Universidad Tecnológica de la Habana

“José Antonio Echeverría”

Facultad de Ingeniería Informática

Evaluación de ORTools para la optimización de rutas de vehículos

Autor

Dayana Vázquez Rodríguez

Tutores

Dr. C. Alejandro Rosete Suárez

Dr. C. Isis Torres Pérez

La Habana, 2024

Resumen

El presente informe aborda el Problema de Planificación de Rutas de Vehículos en el campo de la optimización. El principal propósito consiste en evaluar la capacidad de la biblioteca ORTools para abordar el Problema del Viajante de Comercio y variantes del Problema de Planificación de Rutas de Vehículos.

En el transcurso de este estudio, se profundiza en el estado del arte de la optimización de rutas, explorando las soluciones obtenidas en trabajos previos y las diversas bibliotecas disponibles para abordar este desafío. Además, se examina la eficacia de diferentes enfoques y algoritmos utilizados en la resolución de los problemas. En particular, se presta atención a la biblioteca ORTools debido a su robustez y versatilidad en la resolución de problemas combinatorios. Se plantea una solución utilizando los módulos que brinda esta biblioteca para la resolución de los problemas mencionados anteriormente. Los resultados de esta evaluación se comparan con soluciones de trabajos anteriores y se analizan en detalle para determinar las ventajas y desventajas de utilizar la biblioteca en este contexto.

Palabras clave: optimización, Problema del Viajante de Comercio, Problema de Planificación de Rutas de Vehículos, ORTools.

Abstract

This report addresses the Vehicle Route Planning Problem in the field of optimization. The main purpose is to evaluate the ability of the ORTools library to address the Trade Traveling Salesman Problem and variants of the Vehicle Route Planning Problem.

Throughout this study, we delve into the state of the art in route optimization, exploring the solutions obtained in previous works and the various libraries available to address this challenge. Additionally, we examine the effectiveness of different approaches and algorithms used in solving these problems. In particular, attention is given to the ORTools library due to its robustness and versatility in solving combinatorial problems. A solution is proposed using the modules provided by this library to solve the aforementioned problems. The results of this evaluation are compared with solutions from previous works and are analyzed in detail to determine the advantages and disadvantages of using the library in this context.

Keywords: optimization, Traveling Salesman Problem, Vehicle Routing Problem, ORTools.

Índice de contenido

Índice de ilustraciones

Índice de ecuaciones

Índice de tablas

Introducción

La optimización es un campo de estudio que busca encontrar las mejores soluciones posibles para problemas complejos, maximizando o minimizando una función objetivo sujeta a ciertas restricciones. En diversas áreas, como la logística, la planificación y el transporte, la optimización juega un papel fundamental para mejorar la eficiencia y reducir los costos operativos [51].

Dentro de este amplio panorama de optimización, uno de los desafíos más relevantes es la optimización de rutas de vehículos. Este problema se presenta en numerosos escenarios, desde empresas de transporte que necesitan planificar las rutas de sus flotas, hasta servicios de entrega que buscan minimizar los tiempos de entrega y los costos asociados [2].

Debido a lo anterior, se tiene como **objeto de estudio** la optimización haciendo uso de herramientas para resolver problemas de optimización combinatoria, mientras que el **campo de acción** se enfoca en la optimización de rutas de vehículos y la biblioteca ORTools.

Este trabajo se encarga de abordar este desafiante problema, específicamente, la resolución de los problemas conocidos como el Problema del Viajante de Comercio, TSP (*Traveling Salesman Problem*) [3] y variantes del Problema de Planificación de Rutas de Vehículos, VRP (*Vehicle Routing Problem*) [4]. Para la continuación de la Tesis luego de estas Prácticas Profesionales se espera resolver el Problema de Rutas de Autobuses Escolares SBRP (*School Bus Routing Problem*) [17], que se encarga de planificar rutas eficientes para autobuses escolares y el Subproblema de Selección de Paradas de Autobús, BSS (*Bus Stop Selection*) [17], encargado de ubicar una serie de paradas en diferentes locaciones para asignar a los pasajeros más cercanos a dichas paradas.

El problema TSP se centra en encontrar la ruta más corta que un vendedor ambulante debe tomar para visitar un conjunto de ciudades una sola vez y regresar al punto de partida. Por otro lado, el VRP implica la planificación de rutas eficientes para una flota de vehículos que debe visitar múltiples destinos para satisfacer la demanda de los clientes, minimizando costos como la distancia recorrida, el tiempo de viaje, entre otro. Dentro de los VRP se

encuentran múltiples variantes, algunas de ellas son; el Problema de Planificación de Rutas de Vehículos con Restricciones de Distancia, DVRP (*Distance Vehicle Routing Problem*) [50], que agrega una limitación de distancia máxima a recorrer por los vehículos en su ruta, el Problema de Planificación de Rutas de Vehículos con Capacidad, CVRP (*Capacitated Vehicle Routing Problem*) [4, 16], que agrega una limitación de capacidad a los vehículos, el Problema de Planificación de Rutas de Vehículos con Ventanas de Tiempo, VRPTW (*Vehicle Routing Problem with Time Windows*) [19], que incorpora ventanas de tiempo para las entregas, el Problema Planificación de Rutas de Vehículos con Recogida y Entrega, VRPPD (*Vehicle Routing Problem with Pickup and Delivery*) [20], que involucra la recogida y entrega de mercancías en diferentes ubicaciones, y el Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos, MDVRP (*Multi-Depot Vehicle Routing Problem*) [18], en el cual se considera la existencia de múltiples depósitos desde los cuales los vehículos pueden partir y regresar [3, 4][51].

La planificación de rutas manualmente para resolver estos problemas es una tarea que, si bien ha sido realizada durante años, presenta limitaciones significativas en términos de eficiencia y efectividad. El proceso manual no permite una optimización adecuada de los recursos ni del tiempo, lo que resulta en rutas ineficientes y en la asignación inadecuada de vehículos a destinos específicos. Además, la planificación manual puede ser tediosa y propensa a errores, lo que aumenta la posibilidad de retrasos y costos adicionales. Por esto, la automatización y optimización de este procedimiento surge como una solución a este problema. Al utilizar herramientas y algoritmos avanzados es posible reducir significativamente los costos y los recursos necesarios para la planificación de rutas, así como minimizar el tiempo requerido para realizar esta tarea [2, 5].

Para resolver los problemas TSP y las variantes VRP existen algunas técnicas comunes como los algoritmos heurísticos. Estos algoritmos ofrecen soluciones aproximadas utilizando métodos inteligentes basados en reglas y estrategias específicas como el Algoritmo del Vecino más Cercano [6], el Algoritmo de Inserción más Cercana [6], entre otros [7]. También se encuentran los algoritmos de optimización exacta [8], que buscan encontrar la solución óptima al explorar exhaustivamente todas las posibles combinaciones. Finalmente,

aparecen las metaheurísticas [9], basadas en principios generales que pueden explorar de manera eficiente el espacio de soluciones en comparación con los algoritmos exactos. Ejemplos de estas son el Recocido Simulado, la Búsqueda Tabú, los Algoritmos Genéticos y los Escaladores de Colinas [5, 9].

Existen diversas bibliotecas y herramientas de optimización que implementan las técnicas descritas anteriormente. Algunos ejemplos incluyen FICO Xpress [10]; potente conjunto de solvers de pago y herramientas de modelado para problemas de optimización lineal, no lineal y entera mixta. Otra alternativa es PuLP [11]; biblioteca de código abierto para modelar y resolver problemas de optimización lineal y entera mixta de forma sencilla. En esta herramienta para la resolución de problemas complejos se requiere la incorporación de solvers externos, al contrario de Xpress. También se encuentra SciPy [12]; biblioteca de código abierto que incluye módulos para optimización de forma general de problemas poco complejos. Otro ejemplo es la Biblioteca de Heurísticas de Construcción para Problemas de Planificación de Rutas de Vehículos (BHCVRP) [49]; biblioteca de código abierto en Java que resuelve algunas variantes VRP pero que no implementa metaheurísticas, se restringe solamente a heurísticas de construcción y tiene dependencias de un componente llamado BHAVRP para resolver la variante MDVRP. Por último, aparece Google ORTools [13], una biblioteca con una amplia gama de funcionalidades, excelente documentación y ejemplos de implementación. Esta ofrece soluciones para problemas de optimización combinatoria complejos, pero no proporciona una guía que indique en qué problemas de optimización de rutas de vehículos es adecuado utilizarla y en cuáles no, detallando sus fortalezas y debilidades en este aspecto para aprovecharla al máximo [13].

Debido a lo anterior, surge como **situación problemática**; la necesidad de proporcionar una guía donde se analicen las ventajas y desventajas, en el contexto de optimización de rutas, que aporta la biblioteca ORTools, seleccionada gracias a su versatilidad y robustez en comparación con el resto. Para darle solución a esta problemática se propone como **objetivo general**; definir una guía que refleje las ventajas y desventajas de utilizar la biblioteca en la resolución de los problemas de optimización de rutas de vehículos, evaluando tiempos de ejecución y calidad de las soluciones.

Se pretende cumplir los siguientes **objetivos específicos** y sus **tareas** derivadas para dar lugar a la realización del objetivo general:

1. Identificar los conceptos esenciales y el estado del arte de la optimización de rutas de vehículos.

1.1. Obtener información y conceptos clave sobre la optimización de rutas de vehículos.

1.2. Recopilar soluciones de los problemas en trabajos anteriores obtenida de artículos y tesis en sitios como Scielo, Google Scholar, entre otros.

1.3. Adquirir información y realizar una búsqueda del estado del arte del TSP y las variantes VRP.

1.4. Adquirir información y realizar una búsqueda del estado del arte de los problemas BSS y SBRP (para la continuación de la Tesis luego de las Prácticas Profesionales).

2. Experimentar con la biblioteca ORTools para la resolución de los problemas de optimización de rutas de vehículos.

2.1. Asimilar la biblioteca, incluyendo qué hace y hasta dónde es su alcance en la optimización de rutas, así como los problemas que puede resolver dentro de este contexto.

2.2. Asimilar la importación y exportación de los problemas.

2.3. Experimentar con los módulos de optimización de rutas de vehículos de la biblioteca.

2.4. Incorporar distancias aproximadas a la biblioteca para la comparación con la biblioteca BHCVRP.

2.5. Resolver los problemas TSP y las variantes VRP utilizando la biblioteca.

2.6. Evaluar los resultados obtenidos del punto anterior para analizar que técnicas de optimización resultaron ser más eficientes para las distintas instancias de los problemas.

2.7. Combinar la utilización de los módulos brindados por la biblioteca con algoritmos propios para la resolución de problemas como BSS y SBRP (para la continuación de la Tesis luego de las Prácticas Profesionales).

3. Validar y realizar pruebas a los resultados obtenidos, así como comparar con soluciones anteriores.

3.1. Obtener soluciones anteriores de la biblioteca BHCVRP [49] para comparar con los resultados obtenidos.

3.2. Obtener soluciones anteriores del trabajo de Homberger y Gehring del año 1999 [33] para comparar con los resultados obtenidos.

3.3. Realizar las comparaciones de las soluciones obtenidas del problema CVRP en términos de calidad y tiempo de ejecución con BHCVRP [49]. Luego realizar la prueba estadística de Wilcoxon [58] para comparar las soluciones obtenidas.

3.4. Realizar las comparaciones de las soluciones de la variante VRPTW en términos de calidad con los resultados obtenidos en el trabajo de Homberger y Gehring del año 1999 [33]. Luego realizar la prueba estadística de Wilcoxon [58] para comparar las soluciones obtenidas.

3.5. Realizar las comparaciones de las soluciones obtenidas de los problemas BSS y SBRP (para la continuación de la Tesis luego de las Prácticas Profesionales).

Como valor práctico, al evaluar y comparar diferentes soluciones obtenidas con la biblioteca ORTools, será posible determinar su eficacia para resolver problemas de optimización de rutas de vehículos. Esto se logra mediante la creación de demos que demuestren su funcionalidad y usabilidad. Lo anterior permitirá encontrar soluciones eficientes y cercanas a la óptima, considerando las restricciones específicas y las métricas de rendimiento deseadas en cada caso, beneficiando a las empresas de transporte y servicios de entrega al reducir los costos y minimizar los tiempos.

Según los tipos de artefactos definidos en [14], el resultado fundamental de este trabajo es una Guía (*Guideline*) que aclara las ventajas y desventajas de usar la biblioteca en el campo de la optimización de rutas de vehículos, de acuerdo a las restricciones del problema, el tamaño de las instancias, los tiempos de ejecución y la calidad de las soluciones.

Para cumplir con los objetivos mencionados el informe se compone de 3 capítulos, conclusiones, recomendaciones y referencias bibliográficas.

En el **“Capítulo 1: Introducción a la optimización de rutas de vehículos”** se profundiza en el estado del arte de los problemas de optimización de rutas, así como los conceptos fundamentales y soluciones de trabajos anteriores.

En el **“Capítulo 2: Guía para la evaluación de ORTools”** se presenta la propuesta de solución a la problemática con la resolución de los problemas de optimización de rutas para la evaluación de la biblioteca de Google, ORTools.

Finalmente, en el **“Capítulo 3: Comparación de los resultados”** se validan las soluciones obtenidas comparando con la biblioteca BHCVRP y con los resultados del artículo de Homberger y Gehring del año 1999.

Capítulo 1: Introducción a la optimización de rutas de vehículos

1.1. Introducción

En este capítulo, se proporciona una introducción a los conceptos teóricos clave de la optimización de rutas de vehículos, seguido de un análisis exhaustivo de las distintas variantes de problemas en este campo y los métodos de solución más relevantes, con un enfoque en la aplicación de heurísticas y metaheurísticas.

Además, se revisa la literatura existente para contextualizar los avances en el ámbito de la optimización de rutas, destacando las herramientas y bibliotecas de optimización más utilizadas en la actualidad. Finalmente, se justifica la selección de una de estas herramientas para abordar los problemas propuestos en esta investigación.

1.2. Problemas de optimización de rutas

En el ámbito de la investigación operativa y la logística, la optimización es un proceso matemático y computacional que busca encontrar la mejor solución posible dentro de un conjunto de alternativas disponibles. Este proceso se aplica a una amplia gama de problemas en diversas áreas, incluyendo la planificación, la asignación de recursos y la gestión de sistemas complejos. La optimización busca maximizar o minimizar una función objetivo, que puede representar costos, beneficios, eficiencia, entre otros, sujeta a una serie de restricciones específicas del problema a tratar [1].

