UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA

MAESTRÍA EN OPTIMIZACIÓN Y CÓMPUTO APLICADO

SOLUCIÓN HEURÍSTICA PARA EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON MÚLTIPLES DEPÓSITOS, VENTANAS DE TIEMPO Y REQUERIMIENTOS DE ARCOS

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRÍA EN OPTIMIZACIÓN Y CÓMPUTO APLICADO

PRESENTA

YASER RODRÍGUEZ MILÁN

DIRECTOR DE TESIS
DR. FEDERICO ALONSO PECINA

CO-DIRECTOR
DRA. JESUS DEL CARMEN PERALTA ABARCA

REVISORES:
DR. FEDERICO ALONSO PECINA
DRA. JESÚS DEL CARMEN PERALTA ABARCA
DR. MARCO ANTONIO CRUZ CHÁVEZ
DRA. IRMA YAZMÍN HERNÁNDEZ BÁEZ
DR. MARTÍN GERARDO MARTÍNEZ RANGEL

CUERNAVACA, MORELOS. MARZO, 2026

Facultad de Contaduría, Administración e Informática

Agradecimientos

Agradezco a la Secretaría de Ciencias, Humanidades, Tecnología e Innovación (SECIHTI) por el apoyo otorgado para la realización de esta maestría, al claustro docente y administrativo de la Maestría en Optimización y Cómputo Aplicado (MOCA) por su acompañamiento y formación, y de manera especial a mi familia y amigos, cuyo respaldo incondicional fue esencial para alcanzar este logro.

Esta maestría fue realizada gracias a una beca otorgada por la Secretaría de Ciencias, Humanidades, Tecnología e Innovación

(SECIHTI)

Ciencia y Tecnología
Secretaría de Ciencia, Humanidades, Tecnología e Innovación

Resumen

El Problema de Ruteo de Vehículos (VRP) es una de las temáticas centrales en la investigación de operaciones debido a su impacto directo en la eficiencia logística y la reducción de costos. En particular, la variante que considera múltiples depósitos, ventanas de tiempo y requerimientos de arcos (MDVRPTWAR) representa un desafío de gran complejidad, pues integra simultáneamente decisiones de asignación, sincronización temporal y cumplimiento de los requerimientos de arco.

El objetivo de esta investigación es proponer un algoritmo híbrido de tipo metaheurístico capaz de generar soluciones factibles y competitivas para el MDVRPTWAR. El diseño metodológico combina recocido simulado, búsqueda tabú y operadores de vecindad, estructurados en tres fases: construcción de soluciones iniciales, búsqueda de factibilidad y refinamiento mediante técnicas locales. Este enfoque permite incrementar la probabilidad de obtener resultados de alta calidad en instancias de mediana y gran escala.

La implementación en C++ y las pruebas sobre instancias de referencia confirman que el algoritmo propuesto logra soluciones eficientes, reduciendo tanto la distancia recorrida como el número de vehículos empleados. De esta forma, se contribuye a la mejora del desempeño logístico, abriendo además la posibilidad de extender la propuesta hacia variantes emergentes como el ruteo de vehículos eléctricos o escenarios con flotas heterogéneas.

Este trabajo se probó con 80 instancias de la literatura, obteniendo soluciones factibles en el 97.5% de ellas. En 7 de las instancias utilizadas se obtuvo el óptimo reportado.

Abstract

The Vehicle Routing Problem (VRP) is one of the central topics in operations research due to its direct impact on logistics efficiency and cost reduction. In particular, the variant that considers multiple depots, time windows, and arc requirements (MDVRPTWAR) represents a highly complex challenge, as it simultaneously integrates allocation decisions, time synchronisation, and compliance with arc requirements.

The objective of this research is to propose a hybrid metaheuristic algorithm capable of generating feasible and competitive solutions for the MDVRPTWAR. The methodological design combines simulated annealing, tabu search and neighbourhood operators, structured in three phases: construction of initial solutions, feasibility search and refinement using local techniques. This approach increases the probability of obtaining high-quality results in medium- and large-scale instances.

The implementation in C++ and tests on reference instances confirm that the proposed algorithm achieves efficient solutions, reducing both the distance travelled and the number of vehicles used. This contributes to improving logistics performance and opens up the possibility of extending the proposal to emerging variants such as electric vehicle routing or scenarios with heterogeneous fleets.

This work was tested with 80 instances from the literature, obtaining feasible solutions in 97.5% of them. In 7 of the instances used, the reported optimum was obtained.

Índice

Capítulo 1. Introducción

1.1 Antecedentes

La logística representa un pilar esencial para el desarrollo económico global, al facilitar el flujo eficiente de bienes y servicios a través de cadenas de suministro cada vez más interconectadas. De acuerdo con estimaciones recientes, el sector logístico contribuye entre el 10% y el 15% del producto interno bruto (PIB) mundial, evidenciando su rol estratégico en la competitividad empresarial y la integración de mercados (Beetrack, 2024). Este impacto se acentúa en el contexto del comercio electrónico, cuyo crecimiento acelerado ha intensificado los desafíos asociados a la denominada última milla, etapa que puede concentrar hasta el 60% de los costos logísticos totales (Quadminds, 2022).

En paralelo, la logística sostenible emerge como una respuesta clave frente a las crecientes presiones ambientales y regulatorias. La adopción de tecnologías como vehículos eléctricos y algoritmos de optimización de rutas ha permitido reducir hasta un 20% las emisiones de CO_2 en entornos urbanos (Quadminds, 2022). Además de mejorar la eficiencia operativa, tales progresos incrementan la resiliencia de las cadenas de suministro, permitiéndoles responder con mayor solidez a variaciones económicas, eventos disruptivos y cambios en la normativa.

No obstante, persisten desafíos críticos que limitan el aprovechamiento pleno del potencial logístico. Factores tales como la congestión vial, la falta de integración tecnológica y la volatilidad en la demanda complican la planificación de rutas, contribuyendo a pérdidas anuales estimadas en más de 50 000 millones de dólares a escala global (ILS, 2024). En la última milla, la escasa visibilidad en tiempo real y la gestión deficiente de flotas originan retrasos en aproximadamente el 25% de las entregas, lo que repercute directamente en la experiencia del cliente: el 89% de los consumidores afectados por fallos logísticos no vuelve a comprar a la misma empresa (DispatchTrack, 2023).

En este escenario, la planificación eficiente de rutas de distribución se posiciona como un componente clave para mejorar el desempeño logístico. El Problema de Ruteo de Vehículos (en ingles Vehicle Routing Problem o VRP) constituye uno de los problemas centrales en investigación de operaciones y logística computacional. Se trata de un problema de optimización combinatoria que busca determinar rutas eficientes para una flota de vehículos encargada de atender un conjunto de clientes, minimizando costos asociados a distancia, tiempo o consumo energético, entre otros.

1.2 Planteamiento del Problema

La relevancia del VRP se manifiesta en tres dimensiones fundamentales:

- La reducción de costos operativos, al disminuir distancias recorridas y maximizar el aprovechamiento de la capacidad vehicular.
- La mejora en la calidad del servicio, mediante el cumplimiento de ventanas de tiempo que inciden en la satisfacción del cliente.
- La sostenibilidad ambiental, al reducir la huella de carbono del transporte terrestre.

Dentro de sus numerosas variantes, el VRP con múltiples depósitos, ventanas de tiempo y requerimientos de arco (Vehicle Routing Problem with Multiple Depots, Time Windows and Arc Requirements, MDVRPTWAR, por sus siglas en inglés) destaca por su elevada complejidad y aplicabilidad práctica. Esta variante considera múltiples centros de distribución, restricciones estrictas en los horarios de atención a los clientes, y requerimientos específicos sobre ciertas aristas. Estas condiciones incrementan significativamente la dificultad del problema, haciendo inviable su resolución exacta para instancias de tamaño medio o grande, incluso con técnicas avanzadas de programación matemática.

El MDVRPTWAR integra simultáneamente tres dimensiones críticas: la asignación eficiente de vehículos a depósitos distribuidos geográficamente, el respeto a ventanas de tiempo definidas para cada cliente (lo cual exige una sincronización precisa de los recorridos), y la obligación de visitar ciertos arcos, ya sea por condiciones logísticas, de seguridad o por políticas regulatorias. La primera dimensión, relativa a la presencia de

múltiples depósitos, introduce una complejidad adicional al requerir no solo la planificación de rutas óptimas, sino también la determinación estratégica de qué depósito debe atender a cada cliente, lo que impacta directamente en los costos y la factibilidad operativa del sistema. En cuanto a las ventanas de tiempo, su incorporación transforma el problema en uno de carácter fuertemente temporal, donde las decisiones de ruteo deben alinearse con restricciones estrictas de servicio, haciendo necesario considerar aspectos como el tiempo de tránsito, esperas y secuenciación horaria. Finalmente, los requerimientos de arco confieren una estructura dirigida al grafo del problema, obligando a considerar no solo los nodos, sino también las características y condiciones impuestas sobre los trayectos que los conectan.

En contextos reales, este tipo de problemas aparece en redes de distribución urbana donde existen zonas con restricciones de circulación, rutas obligatorias o corredores logísticos predefinidos. Estas particularidades hacen que la formulación y solución del MDVRPTWAR requieran modelos altamente especializados, capaces de equilibrar múltiples objetivos conflictivos y respetar simultáneamente restricciones topológicas, temporales y operacionales.

1.3 Objetivos

1.3.1 General

Diseñar e implementar un algoritmo híbrido de naturaleza metaheurística que permita generar soluciones factibles para el MDVRPTWAR, evaluando su desempeño en instancias de referencia reportadas en la literatura (Li y otros, 2019).

1.3.2 Específicos

- Analizar el estado del arte relacionado con el VRP, identificando los principales enfoques metodológicos, variantes estructurales del problema y las técnicas de solución más relevantes reportadas en la literatura.
- Diseñar un algoritmo híbrido de tipo metaheurístico, integrando componentes de generación de soluciones iniciales, mecanismos de mejora local y estrategias de

manejo de restricciones, con el fin de abordar de manera eficiente la complejidad combinatoria del MDVRPTWAR.

- Implementar computacionalmente el algoritmo propuesto, garantizando su modularidad, flexibilidad y capacidad para adaptarse a diversas configuraciones.
- Evaluar experimentalmente el desempeño del algoritmo, aplicándolo sobre instancias de prueba reconocidas en la literatura (Li y otros, 2019) y analizando los resultados en términos de calidad de las soluciones y eficiencia computacional.
- Comparar y discutir los resultados obtenidos, identificando fortalezas y limitaciones del enfoque propuesto, así como oportunidades de mejora o líneas de trabajo futuro que contribuyan al avance en la resolución de problemas de ruteo complejos.

1.4 Hipótesis

Se postula que la implementación de un algoritmo híbrido de naturaleza metaheurística, que utilice el Recocido Simulado para resolver el MDVRPTWAR permitirá obtener soluciones factibles en al menos el 90% de las instancias de la literatura.

1.5 Justificación

La optimización del ruteo vehicular en el MDVRPTWAR persigue, en primer lugar, mejoras sustanciales en la eficiencia operativa. Estudios demuestran que una planificación óptima de rutas puede reducir hasta en un 25 % los costos de combustible y mantenimiento, al mismo tiempo que garantiza entregas puntuales (Fernández y otros, 2018). Este ahorro permite a las empresas atender un mayor volumen de pedidos sin necesidad de ampliar su flota, fortaleciendo así su competitividad y capacidad de respuesta en mercados cada vez más exigentes.

Adicionalmente, esta investigación aporta a la sostenibilidad ambiental, pues el transporte de mercancías representa entre el 8 % y el 10 % de las emisiones globales de CO_2 (Organisation for Economic Co-operation and Development (OECD), 2021). La

eliminación de trayectos innecesarios y el cumplimiento de restricciones de arcos contribuyen a reducir las emisiones en un rango estimado del 5 % al 25 % (International Transport Forum, 2023). De esta manera, el desarrollo de un algoritmo híbrido que aborde de forma efectiva múltiples depósitos, ventanas de tiempo y requerimientos de arco no solo incrementa la rentabilidad empresarial, sino que también promueve prácticas logísticas más responsables con el medio ambiente.

1.6 Alcances y Limitaciones

1.6.1 Alcance

Se utilizará el lenguaje C++ para la implementación del algoritmo híbrido metaheurístico para el MDVRPTWAR, evaluado sobre las instancias estándar de la literatura (Li y otros, 2019).

1.6.2 Limitación

Al centrarse solo en estas instancias y en la variante mencionada, los resultados no son extrapolables a otros escenarios ni cubren otras variantes del VRP.

1.7 Organización de la Tesis

El resto del trabajo está organizado de la siguiente manera:

El Capítulo 2 está dedicado al Marco Teórico y Estado del Arte, en el cual se aborda la definición del Problema de Ruteo de Vehículos, sus principales variantes, su complejidad computacional y los enfoques de solución propuestos en la literatura.

En el Capítulo 3 se presenta la Formulación del Problema, detallando el modelo matemático, las restricciones y las características de las instancias empleadas en la experimentación.

El Capítulo 4 desarrolla la Metodología de Trabajo, describiendo el diseño del algoritmo propuesto.

En el Capítulo 5 se expone la Experimentación y Análisis de Resultados, incluyendo los parámetros, los resultados obtenidos, comparaciones con soluciones de referencia y un análisis del rendimiento del enfoque propuesto.

Finalmente, el Capítulo 6 presenta las Conclusiones y Líneas de Trabajo Futuro, donde se reflexiona sobre los principales logros, se evalúa el cumplimiento de los objetivos planteados y se proponen posibles extensiones o mejoras para investigaciones posteriores.

Capítulo 2. Marco Teórico y Estado del Arte

2.1 Problema del Ruteo de Vehículos (VRP)

El VRP constituye uno de los problemas clásicos y más estudiados en el ámbito de la optimización combinatoria. Introducido formalmente por (Dantzig & Ramser, 1959), su formulación original abordaba la planificación de rutas para la distribución de gasolina desde un depósito central hacia una serie de estaciones de servicio, con el objetivo de minimizar la distancia total recorrida. Desde entonces, el VRP ha evolucionado en múltiples variantes que responden a escenarios logísticos reales de creciente complejidad.

En su versión canónica, el VRP se define sobre un grafo dirigido o no $G = (V, A)$, donde $V = \{v_0,\ v_1, \dots ,\ v_n\}$ representa el conjunto de nodos o vértices (un nodo depósito y n clientes), y $A = \{(v_i, v_j)\ |v_i, v_j\ \in V\}$ el conjunto de aristas o arcos con costos asociados, típicamente proporcionales a la distancia o al tiempo de viaje. El objetivo consiste en determinar un conjunto de rutas, una por cada vehículo disponible, que partan y regresen al depósito v_0, cubriendo a todos los clientes exactamente una vez y respetando restricciones operativas Figura 2.1.

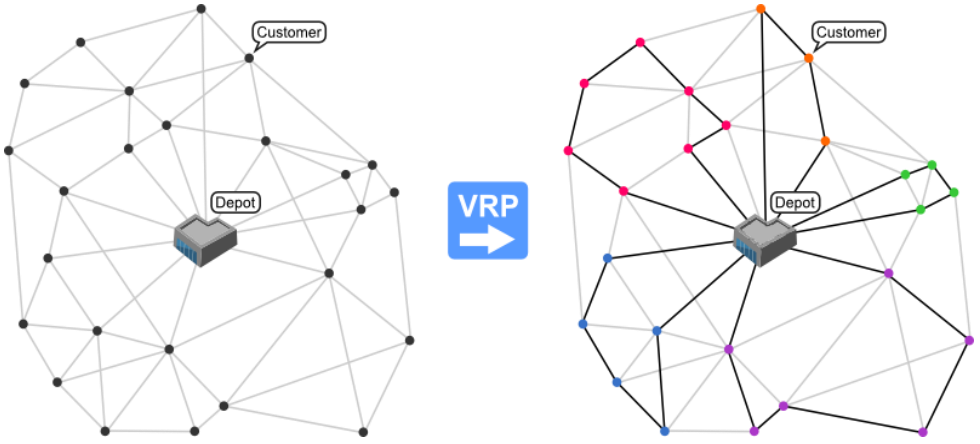

Figura 2.1. Un ejemplo de VRP (izquierda) y su solución (derecha). https://neo.lcc.uma.es/vrp/vehicle-routing-problem/

Su estructura básica sirve como punto de partida para el desarrollo de variantes más complejas que modelan condiciones del mundo real. La estructura básica se asienta sobre los siguientes elementos fundamentales:

- Depósito(s): Nodo central de operaciones donde los vehículos inician/finalizan rutas.
- Clientes: Puntos de demanda que requieren servicio.
- Vehículos: Unidades de transporte que ejecutan rutas.
- Rutas: Secuencia ordenada de visitas a clientes por un vehículo.

Interacciones críticas:

Tabla 2.1. Interacciones críticas de los elementos del VRP

Elemento	Influencia en el sistema	Restricciones típicas
Depósito	Define topología de rutas	Horarios operativos, capacidad almacén
Clientes	Generan demanda y restricciones	Ventanas tiempo, demanda, servicio
Vehículos	Limitan factibilidad de soluciones	Capacidad, autonomía, costos
Rutas	Determinan eficiencia operacional	Circuitos válidos, optimización

2.2 Variantes del VRP

Debido a la diversidad en el comportamiento de estos componentes como muestra la Figura 2.2, se originan una gran cantidad de variantes a este problema, la Figura 2.3 muestra algunas de las principales.

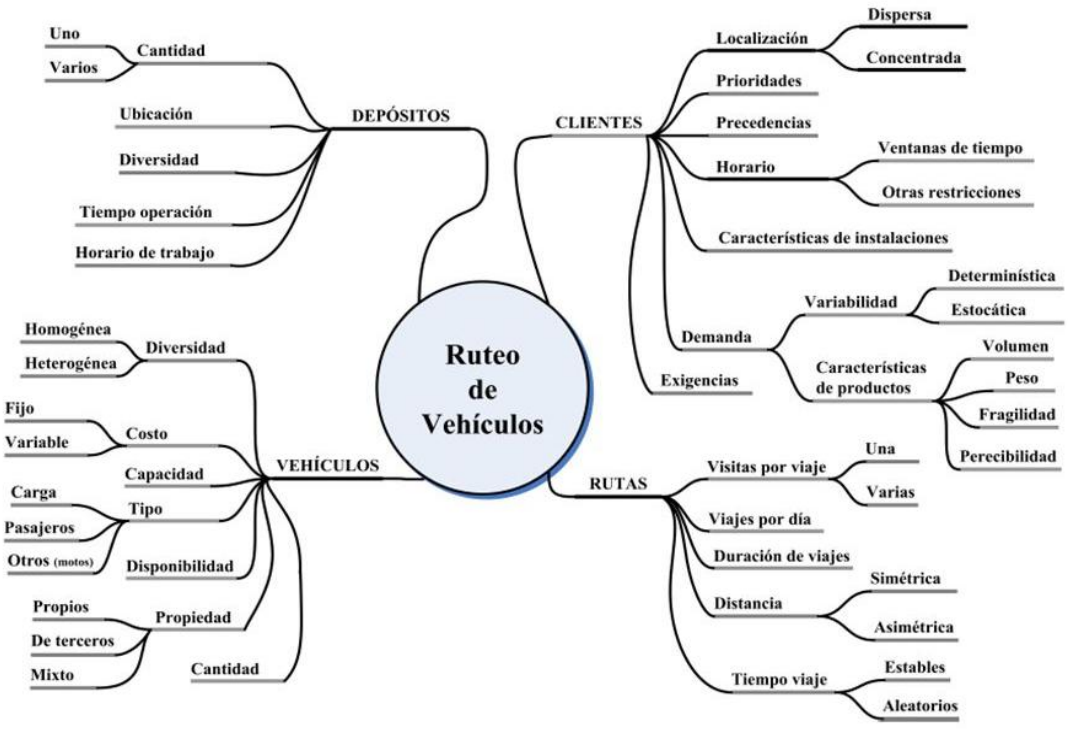

Figura 2.2. Componentes del VRP. Recuperado de: https://shorturl.at/99QFB

2.2.1 VRP con Vehículos Capacitados (CVRP)

El CVRP (Capacitated Vehicle Routing Problem) constituye la formulación básica del VRP, donde cada vehículo posee una capacidad homogénea limitada y cada cliente presenta una demanda conocida. El objetivo es minimizar la distancia total recorrida (o el costo asociado) al servir a todos los clientes sin exceder la capacidad de ningún vehículo. Esta versión fue introducida por primera vez por (Dantzig & Ramser, 1959).

2.2.2 VRP con Ventanas de Tiempo (VRPTW)

El VRPTW (Vehicle Routing Problem with Time Windows) añade al CVRP restricciones temporales: a cada cliente i se le asigna un intervalo $[e_i, l_i]$ en el cual debe iniciarse el servicio. Los vehículos pueden llegar antes de e_i y esperar, pero no pueden atender después de l_i. (Solomon, 1987) fue el primero en proponer algoritmos específicos para este problema, mostrando la importancia de integrar criterios de sincronización en entornos urbanos y de paquetería.

2.2.3 VRP con Múltiples Depósitos (MDVRP)

En el MDVRP (Multi-Depot Vehicle Routing Problem) existen múltiples depósitos, y la decisión incluye no solo la secuencia de clientes en cada ruta, sino también la asignación de cada cliente a uno de los depósitos disponibles. La primera formulación sistemática de esta variante se atribuye a (Min & Yih, 1989), quienes estudiaron la problemática de programar rutas desde varios centros de distribución, enfatizando su relevancia en redes logísticas regionales.

