cada punto del frente de Pareto obtenido por el algoritmo A en el punto más cercano del conjunto de Referencia.

$$\epsilon = \text{ínf}\{\epsilon \in \mathbb{R} : \forall b \in RS \exists a \in A \text{ tal que } f_k(a) \leq \epsilon \cdot f_k(b) \forall k\}.$$

Por lo tanto, cuanto menor sea el indicador ϵ, mejor.

- La *distancia generacional* (*generational distance*, GD) mide lo lejos que están las soluciones del frente de Pareto obtenidas por el algoritmo A de las soluciones del Conjunto de Referencia (RS).

$$GD = \frac{\sqrt{\sum_{i=1}^{|A|} d_i^2}}{|A|}$$

donde d_i es la distancia euclídea entre cada una de estas soluciones y la solución más cercana del conjunto de Referencia. Si $GD = 0$ indica que todas las

soluciones de las soluciones eficientes encontradas por el algoritmo A están en el conjunto de Referencia.

- La *distancia generacional invertida* (*inverted generational distance*, IGD) es una inversión de la métrica de distancia generacional con el objetivo de medir la distancia desde el conjunto de Referencia al conjunto de soluciones eficientes obtenidas por el algoritmo A.

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{|RS|}.$$

  Cuanto menor sea el valor de IGD, mejor. Obsérvese que la única diferencia entre GD e IGD es que al comparar IGD no se pierde ninguna parte del frente de Pareto mejor/óptimo.

- Además, se presenta el tiempo de computación (CPU). Esta métrica indica el tiempo de procesamiento que un algoritmo necesita para encontrar una solución. Normalmente, se calcula el tiempo empleado para resolver cada instancia de forma individual y luego se reporta el valor medio de los tiempos de CPU para todas las instancias. Si dos algoritmos tienen una calidad media similar (medida, por ejemplo, por su desviación respecto al mejor valor conocido), se considera superior el que requiere menos tiempo para alcanzarla.

  Es importante señalar que, al medir el tiempo de CPU de un algoritmo, no se incluye el tiempo necesario para cargar las instancias en memoria ni el tiempo para mostrar los resultados; solo se computa el tiempo que el algoritmo emplea para encontrar la solución.

4.1.3. Herramientas empleadas

Como se mencionó anteriormente, un factor que influye en los resultados obtenidos durante la experimentación es el ordenador utilizado. Por lo tanto, al comparar diferentes algoritmos, es preferible que todos los resultados se hayan obtenido utilizando el mismo ordenador. Si esto no es posible, se puede comparar las tecnologías utilizadas en cada caso aplicando un factor de corrección a los tiempos. En particular, todos los experimentos descritos en este capítulo se han realizado utilizando un ordenador AMD Ryzen 9 5950x de 16 núcleos (3,4 GHz) con 128 GB de RAM.

Los algoritmos presentados en esta Tesis Doctoral han sido desarrollados utilizando el lenguaje de programación Java, específicamente Java SE 11.

Para la evaluación de la calidad de los resultados obtenidos por los algoritmos trayectoriales propuestos, éstos han sido comparados con los resultados de los algoritmos genéticos obtenidos por el *solver* MOEA Framework. La comparativa se ha realizado usando el *solver* MOMetrics.

4.2. Resultados experimentales de los algoritmos

En esta sección se muestran los resultados experimentales obtenidos por los algoritmos heurísticos propuestos en el Capítulo 3, aplicados a los conjuntos de instancias descritos en la sección 4.1.1. Los resultados presentados en esta sección se dividen en tres subsecciones.

Primero, se realizan una serie de experimentos preliminares (ver Sección 4.2.1) utilizando un pequeño porcentaje de las instancias de algunos de los conjuntos previamente mencionados. Esta decisión se toma para evitar el sobreajuste, excesiva adaptación a la muestra de ejemplo. Al disminuir el número de instancias y que las cuales sean dispares, asegura la genericidad de la parametrización del algoritmo. Estos experimentos ilustran aspectos como el rendimiento de los algoritmos constructivos, el impacto del ajuste de parámetros en las búsquedas locales, y las diferencias entre la combinación de búsquedas en las vecindades en el VND, entre otros, logrando con estos experimentos ajustar los parámetros del algoritmo y encontrar la mejor configuración.

Después de completar la experimentación preliminar, se lleva a cabo la experimentación final (ver Sección 4.2.2) utilizando los conjuntos completos de instancias. Los resultados obtenidos se comparan con los alcanzados por los algoritmos heurísticos del estado del arte, descritos en la Sección 2.2 para probar el rendimiento de nuestra propuesta.

Finalmente, se incluye una subsección donde se muestra y discute el caso de estudio que motiva este trabajo, la aplicación real.

Todas las tablas de la parte de experimentación preliminar y de la experimentación final se componen de una columna inicial que representa el algoritmo de caso de estudio, pudiendo ser representado por valores del parámetro α, vecindades o por el nombre del propio algoritmo. En las siguientes columnas se muestran las distintas métricas utilizadas para medir la calidad de los algoritmos. Dichas métricas han sido explicadas con anterioridad en la Sección 4.1.2, las cuales son cardinalidad, cobertura, dispersión, hipervolumen, indicador ϵ, la distancia generacional, distancia generacional invertida y tiempo de computación.

Las tablas de la parte de experimentación real se componen de una columna inicial con el algoritmo de caso de estudio y cuatro columnas más en representación de los valores de los cuatro objetivos: coste o distancia de recorrido expresado en kilómetros, latencia expresada en horas, el número de clientes servidos y el beneficio expresado en la moneda europea Euro.

4.2.1. Experimentación preliminar

Como se ha comentado anteriormente, en este subapartado se procederá al ajuste de los parámetros de nuestro algoritmo. Para ello, en lugar de utilizar el conjunto total de 231 instancias, y con el fin de evitar el sobreajuste, se ha seleccionado

aleatoriamente un subconjunto representativo con un 25 % de las instancias con diferentes características. Nótese que los valores se promedian entre el conjunto de instancias consideradas.

Ajuste de parámetros del algoritmo constructivo

El algoritmo GRA basa su rendimiento en un delicado equilibrio entre explotación (voracidad) y exploración (aleatoriedad). Este equilibrio se regula mediante el parámetro $\alpha \in [0, 1]$, el cual controla la construcción de soluciones iniciales a través de una lista restringida de candidatos (RCL). En cada iteración de la construcción, los elementos candidatos se ordenan según una función heurística, y se forma una RCL con aquellos cuyo coste es mejor que un umbral definido por α. A continuación, un elemento se selecciona aleatoriamente de esta lista.

El objetivo de este experimento es analizar el impacto del parámetro α en el comportamiento y calidad de las soluciones generadas por la construcción GRA. Para ello, se evalúan distintos valores de α:

- $\alpha = 0{,}00$: construcción completamente determinista, donde se elige siempre el mejor candidato.
- $\alpha = 0{,}25, 0{,}50, 0{,}75$: niveles progresivos de aleatoriedad controlada, que introducen variabilidad en la selección de candidatos.
- $\alpha = 1{,}00$: construcción completamente aleatorizada, sin ninguna preferencia por candidatos de mejor calidad.
- RND: se escoge un valor de α completamente al azar en cada iteración de la construcción de las soluciones.

Esta evaluación permitirá identificar qué valores de α proporcionan un mejor compromiso entre calidad de las soluciones y diversidad del frente obtenido.

La Tabla 4.2 muestra los resultados obtenidos al ajustar el parámetro α del algoritmo constructivo GRA, utilizando un subconjunto representativo del 25 % de las instancias para evitar el sobreajuste.

α	$\lvert\hat{PF}\rvert$	CV	Δ	HV	ϵ	GD	IGD	CPU
0,00	573,7333	**0,0034**	**0,1997**	**0,3136**	**0,0186**	**0,0000**	**0,0003**	**0,0362**
0,25	573,7333	0,9956	0,5208	0,0140	0,7178	0,2378	0,6078	0,0998
0,50	573,7333	0,9979	0,5110	0,0075	0,7714	0,2891	0,6431	0,0960
0,75	573,7333	0,9975	0,5120	0,0065	0,7871	0,3197	0,6514	0,0918
1,00	573,7333	0,9962	0,5109	0,0056	0,8035	0,3530	0,6584	0,0981
RND	573,7333	0,9964	0,5423	0,0124	0,7280	0,2331	0,6115	0,0984

Tabla 4.2: Construcción GRA para diferentes valores de α

Como se observa, el valor $\alpha = 0$ destaca de forma clara y consistente como la mejor configuración, al obtener los mejores resultados en todas las métricas evaluadas. En primer lugar, aunque la cardinalidad del frente de Pareto aproximado

($|\hat{PF}|$) es la misma para todos los valores de α, lo cual indica que todos los escenarios generan un número equivalente de soluciones eficientes, lo verdaderamente relevante es la calidad de dichas soluciones. En este sentido, el valor de cobertura (CV) para $\alpha = 0$ es prácticamente cero (0,0034), lo que implica que casi ninguna solución del conjunto de referencia domina a las soluciones obtenidas por el algoritmo, evidenciando su elevada calidad. Además, la dispersión (Δ) asociada a este valor es la más baja (0,1997), lo que indica una mejor distribución de las soluciones a lo largo del frente de Pareto, sin acumulaciones o vacíos. Esta uniformidad se ve reforzada por el valor más alto del hipervolumen (0,3136), que refleja una cobertura más amplia del espacio de soluciones eficientes.

En cuanto a los indicadores de precisión frente al conjunto de referencia, $\alpha = 0$ vuelve a posicionarse como el mejor valor. El indicador ϵ alcanza su mínimo (0,0186), señalando que las soluciones generadas están extremadamente próximas a las del conjunto óptimo. La distancia generacional (GD) es exactamente cero, lo cual indica que todas las soluciones del algoritmo pertenecen también al conjunto de referencia, es decir, el algoritmo ha logrado replicar el frente de Pareto conocido. Asimismo, la distancia generacional invertida (IGD) es también mínima (0,0003), reforzando la idea de que el frente generado no solo es preciso, sino también completo respecto al óptimo. Finalmente, el tiempo de CPU es el más bajo de todos los casos (0,0362), lo que convierte esta configuración no solo en la más efectiva, sino también en la más eficiente en términos computacionales.

En contraste, el resto de valores de α, incluyendo la variante aleatoria (RND), presentan resultados considerablemente inferiores. Las métricas de calidad (CV, HV, ϵ, GD, IGD) y de distribución (Δ) empeoran de forma significativa, y además, el tiempo de cómputo es mayor. Por tanto, no solo se compromete la calidad de las soluciones generadas, sino que también se incrementa el coste computacional. En consecuencia, puede concluirse que el valor $\alpha = 0$ es claramente la opción óptima para configurar el algoritmo constructivo GRA, ya que maximiza la calidad de las soluciones, garantiza una buena diversidad y cobertura del frente de Pareto, y reduce al mínimo el tiempo de ejecución.

Soluciones obtenidas tras el proceso de construcción y mejora

El primer experimento está dedicado a comprobar si cada estructura de vecindad aplicada por separado es capaz de mejorar el conjunto inicial de soluciones eficientes que forman parte del Frente de Pareto. Para ello, partimos de una aproximación inicial del Frente de Pareto que se ha construido utilizando un algoritmo totalmente voraz. La búsqueda local partirá de cada solución eficiente que forme parte de la aproximación inicial del Pareto y la irá mejorando progresivamente. El algoritmo sigue una estrategia de primera mejora, es decir, realiza el primer movimiento que produce una mejora en la solución inicial. Aquí volvemos a resaltar el hecho de que hemos considerado una mejora si y sólo si la nueva solución encontrada domina estrictamente a la solución hallada hasta el momento. De esta forma, en lugar de comparar la nueva solución encontrada con todas las soluciones eficientes del Frente de Pareto, la comparación se hace sólo entre pares de soluciones y eso consume menos tiempo. Por supuesto, cuando finalice la búsqueda local y se

hayan comparado todos los pares de soluciones del mismo tamaño para comprobar si se ha encontrado una mejora (en el sentido de dominación estricta), se realizará un último paso. Este paso consiste en comprobar si todas las soluciones del Frente de Pareto son soluciones no dominadas.

Según los resultados de la Tabla 4.3 todos los vecindarios por separado son capaces de mejorar la aproximación inicial del Frente de Pareto obtenido a partir del mismo método constructivo totalmente voraz denotado como GRA en la tabla. En el experimento, el VND se ejecuta durante una única iteración hasta alcanzar el vecindario máximo. Por supuesto, cuanto mayor sea la complejidad de los vecindarios, más tiempo consumirá el algoritmo VND. Como podemos ver el operador 2-opt ($\mathcal{N}_3$ mostrado en la última fila) es el que consume más tiempo y además, el que es capaz de obtener mejores resultados en casi todas las métricas, excepto en la dispersión y el número de soluciones eficientes en el Frente de Pareto. Sin embargo, los otros dos vecindarios, $\mathcal{N}_1$ y $\mathcal{N}_2$, también son capaces de mejorar la aproximación inicial del Frente de Pareto obtenida por el GRA ya que todas las métricas de rendimiento son mejores. En particular, la cobertura de GRA es casi 1, lo que indica que la mayoría de las soluciones iniciales han sido mejoradas por cualquiera de los vecindarios y ahora son dominadas, destacando el rendimiento de la exploración de vecindarios. Cabe mencionar que cuando aplicamos las vecindades $\mathcal{N}_1$ y $\mathcal{N}_2$ los indicadores no muestran la misma calidad que aplicar la vecindad $\mathcal{N}_3$, pero sin embargo el tiempo de computación de esta última tarda mucho más.