Dentro de esta área de estudio, la optimización de rutas de vehículos constituye un campo de investigación fundamental debido a su impacto

significativo en la eficiencia operativa y la reducción de costos en la logística y la gestión de flotas de transporte. Se centra en encontrar las rutas más eficientes para uno o varios vehículos que deben realizar una serie de entregas o servicios. El objetivo principal es minimizar el costo total asociado, que puede incluir la distancia recorrida, el tiempo de viaje, el consumo de combustible, o una combinación de estos y otros factores operativos. Es vital para mejorar la eficiencia operativa y reducir costos en diversas industrias, incluyendo [2, 5]:

- **Distribución y Logística**: permite una gestión eficiente de las flotas de entrega, reduciendo el tiempo y el costo de transporte.
- **Transporte Público:** optimiza las rutas de los autobuses y otros servicios de transporte público, mejorando el servicio al usuario y reduciendo costos.
- **Servicios de Emergencia:** mejora la capacidad de respuesta en situaciones de emergencia, optimizando las rutas de ambulancias, bomberos y otros servicios críticos.

Existen varios problemas dentro del campo de la optimización de rutas de vehículos, cada uno con sus propias características y desafíos. Este trabajo se enfoca en resolver los siguientes [51]:

- **Problema del Viajante de Comercio (*Traveling Salesman Problem*, TSP)** [3]: el TSP es uno de los problemas más estudiados en el ámbito de la optimización de rutas. Se centra en encontrar la ruta más corta que permite a un viajante visitar un conjunto de ciudades exactamente una vez y regresar a la ciudad de origen. Aunque parece sencillo, el TSP es un problema NP-Completo [15], como el resto de los problemas que se mencionan en este trabajo, lo que significa que no existe un algoritmo eficiente conocido que pueda resolver todos los casos en tiempo polinómico.

  En la Ilustración 1 se puede apreciar que el VRP es una generalización del TSP, ya que mientras este último solo contiene una ruta en el VRP pueden encontrarse varias. Esto se demuestra en la descripción del subproblema BSS y las variantes VRP que aparecen luego de la figura.

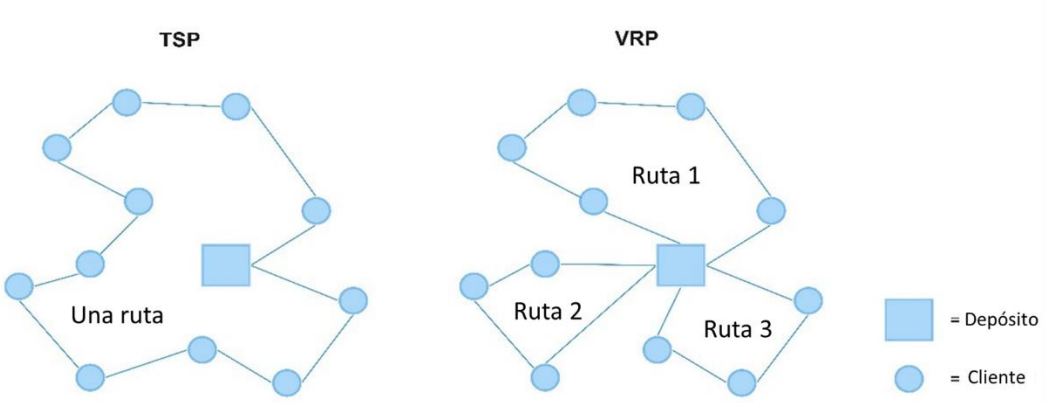

Ilustración 1: Representación gráfica de los problemas TSP y VRP [16].

- **Problema de Planificación de Rutas de Vehículos con Restricciones de Distancia (*Distance Vehicle Routing Problem*, DVRP)** [50]: el DVRP es una variante VRP donde cada vehículo tiene una restricción de distancia para realizar su ruta, satisfaciendo la demanda de los clientes y regresando al punto del que partió (depósito). Su objetivo, como el de todas las variantes VRP, generalmente es minimizar el costo total de las rutas, aunque pueden haber ocasiones en que se requiera maximizar algún aspecto.

- **Problema de Planificación de Rutas de Vehículos con Capacidad (*Capacitated Vehicle Routing Problem,* CVRP)** [4, 16]: el CVRP es una variante del VRP que añade una restricción adicional donde cada vehículo tiene una capacidad limitada que no puede ser excedida. Esto complica la búsqueda de la solución óptima y es crucial en situaciones donde los vehículos tienen limitaciones físicas o de carga.

  En la Ilustración 2 se muestra una representación gráfica de este problema; se visualizan 4 rutas con distintos colores, donde los números que se encuentran dentro de los nodos representan su identificador y los números externos la demanda en ese punto, la cual cada vehículo debe cumplir siempre y cuando se respete su restricción de capacidad máxima.

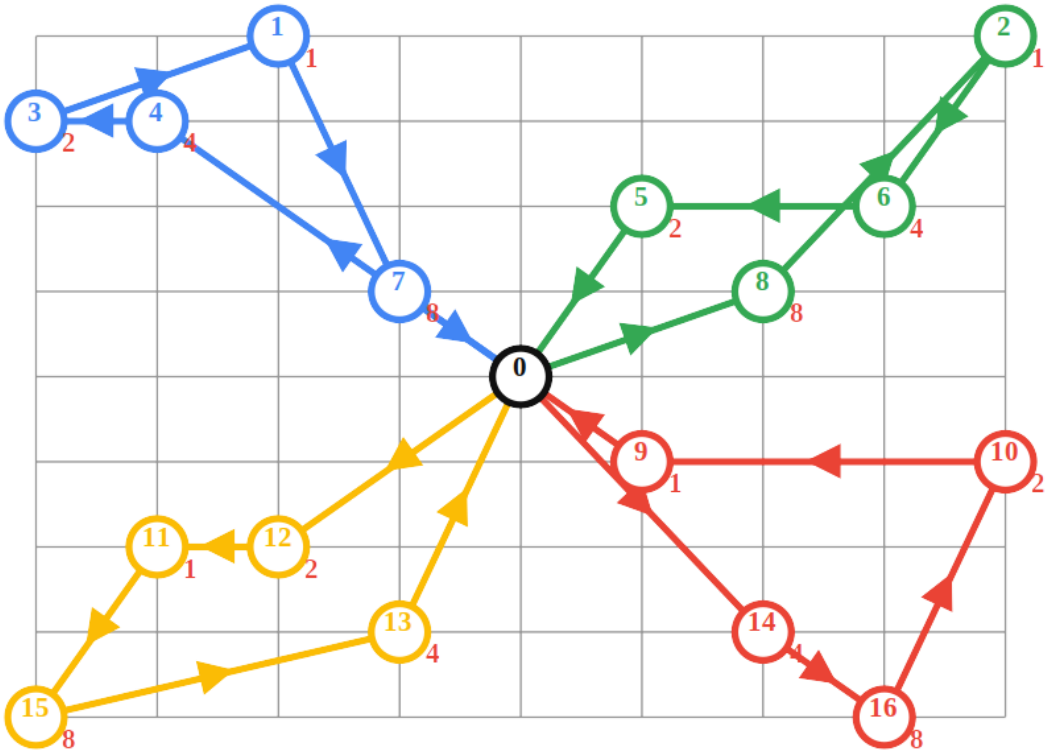

Ilustración 2: Representación gráfica del CVRP [13].

- **Problema de Rutas de Autobuses Escolares (*School Bus Routing Problem,* SBRP)** [17]: es un tipo específico de VRP que se centra en la planificación de las rutas de autobuses escolares. Su objetivo es diseñar rutas eficientes para recoger y dejar a los estudiantes en sus respectivas paradas, minimizando el costo total asociado con las rutas. Este costo puede incluir la distancia recorrida, el tiempo de viaje, la cantidad de autobuses utilizados, o una combinación de estos y otros factores.

- **Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos (*Multi-Depot Vehicle Routing Problem,* MDVRP)** [18]: el MDVRP es una variante VRP que considera múltiples depósitos desde los cuales los vehículos pueden comenzar y terminar sus rutas, al contrario del resto donde solo se toma en cuenta un depósito. Este problema es relevante para grandes empresas de logística que operan en diferentes regiones y necesitan optimizar las rutas considerando varios puntos de partida y llegada.

- **Problema de Planificación de Rutas de Vehículos con Ventanas de Tiempo (*Vehicle Routing Problem with Time Windows*, VRPTW)** [19]: el VRPTW es una variante VRP donde cada cliente tiene un intervalo de tiempo específico (ventana de tiempo) durante el cual debe ser atendido. Esto significa que los vehículos no solo deben planificar sus rutas para minimizar la distancia recorrida, sino también asegurarse de llegar a cada punto de entrega dentro de las horas indicadas por las ventanas de tiempo. Si un vehículo llega antes del inicio de la ventana de tiempo, deberá esperar hasta que esta comience. Si llega después de que la ventana de tiempo haya terminado, la entrega no será aceptada. En este punto la ruta será penalizada o replanificada. Esta variante es particularmente desafiante porque los vehículos deben cumplir con las restricciones temporales además de optimizar la eficiencia de la ruta, lo que añade una capa de complejidad significativa al problema.
- **Problema de Planificación de Rutas de Vehículos con Recogida y Entrega (*Vehicle Routing Problem with Pickup and Delivery*, VRPPD)** [20]: el VRPPD es una variante VRP que se ocupa de la optimización de rutas donde los vehículos deben recoger y entregar productos en diferentes ubicaciones. Este problema es común en servicios de mensajería y logística inversa. La logística inversa se refiere a la gestión y optimización del flujo de productos desde el consumidor final de vuelta al fabricante o al punto de reciclaje para su reutilización, reciclaje o adecuada disposición. En este contexto, los vehículos no solo entregan mercancías, sino que también recogen artículos de retorno o reciclaje. Esto añade una capa adicional de complejidad, ya que las rutas deben planificarse para cumplir con ambas tareas de manera eficiente, asegurando que los vehículos tengan suficiente capacidad para manejar tanto las entregas como las recogidas.
- **Subproblema de Selección de Paradas de Autobús (*Bus Stop Selection,* BSS)** [17]: es un proceso integral y crucial en la planificación y operación de sistemas de transporte público. Este proceso se enfoca en determinar las ubicaciones más adecuadas para las paradas de autobús dentro de una red de transporte, con el objetivo de optimizar la

accesibilidad para los pasajeros, la eficiencia del servicio y la seguridad. La selección de ubicaciones para las paradas debe basarse en criterios estratégicos que consideran la distribución geográfica de la población, la densidad urbana, y los puntos de interés como escuelas, hospitales, centros comerciales y áreas residenciales. Además, deben estar situadas a una distancia razonable para la mayoría de los pasajeros.

También existen otros problemas que no son considerados en el presente trabajo como:

- **Problema de Planificación de Rutas de Vehículos Estocástico (*Stochastic Vehicle Routing Problem,* SVRP)** [21]: variante VRP que considera la incertidumbre en los datos del problema. Los parámetros, como la demanda de los clientes, los tiempos de viaje o los costos, están sujetos a la aleatoriedad o incertidumbre. Esta incertidumbre puede deberse a factores como el tráfico variable, las condiciones climáticas cambiantes o la variabilidad en los tiempos de servicio de los clientes. Su objetivo es encontrar las rutas óptimas para una flota de vehículos dadas las condiciones estocásticas, maximizando la probabilidad de cumplir con los requisitos del servicio y minimizando los costos asociados.

- **Problema de Planificación de Rutas de Vehículos con Entregas Divididas (*Split Delivery Vehicle Routing Problem,* SDVRP)** [22]: variante del VRP que permite que un cliente sea atendido por más de un vehículo en una sola visita. Esto significa que un cliente puede recibir parte de su demanda de un vehículo y el resto de otro vehículo en la misma visita. El objetivo es encontrar las rutas óptimas para una flota de vehículos que minimicen los costos totales, considerando tanto la distancia recorrida como la capacidad de carga de los vehículos, mientras se cumplen todas las demandas de los clientes.

- **Problema de Planificación de Rutas de Camiones y Remolques (*Truck and Trailer Routing Problem*, TTRP)** [49]**:** esta variante considera el empleo de remolques como parte de la flota de vehículos. Además, presenta ciertas particularidades como la definición de tipos de

cliente debido a las restricciones de accesos existentes en las ubicaciones de los mismos. Existen clientes accedidos por camión y remolque, conocidos como clientes de vehículo completo (VC) y otros accedidos solamente por camión sin el remolque denominados clientes de camión puro (TC).

En la Ilustración 3 a continuación se resumen algunos de los problemas antes expuestos derivados del CVRP.

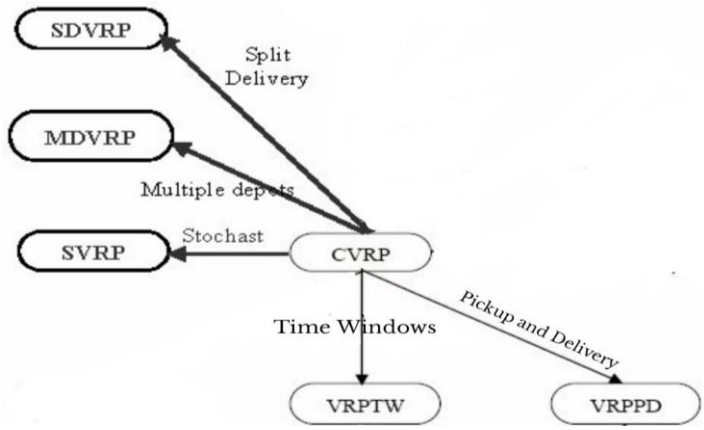

Ilustración 3: Resumen de los problemas derivados del CVRP [23].