2.2.4 VRP con Backhauls (VRPB)

El VRPB (Vehicle Routing Problem with Backhauls), particiona a los clientes en dos conjuntos: entrega y recogida. Se impone la restricción de que, en cada ruta, primero deben completarse todas las entregas antes de iniciar las recogidas. (Toth & Vigo, An exact algorithm for the vehicle routing problem with backhauls, 1997) presentaron el primer algoritmo exacto enmarcado en programación entera para esta variante, destacando la relevancia práctica en sectores donde la carga de retorno no puede mezclarse con la de reparto.

2.2.5 VRP con Entrega Dividida (SDVRP)

En el SDVRP (Split Delivery Vehicle Routing Problem) se permite que un mismo cliente sea servido por varios vehículos (entregas divididas), lo cual puede reducir el número total de viajes o la distancia recorrida. (Dror & Trudeau, 1990) introdujeron este modelo, demostrando mediante experimentos que la flexibilidad de “dividir” la entrega puede generar ahorros significativos, especialmente cuando las demandas individuales exceden un porcentaje crítico de la capacidad vehicular.

2.2.6 VRP con Recogida y Entrega (VRPPD)

El VRPPD (Vehicle Routing Problem with Pickup and Delivery) extiende el CVRP al considerar solicitudes de recogida y entrega interdependientes: cada solicitud define un nodo de origen (recogida) y otro de destino (entrega), vinculados por una precedencia que obliga a que ambos sean servidos por el mismo vehículo en el orden correcto. La primera exposición formal de este problema aparece en el capítulo dedicado al PDPTW

en (Toth & Vigo, Vehicle routing: problems, methods, and applications, 2014), señalando su amplia aplicación en transporte de mercancías y servicios de pasajeros.

2.2.7 VRP Abierto (OVRP)

En el OVRP (Open Vehicle Routing Problem) las rutas son caminos abiertos: los vehículos parten del depósito, atienden a sus clientes y no regresan al punto de partida. (Schrage, 1981) fue el primero en describir este escenario, subrayando su pertinencia para empresas que subcontratan flotas y no requieren retorno al depósito, como mensajería a domicilio o distribución con reparto final a pie.

2.2.8 VRP Verde (GVRP)

El GVRP (Green Vehicle Routing Problem) incorpora criterios ambientales al objetivo tradicional, buscando minimizar emisiones de CO_2 y consumo energético, además de distancia o coste económico. (Bektaş & Laporte, 2011) presentaron el primer modelo sistemático de esta variante, introduciendo funciones objetivo que miden impacto ecológico y promoviendo rutas más “verdes” en la industria del transporte.

2.2.9 VRP de Vehículos Eléctricos (EVRP)

El EVRP (Electric Vehicle Routing Problem) añade las restricciones de autonomía de vehículos eléctricos: nivel de carga de la batería, disponibilidad de estaciones de recarga y tiempo de recarga. (Schneider y otros, 2014) fueron pioneros en formalizar esta variante, ofreciendo un modelo con estaciones intermedias y ventanas de tiempo, esencial para la planificación de flotas eléctricas en ciudades sostenibles.

A partir del análisis de la literatura y considerando el número y tipo de restricciones involucradas, se propone una clasificación cualitativa de las variantes del VRP en función de su dificultad computacional relativa. Dicha clasificación se alinea con observaciones empíricas reportadas por estudios como los de (Vidal y otros, 2020), (Lenstra & Rinnooy Kan, 1981) y (Eksioglu y otros, 2009), quienes reconocen el incremento en la complejidad al incorporar restricciones adicionales.

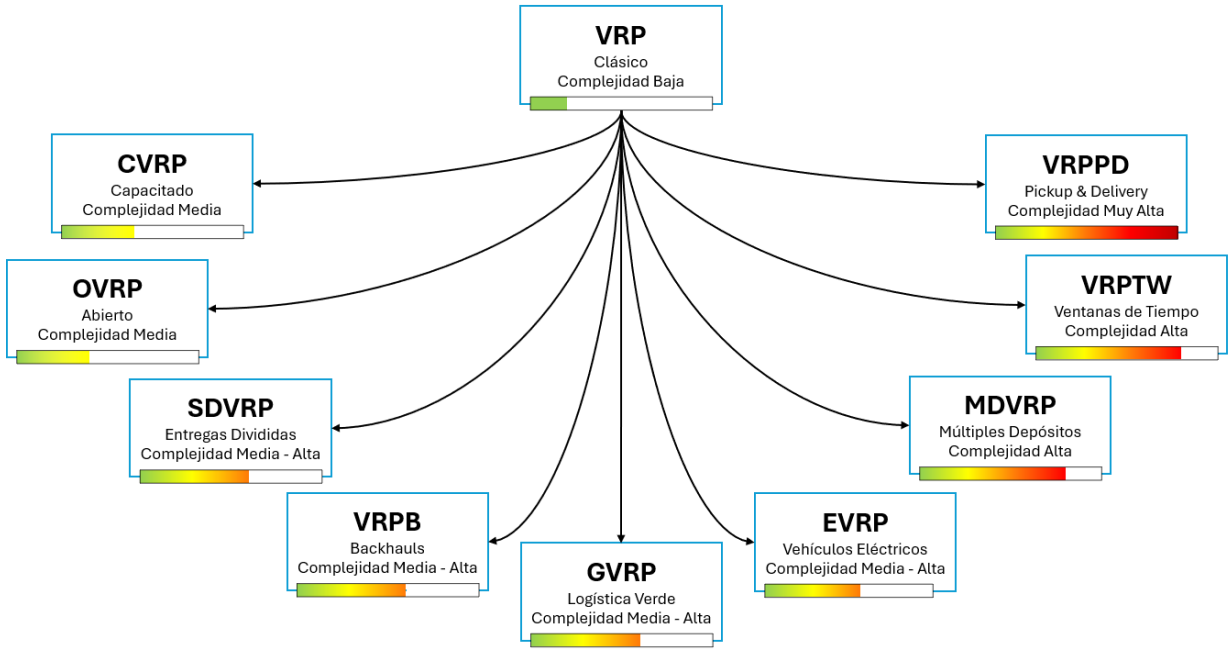

Figura 2.3. Principales Variantes VRP y su complejidad por restricciones

2.3 Complejidad Computacional

La complejidad computacional es una rama de la teoría de la computación que estudia los recursos principalmente tiempo y memoria necesarios para resolver un problema algorítmico, en función del tamaño de su entrada.

Clasificación de problemas según su complejidad computacional (Garey & Johnson, 1979):

- **P** (Polynomial Time): Problemas que pueden resolverse en tiempo polinomial mediante un algoritmo determinista.
- **NP** (Nondeterministic Polynomial Time): Problemas cuyas soluciones pueden verificarse en tiempo polinomial.
- **NP-Complete**: Problemas en NP tan difíciles como cualquier otro en NP (si se resuelve uno, se pueden resolver todos los NP).
- **NP-Hard**: Problemas al menos tan difíciles como los NP-Completo, pero que no necesariamente están en NP (pueden no tener verificación polinomial).

El Problema de Ruteo de Vehículos se clasifica como NP-Hard incluso en su versión más sencilla, el CVRP, debido a su estrecha relación con el Travelling Salesman Problem (TSP). Esta clasificación implica que no existe un algoritmo de tiempo polinomial conocido que resuelva exactamente todas sus instancias en un tiempo razonable (Lenstra & Rinnooy Kan, 1981).

Cuando se introducen restricciones adicionales como ventanas de tiempo, múltiples depósitos, flotas heterogéneas o requerimientos de arco la complejidad se agrava, ya que la estructura de factibilidad y de evaluación de soluciones se vuelve aún más intrincada. Este carácter combinatorio explica por qué el VRP constituye hoy un referente para el desarrollo de técnicas avanzadas de optimización, tanto exactas como aproximadas, y demanda el diseño de métodos de búsqueda balanceados entre calidad de solución y tiempo de cómputo.

2.4 Métodos de Solución

Debido a esta complejidad, el VRP ha sido considerado uno de los benchmarks más importantes en investigación de operaciones (Golden y otros, 2008). Entender su naturaleza computacional es crucial para seleccionar estrategias de solución adecuadas, especialmente cuando se enfrentan instancias reales de gran escala o con restricciones específicas.

Esta naturaleza intratable ha motivado el desarrollo de una amplia gama de estrategias de solución. En términos generales, dichas estrategias se agrupan en las siguientes categorías principales: métodos exactos, heurísticas, metaheurísticas, enfoques híbridos, y algoritmos a la medida Figura 2.4.

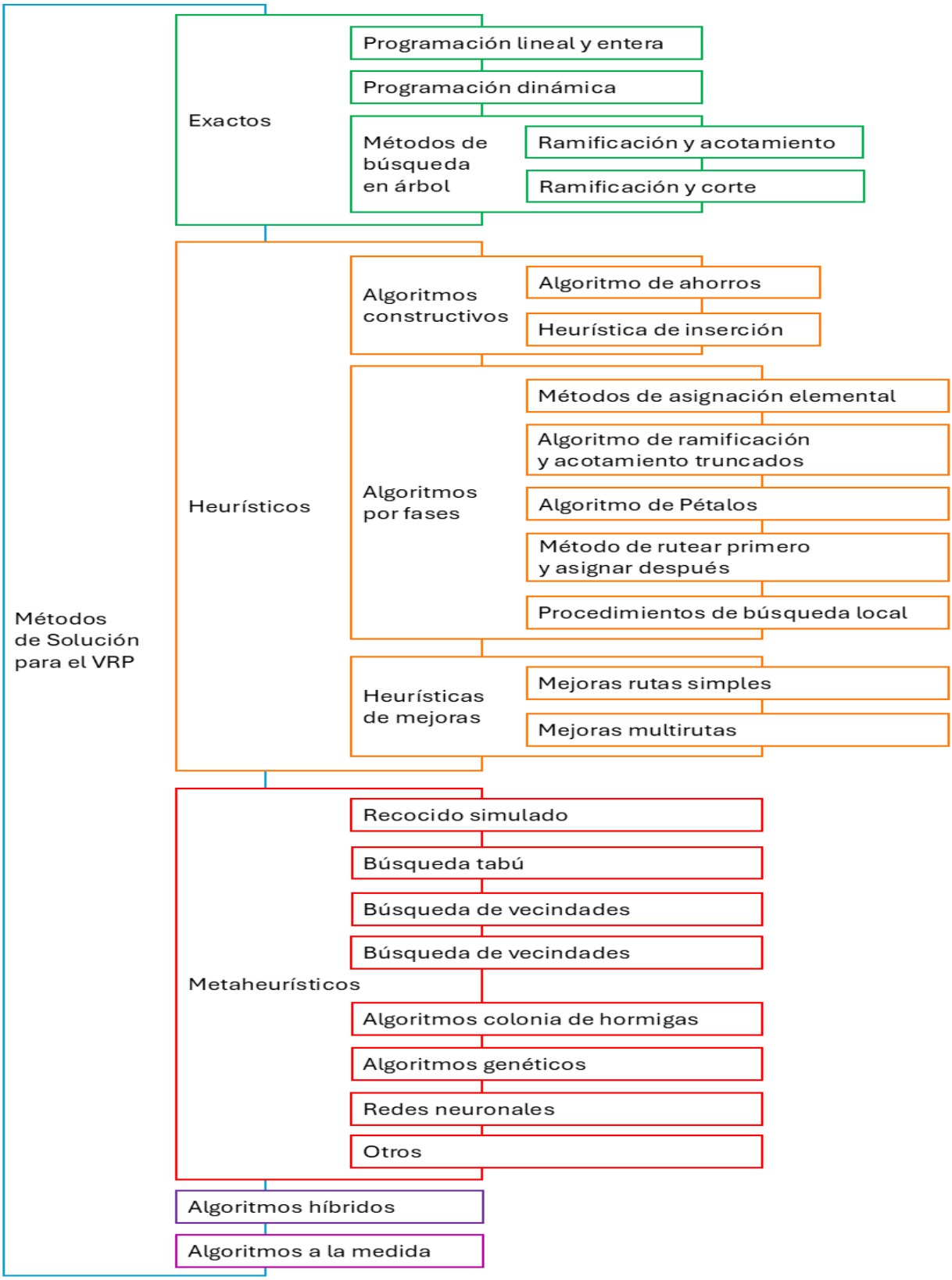

Figura 2.4. Taxonomía según métodos de solución al VRP

Enfoques más relevantes:

2.4.1 Métodos Exactos

Los métodos exactos garantizan la optimalidad de la solución, pero su coste computacional crece exponencialmente con el número de clientes, limitando su aplicación a instancias pequeñas o medianas.

2.4.1.1 Programación Entera Mixta (MIP)

Plantea el VRP como un modelo lineal entero, resoluble mediante solvers comerciales. Permite incorporar de forma directa capacidades y ventanas de tiempo, pero su escalabilidad es reducida (Laporte, The vehicle routing problem: An overview of exact and approximate algorithms, 1992).

2.4.1.2 Branch-and-Bound

Explora sistemáticamente el espacio de soluciones mediante partición recursiva, podando ramas garantizando cotas inferiores y superiores. Es la base de muchos solvers exactos para VRP (Toth & Vigo, The Vehicle Routing Problem, 2002).

2.4.1.3 Generación de Columnas (Column Generation)

Descompone el problema maestro en un subproblema de rutas y un problema de asignación de rutas, iterando entre ambos hasta converger a la solución óptima (Desaulniers y otros, 2002).

2.4.2 Métodos Heurísticos

Estas técnicas generan rápidamente soluciones iniciales factibles mediante reglas sencillas. Su principal ventaja es la velocidad de cómputo, aunque la calidad de las rutas obtenidas suele ser subóptima.

2.4.2.1 Clarke & Wright

Fusiona rutas partiendo de soluciones unitarias (un cliente por ruta) y combinándolas según el ahorro de coste que supone unir dos rutas (Clarke & Wright, 1964).

2.4.2.2 Vecino más Cercano (Nearest Neighbor)

Construye cada ruta añadiendo sucesivamente el cliente más próximo al último atendido, hasta saturar capacidad o cubrir todos los clientes (Golden y otros, 2008).

2.4.2.3 Heurística de Barrido (Sweep)

Ordena clientes por ángulo polar respecto al depósito y los agrupa en “sectores” que se convierten en rutas, respetando capacidad (Gillett & Miller, 1974).

2.4.2.4 Inserción más Barata (Cheapest Insertion)

Parte de una ruta base y en cada paso, inserta el cliente que menos coste adicional produzca (Gendreau y otros, 2002).

2.4.3 Metaheurísticas

Las metaheurísticas equilibran exploración con mecanismos de diversificación para escapar de óptimos locales en el espacio de soluciones, alcanzando alta calidad de la solución en tiempos de cómputo razonables. Entre las más usadas destacan:

2.4.3.1 Recocido Simulado (Simulated Annealing)

Emula el enfriamiento físico de metales para aceptar movimientos de coste superior con cierta probabilidad, controlada por una “temperatura” decreciente (Kirkpatrick y otros, 1983).

2.4.3.2 Búsqueda Tabú (Tabu Search)

Mantiene una memoria de movimientos recientes (lista tabú) para evitar ciclos y fomentar la exploración, cediendo entre intensificación y diversificación (Glover, Future paths for integer programming and links to artificial intelligence, 1986).

2.4.3.3 Algoritmos Genéticos (AG)

Trabajan con una población de soluciones, aplicando operadores de cruce y mutación para generar “descendientes” que heredan características de rutas eficaces (Holland, 1975).

2.4.3.4 GRASP (Greedy Randomized Adaptive Search Procedures)

Itera entre fases de construcción aleatoria controlada y mejora local, adaptando dinámicamente la lista de candidatos (Feo & Resende, 1995).

2.4.3.5 Colonia de Hormigas (Ant Colony Optimization)

Simula el comportamiento de las hormigas reales depositando feromonas en arcos prometedores, reforzando rutas de bajo coste con un proceso estocástico de aprendizaje (Dorigo & Gambardella, 1997).

2.4.4 Métodos Híbridos

Los métodos híbridos combinan distintas técnicas, como algoritmos exactos con metaheurísticas, o múltiples metaheurísticas entre sí, con el fin de mejorar la calidad de las soluciones y reducir los tiempos de cómputo. Esta estrategia busca aprovechar las fortalezas de cada enfoque, adaptándose mejor a variantes complejas del VRP (Xiao y otros, 2024).

2.4.5 Soluciones a la Medida

Las soluciones a la medida se diseñan para contextos específicos donde las restricciones logísticas no pueden representarse con modelos estándar. Estos enfoques se ajustan a características operativas particulares. Aunque no generalizables, ofrecen alta efectividad práctica al alinearse directamente con los requerimientos del entorno (Chen y otros, 2018).

Capítulo 3. Formulación del Problema

La variante del Problema de Ruteo de Vehículos que integra Múltiples Depósitos, Ventanas de Tiempo y Requerimientos de Arco (MDVRPTWAR) apenas ha sido tratada de manera conjunta en la literatura. En análisis del estado del arte constata que la mayoría de los estudios se centran por separado en MDVRPTW o en VRPTW. (Li y otros, 2019) abordan esta variante en el mantenimiento de maquinaria agrícola, donde proponen un algoritmo de luciérnaga con vecindades compuestas para resolverlo.

3.1 Modelo Matemático

El MDVRPTWAR considera múltiples centros de distribución, restricciones estrictas en los horarios de atención a los clientes, y requerimientos específicos sobre ciertas aristas. Estas condiciones incrementan significativamente la dificultad del problema, haciendo inviable su resolución exacta para instancias de tamaño medio o grande, incluso con técnicas avanzadas de programación matemática.

El MDVRPTWAR integra simultáneamente las siguientes características:

- Cada depósito tiene un número reducido de vehículos, todos los vehículos tienen la misma capacidad
- Los clientes se dividen en 2 conjuntos. El primero, clientes “clásicos”, los cuales tienen una demanda específica que debe ser atendido. El segundo, clientes “con requerimiento de arco”, los cuales, al finalizar su visita, deben forzosamente recorrer un arco específico.
- Ventanas de tiempo definidas para cada cliente, las cuales deben respetarse

El MDVRPTWAR puede definirse de la siguiente manera. Sea $G = (V, A)$ un grafo donde $V = \{v_1, \ldots, v_n, \ldots, v_{n+m}\}$ es un conjunto clientes y depósitos, y $A = \{(v_i, v_j) \,|v_i, v_j \in V,\ i \neq j\}$ es un conjunto de arcos. $V_n = \{v_1, \ldots, v_n\}$ representa los clientes o solicitudes, mientras que $V_m = \{v_{n+1}, \ldots, v_{n+m}\}$denota los depósitos, y $V = V_n \cup V_m$. Cada solicitud $i \in V_n$tiene

un tiempo de servicio s_i y una longitud no negativa q_i igual a la distancia desde su nodo inicial hasta su nodo final. Si la solicitud *i* es una solicitud de nodo, se considera que su nodo de inicio es el mismo que su nodo final, es decir, q_i es igual a 0. Cada vehículo se mueve a una velocidad constante H y tiene un límite de duración máxima T y un límite de carga máxima Q. El objetivo de MDVRPTWAR es minimizar la suma ponderada del coste fijo relacionado con el número de vehículos y el coste total de viaje de todos los vehículos y satisfacer las siguientes restricciones.

1. Cada vehículo empieza y termina en el mismo depósito.
2. El número de vehículos utilizados en cada depósito no puede superar el tamaño de la flota.
3. Cada cliente o solicitud debe ser visitada exactamente una vez por un vehículo.
4. La carga y la duración totales (incluidos los tiempos de viaje y los tiempos de servicio) de cada vehículo no superan el límite preestablecido Q y T, respectivamente.
5. El servicio en cada solicitud i debe comenzar dentro de la ventana temporal $[e_i, l_i]$. Si un vehículo llega a la solicitud i antes de la hora e_i esperará.