	$\lvert\hat{PF}\rvert$	CV	Δ	HV
GRA	**573,7333**	0,9785	0,2957	0,2516
$\mathcal{N}_1$	571,3778	0,6562	**0,2703**	0,2654
$\mathcal{N}_2$	573,1556	0,7162	0,2861	0,2512
$\mathcal{N}_3$	546,2000	**0,1293**	0,2928	**0,2904**

	ϵ	GD	IGD	CPU
GRA	0,0881	0,0048	0,0456	**0,0362**
$\mathcal{N}_1$	0,0751	0,0030	0,0335	80,1547
$\mathcal{N}_2$	0,0892	0,0049	0,0453	55,3366
$\mathcal{N}_3$	**0,0593**	**0,0009**	**0,0121**	942,0975

Tabla 4.3: Contribución de las vecindades

Ajuste del orden de las vecindades

Otra decisión importante está relacionada con establecer el orden en el que se aplicarán las vecindades. Para llevar a cabo este experimento se ha partido nuevamente de la aproximación del Frente de Pareto obtenida con el algoritmo totalmente voraz, GRA. La Tabla 4.4 presenta los resultados del experimento llevado a cabo para determinar el orden más adecuado en la aplicación de las vecindades dentro del marco del procedimiento VND. A pesar de que las diferencias entre las configuraciones no son enormes, se puede observar que el orden de aplicación $\mathcal{N}_3+\mathcal{N}_1+\mathcal{N}_2$ proporciona un rendimiento superior en un conjunto amplio de métricas de calidad,

diversidad y eficiencia. Este orden comienza con el operador 2-opt ($\mathcal{N}_3$), que tiene una mayor complejidad computacional, seguido por la eliminación e inserción de un nodo ($\mathcal{N}_1$), y finaliza con el operador 2-intercambio ($\mathcal{N}_2$), el más ligero en términos de coste computacional. Este enfoque contrasta con la hipótesis tradicionalmente aceptada en la literatura, que sugiere comenzar por los operadores más simples y rápidos. Sin embargo, nuestros resultados confirman hallazgos similares reportados por autores como Waste-López [16], en los que se observa que iniciar la búsqueda local con vecindarios más potentes puede permitir un salto de calidad más significativo en las primeras etapas de la búsqueda, generando soluciones que luego se refinan eficientemente con operadores más simples.

	$\lvert\hat{PF}\rvert$	CV	Δ	HV
$\mathcal{N}_1+\mathcal{N}_2+\mathcal{N}_3$	523,3111	0,6079	0,3395	**0,2827**
$\mathcal{N}_1+\mathcal{N}_3+\mathcal{N}_2$	527,2444	**0,5688**	0,3216	0,2831
$\mathcal{N}_2+\mathcal{N}_1+\mathcal{N}_3$	527,4444	0,5916	0,3283	0,2842
$\mathcal{N}_2+\mathcal{N}_3+\mathcal{N}_1$	**529,7111**	0,6008	0,3097	0,2831
$\mathcal{N}_3+\mathcal{N}_1+\mathcal{N}_2$	527,7333	0,5722	**0,3003**	0,2861
$\mathcal{N}_3+\mathcal{N}_2+\mathcal{N}_1$	527,0222	0,5722	0,3113	0,2853

	ϵ	GD	IGD	CPU
$\mathcal{N}_1+\mathcal{N}_2+\mathcal{N}_3$	0,0723	0,0020	0,0270	643,0006
$\mathcal{N}_1+\mathcal{N}_3+\mathcal{N}_2$	0,0718	0,0022	0,0263	734,4080
$\mathcal{N}_2+\mathcal{N}_1+\mathcal{N}_3$	0,0734	0,0021	0,0248	719,8058
$\mathcal{N}_2+\mathcal{N}_3+\mathcal{N}_1$	0,0672	**0,0018**	0,0248	704,8509
$\mathcal{N}_3+\mathcal{N}_1+\mathcal{N}_2$	**0,0639**	**0,0018**	**0,0238**	499,8112
$\mathcal{N}_3+\mathcal{N}_2+\mathcal{N}_1$	0,0648	0,0019	0,0246	**474,2296**

Tabla 4.4: Orden de las vecindades en el marco del VND

Analizando los resultados en detalle, se puede ver que esta configuración alcanza valores especialmente competitivos en métricas clave como la dispersión (0,3003), el hipervolumen (0,2861), el indicador ϵ (0,0639), la distancia generacional (0,0018) y la distancia generacional invertida (0,0238), véase Tabla 4.4 fila 5. Estos valores evidencian que el frente de Pareto aproximado es no solo diverso y bien distribuido, sino también cercano al conjunto de referencia y con buena cobertura del espacio de soluciones. Además, y de forma destacable, el tiempo de CPU es uno de los más bajos (499,8112), lo cual subraya la eficiencia de esta estrategia en comparación con otros órdenes evaluados, que aunque puedan obtener rendimientos similares en ciertas métricas, lo hacen con un coste computacional notablemente mayor.

Por tanto, aunque las diferencias entre configuraciones no son grandes, el orden $\mathcal{N}_3+\mathcal{N}_1+\mathcal{N}_2$ logra un buen equilibrio entre calidad, diversidad y eficiencia, consolidándose como la mejor alternativa para estructurar la exploración de vecindarios dentro del VND. Esta elección no solo mejora la calidad de las soluciones obtenidas, sino que también optimiza el uso de los recursos computacionales, lo cual es especialmente relevante en escenarios de alta exigencia computacional como los tratados en este trabajo.

Ajuste de parámetros del constructivo GRA junto al algoritmo VND

En esta sección se analiza el impacto que tiene la calidad de la solución inicial generada por el algoritmo constructivo Greedy Randomized Adaptive (GRA) sobre el rendimiento del algoritmo híbrido GRA+VND aplicado para resolver problemas similares por [16]. Para ello, se han evaluado diferentes configuraciones del parámetro α, que controla el equilibrio entre la voracidad y la aleatoriedad en la fase de construcción inicial. Como puede observarse en la Tabla 4.5, los resultados indican de forma concluyente que la calidad del Frente de Pareto aproximado depende de manera muy significativa del valor de α utilizado durante la construcción. Gracias a la función voraz definida en la Sección 3.2.1, el mejor rendimiento se obtiene cuando $\alpha = 0$, es decir, cuando la construcción es completamente voraz, lo que se traduce en una selección estricta del nodo más prometedor en cada paso. Esta configuración supera ampliamente al resto en todos los indicadores considerados, obteniendo el mayor número de soluciones no dominadas (527,7333), la mejor cobertura (0,0622), una excelente dispersión (0,2856), el mayor hipervolumen (0,2949), así como los mejores valores para los indicadores de proximidad al frente óptimo: un ϵ de 0,0341, una distancia generacional de 0,0004 y una distancia generacional invertida de 0,0041. Además, es notable que esta configuración también requiere el menor tiempo de cómputo (499,8112), consolidando su superioridad no solo en términos de calidad de soluciones, sino también de eficiencia computacional.

α	$\lvert\hat{PF}\rvert$	CV	Δ	HV
0,00	**527,7333**	**0,0622**	**0,2856**	**0,2949**
0,25	401,1556	0,8942	0,5094	0,1805
0,50	406,0444	0,9060	0,5181	0,1743
0,75	411,6222	0,9229	0,5192	0,1712
1,00	411,2667	0,9131	0,5121	0,1699
RND	401,9778	0,8806	0,5291	0,1812

α	ϵ	GD	IGD	CPU
0,00	**0,0341**	**0,0004**	**0,0041**	**499,8112**
0,25	0,2729	0,2056	0,1339	1706,6703
0,50	0,2873	0,2459	0,1447	1752,7561
0,75	0,3020	0,2764	0,1508	1759,8881
1,00	0,3030	0,2980	0,1531	1769,9987
RND	0,2778	0,2039	0,1356	1738,4523

Tabla 4.5: Construcción GRA combinado con el algoritmo VND

En contraste, mayores valores de α, que introducen distintos grados de aleatoriedad en la construcción inicial, conducen sistemáticamente a resultados peores. A medida que se incrementa la aleatoriedad, desde $\alpha = 0{,}25$ hasta $\alpha = 1$, e incluso con el valor aleatorio RND, se observa una degradación clara en todas las métricas. Por ejemplo, los valores de CV superan el 0,88, lo que indica una pobre cobertura del espacio objetivo, y tanto Δ como HV muestran que las soluciones son más dispersas y menos representativas del frente real. Los valores de GD e IGD también son significativamente más altos, reflejando que las soluciones finales están más

alejadas del conjunto óptimo. Además, el tiempo de cómputo se multiplica por más de tres en estas configuraciones, alcanzando valores superiores a 1700 segundos, lo cual representa un coste computacional muy elevado que, además, no se traduce en mejoras de rendimiento.

Estos resultados ponen de manifiesto la importancia de una buena inicialización del algoritmo. La construcción inicial determina no solo el punto de partida desde el que se explora el espacio de soluciones mediante el VND, sino que también condiciona la eficacia con la que este puede refinar las soluciones. Así, una construcción sólida y orientada por criterios voraces permite comenzar la búsqueda local desde soluciones prometedoras, facilitando que el VND profundice eficazmente en regiones de alta calidad. En cambio, las inicializaciones más aleatorias no logran alcanzar regiones competitivas del espacio de soluciones, por lo que el VND actúa sobre soluciones de menor calidad y, por tanto, con menor margen de mejora. En definitiva, la configuración con $\alpha = 0$ no solo proporciona los mejores resultados finales, sino que reafirma la validez de estrategias completamente voraces para la inicialización cuando se combinan con procedimientos locales intensivos como el VND.

Ajuste de parámetros del límite de exploración de las vecindades

En este último experimento se analiza el efecto de limitar el porcentaje de nodos explorados durante la búsqueda en cada vecindad, con el objetivo de reducir el tiempo de ejecución del algoritmo sin comprometer en exceso la calidad de las soluciones. La lógica de esta decisión radica en que, en muchas ocasiones, la evaluación completa de todos los posibles movimientos de todos los nodos puede resultar innecesariamente costosa, especialmente cuando muchos de estos movimientos tienen pocas probabilidades de mejorar la solución, ya que están muy alejados y no resultan prometedores. Así, se ha incorporado un parámetro γ que determina la proporción de nodos más prometedores (según proximidad) que serán considerados como candidatos a moverse, evaluándose los valores $\gamma = 0,25, 0,50, 0,75$ y 1, véase la Tabla 4.6.

γ	$\lvert\hat{PF}\rvert$	CV	Δ	HV
0,25	531,0667	0,5406	**0,2922**	0,2778
0,50	520,6667	0,5459	0,3189	0,2735
0,75	**537,9333**	**0,3827**	0,2985	0,2785
1,00	527,7333	0,4792	0,3020	**0,2804**

γ	ϵ	GD	IGD	CPU
0,25	0,0627	**0,0016**	0,0205	**191,8928**
0,50	0,0696	0,0020	0,0244	365,8388
0,75	0,0642	0,0017	0,0197	452,0900
1,00	**0,0624**	**0,0016**	**0,0194**	499,8112

Tabla 4.6: Limitación porcentual de las vecindades

Los resultados de la Tabla 4.6 confirman que explorar todas las vecindades

($\gamma = 1$) sigue siendo la opción más sólida si el objetivo principal es obtener la mayor calidad posible en la aproximación del Frente de Pareto. En particular, esta configuración alcanza los mejores valores en tres de las métricas más relevantes: el hipervolumen (0,2804), la distancia generacional invertida (0,0194) y la métrica ϵ (0,0624), además de compartir el mejor valor de GD (0,0016). Estos datos sugieren que, como era de esperar, una mayor exploración conduce a soluciones más cercanas al frente óptimo y más representativas en el espacio objetivo. No obstante, esta calidad viene acompañada del mayor tiempo de cómputo (499,8112 segundos), lo que puede resultar prohibitivo en instancias de gran tamaño.

Frente a esto, la opción más restrictiva ($\gamma = 0,25$) destaca por reducir el tiempo de cómputo en un 61,61 %, situándose en tan solo 191,8928 segundos, y sorprendentemente sin un deterioro significativo en la calidad. De hecho, el número de soluciones no dominadas obtenidas (531,0667) es el segundo más alto entre todas las configuraciones, y la métrica de dispersión (0,2922) alcanza su mejor valor. Aunque otras métricas como CV (0,5406), HV (0,2778) o IGD (0,0205) no son las mejores absolutas, sí mantienen un nivel de calidad muy competitivo, sobre todo teniendo en cuenta la drástica reducción en el tiempo de ejecución. Esto sugiere que una reducción del 75 % en el número de nodos evaluados puede ser una estrategia eficaz para obtener soluciones razonablemente buenas en un tiempo mucho menor.

En definitiva, este análisis revela una clara relación entre el porcentaje de nodos explorados y la eficiencia del algoritmo. Si bien la exploración completa garantiza los mejores resultados, la opción de limitar la búsqueda ofrece una alternativa muy atractiva cuando se desea un compromiso entre calidad y eficiencia computacional. Esta opción es especialmente valiosa en problemas de gran escala, donde los tiempos de ejecución pueden ser un factor crítico. Por ello, aunque esta estrategia no se ha incluido como parte integral del algoritmo, se propone como una herramienta de ajuste que puede activarse de forma flexible en función del contexto del problema y de los recursos disponibles.

Ajuste de α con los parámetros finales de VND

Para cerrar esta sección, este experimento tiene como finalidad verificar si la estrategia constructiva más determinista, basada en un enfoque totalmente voraz ($\alpha = 0$), sigue siendo la mejor opción incluso cuando se introduce una limitación en la exploración de vecindades, concretamente restringiendo los movimientos a solo el 25 % de los nodos más cercanos. La hipótesis subyacente es que, aunque la exploración de vecindarios se haya acotado para mejorar la eficiencia computacional, la calidad de la solución inicial sigue jugando un papel fundamental en el rendimiento global del algoritmo híbrido.

Los resultados obtenidos en la Tabla 4.7 refuerzan con claridad esta idea: el valor $\alpha = 0$ continúa destacando en todas las métricas de calidad multiobjetivo analizadas. Se alcanza la mayor cardinalidad del Frente de Pareto, lo que indica un conjunto más amplio de soluciones no dominadas, y se obtienen los mejores valores en cobertura, dispersión, hipervolumen, así como en los indicadores basados en distancia (ϵ, GD e IGD), todos ellos reflejo de una aproximación más precisa y

α	$\lvert\hat{PF}\rvert$	CV	Δ	HV
0,00	**531,0667**	**0,0545**	**0,2833**	**0,3010**
0,25	364,1111	0,8880	0,5227	0,2043
0,50	373,5111	0,9143	0,5272	0,1957
0,75	377,2222	0,9195	0,5206	0,1905
1,00	379,5111	0,9197	0,5296	0,1895
RND	359,4667	0,8931	0,5396	0,2030

α	ϵ	GD	IGD	CPU
0,00	**0,0301**	**0,0003**	**0,0032**	**191,8928**
0,25	0,2251	0,2011	0,1102	1547,5061
0,50	0,2438	0,2444	0,1227	1583,6511
0,75	0,2552	0,2705	0,1286	1601,1642
1,00	0,2584	0,2908	0,1313	1616,5514
RND	0,2261	0,2080	0,1122	1552,2770

Tabla 4.7: Diferentes valores de α para GRA con los movimientos de las vecindades limitados

diversificada del frente óptimo. Además, este rendimiento excepcional se logra con el menor tiempo de cómputo registrado entre todas las configuraciones probadas, gracias a la limitación del tamaño de las vecindades exploradas.