La elección y aplicación de tipos de distancia juegan un papel crucial en la formulación y resolución de estos problemas. La distancia entre ubicaciones influye directamente en la optimización de rutas, afectando tanto la eficiencia operativa como los costos asociados. En este sentido, algunos tipos de distancia son:

- **Distancia Euclideana (*Euclidean*)** [54, 55]: es una medida directa y lineal entre dos puntos en un espacio euclidiano. Se define como la longitud del segmento que une dos puntos en un plano o en un espacio tridimensional. Matemáticamente, la distancia euclidiana entre dos puntos (x_1, y_1) y (x_2, y_2) en un plano se calcula utilizando el teorema de Pitágoras. La Ecuación 1 muestra la fórmula para este cálculo.

$$euclidean = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ecuación 1: Ecuación para el cálculo de la distancia euclideana.

Donde:

(x_1, y_1) => punto de partida en el eje de coordenadas (x, y)

(x_2, y_2) => punto de llegada en el eje de coordenadas (x, y)

- **Distancia Manhattan** [55]: también conocida como distancia rectilínea o distancia de la ciudad, mide la suma de las diferencias absolutas entre las coordenadas de dos puntos en un espacio de coordenadas rectangulares. Es llamada así por la disposición de las calles en el plano de Manhattan, que forman una cuadrícula ortogonal. Matemáticamente, la distancia de Manhattan entre dos puntos (x_1, y_1) y (x_2, y_2) se calcula como se muestra en la Ecuación 2.

$$manhattan = |x_2 - x_1| + |y_2 - y_1|$$

Ecuación 2: Ecuación para el cálculo de la distancia manhattan.

Donde:

(x_1, y_1) => punto de partida en el eje de coordenadas (x, y)

(x_2, y_2) => punto de llegada en el eje de coordenadas (x, y)

- **Distancia de Chebyshev** [56]: define la distancia entre dos puntos como la máxima diferencia entre sus coordenadas a lo largo de cualquier dimensión. Es útil en situaciones donde los vehículos pueden moverse libremente en cualquier dirección. La distancia de Chebyshev se utiliza en diversas aplicaciones, como en problemas de tableros de ajedrez, rutas de movimientos de robots, y en cualquier situación donde se requiera determinar la distancia máxima posible entre dos puntos en un espacio n-dimensional. Se calcula como aparece en la Ecuación 3.

$$chebyshev = \underset{i=1}{maxn}|p_i - q_i|$$

Ecuación 3: Ecuación para el cálculo de la distancia chebyshev.

Donde:

$p = (p_1, p_2, \ldots, p_n)$ => es un vector que representa las coordenadas del punto p en el espacio n-dimensional. Cada p_i (donde i va de 1 a n) es una coordenada del punto p.
$q = (q_1, q_2, \ldots, q_n)$ => es un vector que representa las coordenadas del punto q en el espacio n-dimensional. Cada q_i (donde i va de 1 a n) es una coordenada del punto q.

- **Distancia Haversine** [54]: tiene en cuenta la curvatura de la esfera terrestre y es particularmente útil para calcular distancias entre puntos geográficos cuando se trabaja con coordenadas de latitud y longitud. Es importante notar que la fórmula asume una esfericidad perfecta de la Tierra, por lo que para distancias muy grandes o precisas, se pueden usar aproximaciones más complejas como las que consideran el elipsoide de referencia de la Tierra. Se calcula como aparece en la Ecuación 4.

$$haversine = 2r \times arcsin$$

Ecuación 4: Ecuación para el cálculo de la distancia haversine.

Donde:
r => radio de la esfera (por ejemplo, el radio promedio de la Tierra)
$\Delta\varphi$ => diferencia de latitudes entre los dos puntos en radianes
$\Delta\lambda$ => diferencia de longitudes entre los dos puntos en radianes
φ_1, φ_2 => latitudes de los dos puntos en radianes

- **Distancia Minkowski** [56]: es una medida de distancia utilizada en geometría y análisis numérico para calcular la distancia entre dos puntos en un espacio n-dimensional. Esta distancia generaliza varias otras medidas de distancia comunes, como la distancia Euclidiana y la distancia Manhattan, dependiendo de un parámetro p. Es útil en problemas donde se desea ajustar la sensibilidad a diferentes dimensiones del espacio n-dimensional. Se calcula como se muestra en la Ecuación 5.

$$minkowski = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{\frac{1}{p}}$$

Ecuación 5: Ecuación para el cálculo de la distancia minkowski.

Donde:

p => es un parámetro que determina el tipo de distancia. Cuando p=1, la distancia es equivalente a la distancia de Manhattan. Cuando p=2, la distancia es equivalente a la distancia Euclidiana. Cuando $p \to \infty$, la distancia es equivalente a la distancia de Chebyshev.

$|p_i - q_i|$ => es el valor absoluto de la diferencia entre la coordenada p_i del punto p y la coordenada q_i del punto q.

- **Distancia Geodésica** [57]: en el contexto de teoría de grafos y redes se refiere a la distancia más corta entre dos nodos (vértices) en un grafo, medida a lo largo de los enlaces (aristas) que conectan estos nodos. Es decir, es la longitud mínima de cualquier camino que conecta dos nodos específicos en la red. Se calcula como aparece en la Ecuación 6.

$$geodésica = \min_{\gamma} \sum_{i=1}^{k-1} w(u_i, u_{i+1})$$

Ecuación 6: Ecuación para el cálculo de la distancia geodésica.

Donde:

w (u_i, u_i+1) => peso del borde entre los nodos u_i y u_i+1

k => número de bordes en el camino mínimo que conecta los nodos u y v

Durante años, la planificación de rutas para resolver estos problemas se ha realizado manualmente. Aunque este enfoque ha sido funcional hasta cierto punto, presenta varias limitaciones significativas en términos de eficiencia y efectividad. No permite una optimización adecuada de los recursos disponibles, ya que los planificadores deben tomar decisiones basadas en su experiencia y

juicio, lo que puede resultar en rutas poco eficientes y en la asignación inadecuada de vehículos a destinos específicos. Este proceso consume mucho tiempo, requiere la consideración de múltiples variables y la actualización constante de información. Además, la naturaleza manual del proceso lo hace propenso a errores humanos, lo que puede aumentar la posibilidad de retrasos y costos adicionales. Los errores en la planificación pueden resultar en tiempos de entrega prolongados y una mala asignación de recursos. Adicionalmente, carece de la flexibilidad necesaria para adaptarse rápidamente a cambios inesperados, como alteraciones en el tráfico, cambios en las demandas de los clientes o interrupciones en el servicio. Esto puede resultar en una menor capacidad para responder a contingencias y en una mayor ineficiencia operativa [5].

La solución a estas problemáticas reside en la automatización y optimización de la planificación de rutas mediante herramientas y algoritmos avanzados. La adopción de métodos automatizados ofrece ventajas significativas como [1, 2]:

- Se pueden considerar una amplia gama de variables y restricciones de manera simultánea, permitiendo una asignación más efectiva de los vehículos y la determinación de las rutas más eficientes. Esto resulta en una utilización óptima de los recursos y una reducción de los costos operativos.
- Elimina la mayoría de los errores humanos asociados con la planificación manual. Los algoritmos pueden procesar grandes volúmenes de datos de manera rápida y precisa, lo que reduce significativamente el tiempo requerido para planificar las rutas. Además, las soluciones automatizadas pueden actualizarse en tiempo real para reflejar cambios en las condiciones del tráfico o las demandas de los clientes.
- Al optimizar las rutas y minimizar los tiempos de viaje y el consumo de combustible, las soluciones automatizadas pueden reducir significativamente los costos asociados con el transporte. La mejora en la eficiencia operativa también puede traducirse en un mejor servicio al cliente y una mayor satisfacción de los usuarios finales.

Existen diversas herramientas y técnicas avanzadas muy utilizadas actualmente en este ámbito que logran lo descrito anteriormente. En el siguiente epígrafe se detallan y ejemplifican algunas de ellas.

1.3. Técnicas de optimización y trabajos previos

En esta sección se abordan las técnicas de optimización y algunos trabajos previos encontrados en la literatura. Primero, se describen las principales técnicas de optimización, incluyendo métodos exactos, heurísticas y metaheurísticas, destacando sus características y aplicaciones. Luego, se presentan estudios que han abordado los problemas de planificación de rutas de vehículos, mostrando cómo han sido aplicadas estas técnicas en diversos contextos y los resultados obtenidos.

1.3.1. Técnicas de optimización

En el ámbito de la optimización, existen actualmente diversas técnicas y algoritmos para encontrar soluciones eficientes a los problemas complejos descritos anteriormente, como los métodos exactos y los métodos heurísticos (heurísticas y metaheurísticas) [5, 24].

Los métodos exactos son técnicas utilizadas en la resolución de problemas de optimización que garantizan encontrar la solución óptima dentro de un espacio de búsqueda dado. Exploran exhaustivamente todas las posibles soluciones factibles y evalúan sistemáticamente cada una de ellas para determinar la mejor en términos de un criterio objetivo. A diferencia de las heurísticas, los métodos exactos garantizan la precisión al encontrar la solución global óptima para el problema en cuestión. Sin embargo, debido a la naturaleza de su enfoque, pueden ser computacionalmente costosos en términos de tiempo y recursos, especialmente para problemas grandes y complejos [5, 8].

Un ejemplo de estos métodos es el algoritmo de Ramificación y Acotamiento (*Branch and Bound*, B&B) [25], este busca la solución óptima explorando sistemáticamente todo el espacio de búsqueda utilizando una estrategia de dividir y conquistar, donde el espacio se divide en subconjuntos más pequeños (ramas) que se exploran recursivamente. A medida que se exploran las ramas, se realizan podas para evitar explorar regiones que no pueden contener soluciones óptimas. Esto se logra utilizando cotas inferiores y superiores para evaluar la calidad de una solución parcial y determinar si se debe explorar una rama específica [25].

En el caso de las heurísticas, consisten en buscar soluciones para problemas complejos en un tiempo razonable. Aunque no garantizan la obtención de la solución óptima, proporcionan soluciones suficientemente buenas con una eficiencia computacional aceptable [24]. Se dividen en diferentes categorías según su enfoque y función. Una de ellas son las heurísticas de construcción; se utilizan para construir una solución paso a paso, agregando elementos al problema hasta que se alcanza una solución completa. También se destacan las heurísticas de mejora u optimización; buscan mejorar una solución existente realizando cambios locales que conduzcan a una solución óptima [5, 7].

En el contexto de la optimización de rutas de vehículos, una heurística de construcción muy común es el Algoritmo del Vecino Más Cercano (*Nearest Neighbor Algorithm*) [6], el cual selecciona el nodo más cercano al nodo actual y repite este proceso hasta que se completa una ruta. Este método es apreciado por su simplicidad y rapidez, siendo una opción atractiva para la generación de soluciones iniciales. Sin embargo, su enfoque voraz puede llevar a soluciones que no sean óptimas debido a decisiones locales que no siempre resultan en la mejor solución global. Este algoritmo puede dejar nodos aislados, incrementando el costo total del recorrido. Otro ejemplo notable es la Heurística de Inserción Más Cercana (*Nearest Insertion Heuristic*) [20], que también construye rutas de manera iterativa, insertando en cada paso el nodo que minimiza el incremento de la distancia total del recorrido. A pesar de sus limitaciones, ambas heurísticas de construcción son útiles como puntos de partida efectivos para técnicas más avanzadas, como metaheurísticas y

heurísticas de mejora, que buscan mejorar las soluciones iniciales obtenidas [5].

Las metaheurísticas son estrategias de alto nivel diseñadas para guiar y modificar heurísticas, escapando de los óptimos locales y explorando el espacio de soluciones de manera más efectiva [9]. Algunas de las metaheurísticas más destacadas en la optimización de rutas de vehículos incluyen [5]:

- Búsqueda Local (*Local Search*) [20]: es una técnica que implica explorar el vecindario de una solución existente en busca de mejoras. En este enfoque, se realizan pequeños cambios o modificaciones en la solución actual con el objetivo de mejorarla. Por ejemplo, en el contexto del problema de planificación de rutas de vehículos, un cambio típico podría ser intercambiar la posición de dos clientes en una ruta para reducir la distancia total recorrida por los vehículos. La búsqueda local no garantiza encontrar la mejor solución global, pero puede conducir a soluciones de alta calidad de manera eficiente. Es especialmente útil en problemas combinatorios donde la búsqueda exhaustiva no es factible debido al tamaño del espacio de búsqueda.
- Algoritmo Genético (*Genetic Algorithm*) [27, 28]: inspirado en los principios de la evolución biológica, este algoritmo utiliza operadores como la selección, el cruce y la mutación para evolucionar una población de soluciones y mejorar iterativamente su calidad. Es eficaz para problemas complejos como el VRP debido a su capacidad para explorar un amplio espacio de soluciones.
- Recocido Simulado (*Simulated Annealing*) [29]: se basa en el proceso de recocido en metalurgia. Consiste en aceptar soluciones peores con una cierta probabilidad para evitar quedarse atrapado en óptimos locales. La probabilidad de aceptar peores soluciones disminuye con el tiempo, lo que permite una exploración inicial amplia y una posterior explotación de las mejores soluciones encontradas.
- Búsqueda Tabú *(Tabu Search*) [30]: se basa en una estructura de memoria llamada lista tabú. Esta lista almacena soluciones recientes y

prohíbe que el algoritmo vuelva a visitarlas durante un cierto período de tiempo, con esto se busca evitar ciclos y fomentar la exploración de nuevas áreas del espacio de soluciones.

- Enjambre de Partículas (*Particle Swarm*) [28]: basado en el comportamiento social de las aves y peces, este algoritmo optimiza un problema moviendo una población de soluciones (partículas) alrededor del espacio de soluciones de acuerdo con fórmulas matemáticas basadas en sus posiciones y velocidades.
- Colonia de Hormigas (*Ant Colony Optimization*) [31]: inspirada en el comportamiento de las hormigas en busca de alimento, este algoritmo utiliza un enfoque de búsqueda cooperativa donde las hormigas (agentes) depositan feromonas en las rutas elegidas, guiando a otras hormigas hacia soluciones prometedoras.

1.3.2. Estado del arte de la optimización de rutas de vehículos

En el ámbito de la optimización de rutas de vehículos, numerosos estudios y publicaciones han abordado la resolución de diversos problemas utilizando una amplia gama de técnicas y enfoques, como las comentadas en el epígrafe anterior.