A continuación, se explica la formulación matemática del problema:

Conjuntos

$\boldsymbol{V}$	Conjunto total de nodos, que incluye tanto clientes como depósitos
$\boldsymbol{V_n}$	Subconjunto de V correspondiente a los clientes
$\boldsymbol{V_m}$	Subconjunto de V correspondiente a los depósitos
$\boldsymbol{K}$	Conjunto de vehículos disponibles

Parámetros

$\boldsymbol{c_{ij}}$	Distancia (o coste del arco) entre los clientes i y j
$\boldsymbol{e_i}$	Límite inferior de la ventana de tiempo del cliente i
$\boldsymbol{l_i}$	Límite superior de la ventana de tiempo del cliente i
$\boldsymbol{s_i}$	Tiempo de servicio requerido por el cliente i
$\boldsymbol{q_i}$	Longitud del requerimiento de arco i
$\boldsymbol{d_i}$	Demanda del cliente i
$\boldsymbol{T}$	Tiempo máximo de operación de cada vehículo
$\boldsymbol{H}$	Velocidad de cada vehículo

$\boldsymbol{Q}$	Capacidad máxima de cada vehículo
$\boldsymbol{K_d}$	Número de vehículos disponibles en el depósito d
$\boldsymbol{\alpha}$	Coeficiente que pondera la distancia total recorrida por los vehículos
$\boldsymbol{\beta}$	Coeficiente que pondera cada requerimiento de arco
Υ	Coeficiente que pondera (o costo fijo) de cada vehículo utilizado

Variables de decisión

$\boldsymbol{X_{ij}^k} \in \{\mathbf{0}, \mathbf{1}\}$	- 1 si el vehículo k viaja directamente desde el nodo i al nodo j - 0 en caso contrario
$\boldsymbol{w_{ik}}$	Tiempo de inicio del servicio al cliente i por vehículo k
$\boldsymbol{a_{ik}} \in \{\mathbf{0}, \mathbf{1}\}$	- 1 si el vehículo k finaliza la ruta con la solicitud del cliente i - 0 en caso contrario
$\boldsymbol{b_{ik}} \in \{\mathbf{0}, \mathbf{1}\}$	- 1 si el vehículo k pasa por el cliente i - 0 en caso contrario

Función objetivo

$$Minimizar\ f = \alpha \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}X_{ij}^k + \beta \sum_{i\in V_n} q_i + \Upsilon \sum_{k\in K}\sum_{i\in V_m}\sum_{j\in V_n} X_{ij}^k \quad (1)$$

La función objetivo (1) tiene como propósito minimizar el costo total del diseño de las rutas. El primer sumando representa el costo total de las distancias recorridas por todos los vehículos entre todos los nodos, ponderado por el coeficiente α. El segundo sumando cuantifica el coste fijo asociado al uso de vehículos, ponderado por el coeficiente β. Este término penaliza soluciones que requieren un mayor número de vehículos, promoviendo una utilización más eficiente de la flota disponible. Este enfoque busca un balance entre eficiencia operativa (distancias mínimas) y uso racional de recursos (cantidad de vehículos).

3.2 Restricciones

El modelo matemático formulado contempla un conjunto de restricciones que aseguran la factibilidad operativa del problema MDVRPTWAR.

$$\sum_{k\in K}\sum_{j\in V_n} X_{dj}^{k} \le K_d \,, \forall d \in V_m \quad (2)$$

$$\sum_{d\in V_m}\sum_{j\in V_n} X_{dj}^{k} \le 1 \,, \quad \forall k \in K \quad (3)$$

$$\sum_{d\in V_m}\sum_{j\in V_n} X_{dj}^{k} = \sum_{i\in V_n} a_{ik} \,, \quad \forall k \in K \quad (4)$$

$$\sum_{d\in V_m} X_{id}^{k} = a_{ik} \,, \forall k \in K, i \in V_n \quad (5)$$

$$\sum_{j\in V} X_{ji}^{k} = a_{ik} + b_{ik} \,, \forall k \in K, i \in V_n \quad (6)$$

$$\sum_{j\in V_n} X_{ij}^{k} = b_{ik} \,, \forall k \in K, i \in V_n \quad (7)$$

$$\sum_{k\in K} (a_{ik} + b_{ik}) = 1 \,, \quad \forall i \in V_n \quad (8)$$

$$\sum_{i\in V_n} X_{di}^{k} = \sum_{i\in V_n} X_{id}^{k} \,, \; k \in K, \forall d \in V_m \quad (9)$$

$$\sum_{i\in V_m}\sum_{j\in V_m} X_{ij}^{k} = 0 \,, \forall k \in K \quad (10)$$

$$\sum_{i\in V_n} X_{ii}^{k} = 0 \,, \forall k \in K \quad (11)$$

$$w_{ik} + S_i + \frac{q_i + c_{ij}}{H} - w_{jk} \le M\left(1 - X_{ij}^{k}\right), \forall k \in K, i, j \in V_n, i \ne j \quad (12)$$

$$e_i \le w_{ik} \le l_i \,, \forall i \in V \quad (13)$$

$$w_{ik} + S_i + \frac{q_i + c_{id}}{H} - l_d \le M\left(1 - X_{id}^{k}\right), \forall i \in V_n, k \in K, d \in V_m \quad (14)$$

$$\sum_{i\in V}\sum_{j\in V_n} d_j X_{ij}^{k} \le Q \,, \forall k \in K \quad (15)$$

$$w_{jk} + S_i + \frac{q_j + c_{jd}}{H} - w_{dk} - T \le M\left(1 - X_{jd}^{k}\right), \forall k \in K, d \in D, j \in V_n \quad (16)$$

$$X_{ij}^{k} \in \{0, 1\}, \forall i, j \in V, k \in K \quad (17)$$

$$w_{ik} \ge 0 \,, \forall i \in V_n, k \in K \quad (18)$$

$$a_{ik} \in \{0, 1\}, \forall i \in V_n, k \in K \quad (19)$$

$$b_{ik} \in \{0, 1\}, \forall i \in V_n, k \in K \quad (20)$$

Las restricciones (2) y (3) controlan la disponibilidad y asignación de vehículos. La restricción (2) limita el número de vehículos que pueden salir de cada depósito, de acuerdo con su disponibilidad, mientras que la restricción (3) impide que un mismo vehículo sea asignado simultáneamente a más de un depósito.

Las restricciones de la (4) a la (8) aseguran una correcta asignación de los clientes a los vehículos y el cumplimiento de que cada cliente sea atendido exactamente una vez. En particular, la restricción (4) garantiza la consistencia entre la salida de un vehículo desde un depósito y la asignación del cliente que abre su ruta; las restricciones (5) a (7) establecen la continuidad de la ruta, identificando los clientes atendidos al final o en posición intermedia; finalmente, la restricción (8) impone que cada cliente sea visitado una única vez por algún vehículo.

La restricción (9) establece que los vehículos deben iniciar y finalizar su recorrido en el mismo depósito. Por su parte, las restricciones (10) y (11) prohíben desplazamientos no permitidos, tales como viajes entre depósitos y ciclos triviales en los que un vehículo regrese al mismo cliente sin avanzar en la ruta.

Las restricciones (12) a (16) están asociadas a las condiciones temporales y operativas del sistema. La restricción (12) emplea una formulación para eliminar sub-rutas y garantizar una secuencia lógica entre visitas a los clientes, donde M representa una constante positiva lo suficientemente grande. Las restricciones (13) y (14) imponen que tanto los clientes como los depósitos sean visitados dentro de sus respectivas ventanas de tiempo. La restricción (15) limita la cantidad de carga transportada por vehículo según su capacidad máxima, y la (16) controla que la duración total del recorrido no exceda un umbral de trabajo preestablecido.

Finalmente, las restricciones de la (17) a la (20) definen la naturaleza de las variables de decisión. Se establece el carácter binario de las variables asociadas a los arcos y asignaciones de los clientes (restricciones 17, 19 y 20), así como la no negatividad de las variables temporales (restricción 18).

Este marco formal proporciona el soporte necesario para el desarrollo y validación del algoritmo híbrido propuesto.

3.3 Instancias

Para validar el desempeño del algoritmo híbrido propuesto, se emplean instancias estándar procedentes de la literatura (Li y otros, 2019). Estas instancias han sido diseñadas para reflejar la complejidad real del problema, y proporcionan un formato (Figura 3.1) unificado que facilita la comparación con otros enfoques publicados.

Estructura de los ficheros de instancia:

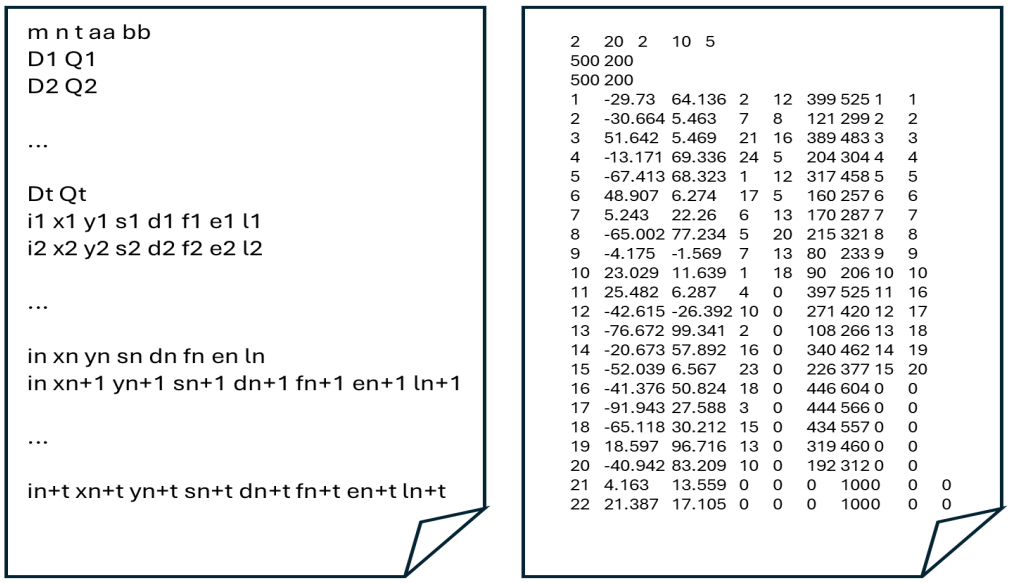

Figura 3.1. Estructura de los ficheros de instancias (izquierda) y ejemplo de instancia A01 (derecha)

La primera línea contiene cinco valores que definen los parámetros globales de la instancia:

m n t aa bb

- **m**: número de vehículos disponibles (valor entero).
- **n**: número de clientes (valor entero).
- **t**: número de depósitos (MDVRP) (valor entero).
- **aa**: número de solicitudes a nivel de nodo (clientes) (valor entero).
- **bb**: número de solicitudes a nivel de arco (clientes con precedencia o requerimientos de arco) (valor entero).

Las siguientes **t** líneas describen las rutas y los vehículos de cada deposito (una línea por cada depósito, con dos valores enteros) con la siguiente información:

D Q

- **D**: duración máxima permitida para la ruta de un vehículo que parte de ese depósito (valor entero).
- **Q**: capacidad máxima de carga de cada vehículo adscrito a ese depósito (valor entero).

Las siguientes **n + t** líneas contienen, para cada cliente y depósito, la siguiente información (los **aa** primeros son clientes, los **bb** siguientes son clientes con precedencia o requerimientos de arco y por los **t** últimos son los depósitos):

i x y s d e l a b

- **i**: identificador del cliente (valor entero).
- **x, y**: coordenadas cartesianas del cliente (valores decimales).
- **s**: tiempo de servicio requerido en el cliente (valor entero).
- **d**: demanda del cliente (valor entero).
- **e, l**: comienzos y fin de la ventana de tiempo para iniciar el servicio (valores enteros).
- **a**: nodo o cliente de origen asociado a la solicitud (valor entero).
- **b**: nodo o cliente de destino asociado a la solicitud (valor entero).

Para la evaluación se emplearon 80 instancias proporcionadas por los autores del articulo base (Li y otros, 2019), que refieren se construyeron a partir de las generadas por (Cordeau y otros, 2001), las que modificaron para adaptarlas al problema MDVRPTWAR, agrupadas en cuatro conjuntos (A, B, C y D), según se detalla en la Tabla 3.1; donde **n** se refiere al total de clientes, **nr** a los clientes normales, **ar** a los requerimientos de arcos o clientes con precedencia, **m** cantidad de depósitos, **Kd** número de vehículos disponibles por cada deposito, **Q** capacidad máxima de los vehículos y **T** la duración máxima de las rutas.

Tabla 3.1. Información básica de las instancias

Instancia	n	nr	ar	m	Kd	Q	T
A01	15	10	5	2	2	200	500
A02	15	10	5	2	2	195	480
A03	15	10	5	2	2	190	750
A04	15	10	5	2	3	185	440
A05	15	10	5	2	2	180	420
A06	15	10	5	2	2	175	800
A07	15	10	5	2	2	200	500
A08	15	10	5	2	2	190	475
A09	15	10	5	2	2	180	450
A10	15	10	5	2	3	170	425
A11	15	10	5	2	2	200	500
A12	15	10	5	2	2	195	480
A13	15	10	5	2	3	190	460
A14	15	10	5	2	2	185	440
A15	15	10	5	2	2	180	750
A16	15	10	5	2	2	175	400
A17	15	10	5	2	2	200	500
A18	15	10	5	2	2	190	475
A19	15	0	5	2	2	180	450
A20	15	10	5	2	2	170	725
B01	36	24	12	4	2	200	550
B02	72	48	24	4	3	195	480
B03	108	72	36	4	5	190	460
B04	144	96	48	4	6	185	590
B05	180	120	60	4	8	180	420
B06	216	144	72	4	9	175	400
B07	54	36	18	6	2	200	650
B08	108	72	36	6	5	190	475
B09	162	108	54	6	6	180	450
B10	216	144	72	6	7	170	425
B11	36	24	12	4	2	200	500
B12	72	48	24	4	3	195	480
B13	108	72	36	4	5	190	460
B14	144	96	48	4	6	185	440
B15	180	120	60	4	7	180	420
B16	216	144	72	4	10	175	400
B17	54	36	18	6	2	200	650
B18	108	72	36	6	4	190	475
B19	162	108	54	6	5	180	450
B20	216	144	72	6	8	170	425

Tabla 3.1. Información básica de las instancias (continuación)

Instancia	n	nr	ar	m	Kd	Q	T
C01	32	16	16	4	2	200	550
C02	64	32	32	4	3	195	480
C03	96	48	48	4	6	190	460
C04	128	64	64	4	7	185	590
C05	160	80	80	4	9	180	420
C06	192	96	96	4	11	175	400
C07	48	24	24	6	3	200	650
C08	96	48	48	6	5	190	475
C09	144	72	72	6	5	180	450
C10	192	96	96	6	7	170	425
C11	32	16	16	4	2	200	500
C12	64	32	32	4	3	195	480
C13	96	48	48	4	5	190	460
C14	128	64	64	4	7	185	440
C15	160	80	80	4	8	180	420
C16	192	96	96	4	9	175	400
C17	48	24	24	6	3	200	650
C18	96	48	48	6	3	190	475
C19	144	72	72	6	7	180	450
C20	192	96	96	6	8	170	425
D01	29	10	19	4	2	200	550
D02	58	20	38	4	3	195	480
D03	86	28	58	4	6	190	460
D04	115	38	77	4	7	185	590
D05	144	48	96	4	9	180	420
D06	173	58	115	4	11	175	400
D07	43	14	29	6	3	200	650
D08	86	28	58	6	5	190	475
D09	130	44	86	6	6	180	450
D10	173	58	115	6	8	170	425
D11	29	10	19	4	2	200	500
D12	58	20	38	4	3	195	480
D13	86	28	58	4	6	190	460
D14	115	38	77	4	8	185	440
D15	144	48	96	4	8	180	420
D16	173	58	115	4	11	175	400
D17	43	14	29	6	3	200	650
D18	86	28	58	6	4	190	475
D19	130	44	86	6	6	180	450
D20	173	58	115	6	8	170	425

Capítulo 4. Metodología y Diseño del Algoritmo

El presente trabajo propone una metaheurística híbrida diseñada para resolver el MDVRPTWAR. La metodología propuesta se articula a partir de un diagrama de flujo (Figura 4.1) que descompone el algoritmo híbrido en tres etapas fundamentales; Solución Inicial, Búsqueda de la Factibilidad y Refinamiento de la Solución.

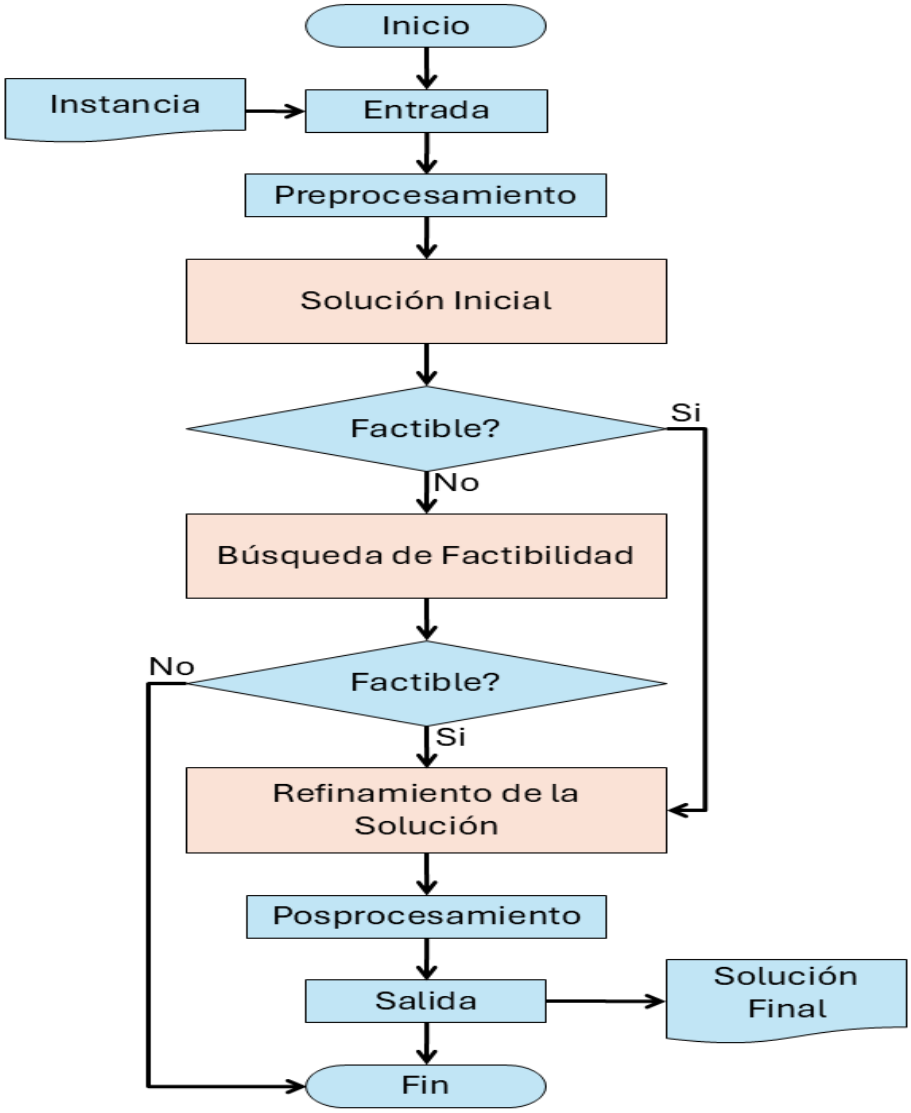

Figura 4.1. Diagrama de flujo del algoritmo

4.1 Solución Inicial

Este módulo tiene como objetivo generar soluciones iniciales a partir de una instancia del problema, las cuales servirán como punto de partida para las fases posteriores del algoritmo. La solución generada no necesariamente cumple todas las restricciones del problema, pero representa una estructura de ruteo razonable que será evaluada y eventualmente ajustada en fases posteriores.

Algoritmo 1. Solución Inicial

```
Entradas:
    •  C: Conjunto de clientes
    •  D: Conjunto de vehículos

Salidas:
    •  S: Solución inicial

Descripción de las variables:
    •  R: Conjunto de rutas
    •  C': Nuevo conjunto de clientes reordenados (o no) a partir de C
    •  D': Nuevo conjunto de vehículos reordenados (o no) a partir de D
    •  vehicleIndex: Índice referente a los vehículos dentro de D'
    •  t: Tiempo de llegada a un cliente
    •  T: Tiempo máximo de duración de la ruta
    •  q: Carga actual del vehículo
    •  Q: Carga máxima soportada por el vehículo

Procedimiento:
    R ← Ø                                   // Cjto. de rutas
    C' ← C
    D' ← D
    Alter(C', D')                           // Reordenar conjuntos
    vehicleIndex ← 1

    WHILE C' ≠ Ø DO
        IF vehicleIndex > |D'| THEN         //Si se necesitan más vehículos
            AddNewVehicle(D')               //Se agrega un nuevo vehículo
        END IF
        vehicle ← D'[vehicleIndex]
        route ← [vehicle]                   //Se crea una ruta para el vehículo
        Add(R, route)                       //Se agrega la nueva ruta a R

        FOR c IN C' DO
            t ← ArrivalTime(r, c)           //Se calcula el tiempo de llegada a v
            q ← RouteLoad(r) + Demand(c)     //Se calcula la carga actual


            IF t ≤ lc         // Si se llega a tiempo al cliente (lc fin de la ventana de tiempo de c)
              Y MeetsArcRequirements(r,c)   // Si r, cumple la precedencia de c
              Y q ≤ Q                        // Si no se sobrepasa el máximo de carga
              Y t < T                        // Si no se excede el límite de tiempo
              THEN
                Add(r, c)                    //Se enruta c en r
                Delete(C',c)                //Se elimina c de los clientes a enrutar
            END IF
        END FOR


        vehicleIndex ← vehicleIndex + 1
    END WHILE


    RETURN S(C',D',R)
```

Funcione auxiliares

- **`Alter(x, y)`**: Dado 2 conjuntos, selecciona uno y reordena sus elementos aleatoriamente. Ejemplo:

  `Alter(C,D) seleccionando al conjunto C`

  $C = \{0,1,2,3,4,5,6,7,8,9\}$ $C' = \{4,7,9,2,0,5,3,6,1,8\}$

- **`AddNewVehicle(x)`**: Crea un nuevo vehículo y lo agrega al conjunto *x*.
- **`Add(x, y)`**: Inserta el elemento *y* en el conjunto *x*.
- **`Delete(x, y)`**: Elimina el elemento *y* del conjunto *x*.
- **`ArrivalTime(x, y):`** Calcula el tiempo de llegada del vehículo de la ruta *x* al cliente *y*.
- **`RouteLoad(x)`**: Calcula la carga actual de la ruta *x*, sumando todas las demandas de sus clientes.
- **`Demand(x)`**: Retorna la demanda del cliente *x*.
- **`MeetsArcRequirements(x, y)`**: Retorna verdadero o falso, chequeando si en la ruta *x* se cumple el requerimiento de arco (o precedencia) para el cliente *y*.