Por el contrario, conforme se incrementa el valor de α, es decir, se introduce un mayor grado de aleatoriedad en el proceso constructivo, se observa un claro empeoramiento de los resultados: se reduce la calidad de las soluciones y aumenta considerablemente el tiempo de ejecución, sin obtener ningún beneficio aparente. Esto pone de manifiesto que la aleatoriedad en la construcción inicial no favorece el proceso de refinamiento posterior realizado por el algoritmo VND, y que, por tanto, una construcción guiada por criterios estrictamente voraces no solo es compatible con estrategias de búsqueda parcial, sino que resulta incluso potenciada por ellas. En consecuencia, puede afirmarse con solidez que el valor óptimo del parámetro α es cero, ya que permite generar una solución inicial de gran calidad sobre la que el algoritmo de mejora local puede actuar de forma más efectiva, maximizando así la calidad de la aproximación final del Frente de Pareto sin incurrir en un coste computacional elevado.

4.2.2. Experimentación final

Tras configurar y parametrizar el algoritmo propuesto utilizando un algoritmo constructivo totalmente voraz ya que el mejor α en el procedimiento GRA es igual a cero, y un VND con un orden de vecindades $\mathcal{N}_3 + \mathcal{N}_1 + \mathcal{N}_2$, y un limite de exploración del 25 % de los nodos más cercanos, en esta sección se presentan y discuten los resultados de las pruebas computacionales realizadas al comparar la propuesta algorítmica frente a tres de los algoritmos evolutivos multiobjetivo más utilizados: elitist Non-dominated Sorting Genetic Algorithm (NSGA-II), Multi-Objective Evolutio-

nary Algorithm based on Decomposition (MOEA/D), y Strength Pareto Evolutionary Algorithm (SPEA2). Estos algoritmos genéticos se han resuelto utilizando un software abierto MOEA Framework (http://moeaframework.org):

Comparativa con los métodos del estado del arte

A continuación, se muestran los resultados computacionales de forma similar a todos los resultados de la sección anterior. Como las instancias de la literatura tienen un amplio rango de tamaños hemos considerado todas las instancias (231 instancias) donde la más pequeña tiene 51 nodos y la más grande 18512 nodos como se introdujo anteriormente.

Como se puede observar en la Tabla 4.8, de media nuestra propuesta algorítmica consume 875 segundos aproximadamente. Para obtener una comparación justa, se ha incluido un límite de tiempo similar como criterio de parada para los algoritmos evolutivos multiobjetivo. La población inicial y el número de evaluaciones se cuadró para que diera el mismo tiempo, y el resto de parámetros se dejaron por defecto.

Los resultados experimentales, mostrados en la Tabla 4.8, permiten extraer conclusiones claras: el algoritmo VND no solo logra generar un número significativamente mayor de soluciones eficientes, sino que también presenta mejoras sustanciales en todas las métricas consideradas. En términos de cobertura (CV), dispersión (Δ), hipervolumen (HV) y los tres indicadores de distancia (ϵ, GD e IGD), el rendimiento del VND supera con holgura al del resto de métodos, reflejando una mayor precisión y diversidad en la aproximación del Frente de Pareto. Esto es especialmente notable en métricas como el hipervolumen o el indicador IGD, que capturan tanto la calidad como la distribución de las soluciones.

En conjunto, los resultados consolidan la propuesta como una alternativa altamente competitiva, especialmente adecuada cuando se dispone de conocimiento estructural del problema y se desea una aproximación de alta calidad en tiempos razonables.

Algoritmo	$\lvert\hat{PF}\rvert$	CV	Δ	HV
VND	**736,4848**	**0,0594**	**0,5038**	**0,4912**
MOEA/D	664,8268	0,5705	0,7572	0,3859
NSGA-II	675,7446	0,5979	0,7599	0,3829
SPEA2	453,5065	0,6884	0,7607	0,2795

Algoritmo	ϵ	GD	IGD	CPU
VND	**0,3607**	**0,0005**	**0,0806**	875,4405
MOEA/D	0,3847	0,0009	0,1713	754,0176
NSGA-II	0,3874	0,0010	0,1723	757,6662
SPEA2	0,4539	0,0085	0,2361	**752,8614**

Tabla 4.8: Comparación de VND con los algoritmos evolutivos multiobjetivo más utilizados.

4.2.3. Experimentación real

Una vez demostrado el rendimiento de nuestra propuesta en el apartado anterior, se procede a resolver la aplicación del mundo real utilizando el algoritmo propuesto donde 103 clientes requieren un servicio.

Como se ha explicado anteriormente, el problema aparece en una situación del mundo real en la que un autónomo, que repara electrodomésticos, tiene interés en planificar la secuencia en la que todos los clientes o un subconjunto de ellos serán atendidos. Para ello, el autónomo proporciona una lista con información sobre los clientes como la localización, para medir distancias y tiempos de desplazamiento, una estimación del tiempo de servicio según la incidencia, es decir, el tiempo de reparación de la avería, y por último, el beneficio obtenido una vez finalizada la reparación. Para mostrar el funcionamiento de la propuesta se ha seleccionado aleatoriamente uno de los días de trabajo del autónomo con más solicitudes. El autónomo necesita un algoritmo para planificar el servicio diario en pocos segundos y, por tanto, nuestra propuesta es una buena herramienta para ello.

Al autónomo le gustaría obtener el máximo beneficio a la mínima distancia, y por otro lado, a los clientes les gustaría ser atendidos esperando lo mínimo posible, por lo que los objetivos entran en conflicto.

El algoritmo es capaz de obtener el Frente de Pareto en 0,204 segundos. Además, el algoritmo presenta 103 soluciones eficientes con todos los tamaños de solución posibles (visitando desde un solo cliente hasta una solución con todos los clientes). A continuación, se representan 4 de las 103 soluciones. Se han seleccionado de acuerdo con 4 escenarios posibles y realistas según las 4 latencias totales posibles (que pueden verse como la duración de la jornada laboral):

- Figura 4.1 muestra la planificación cuando se atiende a 46 clientes ya que su latencia total es cercana a las 24 horas. Por otro lado, el autónomo obtendrá un beneficio total de 2499 unidades monetarias recorriendo 24,262 kilómetros.
- La Figura 4.2 muestra la planificación cuando se atiende a 26 clientes. La latencia total esta entorno a 12 horas. La distancia recorrida sería de 20,516 kilómetros, obteniendo un beneficio total de 1375 unidades monetarias.
- Para una jornada completa, es decir, un turno de 8 horas, la Figura 4.3 muestra la planificación visitando a 20 clientes. En ese caso, la distancia sería de 19,192 kilómetros y el beneficio total de 1029 unidades monetarias.
- Para una jornada parcial, es decir, turno de 4 horas, la Figura 4.4 muestra la planificación en dicha situación visitando 13 clientes, recorriendo 18,638 kilómetros y obteniendo un beneficio más reducido, 720 unidades monetarias.

Comparación algoritmo voraz contra VND

A continuación, se compara el algoritmo voraz con el algoritmo propuesto VND para ver de forma gráfica cual es la aportación del VND respecto de la solución inicial. se incluye la Tabla 4.9 para mostrar cómo nuestro algoritmo VND supera

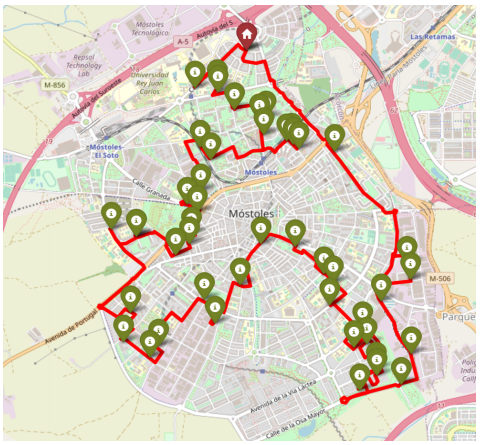


Figura 4.1: Planificación de un turno de 24 horas.

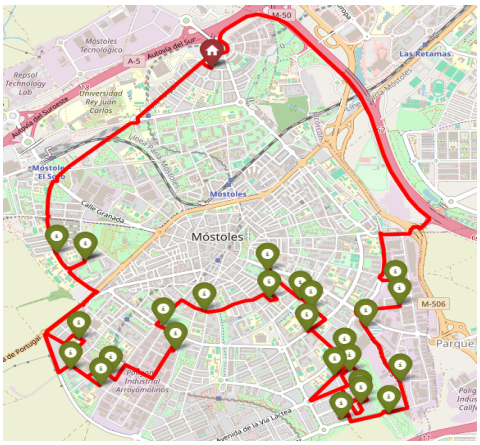


Figura 4.2: Planificación de un turno de 12 horas.

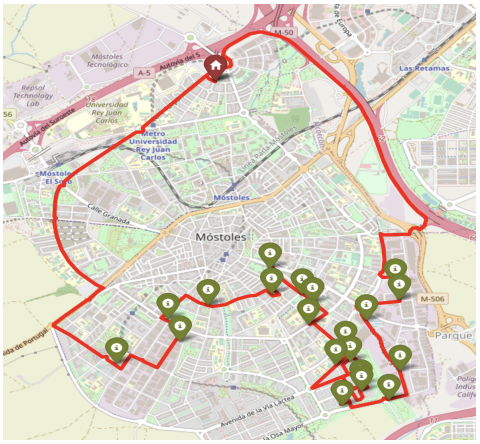


Figura 4.3: Planificación de un turno de 8 horas.

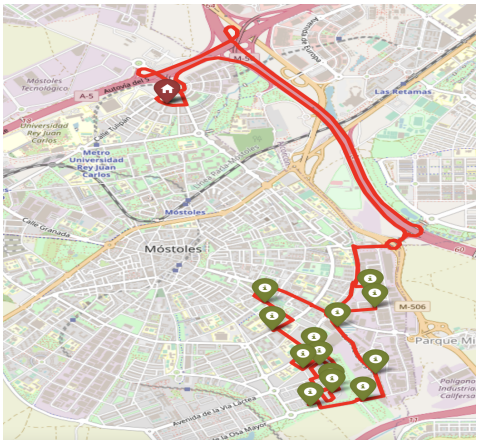


Figura 4.4: Planificación de un turno de 4 horas.

al algoritmo voraz a la hora de resolver el problema del mundo real. A la vista de los resultados obtenidos y como era de esperar, cuanto mayor es el número de clientes atendidos, más difícil es resolver el problema considerado y, por tanto, mejor funciona nuestro algoritmo VND. En todos los casos, el algoritmo VND domina al algoritmo voraz excepto cuando la latencia es de 4 horas que ambos algoritmos obtienen exactamente la misma solución. Esto se ve visualmente en las Figuras 4.5 y 4.6, donde se observa el recorrido de 4 horas del algoritmo voraz y del algoritmo VND.

Seguidamente se muestran representaciones gráficas de las rutas por cada limite de horas.

En la solución estudiada para 8 horas, se observa como al ir complicándose la solución, el algoritmo VND funciona mejor que el algoritmo voraz. Esto se aprecia sobretodo el la Tabla 4.9 filas 3 y 4. Esto representado en un mapa se puede observar en una sutil diferencia entre la Figura 4.7 y la Figura 4.8 en la parte inferior.

Del mismo modo, con soluciones con latencia máxima a 12 horas también se aprecia una mejoría en el algoritmo VND frente al algoritmo voraz en lo que respecta

Algoritmo	Coste (km)	Latencia (h)	Clientes servidos	Beneficio (€)
Voraz	18,638	4	13	720
VND	18,638	4	13	720
Voraz	19,512	8	20	1029
VND	**19,192**	8	20	1029
Voraz	21,306	12	25	1327
VND	**20,516**	12	**26**	**1375**
Voraz	**24,091**	24	37	1988
VND	24,262	24	**43**	**2331**

Tabla 4.9: El algoritmo voraz contra el algoritmo VND.

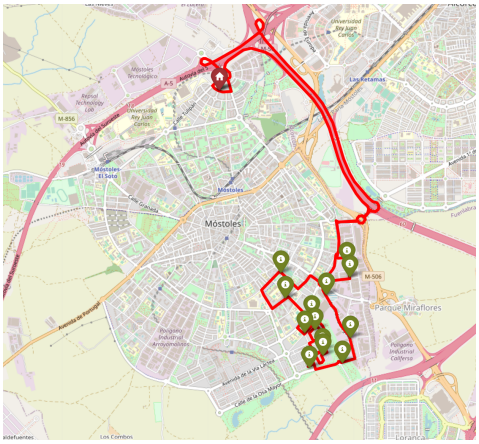


Figura 4.5: Planificación de un turno de 4 horas realizada con el algoritmo voraz.

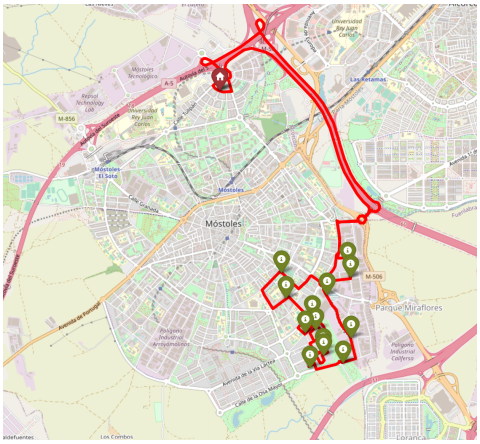


Figura 4.6: Planificación de un turno de 4 horas realizada con el algoritmo VND.

al coste del viaje mostrado en las Figuras 4.9 y 4.10. También da servicio a un cliente más resultando en un mayor beneficio.