Un ejemplo de esto es la solución al problema VRPPD planteada en el artículo titulado “Problema de ruteo de vehículos multi-objetivo con entregas y recogidas simultáneas y minimización de emisiones”. El artículo aborda la importancia del transporte en las cadenas de suministro, destacando su relevancia en términos de costos y su impacto ambiental. Se presenta un modelo VRP multi-objetivo, que considera aspectos como entregas y recogidas simultáneas, flota heterogénea, ventanas de tiempo y múltiples depósitos. Aunque se obtuvieron soluciones óptimas, se reconoce que los tiempos computacionales fueron altos debido a la complejidad del modelo, lo que sugiere la necesidad de desarrollar algoritmos aproximados. Se recomienda el uso de algoritmos híbridos que combinen métodos exactos y heurísticas, como

el algoritmo de búsqueda tabú o genéticos, para mejorar las soluciones iniciales. Además, se sugiere explorar otras variantes del problema VRP como MDVRP o VRPTW [32].

Otro trabajo encontrado en la investigación sobre el estado del arte es el titulado “Una metaheurística evolutiva híbrida paralela para el Problema de Planificación de Rutas de Vehículos con Ventanas de Tiempo” de Homberger y Gehring del año 1999. Este trabajo utiliza una estrategia de evolución (1, λ) en la primera fase para minimizar el número de vehículos, y una búsqueda tabú en la segunda fase para minimizar la distancia total de viaje. La paralelización se basa en el concepto de autonomía cooperativa, donde varias soluciones secuenciales autónomas cooperan intercambiando soluciones bajo ciertas condiciones de aceptación, generando mejores soluciones para instancias grandes del problema. En la primera fase, se generan soluciones iniciales y se mejora iterativamente la solución mediante operadores de movimiento y un operador Or-opt modificado. En la segunda fase, se toma la mejor solución de la primera fase y se optimiza la distancia total de viaje utilizando operadores de movimiento y una lista tabú para evitar ciclos y mejorar la exploración del espacio de soluciones. Este enfoque híbrido y paralelo permite optimizar tanto el número de vehículos como la distancia total de viaje de manera efectiva. Los resultados obtenidos para los seis grupos de problemas indican que la metaheurística evolutiva híbrida secuencial (ES) y sus variantes paralelas (ES4 y ES4C) muestran mejoras significativas en el número total de vehículos y la distancia total recorrida [33].

En la búsqueda sobre trabajos previos también se encontró la tesis titulada “Diseño de un algoritmo metaheurístico para la solución del Problema de Ruteo de Vehículos MultiDepósito”. El algoritmo implementado consta de dos fases y utiliza múltiples vecindades. La primera fase es una heurística constructiva que genera una solución inicial, la cual es luego mejorada en la segunda fase mediante el algoritmo de Recocido Simulado. Se utilizaron 23 instancias de referencia de la literatura, logrando obtener las soluciones óptimas en más del 50% de los casos. Las conclusiones de la tesis destacan que el algoritmo propuesto pudo resolver las instancias de referencia, encontrando la mejor solución conocida en más del 60% de los casos. En cuanto a futuros trabajos,

se sugiere implementar otros algoritmos metaheurísticos, como algoritmos genéticos o búsqueda tabú y abordar otras variantes del VRP con diferentes restricciones, como ventanas de tiempo o recolección y entrega [34].

Otra tesis hallada se titula “Implementación de un algoritmo heurístico de Recocido Simulado para el Problema de Enrutamiento de Vehículos con Capacidades Homogéneas”. Para abordar el CVRP, la tesis propone utilizar el algoritmo metaheurístico Recocido Simulado, utilizando tres vecindarios diferentes para mejorar la solución inicial generada aleatoriamente. La efectividad se evaluó empleando instancias conocidas de la literatura, que varían en tamaño desde 31 hasta 80 clientes. Los resultados de las pruebas mostraron que el algoritmo pudo igualar la solución óptima conocida en más del 60% de las instancias probadas, demostrando ser competitivo en comparación con otros métodos heurísticos. Se plantean varias líneas de investigación futura, como experimentar con instancias de mayor tamaño y explorar otras variantes del problema que incluyan múltiples depósitos o ventanas de tiempo [29].

Se identifica otro informe nombrado “Una implementación eficiente del algoritmo memético para TSP”. Este presenta una implementación del algoritmo memético para resolver el TSP en clústeres de computadoras, utiliza una estrategia de búsqueda local que combina características de 3-opt [36] y Lin-Kernighan [37], operando directamente sobre las permutaciones que representan los tours sin necesidad de traducirlas a conjuntos de aristas. La nueva implementación propuesta muestra una eficiencia mayor que trabajos anteriores, reduciendo los tiempos de ejecución en todas las pruebas realizadas. Se planea continuar usando arquitecturas paralelas diferentes y se considera necesario probar la implementación en instancias más grandes de TSPLIB para evaluar su desempeño en escenarios de mayor complejidad, así como lograr resolver otros tipos de problemas [35].

Para resumir los resultados de estos trabajos se presenta la Tabla 1 donde se muestra en la primera columna los informes consultados y en las siguientes los aspectos a evaluar según los resultados obtenidos en cada trabajo.

Tabla 1: Análisis comparativo de los trabajos estudiados del estado del arte.

Año del Trabajo - Referencia	Variante que resuelve	Obtención de soluciones óptimas en la mayoría de los casos	Tiempos y recursos computacionales bajos	Complejidad en el modelo
1999 - [33]	VRPTW	✓	✓	✓
2014 - [32]	VRPPD	✓	X	✓
2019 - [29]	CVRP	✓	✓	X
2021 - [34]	MDVRP	✓	X	✓
2024 - [35]	TSP	✓	✓	✓

Como se puede observar, gran parte de los trabajos lograron encontrar las soluciones óptimas en la mayoría de los casos (más del 50%), teniendo también buenos tiempos de ejecución e iterando entre modelos más o menos complejos. Sin embargo, tienen en común que resuelven una variante de VRP, proponiendo resolver otras en trabajos a futuro. Siendo este el propósito del presente trabajo; lograr la resolución de la mayoría de estas variantes, incluyendo todas las vistas en estos informes del estado del arte.

1.4. Herramientas de optimización

En el mundo actual, donde la eficiencia y la optimización son esenciales en una amplia gama de aplicaciones, las herramientas o bibliotecas de optimización desempeñan un papel crucial. Estas herramientas permiten a los investigadores, ingenieros y profesionales de diversas disciplinas resolver problemas complejos de manera más rápida y eficiente, encontrando soluciones óptimas o lo suficientemente buenas. Además, ofrecen una amplia gama de algoritmos y métodos, como programación lineal, programación entera, programación no lineal, programación cuadrática, algoritmos genéticos, optimización basada en heurísticas y metaheurísticas, entre otros. Los usuarios pueden formular sus problemas de optimización en términos matemáticos y

resolverlos de manera eficiente utilizando algoritmos avanzados. A continuación, se presentan algunos ejemplos de estas herramientas [38].

1.4.1. Xpress

Xpress, desarrollado por FICO, es una poderosa herramienta de optimización diseñada para resolver una amplia variedad de problemas en áreas industriales y académicas. Los algoritmos que ofrece son de alto rendimiento y se adaptan a diferentes tipos de problemas de optimización. Además, se integra fácilmente con sistemas y lenguajes de programación como Python, C, C++, C#, y Java, lo que permite su uso en diversos entornos y aplicaciones. Sin embargo, puede resultar costoso para usuarios con presupuestos limitados y tiene una curva de aprendizaje compleja. En la optimización de rutas de vehículos, Xpress permite modelar y resolver variantes del VRP, aunque su adquisición y uso implican costos, a diferencia de algunas bibliotecas gratuitas y de código abierto. Xpress ofrece varios tipos de distancia como Euclidean, Manhattan, Chebyshev y la Geodésica para la resolución de variantes VRP como CVRP, VRPPD, HVRP, VRPTW, MVRP, FCVRP, DVRP, SVRP, MDVRP, PVRP, LVRP y RVRP [50]. La documentación disponible de esta herramienta es exhaustiva y proporciona soporte técnico, pero su base de usuarios es más pequeña comparada con herramientas de código abierto [10].

1.4.2. PuLP

PuLP es una biblioteca de optimización de código abierto en Python que ofrece una interfaz fácil de usar para formular y resolver problemas de optimización lineal, entera y mixta. Su popularidad entre científicos de datos, ingenieros y profesionales de optimización se debe a su flexibilidad y simplicidad. Esta herramienta se integra perfectamente con Python y utiliza una sintaxis clara y sencilla. PuLP es lo suficientemente flexible para manejar diversos problemas de optimización y cuenta con documentación detallada y una comunidad activa

que comparte recursos y soluciones. Como biblioteca de código abierto, es gratuita para usar y modificar, sin embargo, puede tener un rendimiento inferior comparado con solvers de alto rendimiento, especialmente en problemas grandes o complejos. Para mejorar el rendimiento de los algoritmos, PuLP puede utilizar solvers externos como Xpress. Aunque permite modelar problemas VRP y la resolución efectiva de estos depende de la integración con los solvers. Se pueden configurar diferentes tipos de distancia como Euclidean, Manhattan y Chebyshev para la resolución de las variantes VRP mencionadas en Xpress [11].

1.4.3. SciPy

SciPy es una biblioteca de código abierto en Python que proporciona algoritmos y funcionalidades para realizar cálculos científicos y matemáticos. La biblioteca se encarga de la resolución de problemas poco complejos de forma general, no contiene módulos especializados para resolver problemas de optimización de rutas, es decir, carece de los algoritmos necesarios para abordar la complejidad de estos problemas. SciPy está construida sobre NumPy, otra biblioteca popular en Python para computación numérica. Sus usos fundamentales están en el área de la física, ingeniería, ciencias de datos y optimización. Los módulos que brinda están dedicados a la optimización, como `scipy.optimize`, que ofrece una variedad de algoritmos para resolver problemas de optimización no lineal, minimización de funciones y ajuste de curvas. Sin embargo, puede no ser tan eficiente en términos de rendimiento como otras bibliotecas. En su módulo `scipy.spatial.distance` ofrece varios tipos de distancia como Euclidean, Manhattan, Chebyshev, Minkowski, entre otras. Su documentación es amplia y compleja cubriendo diversos campos como álgebra lineal, estadísticas y procesamiento de señales, además del módulo de optimización. Aunque es popular y cuenta con una gran comunidad de usuarios y desarrolladores, algunas partes de la biblioteca pueden resultar difíciles de entender y utilizar, especialmente para usuarios principiantes. SciPy tiene varias dependencias, incluida NumPy [40], que deben instalarse correctamente

para su funcionamiento. La gestión de estas dependencias puede ser un desafío, especialmente al configurar el entorno de desarrollo en nuevas máquinas o sistemas. Scipy es útil principalmente para calcular matrices de distancia entre ubicaciones (usando funciones como pdist o cdist), que son esenciales para la formulación de restricciones de distancia en modelos de VRP. Para resolver problemas de VRP específicos, generalmente se usan bibliotecas de optimización especializadas en estos problemas [12].

1.4.4. BHCVRP

La Biblioteca de Heurísticas de Construcción para Problemas de Planificación de Rutas de Vehículos está desarrollada en el lenguaje de programación Java y se está desarrollando una versión en Python que ofrece un buen rendimiento. BHCVRP brinda la adaptación de siete heurísticas de construcción clásicas de la literatura, como son el Algoritmo de Barrido, el Algoritmo de Ahorros en su versión paralela y secuencial, la Heurística del Vecino más Cercano con Lista de Candidatos Restringidos, la Heurística de Mole & Jameson, la Heurística de Inserción de Christofides, Mingozzi y Toth (CMT) y un método aleatorio. Además, ofrece cuatro tipos de distancia (Euclidean, Manhattan, Haversine, Chebyshev) para el cáculo de la distancia en la resolución de variantes VRP como CVRP, HFVRP, MDVRP y TTRP [50]. A pesar de resolver algunos de los problemas de optimización de rutas utilizando heurísticas de construcción, no implementa metaheurísticas. Esto puede ser una desventaja al no abarcar todas las posibles técnicas para encontrar la solución óptima, mejorar la solución inicial y evitar quedarse en óptimos locales, teniendo en cuenta que las metaheurísticas presentan más posibilidades de lograr esto. Por otra parte, no existe una documentación detallada ni una gran comunidad de desarrolladores, además depende del componente BHAVRP para resolver la variante MDVRP [49].

1.4.6. ORTools

Google ORTools es una biblioteca de optimización completamente gratuita y de código abierto desarrollada por Google, diseñada para resolver una amplia gama de problemas de optimización combinatoria, incluidos los problemas de optimización de rutas de vehículos. Escrito principalmente en C++, ORTools también proporciona interfaces para varios lenguajes de programación como Python, C# y Java. Esta herramienta ofrece una variedad de algoritmos para problemas de programación lineal [41], programación entera [42], programación lineal entera mixta [43], entre otros. Es reconocida por su potencia y flexibilidad en la resolución de problemas de optimización en industrias como la logística y el transporte, gracias a su eficiencia y capacidad para manejar problemas de gran escala. Se destaca por su alto rendimiento, permitiendo resolver problemas complejos en tiempos razonables, y por su diversidad de algoritmos que facilitan a los usuarios elegir la mejor opción para su tarea específica. ORTools proporciona documentación detallada, ejemplos de uso y cuenta con una comunidad activa de usuarios que comparten conocimientos y experiencias. La biblioteca ofrece 2 tipos de distancia (Euclidean, Manhattan) para resolver el TSP y variantes VRP como CVRP, DVRP, VRPTW, MDVRP, VRPPD [50] y diferentes variantes de VRPTW con restricciones adicionales como tiempos de descanso, entre otros. Debido a su eficiencia y capacidad para manejar problemas de gran escala, ORTools es adecuada para aplicaciones en empresas de transporte y logística [13].

1.4.6. Comparación entre las herramientas

A modo de resumen sobre las ventajas y desventajas de estas herramientas, se presenta la Tabla 2 para una mejor comprensión. Esta tabla está confeccionada con los aspectos más importantes a considerar, con el objetivo de seleccionar la herramienta más adecuada para la investigación.

Tabla 2: Comparación entre las herramientas de optimización.