Para la construcción de la solución inicial, el algoritmo emplea los siguientes pasos:

1. Inicialización de conjuntos
   - $C = \{v_1, \dots, v_n\}$: clientes a enrutar.
   - $D = \{k_1, \dots, k_x\}$: vehículos disponibles.
   - $R = \emptyset$: conjunto de rutas, inicialmente vacío.
2. Aleatorización

   Se generan permutaciones aleatorias de ambos conjuntos, C y D, obteniendo, C' y D', diversificando así los puntos de partida del algoritmo.

3. Construcción iterativa de rutas

   Para cada vehículo k_j en el orden de D':

   a) Se crea una nueva ruta $r_j = [k_j]$ y se añade a R.

   b) Se recorre secuencialmente la lista de clientes C'.

c) Para cada cliente v_i, se verifica si su inserción al final de r_j cumple simultáneamente:

- **Ventana de tiempo**: siendo t el instante de tiempo en el que el vehículo k_j llega al cliente v_i (`t ← ArrivalTime(r, v)`), entonces: $t \leq l_i$, se llega antes del fin de la ventana de tiempo.
- **Requerimientos de arco**: (`MeetsArcRequirements(r, v)`) si se cumple la precedencia de v_i.
- **Capacidad máxima**: siendo q la carga actual incluyendo la demanda de v_i del vehiculo k_j (`q ← RouteLoad(r) + Demand(v)`), entonces $q \leq Q$, la carga actual de la ruta más la demanda del cliente no excede la capacidad de carga del vehículo.
- **Duración de la ruta**: $t \leq T$, el tiempo desde que sale el vehículo del depósito hasta que atiende todos los clientes y regresa al depósito no excede el límite máximo de tiempo para las rutas.

Si todas las restricciones se satisfacen, se agrega v_i a r_j y se elimina de C'; en caso contrario, se continúa con el siguiente cliente.

4. Gestión de clientes no asignados

Al completar el recorrido de todos los vehículos en D', si todavía existen clientes en C', se añaden nuevos vehículos al conjunto D' y se repite el paso 3 hasta que C' quede vacío.

Al finalizar, el conjunto R contiene una ruta para cada vehículo utilizado, cubriendo todos los clientes. Puede ocurrir que $|R| > |D|$, lo cual constituye la única violación a la factibilidad.

La solución inicial generada está ligada intrínsicamente a los conjuntos C' y D', teniendo la siguiente estructura:

Estructura de solución	
$S = \{C', D', R\}$	• C' : Lista de clientes: • D' : Lista de vehículos • R : Lista de Rutas

Ejemplo de una solución para un problema con 10 clientes 2 depósitos con 2 vehículos cada uno:

$$S = \{\{4,7,9,2,0,5,3,6,1,8\}, \{1,3,2,4\}, \{(10,4,7,2,10), (11,9,11), (11,0,5,1,11), (10,3,6,8,10)\}\}$$

Donde:

- $\mathrm{C}' = \{4,7,9,2,0,5,3,6,1,8\}$, clientes ordenados aleatoriamente.
- $\mathrm{D}' = \{1,3,2,4\}$, vehículos ordenados aleatoriamente, donde los vehículos 1 y 2 pertenecen al primer deposito 10, y los vehículos 3 y 4 pertenecen al depósito 11.
- $\mathrm{R} = \{(10,4,7,2,10), (11,9,11), (11,0,5,1,11), (10,3,6,8,10)\}$, rutas que se representan con una secuencia de identificadores, partiendo de un depósito (deposito al cual pertenece el vehículo que ejecuta la ruta), luego la secuencia de clientes a visitar y finalizando en el mismo deposito del cual se partió. Pudiéndose representar de la siguiente forma: **10**,4,7,2,**10**,**11**,9,**11**,**11**,0,5,1,**11**,**10**,3,6,8,**10**. Donde 10 y 11 son los depósitos y representando 4 rutas.

4.2 Búsqueda de Factibilidad

Una vez construida una solución inicial, es común que esta no cumpla completamente con todas las restricciones del problema. Por tanto, resulta indispensable contar con un procedimiento que transforme dicha solución en una alternativa válida según los criterios definidos por la función objetivo. Esta fase tiene como propósito transformar soluciones infactibles, generadas en la fase de solución inicial, en soluciones factibles que cumplan con todas las restricciones del modelo.

Para abordar este proceso, se ha diseñado un algoritmo híbrido que combina la capacidad de exploración global del recocido simulado con la memoria de prohibiciones de la búsqueda tabú. A continuación, se describe brevemente el fundamento de estos algoritmos, así como su hibridación y aplicación específica dentro del marco de esta investigación.

4.2.1 Fundamentos del Recocido Simulado (SA)

El algoritmo de recocido simulado (*Simulated Annealing*, SA) fue introducido por primera vez por (Kirkpatrick y otros, 1983) como una técnica inspirada en el proceso físico de

recocido de metales, donde se busca alcanzar un estado de mínima energía mediante un enfriamiento gradual controlado. Esta metaheurística estocástica se basa en aceptar soluciones subóptimas con una probabilidad decreciente a medida que la temperatura del sistema disminuye, lo que permite escapar de óptimos locales y explorar eficientemente el espacio de soluciones. Su formulación matemática y su eficacia para abordar problemas combinatorios complejos lo convirtieron rápidamente en una herramienta ampliamente utilizada en optimización.

La primera aplicación documentada del recocido simulado al problema de ruteo de vehículos (VRP) fue realizada por (Osman, 1993). En su trabajo, Osman propuso un algoritmo híbrido que integra recocido simulado y búsqueda tabú para resolver el problema de ruteo de vehículos con capacidad limitada (*Capacitated Vehicle Routing Problem*, CVRP). Su enfoque demostró que el SA puede ser altamente efectivo para mejorar soluciones al permitir movimientos que temporalmente empeoran el objetivo, contribuyendo así a una búsqueda más diversificada y robusta. Desde entonces, el recocido simulado ha sido adaptado y extendido a numerosas variantes del VRP, consolidándose como una de las metaheurísticas más utilizadas en este dominio.

Representación del Algoritmo SA

$INPUT(T_0, \alpha, L, T_f)$
$T \leftarrow T_0$
$S_{act} \leftarrow Genera\_solucion\_inicial$
$WHILE\ T \geq T_f\ DO$
    BEGIN
    FOR $cont \leftarrow 1\ TO\ L(T)\ DO$
        BEGIN
        $S_{cand} \leftarrow Seleciona\_solucion\_N(S_{act})$
        $\delta \leftarrow coste(S_{cand}) - coste(S_{act})$
        $IF\ \left(U(0.1) < e^{\left(-\frac{\delta}{T}\right)}\right) OR\ (\delta < 0)\ THEN$
            $S_{act} \leftarrow S_{cand}$
        END
    $T \leftarrow \alpha(T)$
    END
$\{Escribe\ como\ solución, la\ mejor\ de\ las\ S_{act}\ visitadas\}$

- T_0: temperatura inicial
- T_f: temperatura final
- T: temperatura actual
- α: taza de enfriamiento
- L: número de iteraciones
- $e^{\left(-\frac{\delta}{T}\right)}$ probabilidad de aceptar la solución

Tomado de (Díaz y otros, 1996)

A grandes rasgos, SA genera soluciones vecinas iterativamente y utiliza un criterio de aceptación basado en la temperatura para decidir si se acepta o rechaza una solución candidata. Cuando la solución propuesta mejora el costo, se acepta; cuando empeora, se acepta con una probabilidad que depende de la diferencia de costo y de la temperatura

actual. Al inicio la temperatura es alta, permitiendo saltos grandes en el espacio de búsqueda, y gradualmente se enfría siguiendo un plan decreciente, hasta que solo se aceptan mejoras. De este modo, SA explora ampliamente el espacio de búsqueda y puede escapar de mínimos locales, tendiendo hacia el óptimo global cuando la temperatura se aproxima a cero.

4.2.2 Fundamentos de Búsqueda Tabú (TS)

La búsqueda tabú es una metaheurística de optimización combinatoria que extiende las búsquedas locales tradicionales incorporando memoria adaptativa. Fue planteada por (Glover, Future paths for integer programming and links to artificial intelligence, 1986) y formalizada en (Glover, Tabu search part I, 1989), habiéndose aplicado con éxito a problemas NP-Hard.

La búsqueda tabú mejora la búsqueda local al incorporar una memoria adaptativa que registra movimientos recientes para evitar ciclos y explorar nuevas regiones del espacio de soluciones. Su principio fundamental consiste en aceptar movimientos no mejorantes, si no están prohibidos por la lista tabú, con el fin de escapar de óptimos locales. Además, los criterios de aspiración permiten anular prohibiciones si la solución candidata mejora la mejor conocida, y mecanismos de intensificación y diversificación orientan la búsqueda hacia zonas prometedoras o poco exploradas.

En el contexto del problema de ruteo de vehículos, la búsqueda tabú fue aplicada por primera vez por (Willard, 1985), y desarrollada posteriormente por (Osman, 1993) y (Cordeau y otros, 2007), quienes propusieron variantes y extensiones específicas para problemas de ruteo con distintas restricciones. Estas adaptaciones han demostrado ser altamente efectivas para resolver instancias complejas del VRP.

4.2.3 Algoritmo de Búsqueda de Factibilidad

Algoritmo 2. Búsqueda de Factibilidad

```
Entradas:
    • initialSolution: Solución inicial no factible
    • initialTemperature: Temperatura inicial
    • finalTemperature: Temperatura final
    • iterationsNumber: longitud de la cadena de Markov
    • coolingFactor: Tasa de enfriamiento
    • minimumAcceptanceRate: Tasa mínima de aceptación

Salidas:
    • initialSolution: Actualizada con la solución factible encontrada

Descripción de las variables:
    • currentSolution: Solución actual
    • currentTemperature: Temperatura actual
    • generated, accepted: Contadores internos
    • acceptanceRate: Tasa de aceptación
    • localSearchCount: Número de vecinos a explorar en la búsqueda local
    • operatorWeights: Vector de pesos para selección de operadores
    • overheating: Número de recalentamientos permitidos
    • lastTemperatureAcceptanceRate: Ultima temperatura donde se alcanzó la tasa mínima de aceptación
    • maxVehicles: Número máximo de vehículos disponibles

Procedimiento:
    // Inicialización
    currentTemperature ← initialTemperature
    lastTemperatureAcceptanceRate ← initialTemperature
    currentCoolingFactor ← coolingFactor
    overheating ← 2
    currentSolution ← initialSolution
    // Registra la solución en la lista tabú
    IsTabu(currentSolution)

    // Bucle de recocido (enfriamiento)
    WHILE currentTemperature > finalTemperature DO
        generated ← 0
        accepted ← 0

        // Generar y evaluar vecinos
        FOR i ← 1 TO iterationsNumber DO
            // Generar una nueva solución
            neighbourSolution ← GenerateNeighbourSolution(currentSolution)
            // Mientras sea una solución vista
            WHILE IsTabu(neighbourSolution) DO
                // Generar una nueva solución
                neighbourSolution ← GenerateNeighbourSolution(currentSolution)
            END WHILE
            generated ← generated + 1

            IF IsFeasible(neighbourSolution) THEN            // Si es factible, terminar el algoritmo
                initialSolution ← neighbourSolution
                RETURN
            END IF

            IF Cost(neighbourSolution) ≤ Cost(currentSolution) THEN
                // Se encontró una mejor solución, aceptarla
                currentSolution ← neighbourSolution
                accepted ← accepted + 1
            ELSE                                                // La solución vecina es peor
                // Aceptación probabilística de una peor solución
                Δ ← Cost(neighbourSolution) – Cost(currentSolution)
                probabilityAcceptance ← Exp(- Δ / currentTemperature)
```

Algoritmo 2. Búsqueda de Factibilidad (continuación)

```
                probability ← Random(0,1)
                IF probability < probabilityAcceptance THEN
                    // Se acepta una peor solución
                    currentSolution ← neighbourSolution
                    accepted ← accepted + 1
                END IF
            END IF
        END FOR

        // Se actualiza la taza de aceptación
        acceptanceRate ← (generated > 0) ? ((accepted / generated)*100) : 0

        // Si la aceptación es baja, evaluar lanzar búsqueda local
        IF acceptanceRate < 50.0 THEN
            //Determina si límite de exploración
            localSearchCount ← ShouldActivateLocalSearch(currentTemperature,
                                                         initialTemperature,
                                                         finalTemperature)
            // Si se activa la búsqueda local
            IF localSearchCount > 0 THEN
                tempRatio ← currentTemperature / initialTemperature
                costRatio ← (Costo(currentSolution)-Costo(initialSolution))/Costo(initialSolution)
                infeasibilityFactor ← NumberOfVehiclesUsed(currentSolution) / maxVehicles

                // Actualizar vector de pesos de los operadores según estado de la búsqueda
                operatorWeights ← GetOperatorWeights(tempRatio,costRatio,infeasibilityFactor)

                // Búsqueda Local
                LocalSearch(currentSolution,operatorWeights,localSearchCount)

                IF IsFeasible(currentSolution) THEN            //Si es factible, terminar el algoritmo
                    initialSolution ← currentSolution
                    RETURN
                END IF

                IF Cost(currentSolution) ≤ Cost(initialSolution) THEN
                    initialSolution ← currentSolution
                END IF
            END IF
        END IF

        IF acceptanceRate ≥ minimumAcceptanceRate THEN
            lastTemperatureAcceptanceRate ← currentTemperature
        END IF

        // Actualizar temperatura
        IF (currentTemperature = initialTemperature) Y (acceptanceRate < minimumAcceptanceRate) THEN
            currentTemperature ← 2 * initialTemperature
        ELSE
            // Enfriamiento normal
            currentTemperature ← currentTemperature * currentCoolingFactor

            IF (currentTemperature ≤ finalTemperature) Y (overheating > 0) THEN
                // Recalentamiento
                currentTemperature ← lastTemperatureAcceptanceRate
                currentCoolingFactor ← (overheating = 2) ? 0.99 : 0.999
                overheating ← overheating – 1
            ENF IF
        END IF
    END WHILE

RETURN
```

Funcione auxiliares

- **`IsTabu(S)`**: Evitar explorar soluciones previamente evaluadas durante la búsqueda, garantizando diversificación y eficiencia en la exploración del espacio de soluciones. Tiene una memoria estática para todo el programa donde almacena de forma codificada las soluciones ya exploradas. Cuando esta memoria se llena se irán borrando u olvidando los elementos más antiguos.
  - Codificar la solución (solo C' y D').
  - Buscar la codificación en la memoria Tabú.
    - Si existe, retorna false.
    - Si no existe, inserta la codificación en la memoria Tabú y retorna true.
- **`GenerateNeighbourSolution(S)`**: Dada una solución $S = \{C, D, R\}$, se selecciona mediante muestreo uniforme un elemento de $\{C, D\}$. A continuación, se elige de forma independiente y también con distribución uniforme un operador de vecindario de los siguientes:
  - El operador ***Swap***: Dado un conjunto representado como una secuencia $A = (a_1, a_2, \dots, a_n)$, el algoritmo selecciona dos índices distintos i y j mediante un muestreo aleatorio uniforme $(1 \leq i, j \leq n, i \neq j)$, y ejecuta una transposición de elementos $a_i \leftrightarrow a_j$.

    Ejemplo:

    - Secuencia original: $A = (a, b, c, d, e, f, g, h)$
    - Selección aleatoria: $i = 3\ (a_i = c), j = 6\ (a_j = f)$
    - Secuencia tras el Swap: $A = (a, b, f, d, e, c, g, h)$
  - El operador ***Relocate***: Dada una secuencia $A = (a_1, a_2, \dots, a_n)$, el algoritmo selecciona un índice $i\ (1 \leq i \leq n)$ mediante muestreo aleatorio uniforme y extrae el elemento a_i de su posición original. A continuación, se elige un nuevo índice $j\ (1 \leq j \leq n, j \neq i)$ también de forma aleatoria, y se inserta el elemento a_i en la posición j, desplazando al resto de los elementos.

    Ejemplo:

    - Secuencia original: $A = (a, b, c, d, e, f, g, h)$

- Selección aleatoria: $i = 3$ $(a_i = c)$
- Secuencia sin a_i: $A = (a, b, d, e, f, g, h)$
- Selección aleatoria: $j = 6$ $\left(a_j = g\right)$
- Secuencia tras el Relocate: $A = (a, b, d, e, f, c, g, h)$

Después de aplicar uno de estos operadores, se genera una solución dependiendo del conjunto alterado: $S' = \text{SoluciónInicial}(\text{C}', \text{D})$ o $S' = \text{SoluciónInicial}(\text{C}, \text{D}')$.

- **`IsFeasible(S)`**: Dada una solución S, se evalúan todas las restricciones definidas en el capítulo 3, sección 3.2. Si todas se cumplen S es factible retornando *verdadero* o *falso* en caso contrario.
- **`Cost(S)`**: Como se define en el capítulo 3, sección 3.1. Dada una solución S, se calcula como la suma de la distancia total recorrida por todas las rutas más el costo fijo por vehículo utilizado, a lo que se añade una penalización p por cada restricción incumplida. La penalización actúa como un puente entre la optimización del coste operativo y el respeto estricto de las restricciones, facilitando una convergencia progresiva hacia mejores soluciones.
- **`Exp(x)`**: Retorna la probabilidad de aceptar una solución peor, en rango de probabilidad $(0,1]$.
- **`Random(x, y)`**: Retorna un numero aleatorio a $(x \leq a \leq y)$.
- **`NumberOfVehiclesUsed(S)`**: Dada una solución retorna la cantidad de vehículos que usa.
- **`ShouldActivateLocalSearch(Tc, Ti, Tf)`**: Determina si activar la búsqueda local e indica cuántos vecinos explorar. Esta función retorna un número entero $n \geq 0$, cuyo significado es el siguiente:
  - Si $n = 0$, la búsqueda local no se activa en la iteración correspondiente.
  - Si $n > 0$, la búsqueda local se activa y se exploran hasta n soluciones vecinas.

  La función opera bajo el principio de que a medida que el sistema se enfría (es decir, conforme Tc decrece desde Ti hacia Tf), la probabilidad de activar la búsqueda local aumenta, dado que la exploración intensiva es más beneficiosa en

fases tardías del proceso de búsqueda. En consecuencia, cuanto más frío se encuentra el sistema, mayor es también la cantidad de vecinos a explorar, permitiendo un refinamiento más exhaustivo de la solución actual en las etapas finales.

De manera intuitiva, la función se comporta de la siguiente forma: en las primeras etapas, cuando Tc es cercano a Ti, se tiende a priorizar la diversificación y por tanto la búsqueda local rara vez se activa. Conforme Tc desciende y el sistema se aproxima a Tf, la función devuelve valores $n > 0$ crecientes, reflejando la intensificación progresiva de la exploración local alrededor de la solución incumbente.

Este esquema dinámico permite ajustar de forma adaptativa el compromiso entre diversificación e intensificación, en coherencia con la evolución térmica del algoritmo.

Algoritmo 3. Activación de la Búsqueda Local

```
Entradas:
    • currentTemperature: Temperatura actual
    • initialTemperature: Temperatura inicial
    • finalTemperature: Temperatura final

Salidas:
    • neigh: Numero de vecinos a explorar

Descripción de las variables:
    • MIN_PROB = 0.05: Probabilidad mínima de activación (5%)
    • MAX_PROB = 0.95: Probabilidad máxima de activación (95%)
    • MIN_NEIGH = 100: Tamaño mínimo de vecindario
    • MAX_NEIGH = 800: Tamaño máximo de vecindario
    • GAMMA = 2.5: Exponente para no-linealidad
    • nClients: Número de clientes de la instancia

Procedimiento:
    // Calcular "enfriamiento" normalizado (0 cuando caliente, 1 cuando frío)
    IF currentTemperature >= initialTemperature THEN
       cooling ← 0.0
    END IF
    IF currentTemperature <= finalTemperature THEN
        cooling ← 1.0
    ELSE
        cooling ← (ln(initialTemperature) - ln(currentTemperature)) /
                    (ln(initialTemperature) - ln(finalTemperature))
    END IF

    // Calcular probabilidad de activar búsqueda local
    pAct ← MIN_PROB + (MAX_PROB - MIN_PROB) × (cooling ^ GAMMA)

    // Si no se activa, devolver 0
    IF Random(0,1) > pAct THEN
       RETURN 0
    END IF
```

Algoritmo 3. Activación de la Búsqueda Local (continuación)

```
// Calcular factor de tamaño según número de clientes
sizeFactor ← Normalizar(nClients, mínimo=20, máximo=220)

// Calcular tamaño del vecindario a explorar
scale ← cooling × (0.5 + 0.5 × size_factor)
neigh ← MIN_NEIGH + (MAX_NEIGH - MIN_NEIGH) × scale

// Asegurar que neigh esté dentro de los límites
neigh ← limitar_a_rango(neigh, MIN_NEIGH, MAX_NEIGH)

RETURN neigh
```

- **`GetOperatorWeights(tempRatio, costRatio, infeasibilityFactor)`**: Función para la asignación dinámica de pesos a operadores de vecindad. Para guiar la selección de operadores de vecindad en función del estado actual de la búsqueda. Esta función devuelve un vector $W = [w_1, w_2, \dots, w_k]$, donde cada $w_i \geq 0$ representa el peso asignado al operador de vecindad i del conjunto disponible. Los pesos son proporcionales a la conveniencia de aplicar cada operador bajo las condiciones actuales de la solución y del estado del sistema.
    - $tempRatio \in [0,1]$: Indica el nivel de "enfriamiento" del sistema. Valores cercanos a 1 representan estados iniciales (exploración), mientras que valores cercanos a 0 representan fases finales (explotación).
    - $costRatio \geq 0$: Mide la desviación relativa del costo de la solución actual con respecto a la mejor solución conocida.
    - $infeasibilityFactor \geq 1$: Cuantifica el grado de sobreutilización de vehículos en la solución actual, con respecto al máximo permitido por la instancia. Valores mayores a 1 reflejan mayor inviabilidad.