Esta mejora que introduce la exploración de vecindades se vuelve especialmente significativa a medida que aumenta el número de clientes en la solución. Cuando se trabaja con un conjunto reducido de clientes, las posibilidades de mejora son más limitadas, ya que la estructura de la solución inicial suele ser razonablemente buena y las oportunidades de reorganización que aporten un beneficio notable son escasas. Sin embargo, en soluciones con una gran cantidad de clientes, permite que los movimientos exploratorios en las distintas vecindades identifiquen rutas sustancialmente más eficientes. Esta mayor capacidad de mejora se traduce directamente en una reducción significativa del coste total del viaje. Al disminuir este coste, se libera capacidad operativa del autónomo, lo que permite ampliar el alcance del servicio. En consecuencia, el autónomo logra atender a un mayor número de clientes sin comprometer las restricciones impuestas por el problema, lo que impacta positivamente en el beneficio global obtenido por la solución. En definitiva, la exploración de vecindades actúa como un mecanismo refinador que permite escalar el rendimiento del algoritmo a medida que crece la complejidad del problema, demostrando ser una herramienta clave para la obtención de soluciones de alta calidad en contextos reales y de gran escala.

Figura 4.7: Planificación de un turno de 8 horas realizada con el algoritmo voraz.

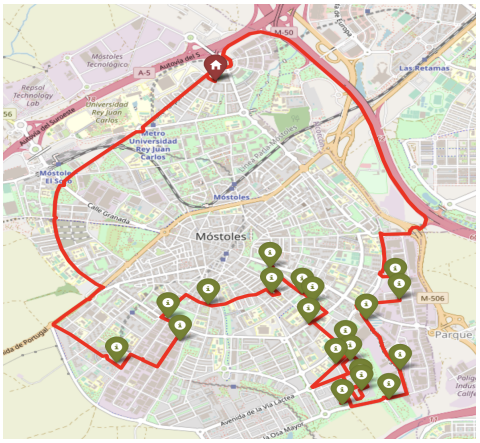


Figura 4.8: Planificación de un turno de 8 horas realizada con el algoritmo VND.

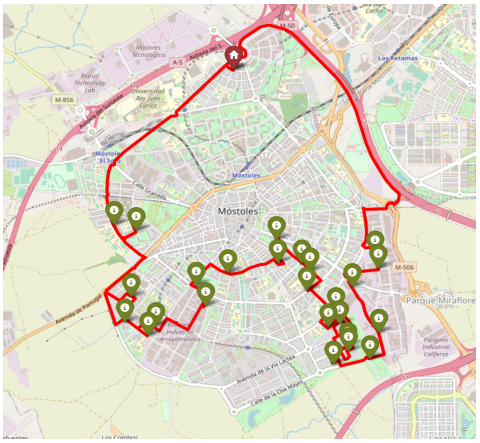


Figura 4.9: Planificación de un turno de 12 horas realizada con el algoritmo voraz.

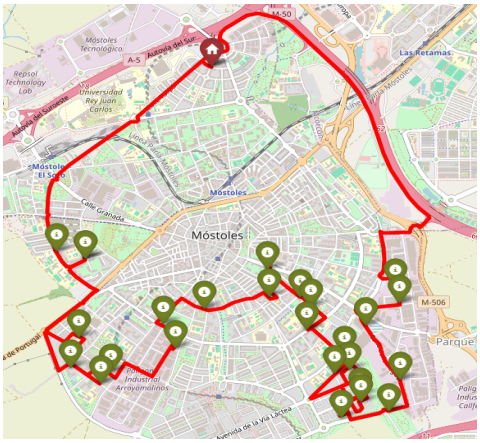


Figura 4.10: Planificación de un turno de 12 horas realizada con el algoritmo VND.

Comparativa gráfica con los métodos del estado del arte

A continuación, se muestran representaciones sobre mapa de las soluciones obtenidas en la ejecución de la instancia real con la propuesta algorítmica y con los algoritmos genéticos antes mencionados. Se han tenido que tomar varias consideraciones debido a la dificultad de comprar un algoritmo trayectorial con algoritmos genéticos para tiempos pequeños de ejecución. Esto viene motivado por los resultados previsibles mostrados en la Tabla 4.10. La resolución de la instancia real por parte del algoritmo propuesto tiene un coste computacional de 0,2 segundos. Para ser justos se utiliza como criterio de parada el tiempo computacional.

Debido a que los resultados de los algoritmos genéticos están muy alejados de los arrojados por el VND, y que la finalidad de esta sección es comparar las soluciones dadas por los algoritmos de una forma visual, se concede una ejecución más prolongada a los algoritmos genéticos, exactamente de 1000 segundos (16 minutos

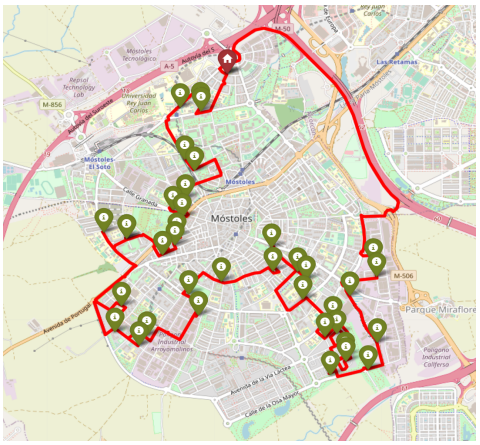


Figura 4.11: Planificación del turno de 24 horas del algoritmo voraz.

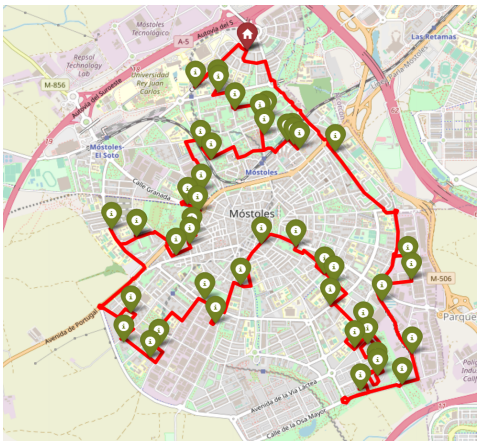


Figura 4.12: Planificación del turno de 24 horas del algoritmo VND.

y 40 segundos). De este modo se puede observar que dejando el suficiente tiempo, los algoritmos genéticos son capaces de arrojar buenas soluciones que dominen en el frente de Pareto. Estos buenos resultados se aprecian sobretodo para soluciones con pocos clientes, debido a la diversidad en las soluciones iniciales que proporcionan los algoritmos genéticos. Cuantos más clientes tienen las soluciones, mayores valores tienen para el coste del viaje y para la latencia. En cambio, el algoritmo VND mantiene unos buenos valores para el coste del viaje aunque haya muchos clientes en la ruta.

El análisis de las soluciones obtenidas por los distintos algoritmos bajo un límite de tiempo de 8 horas no solo permite contrastar resultados numéricos (Tabla 4.11 filas de la 5 a la 8), sino también explorar visualmente las estrategias que cada uno aplica para construir las rutas. La representación gráfica sobre el plano, como se muestra en la Figura 4.14, ofrece información cualitativa muy valiosa que complementa y contextualiza las métricas tradicionales. En este sentido, se observa que el algoritmo VND tiende a construir soluciones de forma secuencial y ordenada, incorporando nodos cercanos a la ruta en curso y evitando giros innecesarios o cruces. Esta estrategia contribuye a generar trayectorias más limpias y eficientes desde el punto de vista logístico, a pesar de que en esta ocasión no sea el que alcanza los mejores valores absolutos en cuanto a clientes atendidos o beneficio.

En contraste, el algoritmo MOEA/D muestra una tendencia a generar rutas con mayores desplazamientos entre puntos, lo cual sugiere una menor prioridad en la proximidad geográfica durante la construcción de la solución. Este comportamiento lleva al algoritmo a servir a un mayor número de clientes (25), lo que se traduce también en un beneficio más alto (1572 €), aunque el coste del recorrido (22,133 km) aumenta de forma considerable en comparación con VND (véase la Tabla 4.11 fila 6). Esta mayor dispersión de las rutas implica que el viajante se cruza consigo mismo en más de una ocasión, lo cual añade ineficiencias.

Los algoritmos NSGA-II y SPEA2, por su parte, evidencian aún más esta tendencia al desorden espacial. Ambas soluciones incluyen múltiples cruces y trayectorias entrelazadas que, si bien pueden permitir alcanzar objetivos como maximizar

Algoritmo	Coste (km)	Latencia (h)	Clientes servidos	Beneficio (€)
VND	18,638	4	**13**	**720**
MOEA/D	16,122	4	8	356
NSGA-II	**13,316**	4	11	544
SPEA2	24,328	4	9	412
VND	**19,192**	8	**20**	**1029**
MOEA/D	30,591	8	12	597
NSGA-II	23,084	8	13	625
SPEA2	31,400	8	11	567
VND	**20,516**	12	**26**	**1375**
MOEA/D	31,462	12	14	633
NSGA-II	29,852	12	16	705
SPEA2	31,556	12	16	791
VND	**24,262**	24	**46**	**2499**
MOEA/D	45,401	24	21	1015
NSGA-II	55,237	24	22	1038
SPEA2	61,211	24	20	991

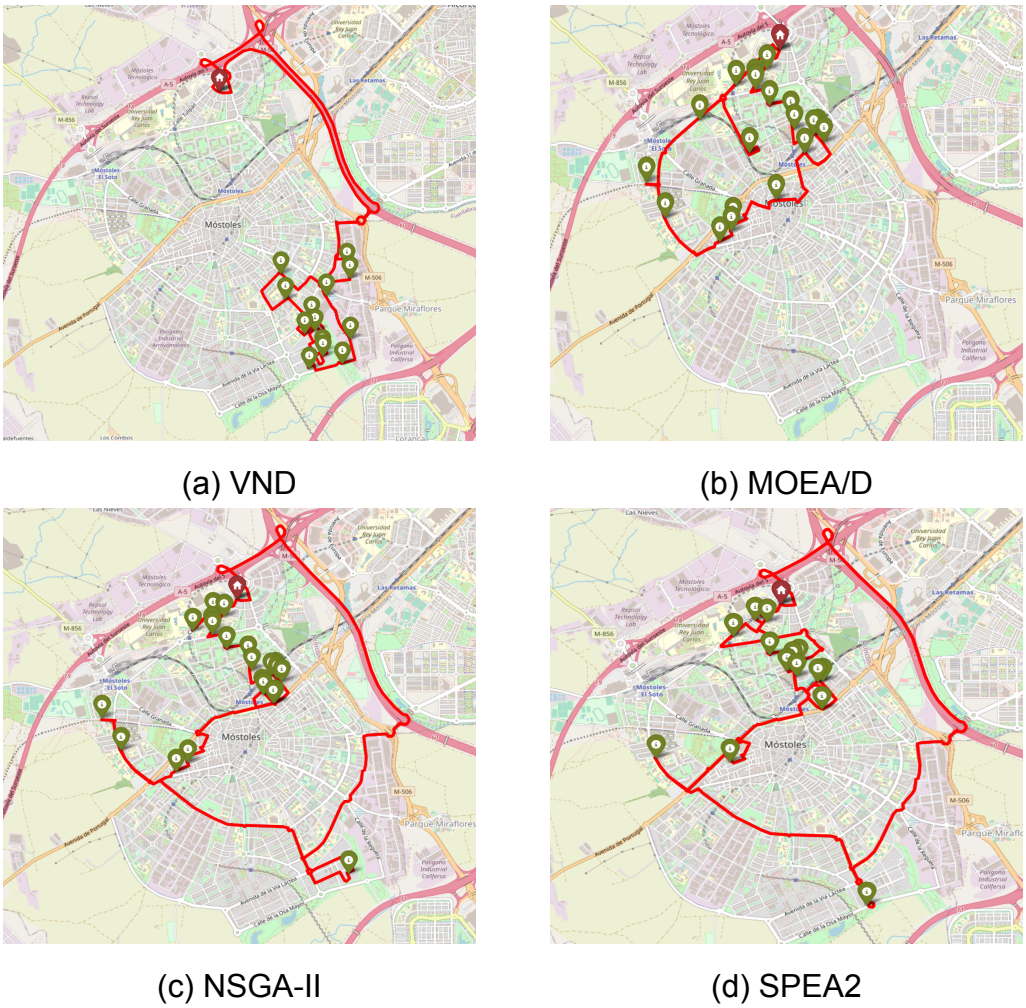
Tabla 4.10: El algoritmo VND contra los algoritmos genéticos con tiempo de CPU de 0,2 segundos.

el número de clientes atendidos (NSGA-II llega hasta 29) o el beneficio económico (1607 € en el mismo caso), véase la Tabla 4.11 fila 7, lo hacen a costa de una estructura de ruta mucho menos coherente y difícilmente interpretable desde una perspectiva operativa. Es interesante notar que, a pesar de sus trayectorias más erráticas, NSGA-II consigue el menor coste en kilómetros (16,092 km), lo que sugiere que su eficiencia en este escenario concreto ha dependido de una combinación afortunada de nodos cercanos y altos beneficios, aunque probablemente en detrimento de la estabilidad de la solución.

En definitiva, esta comparativa deja claro que no existe una única solución ideal cuando se consideran múltiples objetivos, y que cada enfoque presenta ventajas y desventajas según el criterio que se priorice. No obstante, la claridad y coherencia estructural de las rutas obtenidas mediante VND refuerzan su utilidad práctica en contextos donde la planificación logística debe ser ejecutable, comprensible y robusta ante cambios o imprevistos en la operación diaria.

Al analizar las soluciones generadas para un horizonte temporal de 12 horas, se aprecia una creciente complejidad en la configuración de las rutas, especialmente en aquellas construidas por los algoritmos genéticos. A diferencia del algoritmo VND, cuya solución sigue presentando un patrón de recorrido coherente y relativamente fácil de seguir (como se muestra en la Figura 4.15), los métodos evolutivos tienden a generar trayectorias mucho más caóticas, con múltiples cruces, saltos geográficos y decisiones de encaminamiento difíciles de justificar desde una perspectiva operativa.