Herramienta	Gratuita	Resuelve problemas VRP por si sola	Implementa metaheurísticas	Buen rendimiento en problemas complejos	No depende de otras herramientas	Gran comunidad y documentación detallada
Xpress	X	✓	✓	✓	✓	X
PuLP	✓	X	X	X	X	✓
SciPy	✓	X	X	X	X	✓
BHCVRP	✓	✓	X	✓	X	X
ORTools	✓	✓	✓	✓	✓	✓

Todas resuelven la variante MDVRP, excepto SciPy, que no está diseñado para resolver problemas VRP. Cada una considera diferentes restricciones que se detallan en la Tabla 3 .

Tabla 3: Comparación de la configuración de restricciones para MDVRP en las herramientas de optimización.

Biblioteca	Distancia máxima de la ruta	Capacidad máxima de los vehículos	Demandas de los clientes
Xpress	✓	✓	✓
PuLP	✓	✓	✓
BHCVRP	X	✓	✓
ORTools	✓	X	X

Debido a las consideraciones expuestas, donde se analizan tanto ventajas como desventajas en las herramientas disponibles, se puede concluir que ORTools es la mejor opción. Esta biblioteca no solo es gratuita y de código abierto, sino que también ofrece algoritmos especializados en optimización de rutas de vehículos, con funcionalidades avanzadas y un alto rendimiento en problemas complejos. Además, la ausencia de dependencias adicionales, la gran comunidad de usuarios y la detallada documentación que presenta, hacen de ORTools una herramienta robusta. Esta herramienta logra solucionar todos los problemas planteados en los trabajos previos de la sección anterior, y no

solo uno en específico, cumpliendo así con las recomendaciones que se planteaban en esos informes, aunque en la variante MDVRP no presenta restricciones que el resto de herramientas si toma en cuenta como se muestra en la Tabla 3.

Sin embargo, en la búsqueda realizada en plataformas como ResearchGate [44], Google Scholar [45], SciELO [46] y el motor de búsqueda de Google, no se encontraron informes que proporcionaran una guía que refleje las fortalezas y debilidades que presenta la biblioteca en el contexto de la optimización de rutas de vehículos. Específicamente, un informe que indique en qué problemas VRP es adecuado utilizar ORTools y en cuáles no, considerando diversos factores como las restricciones específicas de cada problema, el tamaño de las instancias, la complejidad de los escenarios, el tiempo requerido para la ejecución de los algoritmos y la calidad de las soluciones obtenidas. Este análisis es crucial para maximizar el aprovechamiento de la biblioteca en la resolución de problemas optimización de rutas.

1.5. Biblioteca ORTools

La biblioteca ORTools (*Operations Research Tools*) de Google implementa varias arquitecturas, principios y patrones de diseño para proporcionar una plataforma robusta y extensible para la optimización de problemas combinatorios, en esta sección se profundiza en esos aspectos [13].

1.5.1. Arquitectura

Arquitectura Modular: está diseñada de manera modular, lo que permite a los usuarios utilizar solo los componentes necesarios para su problema específico. Los módulos principales incluyen la planificación de rutas de vehículos (VRP), la programación lineal y mixta, la programación de restricciones (CP), y la optimización de flujo de redes, entre otros [13].

1. **Solver**: el corazón de ORTools es el solver, que se encarga de resolver los problemas de optimización. Existen diferentes solvers para distintos tipos de problemas:
    - Linear Solver: para problemas de programación lineal (LP) y programación lineal entera mixta (MILP).
    - Constraint Solver: para problemas de programación de restricciones (CP).
    - Routing Solver: específicamente diseñado para problemas de planificación de rutas de vehículos como el VRP.
2. **Model Builder:** es responsable de la creación y gestión de los modelos de optimización. Permite a los usuarios definir variables de decisión, restricciones y funciones objetivo de manera estructurada y abstracta.
3. **Search Strategies**: ORTools implementa diversas estrategias de búsqueda y heurísticas para encontrar soluciones iniciales y mejorar las soluciones existentes. Algunas de estas incluyen:
    - First Solution Strategies: estrategias para encontrar soluciones iniciales, como las heurísticas PATH_CHEAPEST_ARC, SAVINGS, SWEEP, entre otras.
    - Local Search Metaheuristics: metaheurísticas para mejorar soluciones, como Guided Local Search, Simulated Annealing, entre otras.

Abstracción y Encapsulación: utiliza capas de abstracción para separar la lógica de alto nivel de la implementación de bajo nivel. Esto permite que los usuarios trabajen con modelos matemáticos sin preocuparse por los detalles de implementación.

1. **Modelo de Datos:** utiliza un modelo de datos abstracto para representar problemas de optimización. Esto incluye:
    - Variables de Decisión: representan las decisiones que se deben tomar (rutas de vehículos, asignación de recursos).

- Restricciones: reglas que las soluciones deben cumplir (capacidad de vehículos, tiempos de entrega).
- Función Objetivo: la función que se desea optimizar (minimizar la distancia total recorrida).

2. **Gestor de Índices**: el gestor de índices abstrae los detalles del mapeo entre los índices de los nodos y las variables del modelo. Esto es particularmente útil en problemas de planificación de rutas de vehículos, donde facilita la conversión entre los índices internos del solver y los nodos del problema.

Interfaz Común: proporciona una interfaz común para diferentes tipos de problemas de optimización. Esto significa que los mismos conceptos y patrones pueden ser aplicados a distintos tipos de problemas, facilitando la curva de aprendizaje.

1.5.2. Patrones y principios de diseño

Los principios y patrones de diseño son fundamentales para desarrollar sistemas de software eficientes y mantenibles [47, 48].

Los principios de diseño, como SOLID, establecen pautas generales que ayudan a los desarrolladores a crear código limpio y estructurado. Estos principios promueven la cohesión, el acoplamiento débil y la flexibilidad en el diseño de software. ORTools implementa los siguientes principios [47]:

- **Principio de Responsabilidad Única (*Single Responsibility Principle*)**: ORTools proporciona una serie de módulos y clases que están diseñados para cumplir una función específica en el proceso de resolución de problemas de optimización. Por ejemplo, contiene módulos para representar problemas de planificación de rutas de vehículos, problemas de programación lineal, entre otros. Cada uno de estos módulos tiene una responsabilidad única y claramente definida en el proceso general de resolución del problema.

- **Principio de Abierto/Cerrado (*Open/Closed Principle*)**: ORTools está diseñado para ser una biblioteca extensible que permite a los usuarios agregar nuevas funcionalidades o adaptar las existentes sin necesidad de modificar el código fuente original. Esto se logra mediante una arquitectura bien definida que utiliza interfaces y patrones de diseño, lo que facilita la extensión de la funcionalidad sin modificar el código existente.
- **Principio de Inversión de Dependencia (*Dependency Inversion Principle*)**: ORTools utiliza una arquitectura que permite la inyección de dependencias y el desacoplamiento de componentes. Esto significa que los módulos y clases están diseñados para depender de abstracciones en lugar de implementaciones concretas. Por ejemplo, los algoritmos de búsqueda local pueden ser intercambiados fácilmente sin modificar el código cliente, lo que permite una mayor flexibilidad y extensibilidad.
- **Principio de Substitución de Liskov (*Liskov Substitution Principle)***: la arquitectura de ORTools permite que las clases derivadas sean sustituibles por sus clases base sin afectar el comportamiento del programa. Esto significa que los algoritmos de resolución y las estructuras de datos proporcionadas por ORTools pueden ser utilizados de manera intercambiable en el código cliente, lo que facilita la composición y la reutilización de código.

Por otro lado, los patrones de diseño son soluciones probadas para problemas comunes en el diseño de software. Ofrecen estructuras y mecanismos reutilizables que permiten a los desarrolladores resolver problemas de diseño específicos de manera eficiente como: la creación de objetos, la composición de clases y la comunicación entre objetos. ORTools implementa los siguientes patrones [48]:

1. **Patrón Fábrica (*Factory Pattern)***: utilizado para crear instancias de modelos de optimización y otras estructuras de datos. Por ejemplo, la creación de variables de decisión y restricciones en la programación de restricciones.

2. **Patrón Estrategia (*Strategy Pattern*)**: permite seleccionar entre diferentes algoritmos de búsqueda y metaheurísticas en tiempo de ejecución. Los usuarios pueden elegir estrategias de solución inicial y metaheurísticas según las necesidades del problema.
3. **Patrón Singleton *(Singleton Pattern*)**: utilizado en componentes que deben tener una única instancia global, como la configuración global del solver o ciertos gestores de recursos.
4. **Patrón Observador (*Observer Pattern*)**: implementado para notificar a los componentes sobre cambios en el estado del problema o en las soluciones durante el proceso de búsqueda.
5. **Patrón Constructor (*Builder Pattern*)**: utilizado para construir instancias complejas de modelos de optimización de manera paso a paso, facilitando la configuración y personalización.

1.6. Conclusiones parciales

En este capítulo se proporcionó un análisis exhaustivo de la historia, los conceptos fundamentales y los antecedentes de la optimización de rutas.

- Existen distintas formas de resolver los problemas de optimización de rutas de vehículos como métodos exactos, heurísticas y metaheurísticas.
- La optimización de rutas de vehículos es importante en una variedad de aplicaciones prácticas (logística, transporte, distribución y planificación de rutas de entrega) porque logra disminuir los costos y el uso eficiente de los recursos, al contrario de los resultados de realizar la planificación de forma manual.
- Existen diversos enfoques y técnicas utilizadas en trabajos anteriores para resolver estos problemas con el uso de algoritmos heurísticos y metaheurísticas.

- ORTools ofrece flexibilidad para ajustar modelos según necesidades específicas, una amplia documentación que facilita su implementación y emplea técnicas avanzadas para encontrar soluciones óptimas o cercanas a la óptima, lo que lo convierte en una herramienta poderosa para resolver problemas de planificación de rutas de vehículos.
- Es fundamental disponer de una guía que explore las ventajas y desventajas de la biblioteca ORTools en el contexto de la optimización de rutas de vehículos.
- ORTools implementa una arquitectura modular, así como diversos patrones y principios de diseño fundamentales para la creación de sistemas eficientes, robustos y mantenibles.

Capítulo 2: Guía para la evaluación de ORTools

2.1. Introducción

Este capítulo aborda la implementación de técnicas de optimización para resolver problemas de planificación de rutas de vehículos, utilizando la biblioteca ORTools.

Se presentan las heurísticas y metaheurísticas disponibles en la biblioteca, así como un diagrama de flujo del proceso de resolución descrito detalladamente. Además, se muestra un diagrama de clases donde se reflejan los paquetes implementados y la relación entre ellos. Se describen los patrones de diseño utilizados en la implementación de la solución para facilitar la adaptabilidad, flexibilidad y extensibilidad. Por último, se evalúa la eficacia de las técnicas que ofrece ORTools en diversas instancias de los problemas y se proporcionan recomendaciones sobre su aplicación.

2.2. Heurísticas y metaheurísticas en ORTools

En el campo de la optimización, las heurísticas y metaheurísticas son métodos aproximados que juegan un papel crucial al proporcionar soluciones eficientes a problemas complejos que son difíciles de resolver de manera exacta en un tiempo razonable. ORTools, la potente biblioteca de optimización de Google mencionada en el capítulo anterior, incorpora una amplia gama de estas técnicas para abordar problemas de optimización combinatoria como problemas de optimización de rutas de vehículos [9].

En este epígrafe, se exploran las diversas heurísticas y metaheurísticas que están integradas en ORTools para proporcionar soluciones eficientes a los

problemas de optimización de rutas de vehículos, adaptándose a diferentes escenarios y restricciones específicas.

2.2.1. Heurísticas

Las heurísticas son estrategias que buscan soluciones factibles rápidamente, aunque no necesariamente óptimas [3]. ORTools implementa varias heurísticas deterministas que se utilizan como puntos de partida para mejorar la eficiencia de las soluciones, por ejemplo:

- **PATH_CHEAPEST_ARC (PCA)**: esta heurística inicia una ruta desde un nodo de inicio y lo conecta al nodo que produce el segmento de ruta más barato. Posteriormente, extiende la ruta iterando sobre el último nodo añadido, siempre eligiendo la conexión más económica. Es una técnica de adición de caminos que busca minimizar el costo de cada segmento de la ruta en cada paso.
- **PATH_MOST_CONSTRAINED_ARC (PMCA)**: similar a PCA, pero esta heurística favorece los arcos más restringidos primero. Utiliza un selector basado en comparaciones que evalúa las restricciones de los arcos, priorizando aquellos con mayores restricciones.
- **EVALUATOR_STRATEGY (ES)**: esta estrategia también se basa en la extensión de la ruta con el arco más barato, pero utiliza una función evaluadora específica para calcular los costos de los arcos. La función evaluadora permite una evaluación personalizada de los costos según el modelo de enrutamiento.
- **SAVINGS (SV)**: el algoritmo de ahorro de Clarke & Wright combina rutas para reducir el costo total. Inicia con cada cliente como una ruta individual y luego combina las rutas que resultan en los mayores "ahorros" de costo, es decir, la mayor reducción en la distancia total recorrida al unir dos rutas en lugar de mantenerlas separadas.

- **SWEEP (SW)**: el algoritmo de barrido ordena los nodos según su ángulo polar alrededor del depósito central. Una vez ordenados, construye rutas en secuencia, asignando los nodos cercanos en el orden de su ángulo polar. Este enfoque es especialmente útil para problemas con nodos distribuidos radialmente alrededor de un punto central.
- **CHRISTOFIDES (CH)**: utiliza una variante del algoritmo de Christofides [20], que originalmente se diseñó para el Problema del Vendedor Viajero (TSP) métrico. En el contexto de problemas de planificación de rutas de vehículos, ORTools extiende una ruta inicial hasta que no se pueden insertar más nodos. A diferencia del algoritmo clásico de Christofides que garantiza un factor de aproximación de 3/2 para el TSP métrico, la variante de ORTools emplea una coincidencia máxima en lugar de una coincidencia mínima, lo que no asegura ese factor de aproximación pero permite su aplicación en modelos de planificación de rutas de vehículos más generales, donde se pueden tener restricciones adicionales o diferentes tipos de nodos y costos asociados.
- **ALL_UNPERFORMED (AU)**: esta heurística desactiva todos los nodos inicialmente y busca una solución donde los nodos son opcionales, es decir, pueden ser atendidos o no según las restricciones de problema. Es útil en situaciones donde no todos los nodos deben ser necesariamente visitados. Esto se aplica típicamente cuando algunos clientes tienen restricciones específicas de disponibilidad o preferencia, o cuando se desea minimizar el número total de nodos visitados mientras se optimiza otra métrica como el costo total o la distancia recorrida.
- **BEST_INSERTION (BI)**: construye una solución iterativamente insertando el nodo más barato en su posición más barata. La elección del nodo y su posición se basa en la función de costo global del modelo de planificación de rutas de vehículos. Este método es efectivo en modelos con nodos opcionales que tienen costos de penalización finitos.
- **PARALLEL_CHEAPEST_INSERTION (PCI)**: similar a BI, pero en lugar de considerar solo la función de costo global, se basa en los costos de

los arcos para insertar el nodo más barato en su posición más económica. Es más rápido que BI debido a su enfoque paralelo.