  La lógica subyacente a la función es la siguiente: en fases tempranas ($tempRatio \approx 1$), se favorecen operadores exploratorios y disruptivos, para recorrer más ampliamente el espacio de soluciones. En fases tardías ($tempRatio \approx 0$), los pesos tienden a concentrarse en operadores refinadores y de intensificación local. Adicionalmente, si la solución actual presenta un costo elevado ($costRatio \gg 0$) o es altamente inviable ($infeasibilityFactor \gg 0$), los pesos se ajustan para priorizar operadores correctivos y de factibilidad.

  Así, `GetOperatorWeights()` permite implementar un esquema adaptativo de selección de operadores en función de la evolución dinámica de la búsqueda.

Algoritmo 4. Asignación Dinámica de Pesos a Operadores de Vecindad

```
Entradas:
    •   tempRatio: Indicador de cuan frio se encuentra el sistema
    •   costRatio: Indicador del costo actual contra el mejor encontrado
    •   infeasibilityFactor: Indicador de infactibilidad de la solución

Salidas:
    •   weights: Vector de pesos probabilísticos para los operadores

Descripción de las variables:

Procedimiento:
    // Pesos base basados en efectividad reportada en literatura VRP
    weights ← [
        8.0,   // SWAP_CLIENTS_SAME_ROUTE
        12.0,  // SWAP_CLIENTS_BETWEEN_ROUTES
        4.0,   // RELOCATE_CLIENT_SAME_ROUTE
        18.0,  // RELOCATE_CLIENT_BETWEEN_ROUTES
        10.0,  // ROUTE_EXCHANGE
        15.0,  // TWO_OPT_SAME_ROUTE
        6.0,   // TWO_OPT_BETWEEN_ROUTES
        22.0   // CROSS_EXCHANGE
    ]

    // Alta infactibilidad (>50% exceso de vehículos)
    IF infeasibilityFactor > 1.5 THEN
       weights[RELOCATE_CLIENT_BETWEEN_ROUTES] ← weights[RELOCATE_CLIENT_BETWEEN_ROUTES] × 1.8
       weights[ROUTE_EXCHANGE] ← weights[ROUTE_EXCHANGE] × 1.6
       weights[SWAP_CLIENTS_SAME_ROUTE] ← weights[SWAP_CLIENTS_SAME_ROUTE] × 0.3
       weights[RELOCATE_CLIENT_SAME_ROUTE] ← weights[RELOCATE_CLIENT_SAME_ROUTE] × 0.4
    END IF

    // Solución cercana al óptimo (alta calidad)
    IF costRatio < 0.1 THEN
        weights[TWO_OPT_SAME_ROUTE] ← weights[TWO_OPT_SAME_ROUTE] × 1.7
        weights[SWAP_CLIENTS_BETWEEN_ROUTES] ← weights[SWAP_CLIENTS_BETWEEN_ROUTES] × 1.4
        weights[CROSS_EXCHANGE] ← weights[CROSS_EXCHANGE] × 1.2
    END IF

    // Fase de explotación (temperatura baja)
    IF tempRatio < 0.4 THEN
       weights[CROSS_EXCHANGE] ← weights[CROSS_EXCHANGE] × 1.5
       weights[TWO_OPT_SAME_ROUTE] ← weights[TWO_OPT_SAME_ROUTE] × 1.3
       weights[SWAP_CLIENTS_BETWEEN_ROUTES] ← weights[SWAP_CLIENTS_BETWEEN_ROUTES] × 1.25
    ENF IF

    // Fase de exploración (temperatura alta)
    IF tempRatio > 0.7 THEN
       weights[TWO_OPT_BETWEEN_ROUTES] ← weights[TWO_OPT_BETWEEN_ROUTES] × 1.6
       weights[ROUTE_EXCHANGE] ← weights[ROUTE_EXCHANGE] × 1.4
    END IF

    // Suavizado de transiciones con los pesos previos
    prevWeights ← weights
    FOR i = 0 TO 7 DO
        weights[i] ← 0.7 × weights[i] + 0.3 × prevWeights[i]
    END FOR
    prevWeights ← weights

    // Normalizar pesos para que sumen 1 (distribución de probabilidad)
    total ← Summation(weights)
    FOR w IN weights DO
        w ← w / total
    END FOR

RETURN weights
```

- **`LocalSearch(sAct, operatorWeights, localSearchCount)`**: Función de búsqueda local con selección probabilística de operadores y control tabú. La mejora de una solución se implementa mediante una función de búsqueda local, diseñada para explorar un conjunto limitado de soluciones vecinas seleccionadas de forma estocástica y ponderada. Para cada una de las soluciones a calcular (`localSearchCount`), se selecciona un operador de vecindad al azar de acuerdo con la distribución de probabilidad definida por el vector de pesos (`operatorWeights`). Este operador se aplica a la solución actual (`sAct`) para generar una solución vecina. Antes de evaluarla, se verifica que no se encuentre en la lista tabú, con el propósito de evitar ciclos y fomentar la diversificación. Si no es tabú, se evalúa su calidad respecto a la función objetivo. En caso de que sea mejor que la solución actual, se actualiza (`sAct`) para reflejar la mejora encontrada.

  Esta función permite introducir intensificación controlada en el algoritmo metaheurístico, priorizando las áreas más prometedoras del espacio de soluciones, y a la vez manteniendo un componente aleatorio ponderado que previene el estancamiento prematuro en óptimos locales. El esquema tabú asegura además que la búsqueda local no recurra a soluciones ya exploradas en las iteraciones recientes, manteniendo así la diversidad en el proceso de búsqueda.

  - **`SelectOperator(w)`**: Función de selección aleatoria ponderada de operadores. Recibe como argumento un vector de pesos normalizados (w) $W = [w_1, w_2, \dots, w_k]$, que define una distribución de probabilidad discreta sobre los k operadores de vecindad disponibles. Su objetivo es seleccionar un índice de $operador \in \{1, \dots, m\}$, de manera que la probabilidad de elegir el operador i sea proporcional a w_i.
  - **`GenerateNeighbourhoodSolution(s, o)`**: Función de aplicación de operador de vecindad. Dada una solución (s) la perturba según el operador (o) y genera una nueva solución vecina.

Algoritmo 5. Búsqueda Local

```
Entradas:
    • currentSolution: solución actual
    • operatorWeights: vector de pesos normalizados (distribución de probabilidad) para    selección
      de operadores
    • evalLimit: límite máximo de evaluaciones de vecindad

Salidas:
    • currentSolution: se actualiza con la mejor solución factible encontrada

Descripción de las variables:
    • tabuList: lista tabú (global)
    • bestNeighborSolution: mejor solución vecina guardada
    • neighborSolution: solución vecina generada en cada iteración

Procedimiento:
    bestNeighborSolution ← currentSolution

    // Explorar hasta evalLimit vecinos
    FOR i ← 1 TO evalLimit DO
        // Selección aleatoria ponderada de operador
        selectedOp ← SelectOperator(operatorWeights)

        // Aplicar el operador, generar solución vecina
        neighborSolution ← GenerateNeighbourhoodSolution(currentSolution, selectedOp)

        // Verificar lista tabú
        IF EsTabu(neighborSolution) THEN
            CONTINUE                              // Saltar esta vecindad
        END IF

        // Evaluar factibilidad y calidad
        IF EsFactible(neighborSolution) THEN
            currentSolution ← neighborSolution
            RETURN                               // Salir en cuanto se encuentra una vecina factible
        END IF
        IF Costo(neighborSolution) ≤ Costo(bestNeighborSolution) THEN
            bestNeighborSolution ← neighborSolution
        END IF
    END FOR

    // Actualizar solución con la mejor vecina encontrada
    IF Costo(bestNeighborSolution) ≤ Costo(currentSolution) THEN
        currentSolution ← bestNeighborSolution
    END IF

    RETURN
```

4.2.4 Operadores de Vecindad en la Búsqueda Local

La fase de búsqueda local explora el entorno de la solución actual aplicando iterativamente operadores de vecindad.

- **SWAP_CLIENTS_SAME_ROUTE**: Intercambia las posiciones de dos clientes dentro de la misma ruta.

$\boldsymbol{R_1}$={D_1, $\boldsymbol{C_1}$, C_2, C_3, C_4, $\boldsymbol{C_5}$, D_1} R_2={D_1, C_6, C_7, C_8, C_9, C_{10}, D_1} R_3={D_2, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, D_2} R_4={D_2, C_{16}, C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, D_2}

Intercambiar en R_1, 1 y 5.

$\boldsymbol{R_1}$={D_1, $\boldsymbol{C_5}$, C_2, C_3, C_4, $\boldsymbol{C_1}$, D_1} R_2={D_1, C_6, C_7, C_8, C_9, C_{10}, D_1} R_3={D_2, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, D_2} R_4={D_2, C_{16}, C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, D_2}

- **SWAP_CLIENTS_BETWEEN_ROUTES**: Intercambia clientes ubicados en rutas distintas.

R_1={D_1, C_1, C_2, C_3, C_4, C_5, D_1} $\boldsymbol{R_2}$={D_1, C_6, C_7, $\boldsymbol{C_8}$, C_9, C_{10}, D_1} $\boldsymbol{R_3}$={D_2, C_{11}, C_{12}, C_{13}, C_{14}, $\boldsymbol{C_{15}}$, D_2} R_4={D_2, C_{16}, C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, D_2}

Intercambiar de R_2, 3 y de R_3 5.

R_1={D_1, C_1, C_2, C_3, C_4, C_5, D_1} $\boldsymbol{R_2}$={D_1, C_6, C_7, $\boldsymbol{C_{15}}$, C_9, C_{10}, D_1} $\boldsymbol{R_3}$={D_2, C_{11}, C_{12}, C_{13}, C_{14}, $\boldsymbol{C_8}$, D_2} R_4={D_2, C_{16}, C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, D_2}

- **RELOCATE_CLIENT_SAME_ROUTE**: Reubica un cliente a otra posición dentro de su propia ruta.

R_1={D_1, C_1, C_2, C_3, C_4, C_5, D_1} R_2={D_1, C_6, C_7, C_8, C_9, C_{10}, D_1} R_3={D_2, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, D_2} $\boldsymbol{R_4}$={D_2, $\boldsymbol{C_{16}}$, C_{17}, C_{18}, $\boldsymbol{C_{19}}$, C_{20}, C_{21}, D_2}

Reinsertar de R_4, 1 en R_4, 4.

R_1={D_1, C_1, C_2, C_3, C_4, C_5, D_1} R_2={D_1, C_6, C_7, C_8, C_9, C_{10}, D_1} R_3={D_2, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, D_2} $\boldsymbol{R_4}$={D_2, $\boldsymbol{C_{17}}$, C_{18}, C_{19}, $\boldsymbol{C_{16}}$, C_{20}, C_{21}, D_2}

- **RELOCATE_CLIENT_BETWEEN_ROUTES**: Extrae un cliente de su ruta actual y lo inserta en una posición específica de otra ruta.

$\boldsymbol{R_1}$={D_1, C_1, $\boldsymbol{C_2}$, C_3, C_4, C_5, D_1} R_2={D_1, C_6, C_7, C_8, C_9, C_{10}, D_1} $\boldsymbol{R_3}$={D_2, C_{11}, C_{12}, $\boldsymbol{C_{13}}$, C_{14}, C_{15}, D_2} R_4={D_2, C_{16}, C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, D_2}

Reinsertar de R_1, 2 en R_3, 3.

$\boldsymbol{R_1}$={D_1, C_1, C_3, C_4, C_5, D_1} R_2={D_1, C_6, C_7, C_8, C_9, C_{10}, D_1} $\boldsymbol{R_3}$={D_2, C_{11}, C_{12}, $\boldsymbol{C_2}$, C_{13}, C_{14}, C_{15}, D_2} R_4={D_2, C_{16}, C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, D_2}

- **ROUTE_EXCHANGE**: Intercambia completamente dos rutas distintas, es decir, asigna las secuencias de una a la otra, de depósitos distintos.

R_1={D_1, C_1, C_2, C_3, C_4, C_5, D_1} $\boldsymbol{R_2}$={D_1, $\boldsymbol{C_6}$, $\boldsymbol{C_7}$, $\boldsymbol{C_8}$, $\boldsymbol{C_9}$, $\boldsymbol{C_{10}}$, D_1} R_3={D_2, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, D_2} $\boldsymbol{R_4}$={D_2, $\boldsymbol{C_{16}}$, $\boldsymbol{C_{17}}$, $\boldsymbol{C_{18}}$, $\boldsymbol{C_{19}}$, $\boldsymbol{C_{20}}$, $\boldsymbol{C_{21}}$, D_2}

Intercambiar R_2 y R_4.

R_1={D_1, C_1, C_2, C_3, C_4, C_5, D_1} $\boldsymbol{R_2}$={D_1, $\boldsymbol{C_{16}}$, $\boldsymbol{C_{17}}$, $\boldsymbol{C_{18}}$, $\boldsymbol{C_{19}}$, $\boldsymbol{C_{20}}$, $\boldsymbol{C_{21}}$, D_1} R_3={D_2, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, D_2} $\boldsymbol{R_4}$={D_2, $\boldsymbol{C_6}$, $\boldsymbol{C_7}$, $\boldsymbol{C_8}$, $\boldsymbol{C_9}$, $\boldsymbol{C_{10}}$, D_2}

- **TWO_OPT_SAME_ROUTE**: Revierte la dirección de un segmento dentro de una misma ruta, eliminando cruces para acortar distancia (clásico 2-opt).

R_1={D_1, C_1, C_2, C_3, C_4, C_5, D_1} R_2={D_1, C_6, C_7, C_8, C_9, C_{10}, D_1} R_3={D_2, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, D_2} $\boldsymbol{R_4}$={D_2, C_{16}, C_{17}, $\boldsymbol{C_{18}}$, $\boldsymbol{C_{19}}$, $\boldsymbol{C_{20}}$, $\boldsymbol{C_{21}}$, D_2}

Revertir en R_4, de 3 a 6.

R_1={D_1, C_1, C_2, C_3, C_4, C_5, D_1} R_2={D_1, C_6, C_7, C_8, C_9, C_{10}, D_1} R_3={D_2, C_{11}, C_{12}, C_{15}, C_{14}, C_{13}, D_2} $\boldsymbol{R_4}$={D_2, C_{16}, C_{17}, $\boldsymbol{C_{21}}$, $\boldsymbol{C_{20}}$, $\boldsymbol{C_{19}}$, $\boldsymbol{C_{18}}$, D_2}

- **TWO_OPT_BETWEEN_ROUTES**: Intercambia dos segmentos terminales de distintas rutas, formando nuevas conexiones.

$\boldsymbol{R_1}$={$D_1, C_1, C_2, C_3, \boldsymbol{C_4}, \boldsymbol{C_5}, D_1$} R_2={$D_1, C_6, C_7, C_8, C_9, C_{10}, D_1$} $\boldsymbol{R_3}$={$D_2, C_{11}, C_{12}, \boldsymbol{C_{13}}, \boldsymbol{C_{14}}, \boldsymbol{C_{15}}, D_2$} R_4={$D_2, C_{16}, C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, D_2$}

Intercambiar de R_1, desde 4 y en R_3 desde 3.

$\boldsymbol{R_1}$={$D_1, C_1, C_2, C_3, \boldsymbol{C_{13}}, \boldsymbol{C_{14}}, \boldsymbol{C_{15}}, D_1$} R_2={$D_1, C_6, C_7, C_8, C_9, C_{10}, D_1$} $\boldsymbol{R_3}$={$D_2, C_{11}, C_{12}, \boldsymbol{C_4}, \boldsymbol{C_5}, D_2$} R_4={$D_2, C_{16}, C_{17}, C_{18}, C_{19}, C_{20}, C_{21}, D_2$}

- **CROSS_EXCHANGE**: Intercambia dos secuencias contiguas de clientes entre rutas diferentes, permitiendo reorganizaciones más extensas y disruptivas.

R_1={$D_1, C_1, C_2, C_3, C_4, C_5, D_1$} $\boldsymbol{R_2}$={$D_1, C_6, \boldsymbol{C_7}, \boldsymbol{C_8}, C_9, C_{10}, D_1$} R_3={$D_2, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, D_2$} $\boldsymbol{R_4}$={$D_2, C_{16}, C_{17}, \boldsymbol{C_{18}}, \boldsymbol{C_{19}}, \boldsymbol{C_{20}}, C_{21}, D_2$}

Intercambiar de R_2, desde 2 a 3 y de R_4, desde 3 a 5.

R_1={$D_1, C_1, C_2, C_3, C_4, C_5, D_1$} $\boldsymbol{R_2}$={$D_1, C_6, \boldsymbol{C_{18}}, \boldsymbol{C_{19}}, \boldsymbol{C_{20}}, C_9, C_{10}, D_1$} R_3={$D_2, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, D_2$} $\boldsymbol{R_4}$={$D_2, C_{16}, C_{17}, \boldsymbol{C_7}, \boldsymbol{C_8}, C_{21}, D_2$}

Con el fin de incrementar la capacidad de busqueda del algoritmo SA, se integró un mecanismo de recalentamiento controlado. Este procedimiento puede activarse hasta en dos ocasiones cuando el sistema alcanza un estado “frío” sin haber logrado aún una solución factible. En el primer recalentamiento, la temperatura se restablece al último valor en el que se observó una alta tasa de aceptación de soluciones (95%), y la tasa de enfriamiento se fija en 0.99. De ser necesario un segundo recalentamiento, el mismo criterio de restauración de temperatura se aplica, pero la tasa de enfriamiento se ajusta a 0.999.

4.3 Refinamiento de la Solución

Una vez concluida la fase inicial de búsqueda de factibilidad, el enfoque propuesto avanza a una segunda fase de refinamiento, orientada a mejorar la calidad de la mejor solución factible hallada. Esta etapa combina de manera complementaria técnicas metaheurísticas clásicas como Recodo Simulado, Búsqueda Local, Búsqueda Tabú y un esquema final de Hill Climbing, priorizando la optimización de la función objetivo mientras se garantiza que las soluciones se mantengan en el espacio factible definido por las restricciones del problema.

4.3.1 Fundamentos de Hill Climbing

El algoritmo Hill Climbing es un método clásico de optimización local, basado en la mejora iterativa de una solución inicial al elegir, en cada paso, la mejor vecina disponible. Este proceso continúa hasta alcanzar un óptimo local, donde ninguna vecina mejora la solución actual (Russell & Norvig, 2021). El término fue popularizado en la década de 1950 en el contexto de la inteligencia artificial, con aportaciones tempranas de (Samuel, 1959) y formalizado en investigaciones de (Minsky, 1961).

En el VRP, Hill Climbing se emplea principalmente como fase de refinamiento final, intensificando la búsqueda en la vecindad de una solución factible para mejorar su calidad sin incurrir en grandes costos computacionales (Laporte, Fifty years of vehicle routing, 2009).

El refinamiento se estructura en dos componentes principales:

4.3.2 Recocido Simulado Adaptado para Soluciones Factibles

El algoritmo de refinamiento implementado en esta etapa está basado en la metaheurística SA, la cual es utilizada para intensificar la búsqueda sobre el espacio factible de soluciones, mejorando progresivamente la calidad de la solución obtenida durante la fase de factibilidad. La técnica combina exploración probabilística con aceptación controlada de soluciones peores y, opcionalmente, fases de búsqueda local más intensivas en función de la dinámica de aceptación observada.

- En cada iteración del SA, se genera un conjunto amplio de vecinos factibles, aplicando sistemáticamente operadores de Swap y Relocate sobre la secuencia actual de clientes y rutas.
- Cada secuencia generada es verificada contra una lista tabú para evitar exploraciones redundantes o ciclos no productivos.
- Si la secuencia no se encuentra en la lista tabú, se construye la solución correspondiente y se comprueba su factibilidad.
- Entre todas las soluciones factibles generadas en la iteración, se identifica y guarda la mejor encontrada en términos de la función objetivo.

- De todas las soluciones factibles descubiertas se selecciona aleatoriamente una solución factible como punto de partida para la siguiente iteración, fomentando así la diversidad.

Adicionalmente, para reforzar la búsqueda, si la tasa de aceptación de soluciones dentro de un ciclo cae por debajo de un umbral (indicativo de estancamiento), se activa una fase opcional de búsqueda local intensiva. Siguiendo el mismo criterio empleado en la fase de factibilidad, aunque ahora restringida únicamente a soluciones factibles.