En esta ocasión, si bien el VND mantiene una buena eficiencia espacial (con un coste de 20,516 km como se muestra en la Tabla 4.11 filas de la 9 a las 12),



(a) VND (b) MOEA/D

(c) NSGA-II (d) SPEA2

Figura 4.13: Comparativa de planificación del turno de 4 horas entre los distintos algoritmos.

su rendimiento en términos de beneficio y clientes atendidos es superado por los algoritmos genéticos. NSGA-II, por ejemplo, alcanza el mayor número de clientes servidos (36) y el mayor beneficio económico (1952 €), con el menor coste total en kilómetros (19,094 km). Esto sugiere que, en esta instancia concreta, ha sido capaz de encontrar una solución altamente eficiente desde el punto de vista multiobjetivo, aunque probablemente a costa de una mayor complejidad en la ruta y una menor claridad operativa.

Por su parte, MOEA/D también logra un buen rendimiento (30 clientes y 1858 €), aunque su coste en kilómetros aumenta ligeramente. SPEA2 vuelve a posicionarse como el algoritmo menos competitivo, con una ruta que cubre solo 20 clientes y presenta el coste más alto (33,527 km), lo que indica una pobre eficiencia tanto logística como económica. Véase Tabla 4.11 fila 10 y 12 respectivamente.

Este incremento en el desorden de las rutas genéticas a medida que se amplía el límite horario puede deberse a que los algoritmos tienen mayor libertad para explorar combinaciones más amplias y diversas de soluciones, lo que en ausencia

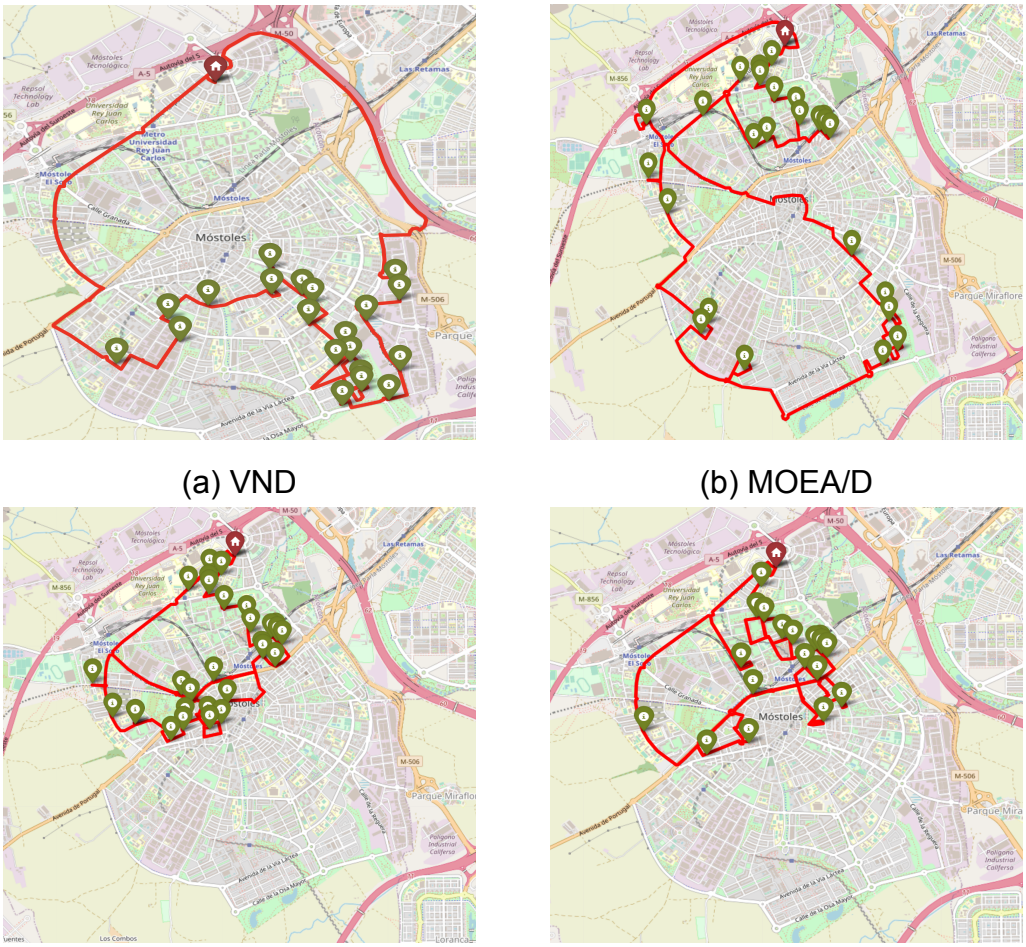
Algoritmo	Coste (km)	Latencia (h)	Clientes servidos	Beneficio (€)
VND	18,638	4	13	720
MOEA/D	**11,071**	4	19	1157
NSGA-II	20,299	4	**21**	**1227**
SPEA2	22,575	4	14	754
VND	19,192	8	20	1029
MOEA/D	22,133	8	**25**	1572
NSGA-II	**16,092**	8	29	**1607**
SPEA2	21,945	8	18	961
VND	20,516	12	26	1375
MOEA/D	22,898	12	30	1858
NSGA-II	**19,094**	12	**36**	**1952**
SPEA2	33,527	12	20	1111
VND	**24,262**	24	46	2449
MOEA/D	33,788	24	37	2245
NSGA-II	26,571	24	**48**	**2645**
SPEA2	40,468	24	29	1529

Tabla 4.11: Comparativa del algoritmo VND respecto a los algoritmos genéticos con un tiempo de CPU de 1000 segundos

de mecanismos específicos de control espacial, como los que incluye el VND, da lugar a soluciones de estructura mucho más dispersa. Este fenómeno pone de manifiesto la importancia de incorporar exploraciones de vecindarios o estrategias de reparación que penalicen explícitamente los solapamientos, las rutas con forma de "Z" o los bucles innecesarios en los algoritmos evolutivos, si se desea alcanzar una solución no solo buena desde el punto de vista cuantitativo, sino también sólida y aplicable en la práctica.

En definitiva, aunque el VND no es siempre el mejor en cuanto a todos los valores objetivos obtenidos, sus soluciones continúan destacando por su claridad estructural y su fiabilidad operativa, factores fundamentales en aplicaciones reales donde la ejecución eficiente de la ruta es tan importante como los resultados numéricos que la justifican.

El análisis final tras una ejecución prolongada de 24 horas revela de forma clara la superioridad del enfoque basado en búsqueda local, concretamente el algoritmo VND, frente a los algoritmos evolutivos multiobjetivo más utilizados en la literatura, como se muestra en la Tabla 4.11 de la fila 13 a la 16. A pesar de que todos los métodos disponen del mismo margen temporal para alcanzar sus mejores soluciones, los algoritmos genéticos no logran igualar el rendimiento del VND en cuanto a la minimización del coste de viaje, que constituye uno de los objetivos clave del problema. En concreto, el algoritmo VND obtiene un coste de viaje de tan solo 24,262 km, una mejora muy significativa respecto a sus competidores, donde el mejor resultado entre los genéticos lo alcanza NSGA-II con 26,571 km, seguido por MOEA/D con 33,788 km, y en última posición SPEA2 con 40,468 km. Esta optimización en el coste no solo supone un ahorro logístico, sino que permite una asignación más eficiente de los recursos, lo que a su vez posibilita atender a un mayor número



(a) VND (b) MOEA/D

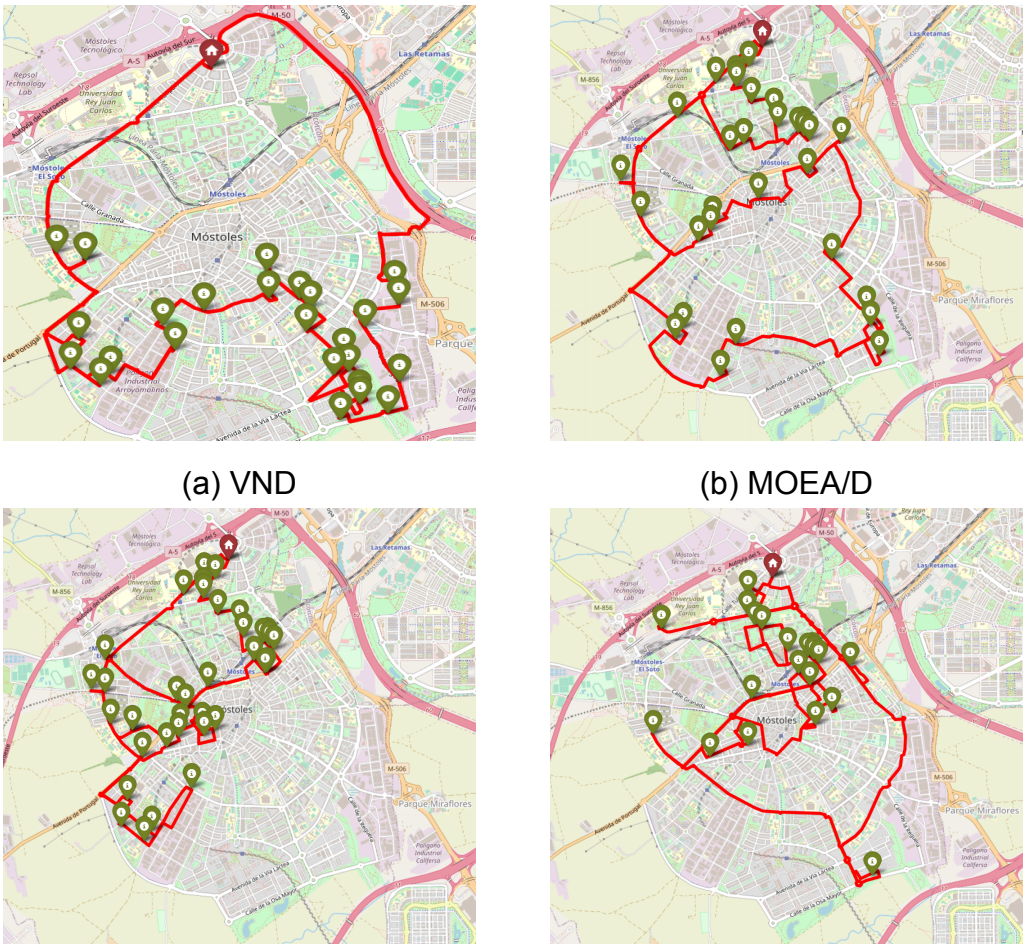
(c) NSGA-II (d) SPEA2

Figura 4.14: Comparativa de planificación del turno de 8 horas entre los distintos algoritmos.

de clientes dentro de los límites temporales y de latencia.

A pesar de que NSGA-II logra servir al mayor número de clientes (48) y alcanzar el beneficio más elevado (2645 €), el resultado del algoritmo VND es mucho más equilibrado, véase Tabla 4.11 filas 13 y 15. Este consigue un beneficio notable (2499 €) sirviendo a 46 clientes, con la ventaja adicional de que el coste es un 30 % más reducido. Esto indica que las soluciones obtenidas por VND son estructuralmente más eficientes, resolviendo mejor los conflictos generados por cruces innecesarios en las rutas o por configuraciones de tipo “dibujo en Z”, que suelen incrementar innecesariamente la distancia total recorrida.

Por tanto, se concluye que las búsquedas locales guiadas por una función voraz enfocada en la minimización del coste permiten alcanzar soluciones de alta calidad que los algoritmos genéticos, con su carácter más exploratorio y generalista, no han logrado alcanzar en este contexto particular. No obstante, este estudio también abre la puerta a posibles líneas de trabajo futuro, como incorporar vecindades específicamente diseñadas para optimizar otros objetivos, como la latencia o el be-

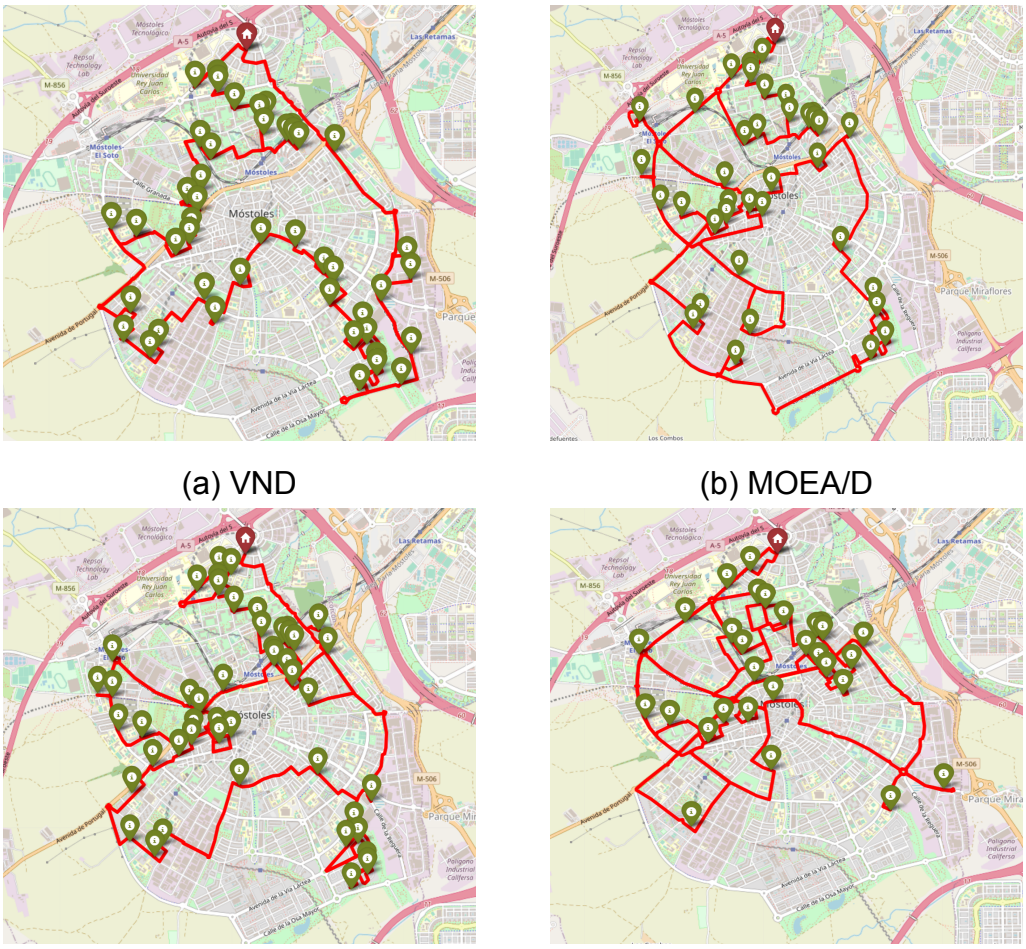


(a) VND (b) MOEA/D

(c) NSGA-II (d) SPEA2

Figura 4.15: Comparativa de planificación del turno de 12 horas entre los distintos algoritmos.

neficio, con la misma eficacia mostrada en el coste. Esto permitiría reforzar aún más la capacidad del enfoque VND y avanzar hacia soluciones todavía más equilibradas en problemas reales de optimización multiobjetivo.