- **SEQUENTIAL_CHEAPEST_INSERTION (SCI)**: construye soluciones de manera secuencial, creando rutas una por una. Para cada ruta, inserta el nodo más barato en su posición más económica hasta completar la ruta. Utiliza los costos de los arcos para determinar la mejor posición de inserción, siendo más rápido que PCI.
- **LOCAL_CHEAPEST_INSERTION (LCI)**: inserta nodos iterativamente en su posición más barata considerando la distancia. Los nodos se seleccionan en orden decreciente de distancia al inicio o final de las rutas, insertando primero los nodos más lejanos. Este enfoque es rápido y se enfoca en minimizar las distancias locales.
- **LOCAL_CHEAPEST_COST_INSERTION (LCCI)**: similar a LCI, pero la inserción se basa en la función de costo del modelo de planificación de rutas de vehículos en lugar de solo los costos de los arcos. Esto permite una evaluación más completa de los costos al decidir dónde insertar cada nodo.
- **GLOBAL_CHEAPEST_ARC (GCA)**: conecta iterativamente los dos nodos que producen el segmento de ruta más barato en cada paso. Es una heurística basada en variables que se enfoca en minimizar el costo global de la ruta seleccionando siempre el arco más económico disponible.
- **LOCAL_CHEAPEST_ARC (LCA)**: selecciona el primer nodo con un sucesor no asignado y lo conecta al nodo que produce el segmento de ruta más barato. Este enfoque simple y local asegura que cada paso minimice el costo del segmento de ruta inmediatamente siguiente.
- **FIRST_UNBOUND_MIN_VALUE (FUMV)**: selecciona el primer nodo con un sucesor no asignado y lo conecta al primer nodo disponible. Combina las estrategias CHOOSE_FIRST_UNBOUND y ASSIGN_MIN_VALUE, asignando valores mínimos en cada paso para construir una solución inicial rápidamente.

2.2.2. Metaheurísticas

Las metaheurísticas son técnicas de búsqueda de alto nivel que guían otras heurísticas para explorar el espacio de soluciones de manera más efectiva, buscando soluciones óptimas o casi óptimas [9]. ORTools implementa varias metaheurísticas robustas como:

- **GREEDY_DESCENT (GD)**: esta metaheurística determinista acepta vecinos de búsqueda local que mejoran (reducen el costo) iterativamente hasta alcanzar un mínimo local. Cada paso de la búsqueda local selecciona la solución adyacente que ofrece la mayor reducción en el costo, continuando este proceso hasta que no se puedan encontrar otras mejoras inmediatas.
- **GUIDED_LOCAL_SEARCH (GLS)**: utiliza la búsqueda local guiada para escapar de los mínimos locales. Introduce penalizaciones dinámicas para ciertas características de la solución, guiando la búsqueda hacia áreas menos exploradas del espacio de soluciones. Esta técnica no determinista es especialmente eficiente para problemas de planificación de rutas de vehículos, ya que ayuda a evitar que la búsqueda se estanque en soluciones que no sean óptimas.
- **SIMULATED_ANNEALING (SA)**: emplea el recocido simulado para escapar de los mínimos locales. Este método no determinista simula el proceso de enfriamiento de un metal, permitiendo movimientos a soluciones peores con una probabilidad que disminuye con el tiempo. Esto permite explorar soluciones que no sean óptimas inicialmente para evitar caer en mínimos locales, buscando así una mejor solución global.
- **TABU_SEARCH (TS)**: utiliza la búsqueda tabú para escapar de los mínimos locales. Este algoritmo no determinista utiliza una lista de movimientos prohibidos (tabú) para evitar ciclos y fomentar la exploración de nuevas áreas del espacio de soluciones. Los movimientos recientes se agregan a esta lista, evitando que la búsqueda vuelva a soluciones recientemente exploradas, lo que facilita la búsqueda de soluciones mejores.

- **GENERIC_TABU_SEARCH (GTS)**: aplica la búsqueda tabú sobre una lista específica de variables para escapar de los mínimos locales. La lista de variables a utilizar permite una mayor flexibilidad en la aplicación de la búsqueda tabú, enfocándose en variables específicas que pueden influir significativamente en la calidad de la solución. Es un algoritmo no determinista.

2.3. Estructura del proyecto

Esta sección abarca la estructura del proyecto, proporcionando un análisis detallado de las tareas realizadas de acuerdo con el flujo general del procedimiento, y una representación exhaustiva de las clases y métodos utilizados en la solución propuesta. Se explica el funcionamiento de cada uno de los módulos o paquetes del proyecto y sus métodos internos, así como las relaciones existentes entre ellos.

2.3.1. Flujo general

Para la resolución de los problemas de planificación de rutas de vehículos utilizando ORTools, se siguen una serie de pasos estructurados. Estos pasos van desde la obtención de los datos de las instancias de cada problema hasta la devolución de los resultados en una estructura legible y fácil de adaptar para su visualización con otras herramientas. Los pasos son:

1. **Obtención de datos:** se adquieren los datos de las instancias del problema que se va a resolver desde archivos con extensión .txt. Estos datos suelen contener información sobre las ubicaciones de los nodos, las demandas de los clientes, las capacidades de los vehículos, las ventanas de tiempo en cada nodo, entre otros, dependiendo del tipo de problema.

2. **Importación y adaptación de datos:** los datos obtenidos se importan en el entorno de desarrollo y se adaptan a la estructura que requiere la biblioteca ORTools para el procesamiento eficiente de las instancias. En este caso es un diccionario con claves como "*demands*" (las demandas de los clientes), "*locations*" (coordenadas X y Y de los nodos), "*vehicles_capacities*" (capacidades de los vehículos), entre otros.

3. **Llamada a módulos de procesamiento:** se utilizan los módulos "*routing_enums_pb2*" y "*pywrapcp*" de "*constraint_solver*" en el procesamiento de los datos necesarios para resolver el problema. El módulo "*routing_enums_pb2*" es un enumerado que contiene todas las metaheurísticas y heurísticas que ofrece la biblioteca. Mientras que "*pywrapcp*" se encarga de la definición del modelo de planificación de rutas de vehículos, la configuración de los parámetros de búsqueda y la gestión de las soluciones encontradas.

4. **Aplicar las técnicas de optimización:** se aplican las heurísticas y metaheurísticas seleccionadas para la resolución del problema. Como primera estrategia de solución se escoge una heurística de las que brinda ORTools y luego una metaheurística para mejorar la solución obtenida.

5. **Obtención de soluciones:** mediante el uso de las metaheurísticas y heurísticas implementadas en la biblioteca, se procesan los datos de las instancias para encontrar soluciones óptimas o cercanas a la óptima según el objetivo definido (minimizar costos, tiempo, distancia).

6. **Exportación de los resultados:** una vez obtenidas las soluciones, se devuelven en una estructura adecuada que permita su fácil interpretación y visualización. Esta estructura es un archivo con extensión .txt que contiene datos como las rutas obtenidas, la distancia recorrida por cada ruta, la distancia total del recorrido y el resultado de la función objetivo.

En la Ilustración 4 se muestra el flujo completo de la resolución de problemas de planificación de rutas vehículos con ORTools, teniendo en cuenta todos los puntos mencionados anteriormente.

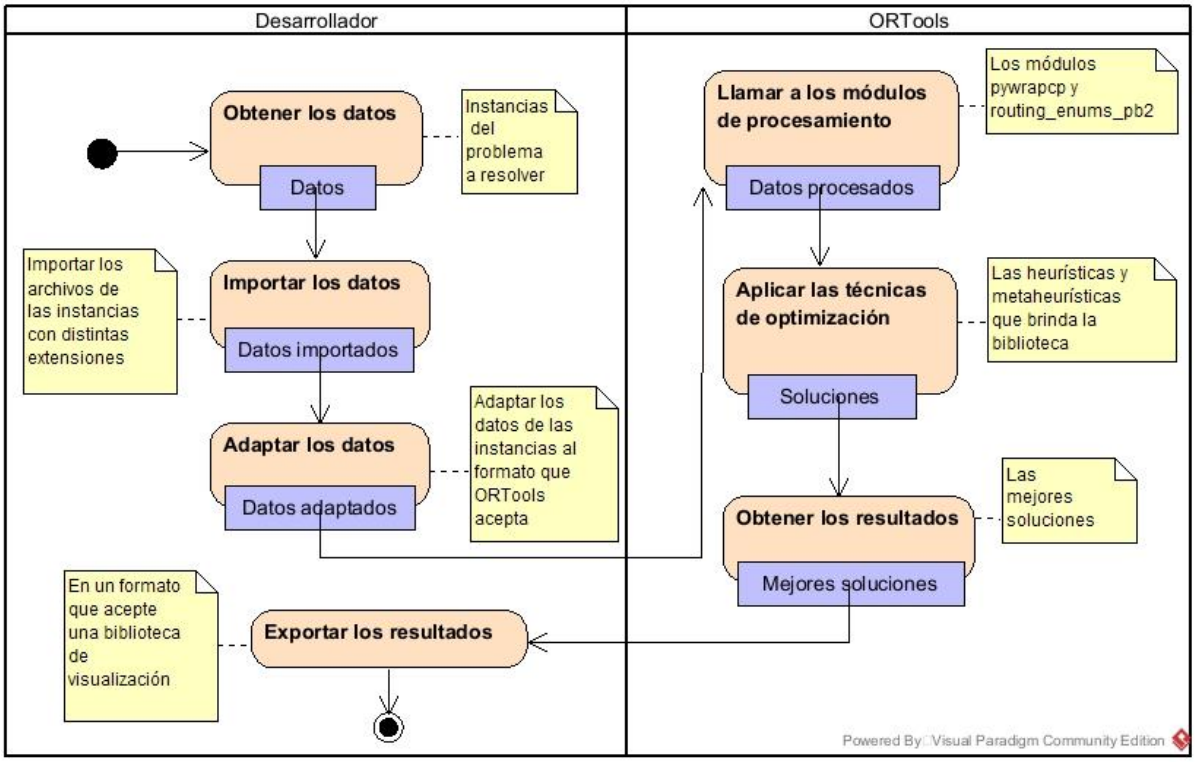

Ilustración 4: Diagrama de actividades del proceso de resolución de los problemas de optimización de rutas de vehículos con ORTools.

2.3.2. Diagrama de clases de la solución

Para la realización de la solución propuesta, se implementaron diversas clases, paquetes y métodos que trabajan en conjunto para asegurar su correcto funcionamiento e interactuar con la biblioteca ORTools. A continuación, se explica detalladamente el funcionamiento de cada elemento.

LoadData: es el paquete encargado de manejar y procesar todos los datos de cada uno de los problemas abordados.

- **InstanceType**: es una clase que hereda de la clase de Python llamada Enum. Contiene un enumerado de los tipos de instancias para ejecutar cada problema de planificación de rutas de vehículos. Esta clase contiene un método que recibe como argumento el tipo de instancia que se quiere cargar y devuelve los datos procesados.

- **ImportData**: se encarga de leer cada tipo de instancia utilizando métodos específicos para cada una. Devuelve los datos de la instancia procesados y adaptados a la estructura correcta que acepta la biblioteca ORTools.

Problems: es el paquete encargado de manejar cada uno de los problemas de planificación de rutas de vehículos.

- **ProblemType**: es una clase que hereda de la clase de Python llamada Enum. Contiene un enumerado de todos los problemas abordados (DVRP, CVRP, VRPTW, VRPMD, VRPPD). Presenta un método que recibe como parámetro el tipo de problema que se desea ejecutar y devuelve los resultados.
- **TSP, DVRP, VRPPD, VRPTW, MDVRP, CVRP**: módulos encargados de manejar su problema. Contienen una función que llama a un método de **InstanceType** para obtener los datos de la instancia a resolver para cada problema. La función resuelve el problema teniendo en cuenta sus restricciones y datos específicos. Llama a los módulos de ORTools para implementar las técnicas de optimización. La solución obtenida se le pasa como argumento a otra función para guardar los resultados en un archivo .txt.

Distances: paquete encargado de manejar los tipos de distancia a calcular para la resolución de los problemas.

- **DistanceType**: es una clase que hereda de la clase de Python llamada Enum. Contiene un enumerado de todos los tipos de distancia abordados (Euclidean, Manhattan, Haversine, Chebyshev). Implementa un método al que se le pasa el tipo de distancia que se quiere calcular y este devuelve el resultado.

ORTools: biblioteca que contiene todos los módulos requeridos para resolver los problemas de optimización de rutas de vehículos (*constraint_solver* y *routing_enums_pb2*).

Run: clase encargada de llamar a la función para resolver la variante VRP pasándole como parámetro el problema que se desea resolver.

Con el propósito de comprender mejor lo mencionado anteriormente, se presenta en la Ilustración 5 el diagrama de clases utilizado para resolver las variantes VRP utilizando la biblioteca ORTools. En el diagrama, los paquetes coloreados en rosado representan las clases y métodos implementados para la interacción con ORTools. El paquete en color morado abarca todo lo relacionado con la biblioteca ORTools, mientras que el paquete en azul celeste representa el módulo de Python llamado Enum, el cual se utiliza en varias de las clases para enumerar y gestionar opciones.