Este esquema de refinamiento combina las fortalezas de las estrategias globales de SA (capacidad de escape de óptimos locales) con la precisión de la Búsqueda Local en entornos prometedores.

Algoritmo 6. Refinamiento SA

```
Entradas:
    • initialSolution: solución inicial factible
    • initialTemperature: temperatura inicial
    • finalTemperature: temperatura final
    • iterationsNumber: longitud de la cadena de Markov
    • coolingFactor: factor de enfriamiento

Salidas:
    • initialSolution: actualizada con la mejor solución factible hallada

Descripción de las variables:
    • currentSolution: solución actual
    • currentTemperature: temperatura actual
    • tabuList: lista tabú (global)
    • generated, accepted: contadores internos
    • acceptanceRate: tasa de aceptación
    • localSearchCount: número de vecinos a explorar en la búsqueda local
    • operatorWeights: vector de pesos para selección de operadores

Procedimiento:
    // Inicialización
    currentSolution ← initialSolution
    currentTemperature ← initialTemperature

    // Bucle de recocido (enfriamiento)
    WHILE currentTemperature > finalTemperature DO
        generated ← 0
        accepted ← 0

        // Generar y evaluar vecinos
        FOR i ← 1 TO iterationsNumber DO
            //Explorar vecindario, retornando el mejor vecino factible encontrado
            bestNeighborhoodsSolution ← Exploration(currentSolution)
            generated ← generated + 1

            IF Cost(bestNeighborhoodsSolution) ≤ Cost(currentSolution) THEN
                // Aceptar mejora
                currentSolution ← bestNeighborhoodsSolution
                accepted ← accepted + 1

                // Actualizar solución inicial con la mejor
                IF Cost(currentSolution) ≤ Cost(initialSolution) THEN
                    initialSolution ← currentSolution
                END IF
            ELSE
                // Aceptación probabilística de una peor solución
                Δ ← Cost(bestNeighborhoodsSolution) – Cost(currentSolution)
                acceptanceProbability ← Exp(–Δ / currentTemperature)
                probability ← Aleatorio(0,1):

                IF probability < acceptanceProbability THEN
                    // Se acepta una peor solución
                    currentSolution ← bestNeighborhoodsSolution
                    accepted ← accepted + 1
                END IF
            END IF
        END FOR

        // Se actualiza la taza de aceptación
        acceptanceRate ← 100 × accepted / generated
```

Algoritmo 6. Refinamiento SA (continuación)

```
        // Si la aceptación es baja, evaluar lanzar búsqueda local
        IF acceptanceRate < 50.0 THEN
            localSearchCount ← ShouldActivateLocalSearch(currentTemperature,
                                                         initialTemperature,
                                                         finalTemperature)
            // Si se activa la búsqueda local
            IF localSearchCount > 0 THEN
                tempRatio ← currentTemperature / initialTemperature
                costRatio ← (Cost(currentSolution)–Cost(initialSolution))/Cost(initialSolution)

                // Actualizar vector de pesos de los operadores según estado de la búsqueda
                operatorWeights ← GetOptimizationOperatorWeights(tempRatio,costRatio,1.0)

                // Búsqueda local
                LocalSearch(currentSolution,operatorWeights,localSearchCount)

                // Actualizar la mejor solución encontrada
                IF Cost(currentSolution) ≤ Cost(initialSolution) THEN
                    initialSolution ← currentSolution
                END IF
            END IF
        END IF

        // Actualizar temperatura (enfriamiento)
        currentTemperature ← currentTemperature × coolingFactor
    END WHILE

    RETURN
```

Funcione auxiliares

- **`Exploration(s)`**: Función de dada una solución (s) explora un conjunto de k soluciones factibles vecinas. Se exploran k/2 soluciones alterando las secuencias de clientes y de rutas (C, D seleccionadas aleatoriamente) con el operador Swap y k/2 soluciones con el operador Relocate. A partir de estos nuevos ordenamientos de (C', D'), se generan nuevas soluciones como se describe en el Algoritmo 1. De estas k soluciones se retorna la mejor.
- **`ShouldActivateLocalSearch(ct, it, ft)`**: Determina si activar la búsqueda local e indica cuántos vecinos explorar. Se describe en el Algoritmo 3, con la diferencia que en esta etapa varían las probabilidades.
- **`GetOptimizationOperatorWeights(t, c, j)`**: Función para la asignación dinámica de pesos a operadores de vecindad. Se describe en el Algoritmo 4, con la diferencia que en esta etapa varían las probabilidades.
- **`LocalSearch(s, o, c)`**: Función de búsqueda local con selección probabilística de operadores y control tabú. Se describe en el Algoritmo 5, con la diferencia que en esta etapa no se termina cuando encuentra factibilidad y que todas las soluciones que tiene en cuenta son factibles.

4.3.3 Hill Climbing Combinado con Operadores de Búsqueda Local

Finalmente, sobre la mejor solución obtenida tras el SA, se aplica un algoritmo de tipo Hill Climbing, que combina de forma iterativa todos los operadores de búsqueda local definidos en la metodología. Este proceso intensifica la búsqueda en la vecindad inmediata de la solución, explorando exhaustivamente las posibilidades de mejora a través de movimientos finos y dirigidos.

Algoritmo 7. Refinamiento Hill Climbing

```
Entrada:
    •   initialSolution: Solución inicial factible

Salida:
    •   initialSolution: Solución mejorada

Descripción de variables:
    •   improvements: Booleano que controla el ciclo de mejora

Procedimiento:
    improvements ← TRUE

    WHILE improvements DO
        improvements ← FALSE

        FOR operador IN { exploreAllSwapClientsSameRoute,
                          exploreAllSwapClientsBetweenRoutes,
                          exploreAllRelocateClientSameRoute,
                          exploreAllRelocateClientBetweenRoute,
                          exploreAllRouteExchange,
                          exploreAllTwoOptSameRoute,
                          exploreAllTwoOptBetweenRoute,
                          exploreAllCrossExchange }
        DO
            IF operador(initialSolution) THEN
                improvements ← TRUE
                CONTINUE WHILE
            END IF
        END FOR
    END WHILE

RETURN
```

Cada una de las funciones **`exploreAll[Operador](S)`** implementa un operador de vecindad específico que define un tipo de movimiento dentro del espacio de soluciones factibles. El objetivo general de estas funciones es explorar completamente la vecindad generada por dicho operador, buscando movimientos que conduzcan a una mejor solución, respetando en todo momento la factibilidad.

Para cada operador, la lógica seguida es la misma:

- Exploración sistemática: Se recorre exhaustivamente el conjunto de soluciones vecinas que se pueden generar mediante ese operador.
- Actualización de la solución: Si se encuentra una solución mejor, se actualiza la solución inicial con esta nueva solución mejorada.
- Criterio de parada: El proceso puede detenerse tan pronto como se detecte una mejora o continuar hasta agotar la vecindad completa.
- Control de factibilidad: Durante todo el proceso, se verifica la validez de las soluciones vecinas generadas. Solo aquellas que cumplan con todas las restricciones del problema son consideradas candidatas válidas para la mejora.

Con la definición de la metodología, se ha estructurado un enfoque sistemático para abordar el problema de ruteo de vehículos, dividiendo el proceso en tres fases fundamentales: generación de una solución inicial, búsqueda de factibilidad y etapa de refinamiento. Cada fase ha sido diseñada para contribuir de forma complementaria al equilibrio entre intensificación y diversificación en el espacio de soluciones.

Este equilibrio se alcanza mediante la hibridación de métodos y heurísticas que, al combinarse, permiten explotar las fortalezas individuales de cada técnica. En particular, el sondeo exhaustivo de los vecindarios definidos por distintos operadores facilita una mejora iterativa guiada por criterios de calidad y factibilidad. Esta sinergia metodológica sienta las bases para una búsqueda robusta y adaptable, que será evaluada empíricamente en los capítulos posteriores.

Capítulo 5. Experimentación y Resultados

Este capítulo presenta los resultados obtenidos al aplicar el algoritmo propuesto para resolver el problema de MDVRPTWAR. Los experimentos se realizaron sobre el conjunto de instancias descritas en la sección 3.3 y bajo la metodología descrita en el capítulo anterior. Se incluye además una comparación cuantitativa con los resultados reportados en (Li y otros, 2019).

5.1 Configuración Experimental

5.1.1 Sintonización

Para la calibración inicial del algoritmo se seleccionaron instancias representativas de diferentes categorías del problema (A04, A07, A08, A12, A14, B04, B05, B08, B09, B10, C05, C06, C10, C12, C20, D04, D06, D10, D15 y D16). Estas instancias fueron escogidas por su diversidad en complejidad, número de clientes, depósitos y vehículos, lo cual permite evaluar exhaustivamente el desempeño del algoritmo en escenarios variados.

Diseño experimental: Para cada prueba se realizaron 30 ejecuciones independientes por instancia, de modo que los resultados obtenidos sean estadísticamente significativos. La repetición de corridas permite medir la variabilidad inherente al carácter estocástico del recocido simulado y obtener estimaciones de rendimiento más fiables.

Procedimiento de sintonización: Se empleó un enfoque one-factor-at-a-time, variando un parámetro a la vez mientras se mantenían constantes los demás. Los pasos principales fueron:

1. Para un parámetro dado k_i, se definió un conjunto de m valores candidatos.
2. Se ejecutó el algoritmo para cada valor de k_i sobre las instancias seleccionadas.

3. Se analizó el desempeño promedio de cada valor de k_i y se seleccionó el que ofreció mejores resultados.
4. Se fijó el valor óptimo de k_i y se repitió el proceso con el siguiente parámetro k_j.

Este método secuencial permite evaluar aisladamente el efecto de cada parámetro, tal como se recomienda en (Arin y otros, 2010). Se definió la media de aceptación de nuevas soluciones como métrica auxiliar para la calibración de parámetros. Algunas de las temperaturas empleadas en el algoritmo se definieron en función del costo de la solución inicial Z. En particular, la temperatura inicial se estableció como un múltiplo de dicha solución, lo que permitió ajustar la escala del enfriamiento de acuerdo con la magnitud del problema.

Parámetros calibrados: Los parámetros ajustados fueron la temperatura inicial, y la temperatura final.

Búsqueda de factibilidad

Temperatura inicial	Z/4	Z/2	100000	Z	**2Z**	4Z
Media Aceptación de nuevas soluciones	62,97	67,39	70,99	70,07	**72,00**	76,00

Temperatura final	0,4	0,2	0,1	0,01	**0,001**
Media Aceptación de nuevas soluciones	2,63	2,49	2,50	2,15	**1,79**

Refinamiento de la solución

Temperatura inicial	Z/2	**100000**	Z	2Z	4Z
Media Aceptación de nuevas soluciones	97,46	**99,88**	98,68	99,33	99,65

La temperatura final utilizada en la fase de refinamiento se estableció con el mismo valor (0,001) que la empleada en el proceso de búsqueda de factibilidad.

5.1.2 Parámetros

Búsqueda de factibilidad

Parámetro	Valor
Temperatura inicial (initialTemperature)	2 veces el costo de la solución inicial
Temperatura final (finalTemperature)	0.001
Longitud de la cadena de Markov (iterationsNumber)	1000
Tasa de enfriamiento (coolingFactor)	0.9
Tasa de aceptación (acceptanceRate)	95%

Refinamiento de la solución

Parámetro	Valor
Temperatura inicial (initialTemperature)	10000
Temperatura final (finalTemperature)	0.001
Longitud de la cadena de Markov (iterationsNumber)	1000
Tasa de enfriamiento (coolingFactor)	0.9
Tasa de aceptación (acceptanceRate)	95%

En cuanto a los parámetros específicos del modelo, la penalización asignada a las restricciones incumplidas se fijó en un valor de 10 000, con el objetivo de desincentivar soluciones inviables. La velocidad de los vehículos H se estableció en 1, mientras que el costo fijo por vehículo ϒ se definió en 100. Adicionalmente, los parámetros de ponderación α y β se configuraron en 1.

5.1.2 Entorno Computacional

Los experimentos se realizaron en un equipo con las siguientes características:

Procesador	Intel Core i7-11800H @ 2.30GHz
Memoria RAM	16 GB
Sistema operativo	Windows 11
Lenguaje de programación	C++ (compilador: GCC 11.3)

5.2 Resultados del Algoritmo Propuesto

En la Tabla 5.1 se presentan los resultados correspondientes a la primera solución factible encontrada por el algoritmo en un total de 100 ejecuciones independientes para cada instancia. La información reportada incluye el mejor costo factible obtenido, el tiempo promedio requerido y el porcentaje de ejecuciones en las que se alcanzó factibilidad. Cabe señalar que estos resultados reflejan únicamente el desempeño de la fase de Búsqueda de factibilidad, sin considerar aún la fase Refinamiento de la solución del algoritmo SA-Hybrid.

Tabla 5.1 Resultados de la fase Búsqueda de factibilidad del algoritmo SA-Hybrid

Instancia	Tiempo Promedio (min)	Factibilidad (%)
A01	0,00	100
A02	0,00	100
A03	0,00	100
A04	0,00	100
A05	0,00	100
A06	0,00	100
A07	0,00	100
A08	0,00	100
A09	0,00	100
A10	0,00	100
A11	0,00	100
A12	0,00	100
A13	0,00	100
A14	0,00	100
A15	0,00	100
A16	0,00	100
A17	0,00	100
A18	0,00	100
A19	0,00	100
A20	0,00	100
Promedio	***0,00***	***100***
B01	0,00	100
B02	0,03	100
B03	0,64	100
B04	0,02	100
B05	0,86	100

Tabla 5.1 Resultados de la fase Búsqueda de Factibilidad del algoritmo SA-Hybrid (continuación)

Instancia	Tiempo Promedio (min)	Factibilidad (%)
B06	-	-
B07	0,00	100
B08	0,00	100
B09	0,03	100
B10	0,87	100
B11	0,00	100
B12	0,02	100
B13	0,28	100
B14	1,04	100
B15	3,83	100
B16	7,99	100
B17	0,00	100
B18	0,00	100
B19	0,16	100
B20	0,04	100
Promedio	***0,83***	***95***
C01	0,00	100
C02	0,07	100
C03	0,04	100
C04	0,00	100
C05	0,12	100
C06	7,51	100
C07	0,00	100
C08	0,00	100
C09	0,21	100
C10	10,67	100
C11	0,00	100
C12	0,02	100
C13	2,10	100
C14	0,08	100
C15	5,03	100
C16	-	-
C17	0,00	100
C18	0,95	100
C19	0,00	100
C20	0,06	100
Promedio	***1,41***	***95***

Tabla 5.1 Resultados de la fase Búsqueda de Factibilidad del algoritmo SA-Hybrid (continuación)

Instancia	Tiempo Promedio (min)	Factibilidad (%)
D01	0,00	100
D02	0,30	100
D03	0,07	100
D04	0,00	100
D05	0,12	100
D06	9,33	100
D07	0,00	100
D08	0,00	100
D09	0,05	100
D10	0,13	100
D11	0,00	100
D12	0,06	100
D13	0,04	100
D14	0,01	100
D15	0,66	100
D16	6,12	100
D17	0,00	100
D18	0,01	100
D19	0,00	100
D20	0,05	100
Promedio	***0,85***	***100***
Promedio total	**0,76**	**97,50**

5.3 Comparación con el Estado del Arte

A continuación, se presentan tablas comparativas entre el algoritmo propuesto y el articulo base (Li y otros, 2019).

En la tabla 5.2 se presentan, para cada instancia del conjunto de prueba, los siguientes indicadores:

- CPLEX: mejor costo de solución y el tiempo de cómputo (en minutos) reportado.
- DFACN: mejor costo de solución y el tiempo de cómputo (en minutos) reportado.
- SA-Hybrid: mejor costo de solución obtenido, el GAP (%), que representa la desviación porcentual respecto al mejor valor conocido entre CPLEX y DFACN, y el tiempo promedio de cómputo (en minutos) en el que se alcanzó la solución.

El cálculo del GAP (%) se realizó de acuerdo con la siguiente ecuación:

$$GAP = \frac{C_{SA-Hybrid} - C_{Best}}{C_{Best}} * 100$$

Donde $C_{SA-Hybrid}$ corresponde al costo de la mejor solución encontrada por el algoritmo propuesto y C_{Best} corresponde al menor costo reportado entre CPLEX y DFACN para la misma instancia.

La tabla 5.2 muestran que CPLEX solo obtuvo soluciones factibles en instancias pequeñas, mientras que el algoritmo propuesto logró resolver la mayoría de las instancias, evidenciando buena capacidad de factibilidad. Sin embargo, sus resultados fueron inferiores al DFACN, lo que sugiere que, si bien el enfoque desarrollado es competitivo en términos de factibilidad, aún presenta margen de mejora en la eficiencia y en la calidad de las soluciones obtenidas.

Tabla 5.2. Resultados de algoritmos de referencia y el algoritmo propuesto SA-Hybrid

Instancia	CPLEX		DFACN		SA-Hybrid		
	Mejor Solución	CPU (min)	Mejor Solución	CPU (min)	Mejor Solución	GAP (%)	CPU (min)
A01	**1455,22**	300,00	**1455,22**	0,01	1457,77	0,18	2,10
A02	**1729,15**	300,00	**1729,15**	0,00	1758,91	1,72	2,75
A03	**1168,85**	300,00	**1168,85**	0,00	**1168,85**	0,00	2,85
A04	**1509,46**	300,00	**1509,46**	0,00	1518,06	0,57	3,64
A05	**1173,01**	300,00	**1173,01**	0,00	1215,98	3,66	4,07
A06	**1267,55**	300,00	**1267,55**	0,01	**1267,55**	0,00	7,66
A07	**1379,71**	300,00	**1379,71**	0,00	1547,9	12,19	5,12
A08	**1474,25**	300,00	**1474,25**	0,00	1586,96	7,65	5,88
A09	**1828,01**	300,00	**1828,01**	0,03	1837,65	0,53	6,32
A10	**1283,31**	300,00	**1283,31**	0,00	**1283,31**	0,00	7,63
A11	**1222,70**	300,00	**1222,70**	0,01	1247,61	2,04	6,90
A12	**1472,06**	300,00	**1472,06**	0,01	1496,2	1,64	7,16
A13	**1125,96**	300,00	**1125,96**	0,00	**1125,96**	0,00	6,58
A14	**1333,91**	300,00	**1333,91**	0,01	1341,81	0,59	6,08
A15	**861,88**	300,00	**861,88**	0,01	**861,88**	0,00	8,07
A16	**1215,15**	300,00	**1215,15**	0,01	**1215,15**	0,00	8,67
A17	**1234,06**	300,00	**1234,06**	0,01	1256,74	1,84	8,70
A18	**1236,64**	300,00	**1236,64**	0,00	**1236,64**	0,00	7,72
A19	**1334,11**	300,00	**1334,11**	0,00	1355,39	1,60	6,54
A20	**1085,18**	300,00	**1085,18**	0,01	1099,22	1.29	6,38
B01	2584,50	300,00	**2459,42**	0,21	2461,68	0,09	7,77
B02		300,00	**4036,97**	9,37	4279,03	6,00	12,72
B03		300,00	**7014,75**	19,16	7528,96	7,33	25,20
B04		300,00	**7579,69**	77,00	8244,38	8,77	28,68
B05		300,00	**8495,91**	40,08	9471,11	11,48	42,67
B07		300,00	**3119,35**	1,09	3252,34	4,26	8,84
B08		300,00	**6204,73**	19,82	7099,19	14,42	35,21
B09		300,00	**8851,83**	16,05	10145,30	14,61	63,60
B10		300,00	**11646,52**	31,75	13185,00	13,21	74,86
B11		300,00	**2164,01**	0,50	2287,81	5,72	18,69
B12		300,00	**3560,85**	18,13	3835,96	7,73	26,98
B13		300,00	**6531,78**	18,85	7218,82	10,52	47,52
B14		300,00	7317,52	77,55	8026,74	9,69	26,31
B15		300,00	7604,43	148,64	8546,75	12,39	37,77

Tabla 5.2. Resultados de algoritmos de referencia y el algoritmo propuesto SA-Hybrid (continuación)