(a) VND (b) MOEA/D

(c) NSGA-II (d) SPEA2

Figura 4.16: Comparativa de planificación del turno de 24 horas entre los distintos algoritmos.

Capítulo 5

Conclusiones y trabajos futuros

En el capítulo final de esta Tesis Doctoral se presentan las conclusiones y las principales contribuciones derivadas de la investigación realizada sobre "el problema multiobjetivo del comerciante reparador con beneficios". Además, se incluyen la conclusiones publicadas durante el proceso de investigación. Finalmente, se proponen varias líneas de trabajo futuras que permitirán continuar con la investigación iniciada en esta Tesis Doctoral.

5.1. Conclusiones

Los problemas $\mathcal{NP}$-Difíciles representan uno de los mayores desafíos actuales en el ámbito de la computación, tanto desde una perspectiva teórica como tecnológica. La resolución óptima de estos problemas se convierte en una barrera insuperable cuando el tamaño del problema es considerable, lo que justifica el término comúnmente utilizado para describirlos: "problemas intratables".

La creciente aparición de problemas con relevancia práctica, que pueden clasificarse como $\mathcal{NP}$-Difíciles, ha llevado al desarrollo de técnicas eficientes para abordarlos. Algunas de estas técnicas permiten encontrar soluciones óptimas cuando el tamaño de las instancias de entrada no es demasiado grande (algoritmos exactos). Sin embargo, para la mayoría de los problemas reales, que son de mayor tamaño, no es posible encontrar la solución óptima en un tiempo de cómputo razonable utilizando algoritmos exactos. Por esta razón, es necesario contar con otro tipo de técnicas que, aunque no puedan garantizar que las soluciones encontradas sean óptimas, logren soluciones de alta calidad en tiempos de cómputo reducidos (algoritmos heurísticos).

En esta Tesis Doctoral se han desarrollado algoritmos heurísticos para abordar el *Multi-objective Traveling Salesman-Repairman Problem with Profits*. En la Sección 5.1.1 se describen las principales contribuciones de esta Tesis Doctoral.

5.1.1. Principales aportaciones

En esta Tesis Doctoral se ha abordado el Mo-TSRPP. Se ha llevado a cabo un exhaustivo estudio del estado del arte del problema y se han propuesto algoritmos heurísticos para su resolución, los cuales constituyen algunas de las principales contribuciones de esta investigación. Para la obtención de soluciones aproximadas, han sido propuestos un algoritmo heurístico constructivo y otros algoritmos de búsqueda local combinados en una búsqueda de vecindades, englobadas dentro de un esquema metaheurístico de VND. A continuación, se presentan, organizadas por capítulos, estas y otras aportaciones destacadas:

En el Capítulo 1 se presenta una introducción al Mo-TSRPP. Se detalla el problema junto con sus aplicaciones prácticas, el contexto en el que está enmarcada esta Tesis Doctoral, la motivación detrás del estudio, la hipótesis de trabajo y los objetivos. También se ofrece un adelanto de la propuesta algorítmica que se desarrollará en los capítulos siguientes.

En el Capítulo 2 se lleva a cabo una revisión detallada del estado del arte del problema tratado, con un enfoque particular en los trabajos que están directamente relacionados con la propuesta de esta Tesis Doctoral: algoritmos heurísticos para resolver el Mo-TSRPP. Específicamente, se ha realizado un análisis exhaustivo de las propuestas heurísticas basadas en las técnicas GRASP y VND. Además, se han recopilado las relaciones conocidas entre el problema en cuestión y otros problemas de optimización.

En el Capítulo 3 se revisan las principales técnicas disponibles para obtener soluciones aproximadas a problemas de optimización. Entre las técnicas analizadas, se han seleccionado las metaheurísticas, y en particular, el esquema VND, para desarrollar un algoritmo que pueda generar soluciones aproximadas de alta calidad para el Mo-TSRPP. Específicamente, se han propuesto una heurística constructiva tipo GRA y tres métodos de búsqueda local. Finalmente, se ha configurado un algoritmo VND utilizando la mejor combinación de método constructivo y búsquedas locales, considerando criterios de calidad, diversidad de las soluciones obtenidas y tiempo de ejecución al probar diferentes combinaciones de los algoritmos mencionados.

En el capítulo 4, se realiza una amplia experimentación que valida los algoritmos heurísticos presentados en el capítulo 3. Para evaluar estos algoritmos, se han utilizado tres conjuntos diferentes de instancias (ver sección 4.1.1). En los experimentos relacionados con los algoritmos heurísticos, se ha seleccionado un pequeño porcentaje de instancias de los conjuntos mencionados para realizar experimentos preliminares. Estos experimentos han demostrado la contribución de los diversos componentes introducidos en los algoritmos y han ayudado a ajustar algunos de sus parámetros. Finalmente, las versiones finales de los algoritmos propuestos se han comparado favorablemente con otras propuestas del estado del arte, en concreto con los algoritmos genéticos MOEA/D, SPEA2 y NSGA-II, demostrando su superioridad y avalando así la calidad de los mismos.

5.1.2. Publicaciones

En esta sección se enumeran las publicaciones derivadas del trabajo realizado en esta Tesis Doctoral. Los resultados finales presentados en esta tesis, y por ende, sus principales contribuciones, han sido revisados por la comunidad científica. En particular, se ha enviado un artículo a una revista indexada en el Journal Citation Reports (JCR), y se encuentra publicada:

- Título: The multiobjective traveling salesman–repairman problem with profits: design and implementation of a variable neighborhood descent algorithm for a real scenario [171]
- Autores: Rubén Morante, Ana Dolores López, Jesús Sanchéz-Oro y Alfredo García.
- Revista: International Transactions in Operational Research.
- Fecha: Noviembre 2023.
- Indicadores de calidad:
  - Factor de impacto: 3,1
  - Posición relativa de la revista: 34 / 106 (Q2)
  - Categoría: Operations Research & Management Science

A su vez, se ha presentado en los siguientes congresos:

- 2-3/6/2022 V Workshop GRAFO, Móstoles, España.
- 12-15/6/2022 VeRoLog, Hamburgo, Alemania.
- 25-28/10/2022 ICVNS, Abu Dhabi, Emiratos Árabes Unidos.
- 24-25/11/2022 III Escuela de Invierno sobre Optimización Heurística y Modelización Matemática, Burgos, España.
- 18-19/9/2024 VII Workshop GRAFO, Móstoles, España.

5.2. Trabajos futuros

Durante la elaboración de una Tesis Doctoral, es común identificar líneas de investigación abiertas, que pueden servir como base para estudios futuros. A partir del proceso de investigación llevado a cabo en esta Tesis Doctoral, se sugieren los siguientes trabajos futuros:

El Mo-TSRPP está íntimamente vinculado con otros problemas de optimización (en el Capítulo 2 se detallan algunas de estas conexiones) y, más específicamente, con otros problemas de enrutamiento. A veces, cuando existen problemas relacionados o una forma clara de transformación entre ellos, las cotas propuestas para un problema pueden ser aplicables a otros. En particular, resulta interesante investigar si las cotas propuestas para el Mo-TSRPP son aplicables a otros problemas

relacionados. De igual manera, es relevante identificar cotas para otros problemas y verificar si son aplicables al Mo-TSRPP.

Esta Tesis Doctoral está enfocada a casos prácticos con una sóla ruta. Un enfoque de múltiples rutas abarcaría un mayor número de casos prácticos, pudiendo derivar a un nuevo problema de tipo *Vehicle Routing Problem* (VRP).

Como se mencionó anteriormente, resolver problemas $\mathcal{NP}$-Difíciles de manera óptima es computacionalmente muy costoso cuando el tamaño del problema es grande. En esta Tesis Doctoral, se han utilizado instancias de hasta 18000 nodos para evaluar los algoritmos propuestos. Para las instancias más grandes, no se ha logrado una resolución óptima usando una CPU convencional dentro de los tiempos de cómputo considerados. Sería interesante explorar la implementación de los algoritmos propuestos en arquitecturas gráficas de procesamiento vertical y evaluar qué tamaño de problema podría resolverse para el Mo-TSRPP utilizando estos algoritmos. Este aspecto podría ser de gran interés para aplicaciones prácticas.

Aunque es comprensible el interés en utilizar una metaheurística trayectorial para obtener soluciones aproximadas al Mo-TSRPP, sería valioso investigar las ventajas que una metaheurística poblacional podría ofrecer para este problema.

En las estrategias de búsqueda local propuestas (ver Sección 3.2.2), uno de los factores más cruciales para la eficacia del método fue la definición del criterio de aceptación de un movimiento, que se basa únicamente en la mejora del valor de la función objetivo. Sería interesante profundizar en un método de evaluación de los movimientos. Esta idea es especialmente relevante cuando se trabaja con problemas que tienen muchas soluciones en las que el valor de la función objetivo no cambia, lo que dificulta determinar la conveniencia de ciertos movimientos.

Las soluciones iniciales dadas por el constructivo determinan en gran medida la evolución de la solución y la exploración de las vecindades en los métodos de mejora propuestos. Sería interesante determinar cómo de buena es una solución en esta fase previa y aplicar métodos multiarranques para la obtención de mejores soluciones antes de pasar a una fase de mejora.

Resulta muy interesante pasar de un enfoque multiobjetivo con frente de pareto al uso de una función objetivo ponderada, función de agregación ponderada o al uso de una función mixtura. Este tipo de función se utiliza en problemas de optimización multiobjetivo, donde se combinan varias funciones objetivo en una sola función mediante la asignación de pesos a cada una de ellas. Los pesos reflejan la importancia relativa de cada objetivo en la solución final.

En el VND, el número de vecindades exploradas es determinante en la evolución de la solución. La literatura proporciona muchas búsquedas locales para el problema del TSP que se podrían abordar.

Bibliografía

[1] E.-G. Talbi, Metaheuristics: from design to implementation. John Wiley & Sons, 2009.

[2] C. A. Floudas and P. M. Pardalos, Encyclopedia of optimization. Springer Science & Business Media, 2008.

[3] E. K. Chong, W.-S. Lu, and S. H. Żak, An Introduction to Optimization: With Applications to Machine Learning. John Wiley & Sons, 2023.

[4] A. Duarte, J. Pantrigo, and M. Gallego, "Metaheurísticas," Madrid: Dykinson, 2007.

[5] C. H. Papadimitriou and K. Steiglitz, Combinatorial optimization: algorithms and complexity. Courier Corporation, 1998.

[6] M. Yagiura and T. Ibaraki, "On metaheuristic algorithms for combinatorial optimization problems," The Transactions of the Institute of Electronics, Information and Communication Engineers, vol. 83, no. 1, pp. 3–25, 2000.

[7] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, Complexity and approximation: Combinatorial optimization problems and their approximability properties. Springer Science & Business Media, 2012.

[8] M. R. Garey, "Computers and intractability: A guide to the theory of np-completeness, freeman," Fundamental, 1997.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, et al., "Introduction to algorithms, chapter 11," 2001.

[10] R. A. Española, Diccionario dela lengua española. Editorial Espasa-Calpe, 1970.

[11] G. Pólya and J. H. Conway, How to solve it: A new aspect of mathematical method. Princeton University Press Princeton, 1957.

[12] O. E. Dictionary, "Oxford english dictionary," Simpson, Ja & Weiner, Esc, vol. 3, 1989.

[13] F. Glover, "Future paths for integer programming and links to artificial intelligence," Computers & operations research, vol. 13, no. 5, pp. 533–549, 1986.

[14] R. Martí, V. Campos, M. G. Resende, and A. Duarte, “Multiobjective grasp with path relinking,” European Journal of Operational Research, vol. 240, no. 1, pp. 54–71, 2015.

[15] S. Pérez-Peló, J. Sánchez-Oro, A. D. López-Sánchez, and A. Duarte, “A multi-objective parallel iterated greedy for solving the p-center and p-dispersion problem,” Electronics, vol. 8, no. 12, 2019.

[16] A. López-Sánchez, A. Hernández-Díaz, F. Gortázar, and M. Hinojosa, “A multiobjective grasp–vnd algorithm to solve the waste collection problem,” International Transactions in Operational Research, vol. 25, no. 2, pp. 545–567, 2018.

[17] J. Sánchez-Oro, A. D. López-Sánchez, and J. M. Colmenar, “A general variable neighborhood search for solving the multi-objective open vehicle routing problem,” Journal of Heuristics, vol. 19, no. 3, p. 423–452, 2020.

[18] K. Ilavarasi and K. S. Joseph, “Variants of travelling salesman problem: A survey,” in International conference on information communication and embedded systems (ICICES2014), pp. 1–7, IEEE, 2014.

[19] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). USA: Princeton University Press, 2007.

[20] E. Angelelli, C. Bazgan, M. G. Speranza, and Z. Tuza, “Complexity and approximation for traveling salesman problems with profits,” Theoretical Computer Science, vol. 531, pp. 54 – 65, 2014.

[21] M. Dell’Amico, F. Maffioli, and P. Värbrand, “On prize-collecting tours and the asymmetric travelling salesman problem,” International Transactions in Operational Research, vol. 2, pp. 297 – 308, 1995.

[22] T. Tsiligirides, “Heuristic methods applied to orienteering,” Journal of the Operational Research Society, vol. 35, pp. 797 – 809, 1984.