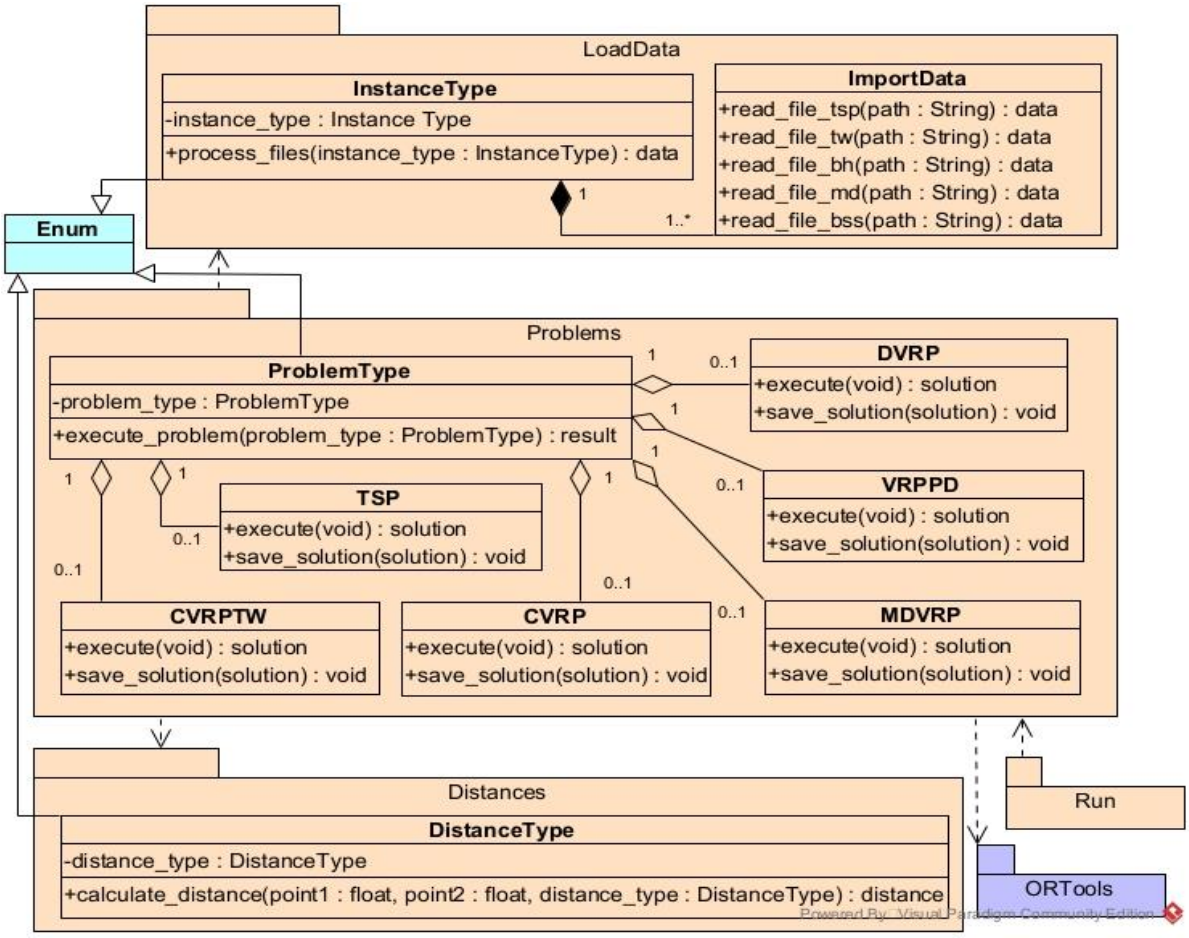

Ilustración 5: Diagrama de clases para la resolución de los problemas de optimización de rutas de vehículos con la biblioteca ORTools.

2.4. Evaluación de efectividad

Se lleva a cabo un experimento implementando diversas combinaciones de las 15 heurísticas y 5 metaheurísticas disponibles en ORTools, con el objetivo de analizar su eficiencia en la resolución de diferentes problemas de optimización (TSP, DVRP, CVRP, VRPTW, VRPPD, MDVRP). Los tipos de distancia utilizados para el cálculo de la distancia de un nodo a otro son Euclidean y Manhattan. Se le asignan identificadores a cada una de las heurísticas y metaheurísticas para su fácil identificación en las tablas de los resultados obtenidos, estos se encuentran en la sección 2.2 donde se explica el funcionamiento de cada una.

A continuación, se presenta el análisis de las heurísticas y metaheurísticas aplicadas en 67 instancias con diferentes tamaños y configuraciones para reutilizar algunas en varios problemas. El objetivo es **minimizar la distancia total** recorrida por las rutas y evaluar el **tiempo de ejecución (segundos)** del algoritmo. Se implementa inicialmente una heurística como estrategia inicial, seguida por la ejecución de una metaheurística para mejorar la solución. Se configura un tiempo máximo de ejecución de **50 segundos** establecido para todos los experimentos, dado que la biblioteca generalmente logró encontrar la mejor solución en menos tiempo. Las conclusiones se derivan de los mejores resultados obtenidos en **10 ejecuciones**, el resto de soluciones se encuentran en Google Drive en el siguiente enlace:

https://drive.google.com/drive/folders/1x0qD0wlnAdd-ohs1VUScC0ylReMJDFOS?usp=drive_link

2.4.1. TSP

En esta sección se presentan los resultados del experimento para la variante TSP. En la Tabla 4 se describen las 13 instancias del problema utilizadas para el experimento, recopiladas de [52].

Tabla 4: Descripción de las instancias TSP.

Instancia	Tamaño	Rango de coordenadas X [mínimo, máximo]	Rango de coordenadas Y [mínimo, máximo]
ulysses16	16	[-5.21, 41.23]	[9.10, 26.15]
ulysses22	22	[33.48, 41.23]	[-5.21 , 26.15]
att48	48	[10, 7762]	[140, 5184]
st70	70	[5, 98]	[1, 100]
pr76	76	[200, 19800]	[800, 12200]
rd100	100	[13.601, 967.650]	[0.998, 979.771]
pr107	107	[8175, 16425]	[4700, 11500]
pr124	124	[4475, 13585]	[1800, 11225]
ch150	150	[10.026, 697.388]	[0.420, 699.538]
d493	493	[3.64360e+03, 1.11630e+03]	[3.64360e+03, 9.52000e+02]
d657	657	[875.10, 3116.60]	[739.00, 2999.70]
u724	724	[605.61, 3075.81]	[707.7, 2444.39]
dsj1000	1000	[-66860, 1079810]	[-10984, 1086172]

En la Tabla 5 se presentan los mejores resultados obtenidos al realizar el experimento con las instancias descritas, utilizando la distancia euclideana. Mientras que en la Tabla 6 se muestran los resultados utilizando la distancia manhattan.

Tabla 5: Mejores resultados de las instancias TSP utilizando la distancia euclideana.

Instancia	Heurística	Metaheurística	Distancia total	Tiempo de ejecución
ulysses16	ES	GLS	72	0.0026
ulysses22	PCA	GLS	72	0.0053
att48	PCA	GD	34 184	0.0601
st70	PCA	GTS	682	0.1232
pr76	PCI	GTS	108 879	0.1412
rd100	PCA	GD	8 221	0.2450

pr107	PCA	GD	44 573	0.2952
pr124	BI	GTS	59 246	0.2858
ch150	PCA	GD	6 733	0.7677
d493	PCA	TS	35 969	20.0003
d657	PCA	GD	51 611	20.0018
u724	PCA	GD	44 270	20.0009
dsj1000	GCA	GTS	20 461 588	15.0019

Rendimiento en instancias de tamaño menor a 50 (3 instancias):

- La metaheurística GLS resultó ser la que mejores resultados brindó en combinación con las heurísticas PCA y ES, aunque PCA también se destacó con la metaheurística GD.

Rendimiento en instancias de tamaño entre 51 y 200 (6 instancias):

- Las metaheurísticas GTS y GD prevalecen en combinación con la heurística PCA.
- Para las instancias `pr76` y `pr124` se obtuvo la mejor solución con las heurísticas PCI y BI, respectivamente, en combinación con la metaheurística GTS.

Rendimiento en instancias de tamaño mayor a 200 (4 instancias):

- La metaheurística GD predomina en combinación con la heurística PCA.
- Las metaheurísticas TS y GTS en combinación con las heurísticas PCA y GCA, respectivamente, lograron las mejores soluciones en las instancias `d493` y `dsj1000`.

En general, se destaca la heurística PCA para todos los tamaños de instancia evaluados junto a las metaheurísticas GTS y GD, y para algunas instancias GLS.

Tabla 6: Mejores resultados de las instancias TSP utilizando la distancia manhattan.

Instancia	Heurística	Metaheurística	Distancia total	Tiempo de ejecución
ulysses16	PMCA	GTS	80	0.0037
ulysses22	LCI	TS	81	0.0054
att48	SCI	TS	42 594	0.0677
st70	PCA	GD	872	0.1296
pr76	LCA	GLS	136 538	0.1295
rd100	PCA	TS	10 280	0.4742
pr107	PCA	SA	50 500	0.2542
pr124	PCA	TS	68 860	0.4479
ch150	PCA	GD	8 229	0.5504
d493	GLA	TS	44 628	13.0226
d657	SV	GLS	63 858	47.6351
u724	PCI	GTS	53 369	48.0367
dsj1000	FUMV	GD	24 485 240	60.0012

Rendimiento en instancias de tamaño menor a 50 (3 instancias):

- La metaheurística TS resultó ser la que mejores resultados brindó en combinación con las heurísticas LCI y SCI, aunque PMCA también se destacó con la metaheurística GTS en la instancia `ulysses16`.

Rendimiento en instancias de tamaño entre 51 y 200 (6 instancias):

- Las metaheurísticas TS, SA y GD prevalecen en combinación con la heurística PCA.
- Para las instancias `st70` y `pr76` se obtuvo la mejor solución con las heurísticas SCI y LCA, en combinación con las metaheurísticas TS y GLS, respectivamente.

Rendimiento en instancias de tamaño mayor a 200 (4 instancias):

- Las metaheurísticas GLS, GD, TS y GTS en combinación con las heurísticas SV, FUMV, GLA y PCI, respectivamente, lograron las mejores soluciones en este grupo de instancias.

En general, se destaca la heurística PCA junto a las metaheurísticas TS y GD en la mayoría de las instancias evaluadas.

2.4.2. DVRP

En esta sección se presentan los resultados del experimento para la variante DVRP. En la Tabla 7 se describen las 10 instancias del problema utilizadas para el experimento obtenidas de [53]. Para cada instancia se mantienen los clientes, cantidad de vehículos y sus capacidades con sus datos originales, sólo se adaptan para trabajar con un único depósito ya que originalmente son instancias para múltiples depósitos.

Como parámetros fijos configurados por defecto se tiene:

- **slack_max:** define la cantidad máxima de tiempo de espera (o slack) permitida en la ruta de un vehículo = (0).
- **fix_start_cumul_to_zero:** indica si el valor acumulado de la distancia para cada vehículo comenzará en cero = (True).
- **capacity:** máxima distancia que deben recorrer los vehículos = (500).

Tabla 7: Descripción de las instancias DVRP.

Instancia	Tamaño	Rango de coordenadas X [mínimo, máximo]	Rango de coordenadas Y [mínimo, máximo]	Cantidad de vehículos
CVRP_100	15	[5.0, 52.0]	[16.0, 64.0]	15
p15	48	[-91.943, 57.404]	[-55.811, 99.341]	4
CVRP_1	50	[5.0, 63.0]	[6.0, 69.0]	5
p1	50	[5, 63]	[6, 69]	4

CVRP_4	75	[6.0, 70.0]	[4.0, 76.0]	9
p3	75	[6, 75]	[5, 76]	5
p4	100	[2, 100]	[3, 77]	2
p5	100	[2, 100]	[2, 77]	2
p6	100	[2, 100]	[3, 77]	4
p19	144	[-91.669, 82.794]	[-88.501, 84.296]	6

En la Tabla 8 se presentan los mejores resultados obtenidos al realizar el experimento con las instancias descritas, utilizando la distancia euclideana. Mientras que en la Tabla 9 se presentan los resultados utilizando la distancia manhattan.

Tabla 8: Mejores resultados de las instancias DVRP utilizando la distancia euclideana.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
CVRP_100	PCA	GD	247	1	0.0160
p15	PCA	GD	1 103	4	0.0231
CVRP_1	LCI	TS	463	2	0.0234
p1	LCI	TS	495	2	0.0343
CVRP_4	SV	SA	662	2	0.0412
p3	LCI	TS	689	2	0.0123
p4	LCCI	TS	713	2	0.1822
p5	LCCI	TS	719	2	0.1391
p6	LCA	GTS	815	2	0.1432
p19	SV	SA	1 803	4	0.1654

Rendimiento en instancias de tamaño menor a 51 (4 instancias):

- Dentro de este grupo, las combinaciones más eficientes resultaron ser las heurísticas PCA y LCI con las metaheurísticas GD y TS, respectivamente.

Rendimiento en instancias de tamaño entre 51 y 200 (6 instancias):

- Las heurísticas LCCI y SV son las predominantes, utilizadas en combinación con las metaheurísticas TS y SA, respectivamente. Excepto en la instancia `p6` que se destaca la heurística LCA con la metaheurística GTS.

En general, para las instancias proporcionadas, las heurísticas PCA, LCI y SV en combinación con las metaheurísticas GD, TS y SA fueron las más eficientes en todas las categorías de tamaño.

Tabla 9: Mejores resultados de las instancias DVRP utilizando la distancia manhattan.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
CVRP_100	PCI	TS	306	2	0.0239
p15	SV	TS	1 342	3	0.1023
CVRP_1	LCA	GTS	652	3	0.0923
p1	LCCI	TS	706	4	0.1129
CVRP_4	CH	GD	814	2	0.1234
p3	CH	GD	836	2	0.1345
p4	CH	GD	930	2	0.0876
p5	LCCI	TS	910	2	0.0989
p6	LCI	TS	982	3	0.1078
p19	SV	SA	2 368	5	0.1264

Rendimiento en instancias de tamaño menor a 51 (4 instancias):

- Dentro de este grupo, las combinaciones más eficientes resultaron ser las heurísticas PCI, SV y LCCI con la metaheurística TS. Excepto en la instancia `CVRP_1` donde la heurística LCA destaca junto a la metaheurística GTS.

Rendimiento en instancias de tamaño entre 51 y 200 (6 instancias):

- La combinación más efectiva resultó ser la heurística CH junto a la metaheurística GD. En las instancias `p5` y `p6` se destaca la metaheurística TS junto a las heurísticas LCCI y LCI, respectivamente.