Instancia	CPLEX		DFACN		SA-Hybrid		
	Mejor Solución	CPU (min)	Mejor Solución	CPU (min)	Mejor Solución	GAP (%)	CPU (min)
C01	**2768,06**	300,00	2762,35	1,26	2781,84	0,71	11,72
C02		300,00	**4566,35**	4,14	4739,37	3,79	15,77
C03		300,00	**8235,48**	28,43	8916,36	8,27	31,58
C04		300,00	**8888,81**	76,75	9530,75	7,22	71,12
C05		300,00	9915,67	30,74	11154,40	12,49	66,82
C06		300,00	14036,02	116,92	15523,30	10,60	80,44
C07	4280,36	300,00	3991,29	0,50	4165,78	4,37	24,83
C08		300,00	7528,70	6,78	8434,38	12,03	35,58
C09		300,00	9770,74	32,98	11016,90	12,75	39,60
C10		300,00	15081,04	32,60	16114,20	6,85	79,83
C11		300,00	2548,17	0,87	2621,51	2,88	18,70
C12		300,00	3868,64	3,21	4151,20	7,30	22,63
C13		300,00	7827,57	6,28	8263,78	5,57	37,47
C14		300,00	8358,05	33,59	9303,78	11,32	102,32
C15		300,00	9229,73	24,17	10134,50	9,80	49,94
C17		300,00	3639,44	1,14	3861,23	6,09	28,03
C18		300,00	6985,42	21,14	7412,36	6,11	27,52
C19		300,00	9204,43	44,34	10342,00	12,36	32,38
C20		300,00	14039,78	94,40	15466,50	10,16	87,49
D01	**3155,54**	300,00	**3155,54**	1,67	3186,60	0,98	9,96
D02		300,00	**5019,83**	7,29	5223,50	4,06	6,84
D03		300,00	**9222,55**	6,87	9726,96	5,47	28,33
D04		300,00	**9644,28**	14,77	10082,80	4,55	51,79
D05		300,00	**11050,40**	21,26	12038,00	8,94	51,17
D06		300,00	**15718,61**	87,32	16709,90	6,31	76,87
D07	4113,90	300,00	4008,20	0,38	4217,46	5,22	22,18
D08		300,00	8195,89	6,16	9010,17	9,94	47,99
D09		300,00	11152,57	3,45	12238,90	9,74	63,26
D10		300,00	15662,80	150,01	16912,50	7,98	61,49
D11		300,00	2913,13	0,87	2960,16	1,61	16,10
D12		300,00	4723,24	6,87	4960,51	5,02	19,12
D13		300,00	8739,15	13,47	9360,36	7,11	26,23
D14		300,00	9392,67	53,42	10247,90	9,11	47,60
D15		300,00	10450,10	65,49	11155,30	6,75	50,94
D16		300,00	14574,36	95,35	15427,50	5,85	119,57
D17		300,00	3880,51	0,51	4098,65	5,62	19,41
D18		300,00	7700,81	4,14	8337,98	8,27	30,23
D19		300,00	10184,12	46,62	11111,70	9,11	34,95
D20		300,00	14682,08	66,29	15967,60	8,76	62,13

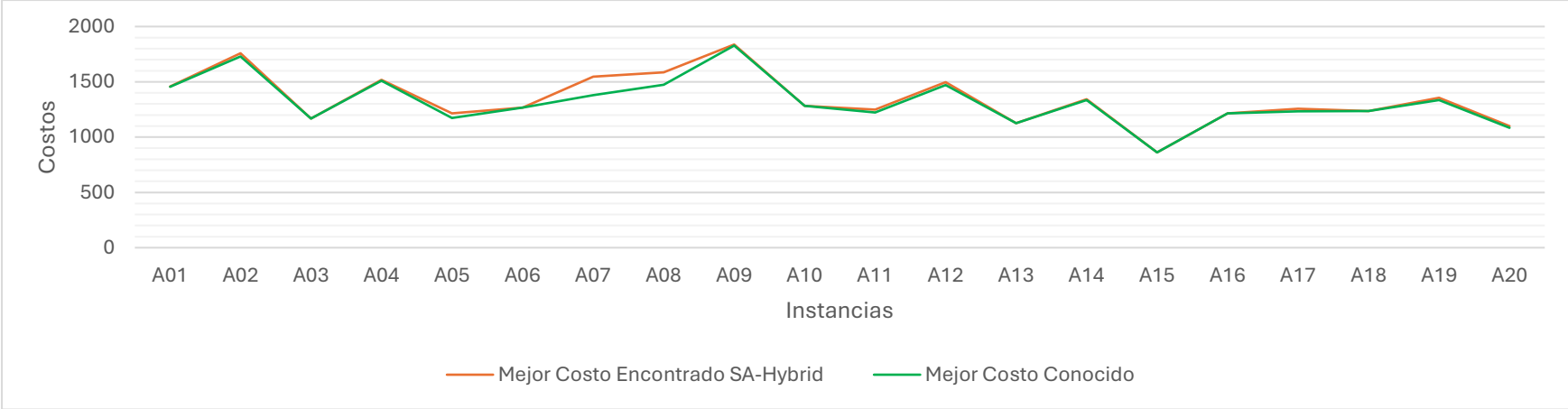

Figura 5.1. Comparación entre el mejor costo conocido y el mejor encontrado por SA-Hybrid (Instancias A).

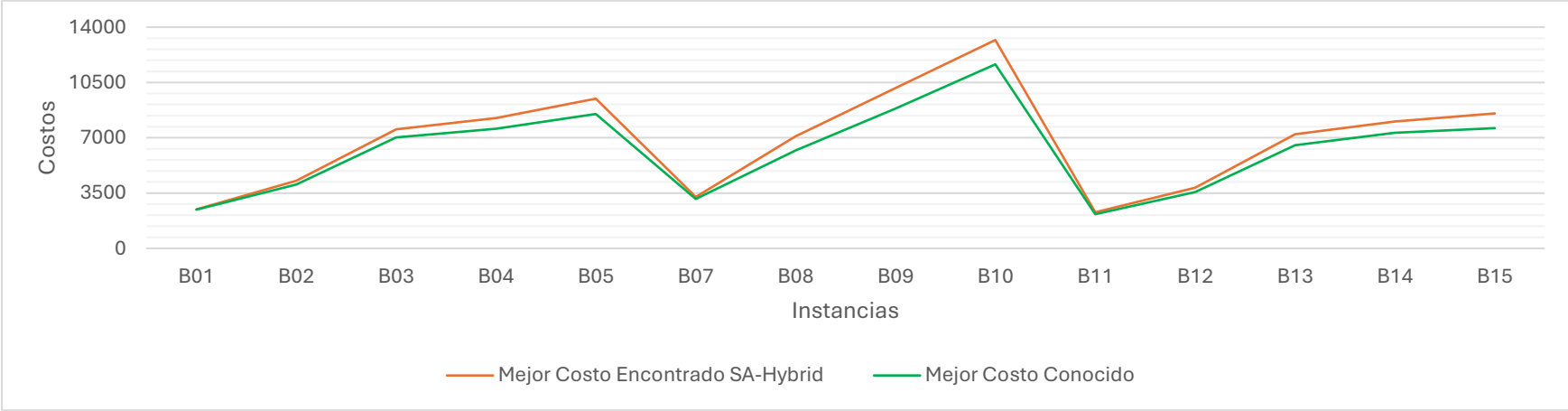

Figura 5.2 Comparación entre el mejor costo conocido y el mejor encontrado por SA-Hybrid (Instancias B).

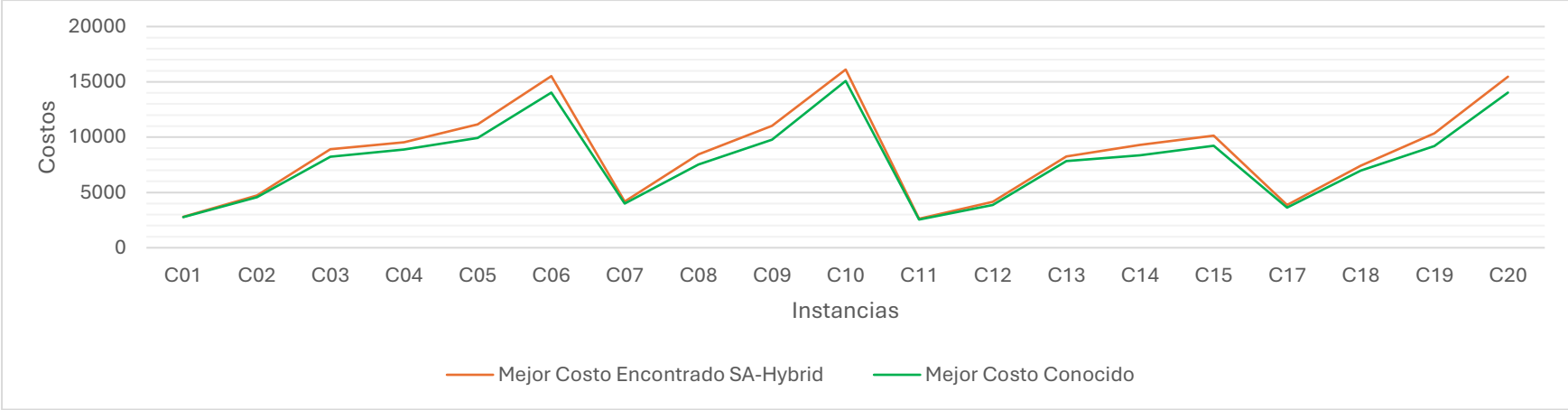

Figura 5.3 Comparación entre el mejor costo conocido y el mejor encontrado por SA-Hybrid (Instancias C).

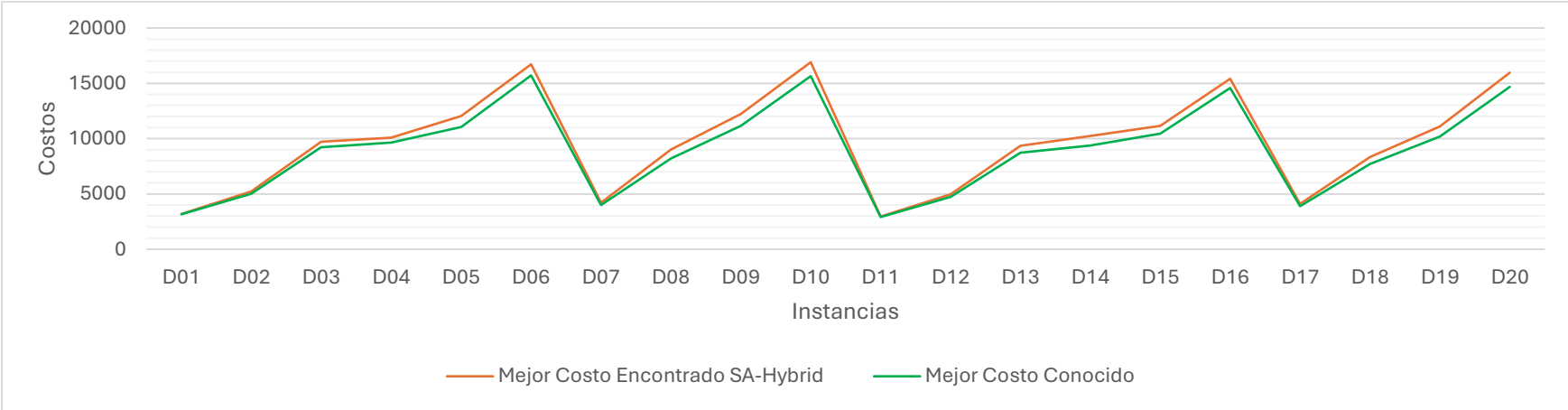

Figura 5.4 Comparación entre el mejor costo conocido y el mejor encontrado por SA-Hybrid (Instancias D).

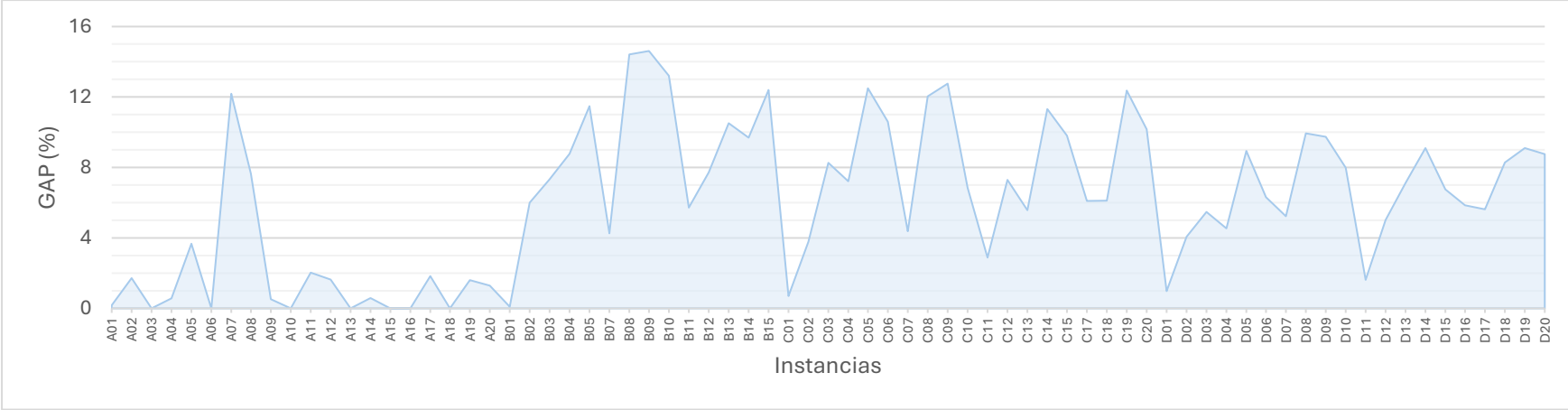

Figura 5.5 Desviación porcentual del mejor costo encontrado por SA-Hybrid respecto al mejor costo conocido.

En las figuras 5.1 a 5.4 se presentan los resultados obtenidos para cada conjunto de instancias, agrupados por familias A, B, C y D, respectivamente. Cada figura muestra, el mejor costo conocido, reportado en la literatura a partir de los resultados de CPLEX y DFACN, y el mejor costo encontrado por el algoritmo SA-Hybrid.

En la Figura 5.5 se presenta, para cada instancia, el GAP (%) o desviación porcentual entre el costo de la mejor solución obtenida por el algoritmo SA-Hybrid y el mejor costo conocido, reportado en la literatura. El eje X representa las instancias de prueba y el eje Y el valor del GAP en porcentaje. Esta representación ofrece una visión general de la magnitud de las desviaciones respecto al mejor valor conocido para cada caso. El valor promedio de GAP calculado sobre todas las instancias fue de 6,07 %, lo que indica que el algoritmo propuesto logra soluciones de alta calidad, manteniéndose en promedio a menos de un 7 % de desviación respecto a los mejores resultados conocidos.

5.4 Evaluación de Robustez

En la Tabla 5.3 se presentan los estadísticos de desempeño y robustez obtenidos a partir de 100 ejecuciones independientes del algoritmo SA-Hybrid para cada instancia. Para cada caso se reportan los siguientes indicadores:

- Media del costo, que refleja el valor promedio alcanzado.
- Costo mínimo y costo máximo, que permiten conocer el rango de soluciones obtenidas.
- Desviación estándar, utilizada como medida de variabilidad para evaluar la estabilidad del algoritmo.
- Porcentaje de factibilidad, que indica la proporción de ejecuciones en las que se obtuvo una solución factible.

Estos indicadores proporcionan una visión integral del comportamiento del algoritmo, permitiendo evaluar su consistencia, calidad de soluciones y capacidad de generar soluciones factibles de manera recurrente.

Tabla 5.3 Indicadores estadísticos de robustez del algoritmo SA-Hybrid

Instancia	Media Costo	Mín. Costo	Máx. Costo	Desv. Estándar	Factibilidad (%)
A01	1466,00	1457,77	1477,87	9,52	100
A02	1758,96	1758,91	1759,97	0,23	100
A03	1186,45	1168,85	1230,23	20,48	100
A04	1539,81	1518,06	1555,59	8,54	100
A05	1217,75	1215,98	1242,09	6,29	100
A06	1273,81	1267,55	1288,98	4,58	100
A07	1551,24	1547,90	1560,73	5,66	100
A08	1586,96	1586,96	1586,96	0,00	100
A09	1837,65	1837,65	1837,65	0,00	100
A10	1313,55	1283,31	1316,40	9,12	100
A11	1264,27	1247,61	1299,23	9,76	100
A12	1506,79	1496,20	1524,07	9,80	100
A13	1135,83	1125,96	1168,81	11,32	100
A14	1360,58	1341,81	1410,77	22,43	100
A15	872,27	861,89	912,10	8,89	100
A16	1221,36	1215,15	1242,80	7,89	100
A17	1275,71	1256,74	1362,13	28,44	100
A18	1268,93	1236,64	1342,44	20,14	100

Tabla 5.3 Indicadores estadísticos de robustez del algoritmo SA-Hybrid (continuación)

Instancia	Media Costo	Mín. Costo	Máx. Costo	Desv. Estándar	Factibilidad (%)
A19	1361,07	1355,39	1397,42	7,61	100
A20	1130,05	1115,22	1173,95	13,54	100
B01	2611,84	2469,61	2785,17	66,78	100
B02	4457,11	4279,03	4674,21	79,84	100
B03	7793,93	7539,01	8160,63	104,76	100
B04	8585,51	8255,60	8808,35	111,18	100
B05	9921,49	9471,11	10307,90	139,98	100
B07	3491,10	3252,34	3715,62	86,92	100
B08	7435,20	7099,19	7695,76	128,86	100
B09	10659,42	10305,20	11083,70	156,60	100
B10	13635,65	13206,90	13904,80	156,13	100
B11	2412,26	2295,45	2549,97	63,55	100
B12	4053,79	3869,74	4211,53	66,46	100
B13	7578,47	7408,23	7843,12	127,72	100
B14	8300,00	8127,04	8520,80	122,21	100
B15	8810,46	8546,75	9051,64	147,98	100
C01	2881,39	2781,84	2984,47	48,39	100
C02	4921,53	4739,37	5184,76	76,37	100
C03	9164,63	8929,74	9473,58	115,17	100
C04	9956,01	9632,20	10294,80	150,00	100
C05	11448,37	11154,40	11847,10	141,27	100
C06	15793,54	15593,70	16063,20	151,82	100
C07	4320,25	4185,96	4487,33	71,96	100
C08	8661,94	8448,76	8817,91	83,93	100
C09	11311,48	11016,90	11650,40	149,36	100
C10	16354,43	16165,00	16664,50	131,56	100
C11	2731,64	2684,16	2775,35	34,04	100
C12	4388,72	4248,18	4465,44	69,40	100
C13	8518,50	8308,90	8698,25	131,87	100
C14	9702,75	9566,03	9949,27	109,38	100
C15	10480,19	10356,50	10612,20	88,86	100
C17	4065,23	3953,62	4212,10	83,64	100
C18	7601,91	7496,49	7709,89	67,88	100
C19	10678,03	10366,00	11001,60	180,94	100
C20	15774,14	15466,50	16017,70	196,74	100
D01	3243,79	3186,60	3338,55	29,53	100
D02	5387,81	5256,57	5530,12	67,06	100
D03	9985,12	9779,45	10202,40	103,92	100
D04	10534,86	10224,00	10870,80	129,15	100
D05	12414,87	12158,40	12764,80	119,68	100

Tabla 5.3 Indicadores estadísticos de robustez del algoritmo SA-Hybrid (continuación)

Instancia	Media Costo	Mín. Costo	Máx. Costo	Desv. Estándar	Factibilidad (%)
D06	16992,78	16737,70	17120,00	149,13	100
D07	4350,94	4237,45	4525,51	61,25	100
D08	9385,86	9094,86	9653,58	134,12	100
D09	12531,08	12287,40	12831,00	177,30	100
D10	17262,94	16992,10	17528,70	130,94	100
D11	3101,59	3003,99	3207,45	43,64	100
D12	5152,51	5025,16	5275,56	65,95	100
D13	9585,91	9391,84	9808,68	109,46	100
D14	10571,23	10420,90	10821,10	119,14	100
D15	11462,43	11304,00	11780,20	140,29	100
D16	15895,07	15558,00	16087,60	168,55	100
D17	4276,23	4148,58	4439,48	101,28	100
D18	8543,81	8337,98	8660,01	98,27	100
D19	11468,46	11111,70	11682,40	181,99	100
D20	16335,03	16128,80	16666,10	161,43	100
					100

En relación con los resultados obtenidos por el método propuesto, la evaluación sobre las 80 instancias de referencia reveló un comportamiento favorable en términos de factibilidad: en 78 instancias se alcanzó solución factible en el 100 % de las ejecuciones, lo que equivale a un 97,5 % de instancias exitosas. Las únicas instancias en las que no se consiguió factibilidad bajo la configuración empleada fueron B06 y C16. Respecto a los tiempos para alcanzar la primera solución factible, las instancias del conjunto A presentaron un tiempo medio de 0,0 minutos (prácticamente inmediato); las instancias del conjunto B, excluida B06, mostraron un tiempo medio de 0,79 minutos; las del conjunto C, excluida C16, un tiempo medio de 1,41 minutos; y las del conjunto D un tiempo medio de 0,85 minutos. Considerando únicamente las instancias donde se logró factibilidad, el tiempo medio general fue de 0,76 minutos. La instancia con mayor tiempo promedio para encontrar factibilidad fue C10, con 10,67 minutos.

En aquellas instancias en las que no fue posible alcanzar factibilidad, el algoritmo continuó ejecutándose hasta agotar el límite de exploración definido. En este escenario, se observó que para la instancia B06 el tiempo promedio de ejecución fue de 1530,51 minutos, mientras que en la instancia C16 se registró un promedio de 1250,58 minutos.

En cuanto a la calidad de las soluciones, el procedimiento encontró 7 óptimos en el subconjunto A; para el resto de las instancias los mejores valores obtenidos resultaron superiores que las mejores soluciones conocidas en la literatura. Si se considera la fase de Refinamiento de la solución en su conjunto, los tiempos medios registrados fueron: 6,04 minutos para las instancias A; 30,21 minutos para las instancias B ejecutadas; 32,55 minutos para las instancias C ejecutadas; y 37,49 minutos para las instancias D ejecutadas.