[23] G. Laporte and S. Martello, “The selective travelling salesman problem,” Discrete Applied Mathematics, vol. 26, no. 2, pp. 193–207, 1990.

[24] B. C. Silva, I. F. Fernandes, M. C. Goldbarg, and E. F. Goldbarg, “Quota travelling salesman problem with passengers, incomplete ride and collection time optimization by ant-based algorithms,” Computers & Operations Research, vol. 120, p. 104950, 2020.

[25] N. Jozefowiez, F. Glover, and M. Laguna, “Multi-objective meta-heuristics for the traveling salesman problem with profits,” J. Math. Model. Algorithms, vol. 7, pp. 177–195, 06 2008.

[26] M. Gendreau, G. Laporte, and F. Semet, “A tabu search heuristic for the undirected selective travelling salesman problem,” European Journal of Operational Research, vol. 106, no. 2, pp. 539–545, 1998.

[27] M. Gendreau, A. Hertz, and G. Laporte, “New insertion and postoptimization procedures for the traveling salesman problem,” Operations Research, vol. 40, no. 6, pp. 1086–1094, 1992.

[28] J.-F. Bérubé, M. Gendreau, and J.-Y. Potvin, “An exact epsilon-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits.,” European Journal of Operational Research, vol. 194, no. 1, pp. 39–50, 2009.

[29] A. Lucena, “Time-dependent traveling salesman problem?the deliveryman case,” Networks, vol. 20, no. 6, pp. 753 – 763, 1990.

[30] M. M. Silva, A. Subramanian, T. Vidal, and L. S. Ochi, “A simple and effective metaheuristic for the minimum latency problem,” European Journal of Operational Research, vol. 221, no. 3, pp. 513 – 520, 2012.

[31] J. Pei, N. Mladenović, D. Urošević, J. Brimberg, and X. Liu, “Solving the traveling repairman problem with profits: A novel variable neighborhood search approach,” Information Sciences, vol. 507, pp. 108–123, 2020.

[32] T. Dewilde, D. Cattrysse, S. Coene, F. C. R. Spieksma, and P. Vansteenwegen, “Heuristics for the traveling repairman problem with profits,” Computers & Operations Research, vol. 40, no. 7, pp. 1700 – 1707, 2013.

[33] M. Avci and M. G. Avci, “A grasp with iterated local search for the traveling repairman problem with profits,” Computers & Industrial Engineering, vol. 113, pp. 323 – 322, 2017.

[34] M. E. Bruni, S. Khodaparasti, and S. Nucamendi-Guillén, “The bi-objective minimum latency problem with profit collection and uncertain travel times.,” in ICORES, pp. 109–118, 2020.

[35] N. Arellano-Arriaga, J. Molina, S. E. Schaeffer, and I. Álvarez Socarrás, A.M. amd Martínez-Salazar, “A bi-objective study of the minimum latency problem,” Journal of Heuristics, vol. 25 (3), pp. 431–454, 2019.

[36] J. Molina, A. Lopez, A. Hernández-Díaz, and I. Martínez-Salazar, “A multi-start algorithm with intelligent neighborhood selection for solving multi-objective humanitarian vehicle routing problems,” Journal of Heuristics, vol. 24, 04 2018.

[37] S. Bock and K. Klamroth, “Combining traveling salesman and traveling repairman problems: A multi-objective approach based on multiple scenarios,” Computers & Operations Research, vol. 112, p. 104766, 2019.

[38] A. Mjirda, R. Todosijević, S. Hanafi, P. Hansen, and N. Mladenović, “Sequential variable neighborhood descent variants: an empirical study on the traveling salesman problem,” International Transactions in Operational Research, vol. 24, no. 3, pp. 615–633, 2017.

[39] Y. Wang, Y. Chen, and Y. Lin, “Memetic algorithm based on sequential variable neighborhood descent for the minmax multiple traveling salesman problem,” Computers & Industrial Engineering, vol. 106, pp. 105–122, 2017.

[40] B. Issaoui, I. Zidi, E. Marcon, and K. Ghedira, “New multi-objective approach for the home care service problem based on scheduling algorithms and variable neighborhood descent,” Electronic Notes in Discrete Mathematics, vol. 47, pp. 181–188, 2015. The 3rd International Conference on Variable Neighborhood Search (VNS’14).

[41] H. Hernández-Pérez, I. Rodríguez-Martín, and J. J. Salazar-González, “A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem,” Computers & Operations Research, vol. 36, no. 5, pp. 1639 – 1645, 2009.

[42] T. A. Feo, M. G. C. Resende, and S. H. Smith, “A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set,” Operations Research, vol. 42, no. 5, pp. 860–878, 1994.

[43] T. A. Feo and M. G. C. Resende, “Greedy Randomized Adaptive Search Procedures,” Journal of Global Optimization, vol. 6, no. 2, pp. 109–133, 1995.

[44] N. Mladenović and P. Hansen, “Variable neighborhood search,” Computers & operations research, vol. 24, no. 11, pp. 1097–1100, 1997.

[45] T. A. Feo and M. G. Resende, “Greedy randomized adaptive search procedures,” Journal of global optimization, vol. 6, pp. 109–133, 1995.

[46] M. G. Resende and J. L. G. Velarde, “Grasp: Procedimientos de búsqueda miopes aleatorizados y adaptativos,” Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 7, no. 19, p. 0, 2003.

[47] T. A. Feo and M. G. Resende, “A probabilistic heuristic for a computationally difficult set covering problem,” Operations research letters, vol. 8, no. 2, pp. 67–71, 1989.

[48] H. Hernández-Pérez, I. Rodríguez-Martín, and J. J. Salazar-González, “A hybrid grasp/vnd heuristic for the one-commodity pickup-and-delivery traveling salesman problem,” Computers & Operations Research, vol. 36, no. 5, pp. 1639–1645, 2009.

[49] M. Mestria, L. S. Ochi, and S. de Lima Martins, “Grasp with path relinking for the symmetric euclidean clustered traveling salesman problem,” Computers & Operations Research, vol. 40, no. 12, pp. 3218–3229, 2013.

[50] R. Cordone and R. W. Calvo, “A heuristic for the vehicle routing problem with time windows,” Journal of Heuristics, vol. 7, pp. 107–129, 2001.

[51] E. K. Burke, P. I. Cowling, and R. Keuthen, “Effective local and guided variable neighbourhood search methods for the asymmetric travelling salesman problem,” in Workshops on Applications of Evolutionary Computation, pp. 203–212, Springer, 2001.

[52] S. A. Cook, “The complexity of theorem-proving procedures,” Proceedings of the third annual ACM symposium on Theory of computing, 1971.

[53] R. M. Karp, Reducibility among combinatorial problems. Springer, 2010.

[54] T. Bektas, “The multiple traveling salesman problem: an overview of formulations and solution procedures,” omega, vol. 34, no. 3, pp. 209–219, 2006.

[55] G. Laporte, “The traveling salesman problem: An overview of exact and approximate algorithms,” European Journal of Operational Research, vol. 59, no. 2, pp. 231–247, 1992.

[56] J.-F. Bérubé, M. Gendreau, and J.-Y. Potvin, “An exact epsilon-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits,” European journal of operational research, vol. 194, no. 1, pp. 39–50, 2009.

[57] K. Savla, E. Frazzoli, and F. Bullo, “Traveling salesperson problems for the dubins vehicle,” IEEE Transactions on Automatic Control, vol. 53, no. 6, pp. 1378–1391, 2008.

[58] T. S. Alemayehu and J.-H. Kim, “Efficient nearest neighbor heuristic tsp algorithms for reducing data acquisition latency of uav relay wsn,” Wireless Personal Communications, vol. 95, pp. 3271–3285, 2017.

[59] K. Sundar and S. Rathinam, “Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots,” IEEE Transactions on Automation Science and Engineering, vol. 11, no. 1, pp. 287–294, 2013.

[60] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, “The orienteering problem: A survey,” European Journal of Operational Research, vol. 209, no. 1, pp. 1–10, 2011.

[61] R. Agarwala, D. L. Applegate, D. Maglott, G. D. Schuler, and A. A. Schäffer, “A fast and scalable radiation hybrid map construction and integration strategy,” Genome Research, vol. 10, no. 3, pp. 350–364, 2000.

[62] B. L. Golden, A. A. Assad, and E. A. Wasil, “Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries,” in The vehicle routing problem, pp. 245–286, SIAM, 2002.

[63] P. P. Repoussis and C. D. Tarantilis, “Solving the fleet size and mix vehicle routing problem with time windows via adaptive memory programming,” Transportation Research Part C: Emerging Technologies, vol. 18, no. 5, pp. 695–712, 2010.

[64] D. Jin, Q. Li, and M. Lu, “A heuristic search algorithm for hamiltonian circuit problems in directed graphs,” Wireless Networks, vol. 28, no. 2, pp. 979–989, 2022.

[65] O. C. Sokmen, S. Emec, M. Yilmaz, and G. Akkaya, “An overview of chinese postman problem,” in 3rd International Conference on Advanced Engineering Technologies, vol. 10, 2019.

[66] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, “The vehicle routing problem: State of the art classification and review,” Computers & industrial engineering, vol. 99, pp. 300–313, 2016.

[67] R. Matai, S. P. Singh, and M. L. Mittal, “Traveling salesman problem: an overview of applications, formulations, and solution approaches,” Traveling salesman problem, theory and applications, vol. 1, no. 1, pp. 1–25, 2010.

[68] B. Bontoux, D. Feillet, and C. Artigues, “Large neighborhood search for variants of tsp,” in MIC 2007: The Seventh Metaheuristics International Conference, pp. CD–Rom, 2007.

[69] D. Feillet, P. Dejax, and M. Gendreaux, “Travelling salesman problems with profits: an overview,” Laboratoire Informatique d’Avignon, to appear in Transp. Sc, 2004.

[70] D. Feillet, P. Dejax, and M. Gendreau, “Traveling salesman problems with profits,” Transportation science, vol. 39, no. 2, pp. 188–205, 2005.

[71] B. L. Golden, Q. Wang, and L. Liu, “A multifaceted heuristic for the orienteering problem,” Naval Research Logistics (NRL), vol. 35, no. 3, pp. 359–366, 1988.

[72] T. Tsiligirides, “Heuristic methods applied to orienteering,” Journal of the Operational Research Society, vol. 35, no. 9, pp. 797–809, 1984.

[73] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala, “Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen,” in Proceedings of the twenty-seventh annual ACM symposium on Theory of computing, pp. 277–283, 1995.

[74] M. Dell’Amico, F. Maffioli, and P. Värbrand, “On prize-collecting tours and the asymmetric travelling salesman problem,” International Transactions in Operational Research, vol. 2, no. 3, pp. 297–308, 1995.

[75] T. Volgenant and R. Jonker, “On some generalizations of the travelling-salesman problem,” Journal of the Operational Research Society, vol. 38, no. 11, pp. 1073–1079, 1987.

[76] J. Riera-Ledesma and J. J. Salazar-González, “A heuristic approach for the travelling purchaser problem,” European Journal of Operational Research, vol. 162, no. 1, pp. 142–152, 2005.

[77] D. Manerba, R. Mansini, and J. Riera-Ledesma, “The traveling purchaser problem and its variants,” European Journal of Operational Research, vol. 259, no. 1, pp. 1–18, 2017.

[78] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff, “Approximation algorithms for orienteering and discounted-reward tsp,” SIAM Journal on Computing, vol. 37, no. 2, pp. 653–670, 2007.

[79] E. Balas, “The prize collecting traveling salesman problem,” Networks, vol. 19, no. 6, pp. 621–636, 1989.

[80] N. Mladenović, R. Todosijević, and D. Urošević, “Two level general variable neighborhood search for attractive traveling salesman problem,” Computers & Operations Research, vol. 52, pp. 341–348, 2014.

[81] Y. Dumas, J. Desrosiers, E. Gelinas, and M. M. Solomon, “An optimal algorithm for the traveling salesman problem with time windows,” Operations research, vol. 43, no. 2, pp. 367–371, 1995.

[82] I. Kara, O. N. Koc, F. Altıparmak, and B. Dengiz, “New integer linear programming formulation for the traveling salesman problem with time windows: minimizing tour duration with waiting times,” Optimization, vol. 62, no. 10, pp. 1309–1319, 2013.

[83] R. Burink, Precedence-Constrained Traveling Salesman Problem. PhD thesis, Faculty of Science and Engineering, 2018.

[84] J. Gan and G. Zhang, “The k-delivery traveling salesman problem: revisited,” in Combinatorial Optimization and Applications: 13th International Conference, COCOA 2019, Xiamen, China, December 13–15, 2019, Proceedings 13, pp. 197–209, Springer, 2019.

[85] M. Gendreau, G. Laporte, and D. Vigo, “Heuristics for the traveling salesman problem with pickup and delivery,” Computers & Operations Research, vol. 26, no. 7, pp. 699–714, 1999.

[86] C. E. Noon and J. C. Bean, “A lagrangian based approach for the asymmetric generalized traveling salesman problem,” Operations Research, vol. 39, no. 4, pp. 623–632, 1991.

[87] M. Fischetti, J. J. S. González, and P. Toth, “The symmetric generalized traveling salesman polytope,” Networks, vol. 26, no. 2, pp. 113–123, 1995.

[88] J. Monnot, V. T. Paschos, and S. Toulouse, “Approximation algorithms for the traveling salesman problem,” Mathematical methods of operations research, vol. 56, pp. 387–405, 2003.

[89] G. Carpaneto, S. Martello, and P. Toth, “An algorithm for the bottleneck traveling salesman problem,” Operations Research, vol. 32, no. 2, pp. 380–389, 1984.

[90] S. Anily and J. Bramel, “Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries,” Naval Research Logistics (NRL), vol. 46, no. 6, pp. 654–670, 1999.

[91] A. N. Letchford, S. D. Nasiri, and D. O. Theis, “Compact formulations of the steiner traveling salesman problem and related problems,” European Journal of Operational Research, vol. 228, no. 1, pp. 83–92, 2013.

[92] A. L. Ottoni, E. G. Nepomuceno, M. S. d. Oliveira, and D. C. d. Oliveira, “Reinforcement learning for the traveling salesman problem with refueling,” Complex & Intelligent Systems, vol. 8, no. 3, pp. 2001–2015, 2022.

[93] J. R. Current and D. A. Schilling, “The covering salesman problem,” Transportation science, vol. 23, no. 3, pp. 208–213, 1989.

[94] V. Bhavani and M. S. Murthy, “Time-dependent travelling salesman problem,” Opsearch, vol. 42, pp. 199–227, 2005.

[95] P. Chalasani, R. Motwani, and A. Rao, “Algorithms for robot grasp and delivery,” in Proc. 2nd Int. Workshop Algorithmic Found. Robot.(WAFR), p. 347, 1997.

[96] C. S. Helvig, G. Robins, and A. Zelikovsky, “The moving-target traveling salesman problem,” Journal of Algorithms, vol. 49, no. 1, pp. 153–174, 2003.

[97] M. Hammar and B. J. Nilsson, “Approximation results for kinetic variants of tsp,” in Automata, Languages and Programming: 26th International Colloquium, ICALP’99 Prague, Czech Republic, July 11–15, 1999 Proceedings 26, pp. 392–401, Springer, 1999.

[98] K. Kučerová, P. Váňa, and J. Faigl, “Variable-speed traveling salesman problem for vehicles with curvature constrained trajectories,” in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4714–4719, IEEE, 2021.

[99] E. Benavent and A. Martínez, “Multi-depot multiple tsp: a polyhedral study and computational results,” Annals of Operations Research, vol. 207, pp. 7–25, 2013.

[100] P. Jaillet, Probabilistic traveling salesman problems. PhD thesis, Massachusetts Institute of Technology, 1985.

[101] A. Henchiri, M. Bellalouna, and W. Khaznaji, “A probabilistic traveling salesman problem: a survey.,” in FedCSIS (Position Papers), pp. 55–60, 2014.

[102] M. Wagner, M. Lindauer, M. Mısır, S. Nallaperuma, and F. Hutter, “A case study of algorithm selection for the traveling thief problem,” Journal of Heuristics, vol. 24, pp. 295–320, 2018.

[103] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson, “A note on the prize collecting traveling salesman problem,” Mathematical programming, vol. 59, no. 1, pp. 413–420, 1993.

[104] G. Gutin and A. P. Punnen, The traveling salesman problem and its variations, vol. 12. Springer Science & Business Media, 2006.

[105] J. A. Bondy and U. S. R. Murty, Graph theory. Springer Publishing Company, Incorporated, 2008.

[106] N. Christofides, “Worst-case analysis of a new heuristic for the travelling salesman problem,” in Operations Research Forum, vol. 3, p. 20, Springer, 2022.

[107] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to algorithms. MIT press, 2022.

[108] S. N. Kabadi, “Polynomially solvable cases of the tsp,” The traveling salesman problem and its variations, pp. 489–583, 2007.

[109] M. O. Ball, T. Magnanti, C. L. Monma, and G. L. Nemhauser, Handbooks in Operations Research and Management Science: Network Models. North-Holland, 1995.

[110] Y. Shiloach and S. Even, “An on-line edge-deletion problem,” Journal of the ACM (JACM), vol. 28, no. 1, pp. 1–4, 1981.

[111] Ö. Özpeynirci and M. Köksalan, “Multiobjective traveling salesperson problem on halin graphs,” European journal of operational research, vol. 196, no. 1, pp. 155–161, 2009.

[112] C. Keller and M. Goodchild, “The multiobjective vending problem: a generalization of the travelling salesman problem,” Environment and Planning B: Planning and Design, vol. 15, no. 4, pp. 447–460, 1988.

[113] M. Gendreau, G. Laporte, and F. Semet, “A tabu search heuristic for the undirected selective travelling salesman problem,” European Journal of Operational Research, vol. 106, no. 2-3, pp. 539–545, 1998.

[114] Ö. Şimşek, “The biobjective traveling salesman problem with profit,” Master’s thesis, Middle East Technical University, 2007.

[115] S. Karademir, “A genetic algorithm for the biobjective traveling salesman problem with profits,” Master’s thesis, Middle East Technical University, 2008.

[116] N. Jozefowiez, F. Glover, and M. Laguna, “Multi-objective meta-heuristics for the traveling salesman problem with profits,” Journal of Mathematical Modelling and Algorithms, vol. 7, pp. 177–195, 2008.

[117] A. Piwońska, “Genetic algorithm finds routes in travelling salesman problem with profits,” Zeszyty Naukowe Politechniki Białostockiej. Informatyka, no. 5, pp. 51–65, 2010.

[118] C. Filippi and E. Stevanato, “A two-phase method for bi-objective combinatorial optimization and its application to the tsp with profits,” Algorithmic Operations Research, vol. 7, no. 2, pp. 125–139, 2012.

[119] N. Labadie, J. Melechovsky, and C. Prins, “An evolutionary algorithm with path relinking for a bi-objective multiple traveling salesman problem with profits,” Applications of Multi-Criteria and Game Theory Approaches: Manufacturing and Logistics, pp. 195–223, 2014.

[120] E. Angelelli, C. Bazgan, M. G. Speranza, and Z. Tuza, “Complexity and approximation for traveling salesman problems with profits,” Theoretical Computer Science, vol. 531, pp. 54–65, 2014.

[121] M. Zhang, J. Qin, Y. Yu, and L. Liang, “Traveling salesman problems with profits and stochastic customers,” International Transactions in Operational Research, vol. 25, no. 4, pp. 1297–1313, 2018.

[122] R. Lahyani, M. Khemakhem, and F. Semet, “A unified matheuristic for solving multi-constrained traveling salesman problems with profits,” EURO Journal on Computational Optimization, vol. 5, no. 3, pp. 393–422, 2017.

[123] O. Osicka, M. Guajardo, and K. Jörnsten, “Cooperation of customers in traveling salesman problems with profits,” Optimization Letters, vol. 14, no. 5, pp. 1219–1233, 2020.

[124] M. Namazi, M. Newton, A. Sattar, and C. Sanderson, “A profit guided coordination heuristic for travelling thief problems,” in Proceedings of the International Symposium on Combinatorial Search, vol. 10, pp. 140–144, 2019.

[125] P. He, J.-K. Hao, and Q. Wu, “Hybrid genetic algorithm for undirected traveling salesman problems with profits,” Networks, vol. 82, no. 3, pp. 189–221, 2023.

[126] E. E. Işık and M. Şimşir, “Maximizing total net profit for traveling salesman problem with profits using metaheuristic algorithms,” The European Journal of Research and Development, vol. 3, no. 1, pp. 46–59, 2023.

[127] S. Coene and F. C. Spieksma, “A latency problem with profits.,” in CTW, pp. 29–32, 2007.

[128] F. Afrati, S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou, “The complexity of the travelling repairman problem,” RAIRO-Theoretical Informatics and Applications, vol. 20, no. 1, pp. 79–87, 1986.

[129] S. Coene and F. C. Spieksma, “Profit-based latency problems on the line,” Operations Research Letters, vol. 36, no. 3, pp. 333–337, 2008.

[130] T. Dewilde, D. Cattrysse, S. Coene, F. C. Spieksma, and P. Vansteenwegen, “Heuristics for the traveling repairman problem with profits,” Computers & Operations Research, vol. 40, no. 7, pp. 1700–1707, 2013.

[131] M. Avci and M. G. Avci, “A grasp with iterated local search for the traveling repairman problem with profits,” Computers & Industrial Engineering, vol. 113, pp. 323–332, 2017.

[132] P. Beraldi, M. E. Bruni, D. Laganà, and R. Musmanno, “The risk-averse traveling repairman problem with profits,” Soft Computing, vol. 23, pp. 2979–2993, 2019.

[133] Y. Lu, J.-K. Hao, and Q. Wu, “Hybrid evolutionary search for the traveling repairman problem with profits,” Information Sciences, vol. 502, pp. 91–108, 2019.

[134] N. A. Arellano-Arriaga, J. Molina, S. E. Schaeffer, A. Álvarez-Socarrás, and I. A. Martínez-Salazar, “A bi-objective study of the minimum latency problem,” Journal of Heuristics, vol. 25, pp. 431–454, 2019.

[135] M. E. Bruni, S. Khodaparasti, and S. Nucamendi-Guillén, “The bi-objective minimum latency problem with profit collection and uncertain travel times.,” in ICORES, pp. 109–118, 2020.

[136] J. Ren, J.-K. Hao, F. Wu, and Z.-H. Fu, “Intensification-driven local search for the traveling repairman problem with profits,” Expert Systems with Applications, vol. 202, p. 117072, 2022.

[137] D. S. Hochba, “Approximation algorithms for np-hard problems,” ACM Sigact News, vol. 28, no. 2, pp. 40–52, 1997.

[138] C. Blum, A. Roli, and E. Alba, "An introduction to metaheuristic techniques," Parallel Metaheuristics: A New Class of Algorithms, vol. 47, p. 1, 2005.

[139] M. Birattari, L. Paquete, T. Stützle, and K. Varrentrapp, "Classification of metaheuristics and design of experiments for the analysis of components," Tech. Rep. AIDA-01-05, 2001.

[140] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," ACM computing surveys (CSUR), vol. 35, no. 3, pp. 268–308, 2003.

[141] M. Gendreau and J.-Y. Potvin, "Metaheuristics in combinatorial optimization," Annals of Operations Research, vol. 140, pp. 189–213, 2005.

[142] B. Melián, J. A. M. Pérez, and J. M. M. Vega, "Metaheurísticas: Una visión global," Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 7, no. 19, p. 0, 2003.

[143] F. Glover, "Heuristics for integer programming using surrogate constraints," Decision sciences, vol. 8, no. 1, pp. 156–166, 1977.

[144] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," science, vol. 220, no. 4598, pp. 671–680, 1983.

[145] J. Holland, "Adaptation in natural and artificial systems, univ. of mich. press," Ann Arbor, vol. 7, pp. 390–401, 1975.

[146] P. Moscato et al., "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Caltech concurrent computation program, C3P Report, vol. 826, no. 1989, p. 37, 1989.

[147] F. Glover, "A template for scatter search and path relinking," in European conference on artificial evolution, pp. 1–51, Springer, 1997.

[148] F. Glover, "Tabu search and adaptive memory programming—advances, applications and challenges," Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies, pp. 1–75, 1997.

[149] A. Tuson, "Tabu search: F glover and m laguna. kluwer academic press, london, 1997. xix+ 382 pp.£ 54.50. isbn 0 7923 8187 4," Journal of the Operational Research Society, vol. 50, no. 1, pp. 106–107, 1999.

[150] M. Gendreau, J.-Y. Potvin, et al., Handbook of metaheuristics, vol. 2. Springer, 2010.

[151] F. W. Glover and G. A. Kochenberger, Handbook of metaheuristics, vol. 57. Springer Science & Business Media, 2006.

[152] A. Casado, S. Pérez-Peló, J. Sánchez-Oro, and A. Duarte, "A grasp algorithm with tabu search improvement for solving the maximum intersection of k-subsets problem," Journal of Heuristics, vol. 28, no. 1, pp. 121–146, 2022.

[153] G. A. Croes, "A method for solving traveling-salesman problems," Operations research, vol. 6, no. 6, pp. 791–812, 1958.

[154] S. Lin and B. W. Kernighan, “An effective heuristic algorithm for the traveling-salesman problem,” Operations research, vol. 21, no. 2, pp. 498–516, 1973.

[155] G. Clarke and J. W. Wright, “Scheduling of vehicles from a central depot to a number of delivery points,” Operations research, vol. 12, no. 4, pp. 568–581, 1964.

[156] M. Gendreau, A. Hertz, and G. Laporte, “A tabu search heuristic for the vehicle routing problem,” Management science, vol. 40, no. 10, pp. 1276–1290, 1994.

[157] G. Laporte and I. H. Osman, “Routing problems: A bibliography,” Annals of operations research, vol. 61, pp. 227–262, 1995.

[158] S. Lin, “Computer solutions of the traveling salesman problem,” Bell System Technical Journal, vol. 44, no. 10, pp. 2245–2269, 1965.

[159] D. S. Johnson and L. A. McGeoch, “The traveling salesman problem: a case study,” Local search in combinatorial optimization, pp. 215–310, 1997.

[160] P. Hansen and N. Mladenović, “Variable neighborhood search: Principles and applications,” European journal of operational research, vol. 130, no. 3, pp. 449–467, 2001.

[161] P. Hansen, N. Mladenović, R. Todosijević, and S. Hanafi, “Variable neighborhood search: basics and variants,” EURO Journal on Computational Optimization, vol. 5, no. 3, pp. 423–454, 2017.

[162] A. Duarte, N. Mladenovic, J. Sánchez-Oro, and R. Todosijević, “Variable neighborhood descent,” Handbook of heuristics, pp. 341–367, 2018.

[163] A. Hertz and M. Mittaz, “A variable neighborhood descent algorithm for the undirected capacitated arc routing problem,” Transportation science, vol. 35, no. 4, pp. 425–434, 2001.

[164] A. Casado, N. Mladenović, J. Sánchez-Oro, and A. Duarte, “Variable neighborhood search approach with intensified shake for monitor placement,” Networks, 2022.

[165] S. Pérez-Peló, J. Sanchez-Oro, A. Gonzalez-Pardo, and A. Duarte, “A fast variable neighborhood search approach for multi-objective community detection,” Applied Soft Computing, vol. 112, p. 107838, 2021.

[166] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.

[167] Q. Zhang and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, pp. 712–731, 2007.

[168] E. Zitzler, M. Laumanns, and L. Thiele, “Spea2: Improving the strength pareto evolutionary algorithm,” TIK-report, vol. 103, 2001.

[169] J. Durillo and A. Nebro, “Jmetal: A java framework for multi-objective optimization,” Advances in Engineering Software, vol. 42, pp. 760–771, 10 2011.

[170] M. Li and X. Yao, “Quality evaluation of solution sets in multiobjective optimisation: A survey,” ACM Computing Surveys (CSUR), vol. 52, no. 2, p. 26, 2019.

[171] R. Morante-González, A. López-Sánchez, J. Sánchez-Oro, and A. Hernández-Díaz, “The multiobjective traveling salesman–repairman problem with profits: design and implementation of a variable neighborhood descent algorithm for a real scenario,” International Transactions in Operational Research, vol. 32, no. 1, pp. 221–243, 2025.