Estos resultados dependen directamente del conjunto de instancias analizado y de la parametrización experimental aplicada. Asimismo, la naturaleza estocástica del método introduce variabilidad entre ejecuciones; por tanto, otros conjuntos de instancias o configuraciones alternativas podrían modificar tanto la tasa de factibilidad como la calidad y los tiempos de cómputo reportados. Estas consideraciones aconsejan interpretar los hallazgos como representativos del diseño experimental aquí empleado y como base para posteriores refinamientos y análisis estadísticos más exhaustivos.

Capítulo 6. Conclusiones y Trabajos Futuros

6.1 Conclusiones

El algoritmo híbrido propuesto ha demostrado ser efectivo para generar soluciones factibles, cumpliendo el objetivo general de la investigación. La verificación experimental muestra que la implementación alcanza una tasa de factibilidad cercana al 97,5% sobre el conjunto de instancias, lo que respalda la hipótesis planteada. La fase de búsqueda de factibilidad se caracteriza por tiempos de cómputo reducidos (del orden de segundos a pocos minutos en la mayoría de los casos), indicando que la estrategia inicial permite obtener soluciones factibles de forma eficiente.

Desde la perspectiva de robustez, las estadísticas reflejan una alta consistencia en la mayor parte de las instancias, aunque existen algunos casos con mayor dispersión que requieren análisis adicional. Finalmente, en las comparaciones preliminares con resultados reportados en la literatura, SA-Hybrid resulta competitivo en términos de rapidez para alcanzar factibilidad, si bien en determinadas instancias los costos finales aún quedan por encima de los mejores valores publicados, lo que sugiere oportunidades de mejora en la fase de Refinamiento de la solución.

El algoritmo mostró buen comportamiento y robustez, pero alcanzó factibilidad en 78 de 80 instancias (97,5%); no se encontraron soluciones factibles para B06 y C16 bajo la configuración empleada. El valor promedio de GAP calculado sobre todas las instancias fue de 6,07 %, lo que indica que el algoritmo propuesto logra soluciones de alta calidad, manteniéndose en promedio a menos de un 7 % de desviación respecto a los mejores resultados conocidos en la literatura. En el contexto de metaheurísticas aplicadas a problemas de ruteo vehicular, un GAP de este orden se considera competitivo y aceptable. Este resultado respalda la eficiencia del enfoque SA-Hybrid para resolver instancias de tamaño y complejidad similares a las evaluadas.

6.2 Trabajos Futuros

Como trabajo futuro se propone ampliar la validación del método mediante el uso de otras colecciones de la literatura y la generación de instancias más exigentes (mayor número de clientes, depósitos y restricciones) para evaluar con mayor rigor su robustez y competitividad. Además, de explorar hibridaciones con otras metaheurísticas y esquemas de búsqueda local, por ejemplo, Búsqueda Tabú, Búsqueda Local Iterada y Algoritmos Genéticos con el objetivo de acercar los resultados al óptimo en instancias de alta complejidad y abordar los casos problemáticos detectados (las instancias B6 y C16).

Referencias

Arin, A., Rabadi, G., & Unal, R. (2010). *Tuning Parameters In Heuristics By Using Design Of Experiments Methods.* In Selected Papers Presented at MODSIM World 2009 Conference and Expo.

Beetrack. (24 de diciembre de 2024). *La importancia de la logística en el mundo de hoy*. https://www.beetrack.com/es/blog/logistica

Bektaş, T., & Laporte, G. (2011). *The pollution-routing problem.* https://doi.org/10.1016/j.trb.2011.02.004

Chen, S., Chen, R., Wang, G. G., Gao, J., & Sangaiah, A. K. (2018). *An adaptive large neighborhood search heuristic for dynamic vehicle routing problems.* Computers & Electrical Engineering, 67, 596-607. https://doi.org/10.1016/j.compeleceng.2018.02.049

Clarke, G., & Wright, J. W. (1964). *Scheduling of vehicles from a central depot to a number of delivery points.* Operations research, 12(4), 568-581. https://doi.org/https://doi.org/10.1287/opre.12.4.568

Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y., & Semet, F. (2007). *VRP with time windows. In Toth, P. & Vigo, D. (Eds.).). The vehicle routing problem.* https://doi.org/10.1137/1.9780898718515.ch6

Cordeau, J. F., Laporte, G., & Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*.

Dantzig, G. B., & Ramser, J. H. (1959). *The truck dispatching problem.* Management science, 6(1), 80-91. https://doi.org/10.1287/mnsc.6.1.80

Desaulniers, G., Desrosiers, J., & Solomon, M. M. (2002). *Column generation.* Springer Science & Business Media. https://doi.org/10.1007/b135457

Díaz, A., Glover, F., Ghaziri, H. M., Gonazáles, J. L., Laguna, M., Moscato, P., & Tseng, F. T. (1996). *Optimización Heurísticay Redes Neuronales.* Madrid, España: PARAINFO. S.A.

DispatchTrack. (3 de marzo de 2023). *Los 7 principales desafíos de los operadores logísticos (3PL)*. https://www.dispatchtrack.com/es/blog/operadores-logisticos-3pl

Dorigo, M., & Gambardella, L. M. (1997). *Ant colony system: A cooperative learning approach to the traveling salesman problem.* IEEE Transactions on evolutionary computation, 1(1), 53-66. https://doi.org/10.1109/4235.585892

Dror, M., & Trudeau, P. (1990). *Split delivery routing.* Naval Research Logistics (NRL), 37(3), 383-402. https://doi.org/10.1002/nav.3800370304

Eksioglu, B., Vural, A. V., & Reisman, A. (2009). *The vehicle routing problem: A taxonomic review.* Computers & Industrial Engineering, 57(4), 1472-1483. https://doi.org/10.1016/j.cie.2009.05.009

Feo, T. A., & Resende, M. G. (1995). *Greedy randomized adaptive search procedures.* Journal of global optimization, 6(2), 109-133. https://doi.org/10.1007/BF01096763

Fernández, A., García, M., & López, J. (2018). *Optimización de rutas logísticas y reducción de costos operativos*.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness.* The Journal of Symbolic Logic, 48(2), 498-500. https://doi.org/10.2307/2273574

Gendreau, M., Laporte, G., & Potvin, J. Y. (2002). *Metaheuristics for the capacitated VRP. In The vehicle routing problem (pp. 129-154).* Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718515.ch6

Gillett, B. E., & Miller, L. R. (1974). *A heuristic algorithm for the vehicle-dispatch problem.* Operations research, 22(2), 340-349. https://doi.org/10.1287/opre.22.2.340

Glover, F. (1986). *Future paths for integer programming and links to artificial intelligence.* Computers & operations research, 13(5), 533-549. https://doi.org/10.1016/0305-0548(86)90048-1

Glover, F. (1989). *Tabu search part I.* ORSA Journal on computing, 1(3), 190-206. https://doi.org/10.1287/ijoc.1.3.190

Golden, B., Raghavan, S., & Wasil, E. A. (2008). *The vehicle routing problem: Latest advances and new challenges (Vol. 43).* Springer Science & Business Media. https://doi.org/10.1007/978-0-387-77778-8

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* University of Michigan Press google schola, 2, 29-41.

ILS. (4 de septiembre de 2024). *Logística en la economía global: Impulsores y desafíos en un mundo interconectado*. https://www.logistica-ils.com/noticias/logistica-en-la-economia-global-impulsores-y-desafios-en-un-mundo-interconectado

International Transport Forum. (2023). *Perspectivas del Transporte del ITF 2023*. https://www.itf-oecd.org/perspectivas-del-transporte-del-itf-2023

Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). *Optimization by simulated annealing.* https://doi.org/10.1126/science.220.4598.671

Laporte, G. (1992). *The vehicle routing problem: An overview of exact and approximate algorithms.* European journal of operational research, 59(3), 345-358. https://doi.org/10.1016/0377-2217(92)90192-C

Laporte, G. (2009). *Fifty years of vehicle routing.* Transportation science, 43(4), 408-416. https://doi.org/10.1287/trsc.1090.0301

Lenstra, J. K., & Rinnooy Kan, A. H. (1981). *Complexity of vehicle routing and scheduling problems.* Networks, 11(2), 221-227. https://doi.org/10.1002/net.3230110211

Li, J., Li, T., Yu, Y., Zhang, Z., Pardalos, P. M., Zhang, Y., & Ma, Y. (2019). *Li, J., Li, T., Yu, Y., Zhang, Z., Pardalos, P. M., Zhang, Y., & Ma, Y. (2019). Discrete firefly algorithm with compound neighborhoods for asymmetric multi-depot vehicle routing problem in the maintenance of farm machinery.* Applied soft computing, 81, 105460. https://doi.org/10.1016/j.asoc.2019.04.030

Min, H., & Yih, P. (1989). *The multiple depot vehicle scheduling problem.* https://doi.org/10.1016/0191-2607(89)90022-5

Minsky, M. (1961). *Steps toward artificial intelligence.* Proceedings of the IRE. https://doi.org/10.1109/JRPROC.1961.287775

Organisation for Economic Co-operation and Development (OECD). (2021). *Perspectivas del Transporte 2021*.

Osman, I. H. (1993). *Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem.* Annals of operations research, 41(4), 421-451. https://doi.org/10.1007/BF02023004

Quadminds. (4 de enero de 2022). *3 grandes problemas de la entrega de pedidos y sus consecuencias*. https://www.quadminds.com/blog/problemas-en-la-entrega-de-pedidos/

Quadminds. (22 de marzo de 2022). *Los problemas logísticos más comunes en la entrega del producto*. https://www.quadminds.com/blog/problemas-logisticos-2/

Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach.*

Samuel, A. L. (1959). *Some studies in machine learning using the game of checkers.* IBM Journal of Research and Development. https://doi.org/10.1147/rd.33.0210

Schneider, M., Stenger, A., & Goeke, D. (2014). *A hybrid simulation approach for estimating the market share evolution of electric vehicles.* Transportation Science, 48(4), 651-670. https://doi.org/10.1287/trsc.2014.0526

Schrage, L. (1981). *A classification of VRPs encountered in practice.*

Solomon, M. M. (1987). *Algorithms for the vehicle routing and scheduling problems with time window constraints.* Operations research, 35(2), 254-265. https://doi.org/10.1287/opre.35.2.254

Toth, P., & Vigo, D. (1997). *An exact algorithm for the vehicle routing problem with backhauls.* Transportation science, 31(4), 372-385. https://doi.org/10.1287/trsc.31.4.372

Toth, P., & Vigo, D. (2002). *The Vehicle Routing Problem.* Society for Industrial and Applied Mathematics.

Toth, P., & Vigo, D. (2014). *Vehicle routing: problems, methods, and applications.* Society for industrial and applied mathematics. https://doi.org/10.1137/1.9781611973594

Vidal, T., Laporte, G., & Matl, P. (2020). *A concise guide to existing and emerging vehicle routing problem variants.* European Journal of Operational Research, 286(2), 401-416. https://doi.org/10.1016/j.ejor.2019.10.010

Willard, J. (1985). *A tabu search heuristic for the vehicle routing problem.* Technical report, University of Colorado.

Xiao, S., Peng, P., Zheng, P., & Wu, Z. (2024). *A hybrid adaptive simulated annealing and tempering algorithm for solving the half-open multi-depot vehicle routing problem.* Mathematics, 12(7), 947. https://doi.org/10.3390/math12070947

Anexos

Soluciones óptimas encontradas por SA-Hybrid

Instancia	Rutas	Costo
A03	{(21, 11, 16, 12, 17, 3, 14, 19, 10, 1, 13, 18, 21), (21, 15, 20, 2, 7, 9, 5, 21), (22, 8, 4, 6, 22)}	1168,85
A06	{(21, 8, 14, 19, 12, 17, 10, 15, 20, 11, 16, 21), (22, 2, 1, 6, 9, 7, 13, 18, 22), (22, 5, 4, 3, 22)}	1267,55
A10	{(22, 8, 3, 5, 12, 17, 22), (22, 13, 18, 10, 22), (22, 9, 14, 19, 4, 22), (21, 15, 20, 2, 1, 7, 11, 16, 6, 21)}	1283,31
A13	{(21, 11, 16, 5, 15, 20, 1, 21), (21, 3, 14, 19, 8, 4, 6, 10, 12, 17, 21), (21, 13, 18, 7, 9, 2, 21)}	1125,96
A15	{(21, 8, 6, 9, 3, 12, 17, 7, 2, 14, 19, 21), (21, 11, 16, 1, 4, 15, 20, 5, 10, 13, 18, 21)}	861,88
A16	{(21, 14, 19, 9, 7, 2, 13, 18, 21), (22, 15, 20, 8, 3, 4, 5, 10, 22), (22, 11, 16, 1, 6, 12, 17, 22)}	1215,15
A18	{(22, 13, 18, 4, 9, 12, 17, 3, 22), (21, 7, 11, 16, 8, 10, 21), (22, 5, 15, 20, 2, 1, 6, 14, 19, 22)}	1236,64

Cuernavaca, Morelos a 21 de octubre del 2025.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
DIRECTOR DE LA FACULTAD DE CONTADURÍA,
ADMINISTRACIÓN E INFORMÁTICA.
PRESENTE

En mi carácter de revisor de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Yaser Rodríguez Milán, con matrícula 10072390, con el título SOLUCIÓN HEURÍSTICA PARA EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON MÚLTIPLES DEPÓSITOS, VENTANAS DE TIEMPO Y REQUERIMIENTOS DE ARCOS, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Dr. Federico Alonso Pecina
Profesor- investigador
Facultad de Contaduría, Administración e Informática

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANÓ INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

FEDERICO ALONSO PECINA | Fecha:2025-10-21 11:33:18 | FIRMANTE

gm3KrIgWxnStJUfPnG2CssfE9EqR3OVCBhUqEtrTfA4CP/fyikZSPrSxeaxixLyHZR52r/UUtUBk5PekjYOrwevCbcnfDE37gGkBEPJFcZI0qDsSjSTu+puxdfFEwnM/eSZRHQBHMgtH8dEKSVwLjCS7Ur+JvZHRJSzMi1lodMGFkligh6t+Tvy5eru/dVDHXVE3HSPsI+vxSwfvxUo2hfmf593+2GxfS8UxSA8vxxsIHJPRlUeGZLzuxt7XDYQJ99Ys6+CzsXCABg2g7lMW1mUmXz02RmcMCpmOlQaesPQOIzOt10dVjxZSDpDd4naO5HsoMxqQjVY+RL0Fd7s12Q==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

HVycZrPuE

https://efirma.uaem.mx/noRepudio/NSv2KLk6RSLghU58c7M0hWiKdN1qIc2H

Cuernavaca, Morelos a 20 de octubre del 2025.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
DIRECTOR DE LA FACULTAD DE CONTADURÍA,
ADMINISTRACIÓN E INFORMÁTICA.
PRESENTE

En mi carácter de revisor de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Yaser Rodríguez Milán, con matrícula 10072390, con el título SOLUCIÓN HEURÍSTICA PARA EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON MÚLTIPLES DEPÓSITOS, VENTANAS DE TIEMPO Y REQUERIMIENTOS DE ARCOS, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Firmado Electrónicamente
Dra. Jesús del Carmen Peralta Abarca
Profesor- investigador
Facultad de Ciencias Químicas e Ingeniería

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANÓ INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

JESUS DEL CARMEN PERALTA ABARCA | Fecha:2025-11-10 20:25:18 | FIRMANTE

ZghwCpdizWQFD43CgEczw+N95uIGhaY97PG1ElBagx1ffDfeFg7D+cEhKFdj7GXG7LuB1wNY18zyoHSaBGU71XykG1SBPoUqufiApYdHJhuuMTsYoQ8b0cM4lk9w2oBW0Z/NVBItfIDB2h7Ss6zENK/bjfXU8AitkVV+wzdyAb2g/YqDBief5AcM5pWz3xn20wpBWwfc+8YX/uyFdYYhzxJdFBGjk3LPKMiQch4vCrU0ARmfaCKMY5gre2hbwg4ynnFA1AQcSxLiNraP+rIaz8PHDD3t6kVLvMjdZUG1JpUpAuvY0iw/llG/+HJGd8SwAxUjzxPAMejrPvZM1aLinA==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

HmeJ1y3N8

https://efirma.uaem.mx/noRepudio/IlxqqpaTFGdn6ZxpyZYklYsfFgeW4mlO

Cuernavaca, Morelos a 20 de octubre del 2025.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
DIRECTOR DE LA FACULTAD DE CONTADURÍA,
ADMINISTRACIÓN E INFORMÁTICA.
PRESENTE

En mi carácter de revisor de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Yaser Rodríguez Milán, con matrícula 10072390, con el título SOLUCIÓN HEURÍSTICA PARA EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON MÚLTIPLES DEPÓSITOS, VENTANAS DE TIEMPO Y REQUERIMIENTOS DE ARCOS, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Dr. Marco Antonio Cruz Chávez
Profesor- investigador
Centro de Investigación en Ingeniaría y Ciencias Aplicadas

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANÓ INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

MARCO ANTONIO CRUZ CHAVEZ | Fecha:2025-11-11 22:46:33 | FIRMANTE

Bu6u5s7sPyn1RlXwBxCXGE33dkoYrvkWY7T8J0RTqXtMh1G2IvL2y4dzLs/HEeQVTiZJSxjYeAflPoVWqR2omGKBXv4LWF5A2THCTkrTWLg6pfxthVoHFyNjv9qckLWM3QcvpdHkPDnGHPOXPB2i11zQUEFZbDW2QPoJwCV/sj+QUymwdFiZ4kdSigFM26I8gHWdfvAMG5Wo2WmJTqOdy0Sfg/2KvBOACNUWcK94qYUm8T7JEgs+JrERo9GHeAA/uXgExccGbyOqW71HLIiS2WlnbyW9nPF6fZ7+ExSWYtrCI56P08KWgnzBssMacETSRq3v0HU1im5wAwceAgQb7w==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

mcl7vqnKt

https://efirma.uaem.mx/noRepudio/eUnqGkDVUYCYecUKWEf2zetfnZHPi4bt

Cuernavaca, Morelos a 20 de octubre del 2025.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
DIRECTOR DE LA FACULTAD DE CONTADURÍA,
ADMINISTRACIÓN E INFORMÁTICA.
PRESENTE

En mi carácter de revisora de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Yaser Rodríguez Milán, con matrícula 10072390, con el título SOLUCIÓN HEURÍSTICA PARA EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON MÚLTIPLES DEPÓSITOS, VENTANAS DE TIEMPO Y REQUERIMIENTOS DE ARCOS, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Dra. Irma Yazmín Hernández Báez
Profesor de Tiempo Completo
Universidad Politécnica del Estado de Morelos

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANÓ INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

IRMA YAZMIN HERNANDEZ BAEZ | Fecha:2025-11-03 10:38:49 | FIRMANTE

LZFkKBw4ZVjw1bGZ6n66cVAabbi8ZyOpFufZTTNgoe7zcfTO3mq+zbxnRSP8ld1d02yFpgn5tkztSCoV/7CJ1sz/Zark5W3ZzI7QlIAP+ll3OQfjOO6tXXeym+xQaiGVKTDqJsG6pBGtoY6E3nOi/bg2Xpn2s+9ZuqKihqxHXVm8Lle92TaFTndchlnIc76zSi/UAvccDkPLuxfofHtCiRxVcY+hpJalCbrr6gvU+lkgfvMVdQa17W+Kp7vx2rE6vDHbQAAPrWg90SN90JfxbZiomwaelrvvloCAsr0dflYlkNEYBNljurF4V7+J4HAzCJN0YwMONkUL6sfvXABIqw==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

6na2oiAxN

https://efirma.uaem.mx/noRepudio/ThsXoI0TqfrK3cki56lHLWbN2WMYQdNr

Cuernavaca, Morelos a 20 de octubre del 2025.

DR. FELIPE DE JESÚS BONILLA SÁNCHEZ
DIRECTOR DE LA FACULTAD DE CONTADURÍA,
ADMINISTRACIÓN E INFORMÁTICA.
PRESENTE

En mi carácter de revisor de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Yaser Rodríguez Milán, con matrícula 10072390, con el título SOLUCIÓN HEURÍSTICA PARA EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON MÚLTIPLES DEPÓSITOS, VENTANAS DE TIEMPO Y REQUERIMIENTOS DE ARCOS, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Dr. Martín Gerardo Martínez Rangel
Profesor- investigador
Facultad de Contaduría, Administración e Informática

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANÓ INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

MARTIN GERARDO MARTINEZ RANGEL | Fecha:2025-11-18 21:19:15 | FIRMANTE

e4KoAPoGl7aEKcv/koxJPJsTQv7Hu8svew895MObj5wP1mSCR2bFhQof85KORYhuU4uO2OhJl43YOCu9tGGImQ9JnnOme+eJ5GTehS0y24iws6AIZU6zZusiykqdiAEivc1DWcmu5FfAc32MT6wZ12R8BXuLF9Es37ute/Q/2Ace4FyFspFGk/90BhVcE2EFJxIA09B6QM5p9GIyf34kRal3iQVkFZcD1/kEovBaF/kqcn61ZTENojoCejjzimnIO/a3rHSC+WBo02vsPZwPahlKPaQ6CYZ5lhDQoLUGJRDl/XQnsnOPI/z6erBPyr+7eMUlYUtWIDVIgUMT9N62TA==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

I4dSl9nTy

https://efirma.uaem.mx/noRepudio/I7TPjK0PNAu9qjqso8ux46F6OkqDpJfA