En general, las combinaciones que más resaltan son la heurística CH y la metaheurística GD, así como la metaheurística TS junto a las heurísticas LCCI, LCI, SV y PCI.

2.4.3. CVRP

En esta sección se presentan los resultados del experimento para la variante CVRP. En la Tabla 10 se describen las 20 instancias del problema utilizadas para el experimento. De estas, 10 fueron obtenidas de [33] donde originalmente fueron utilizadas para VRPTW. En este experimento se ignoran las ventanas de tiempo para poder utilizar las instancias en CVRP.

Las otras 10 se recopilaron de [53] con ligeras transformaciones. Para cada instancia se mantienen los clientes, cantidad de vehículos y sus capacidades, sólo se adaptan para trabajar con un único depósito. En el caso particular de las instancias HFVRP (últimas 5 instancias en las tablas), para cada instancia se mantienen los clientes y cantidad de vehículos con sus datos originales, mientras que sus capacidades se distribuyen de manera aleatoria entre los vehículos, siempre respetando la capacidad total original. Con este procedimiento se obtiene una flota de vehículos con varios tipos de vehículos diferentes según su capacidad.

Como parámetros fijos configurados por defecto se tiene:

- **slack_max:** define la cantidad máxima de tiempo de espera (o slack) permitida en la ruta de un vehículo = (0).
- **fix_start_cumul_to_zero :** indica si el valor acumulado de la distancia para cada vehículo comenzará en cero = (True).

Tabla 10: Descripción de las instancias CVRP.

Instancia	Tamaño	Rango de coordenadas X [mínimo, máximo]	Rango de coordenadas Y [mínimo, máximo]	Rango de demandas [mínimo, máximo]	Capacidad de los vehículos	Cantidad de vehículos
CVRP_1	50	[5, 63]	[6, 69]	[3, 41]	200	5
CVRP_4	75	[6, 70]	[4, 76]	[1, 37]	250	9
CVRP_p3	75	[6, 76]	[4, 76]	[1, 37]	480	3
CVRP_p5	100	[2, 101]	[3, 77]	[1, 41]	400	5
CVRP_p14	360	[-160, 160]	[-160, 160]	[1, 12]	500	5
C1_2_1	200	[0, 139]	[0, 139]	[10, 40]	200	50
C1_2_4	200	[1, 137]	[3, 140]	[10, 40]	200	50
C1_2_5	200	[0, 139]	[0, 139]	[10, 40]	200	50
C1_2_6	200	[0, 139]	[0, 139]	[10, 40]	200	50
C1_2_8	200	[0, 139]	[0, 139]	[10, 40]	200	50
C1_4_1	400	[1, 200]	[0, 200]	[10, 40]	200	100
C1_4_4	400	[100, 180]	[50, 150]	[5, 50]	200	100
C1_6_1	600	[0, 287]	[0, 292]	[10, 40]	200	150
C1_6_2	600	[0, 287]	[4, 292]	[10, 40]	200	150
C1_6_3	600	[0, 287]	[4, 294]	[10, 40]	200	150
HFVRP_1	50	[5, 63]	[6, 69]	[3, 41]	200, 400, 100, 250, 50	5
HFVRP_4	75	[6, 70]	[4, 76]	[1, 37]	250, 100, 455, 90, 250, 50, 100, 500, 455	9
HFVRP_p3	75	[6, 76]	[4, 76]	[1, 37]	500, 470, 470	3
HFVRP_p5	100	[2, 101]	[3, 77]	[1, 41]	400, 100, 200, 650, 650	5
HFVRP_p14	360	[-160, 160]	[-160, 160]	[1, 12]	500, 600, 400, 500, 500	5

Se presentan los mejores resultados obtenidos al realizar el experimento con las instancias descritas con tamaños que varían entre 50 y 600 nodos, así como capacidades homogéneas (primeras 15 instancias) y heterogéneas (últimas 5 instancias). Se utiliza la distancia euclideana en la Tabla 11 y la distancia manhattan en la Tabla 12.

Tabla 11: Mejores resultados de las instancias CVRP utilizando la distancia euclideana.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
CVRP_1	PCA	SA	2 152	4	0.1320
CVRP_4	FUMV	TS	3 916	6	0.3265
CVRP_p3	ES	GLS	4 133	3	0.2628
CVRP_p5	PCA	SA	4 534	4	0.5498
CVRP_p14	FUMV	GTS	9 300	4	14.4430
C1_2_1	CH	TS	2 793	18	1.5212
C1_2_4	BI	TS	2 793	18	1.5450
C1_2_5	LCCI	TS	2 793	18	1.5280
C1_2_6	AU	SA	2 793	18	1.5388
C1_2_8	LCA	GLS	2 793	18	1.5303
C1_4_1	ES	GTS	7 245	36	8.6883
C1_4_4	PMCA	GLS	7 245	36	8.9911
C1_6_1	PMCA	GD	14 148	56	19.0884
C1_6_2	PMCA	GD	14 148	56	19.1798
C1_6_3	FUMV	SA	14 148	56	19.0917
HFVRP_1	PCA	TS	3 225	4	0.1427
HFVRP_4	PCA	GLS	3 730	4	0.2995
HFVRP_p3	PCA	GD	4 030	3	0.2144
HFVRP_p5	PCA	TS	1 512	3	0.6948
HFVRP_p14	SV	SA	9 325	4	12.6994

Rendimiento en instancias con capacidades homogéneas (15 instancias):

- En instancias de tamaño entre 50 y 100 se destaca la combinación de la heurística PCA y la metaheurística SA. Excepto en las instancias `CVRP_4` y `CVRP_p3` donde resaltan las heurísticas FUMV y ES con las metaheurísticas TS y GLS, respectivamente.
- En instancias de tamaño entre 200 y 360 se destaca la metaheurística TS en combinación con las heurísticas CH, BI y LCCI en 3 instancias. En las otras instancias la combinación de las heurísticas FUMV, AU y LCA

con las metaheurísticas GTS, SA y GLS, respectivamente, lograron las mejores soluciones.

- En instancias de tamaño entre 400 y 600 resalta la heurística PMCA con las metaheurísticas GLS y GD. Excepto en las instancias `C1_4_1` y `C1_6_3` donde las combinaciones efectivas resultaron ser con las heurísticas ES y FUMV junto a las metaheurísticas GTS y SA, respectivamente.

Rendimiento en instancias con capacidades heterogéneas (5 instancias):

- En este grupo se destaca la heurística PCA con varias metaheurísticas como TS, GLS y GD, excepto en la instancia `HFVRP_p14` donde la mejor combinación resultó ser la heurística SV con la metaheurística SA.

En general, las heurísticas más destacadas fueron PCA, PMCA y FUMV, mientras que las metaheurísticas fueron GLS, TS y SA.

Tabla 12: Mejores resultados de las instancias CVRP utilizando la distancia manhattan.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
CVRP_1	PCA	SA	2 972	4	0.1075
CVRP_4	PMCA	GLS	5 526	6	0.1359
CVRP_p3	AU	GLS	5 726	3	0.2703
CVRP_p5	PMCA	GD	6 322	4	0.4790
CVRP_p14	SV	SA	12 580	4	6.1363
C1_2_1	PCA	GLS	3 684	18	1.5836
C1_2_4	PCA	GLS	3 684	18	1.5775
C1_2_5	PCA	GLS	3 684	18	1.5804
C1_2_6	PCA	GLS	3 684	18	1.5866
C1_2_8	PCA	GLS	3 684	18	1.5789
C1_4_1	CH	GLS	9 112	36	7.0906
C1_4_4	ES	GTS	9 112	36	7.1388
C1_6_1	PCA	SA	18 760	56	17.7215
C1_6_2	PCA	SA	18 760	56	17.6531

C1_6_3	PCA	SA	18 760	56	17.5731
HFVRP_1	SCI	GLS	3 676	4	0.2021
HFVRP_4	PCA	GD	5 086	5	0.3728
HFVRP_p3	ES	SA	5 612	3	0.2440
HFVRP_p5	SV	GTS	1 938	3	0.4945
HFVRP_p14	PCA	GLS	12 660	4	6.3818

Rendimiento en instancias con capacidades homogéneas (15 instancias):

- En instancias de tamaño entre 50 y 100 se destaca la combinación de las heurísticas PMCA y AU con la metaheurística GLS. Excepto en las instancias `CVRP_1` y `CVRP_p5` donde resaltan las heurísticas PCA y PMCA con las metaheurísticas SA y GD, respectivamente.
- En instancias de tamaño entre 200 y 360 se destaca la metaheurística GLS en combinación con la heurística PCA. Excepto en la instancia `CVRP_p14` donde resalta la heurística SV con la metaheurística SA.
- En instancias de tamaño entre 400 y 600 resalta la heurística PCA con la metaheurística SA. Excepto en las instancias `C1_4_1` y `C1_4_4` donde las combinaciones efectivas resultaron ser con las heurísticas CH y ES junto a las metaheurísticas GLS y GTS, respectivamente.

Rendimiento en instancias con capacidades heterogéneas (5 instancias):

- En este grupo se destaca la heurística PCA con las metaheurísticas GLS y GD.
- También se destacan las heurísticas SCI, ES y SV en combinación con las metaheurísticas GLS, SA y GTS, respectivamente.

En general, las heurísticas más destacadas fueron PCA, PMCA y SV, mientras que las metaheurísticas fueron GLS y SA.

2.4.4. VRPTW

En esta sección se presentan los resultados del experimento para la variante VRPTW. En la Tabla 13 se describen las 23 instancias del problema utilizadas para el experimento, obtenidas de [33]. Como parámetros fijos configurados por defecto se tiene:

- **máxima velocidad de los vehículos** = 83.33 km/h
- **máxima distancia que un vehículo debe recorrer** = 1000
- **máximo tiempo en que un vehículo debe completar su ruta** = 1500 minutos

Tabla 13: Descripción de las instancias VRPTW.

Instancia	Tamaño	Rango de coordenadas X [mínimo, máximo]	Rango de coordenadas Y [mínimo, máximo]	Rango de demandas [mínimo, máximo]	Capacidad de los vehículos	Cantidad de vehículos	Rango de intervalo en ventanas de tiempo
C1_2_1	200	[0, 139]	[0, 139]	[10, 40]	200	50	[1, 1100]
C1_2_2	200	[0, 139]	[0, 139]	[10, 40]	200	50	[32, 1158]
C1_2_3	200	[1, 139]	[0, 139]	[10, 40]	200	50	[62, 1113]
C1_2_4	200	[1, 137]	[3, 140]	[10, 40]	200	50	[59, 711]
C1_2_5	200	[0, 139]	[0, 139]	[10, 40]	200	50	[50, 200]
C1_2_6	200	[0, 139]	[0, 139]	[10, 40]	200	50	[53, 1153]
C1_2_7	200	[0, 139]	[0, 139]	[10, 40]	200	50	[4, 1609]
C1_2_8	200	[0, 139]	[0, 139]	[10, 40]	200	50	[71, 1183]
C1_2_9	200	[0, 139]	[0, 139]	[10, 40]	200	50	[0, 1110]
C1_2_10	200	[1, 137]	[0, 139]	[10, 40]	200	50	[2, 1122]
C1_4_1	400	[1, 200]	[0, 200]	[10, 40]	200	100	[30, 1192]
C1_4_2	400	[3, 200]	[0, 198]	[10, 40]	200	100	[28, 1242]
C1_4_3	400	[3, 197]	[1, 198]	[10, 50]	200	100	[71, 545]
C1_4_4	400	[100, 180]	[50, 150]	[5, 50]	200	100	[1, 10]
C1_4_5	400	[3, 200]	[0, 198]	[10, 40]	200	100	[1, 1150]
C1_4_6	400	[0, 200]	[0, 200]	[10, 40]	200	100	[2, 934]
C1_4_7	400	[0, 200]	[0, 200]	[10, 40]	200	100	[18, 1118]
C1_4_8	400	[0, 200]	[0, 198]	[10, 50]	200	100	[17, 1025]
C1_4_9	400	[3, 200]	[0, 198]	[10, 50]	200	100	[4, 1351]

C1_4_10	400	[0, 200]	[0, 200]	[10, 50]	200	100	[1, 1280]
C1_6_1	600	[0, 287]	[0, 292]	[10, 40]	200	150	[1, 1150]
C1_6_2	600	[0, 287]	[4, 292]	[10, 40]	200	150	[2, 1394]
C1_6_3	600	[0, 287]	[4, 294]	[10, 40]	200	150	[10, 1403]

En la Tabla 14 se presentan los mejores resultados obtenidos al realizar el experimento con las instancias descritas, utilizando la distancia euclideana. Mientras que en la Tabla 15 se muestran los resultados utilizando la distancia manhattan,

Tabla 14: Mejores resultados de las instancias VRPTW utilizando la distancia euclideana.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
C1_2_1	PMCA	GD	2 990	21	4.9714
C1_2_2	PCI	GTS	2 761	20	13.3402
C1_2_3	SW	GD	2 828	19	10.6413
C1_2_4	PMCA	GTS	2 815	20	10.9510
C1_2_5	SW	GTS	3 105	21	10.2025
C1_2_6	SV	TS	2 859	21	7.2663
C1_2_7	SV	GD	2 811	21	8.6596
C1_2_8	BI	GLS	2 708	20	7.6467
C1_2_9	PMCA	GD	2 963	20	12.4238
C1_2_10	GCA	GTS	2 861	20	9.6781
C1_4_1	PCA	GD	7 242	40	39.4239
C1_4_2	BI	TS	7 427	40	55.6553
C1_4_3	LCI	GLS	7 711	41	56.3412
C1_4_4	LCCI	GD	7 491	38	54.8672
C1_4_5	GCA	TS	7 440	41	46.3954
C1_4_6	FUMV	GTS	7 478	42	59.1284
C1_4_7	PCI	GLS	7 477	40	41.9400
C1_4_8	LCA	GTS	7 557	42	44.9334
C1_4_9	CH	GD	7 521	39	47.6487
C1_4_10	LCCI	GLS	7 343	38	53.3252

C1_6_1	BI	GLS	15 364	64	60.0050
C1_6_2	LCCI	TS	16 759	63	60.0042
C1_6_3	FUMV	TS	15 778	59	60.0049

Rendimiento en instancias de tamaño 200 (10 instancias):

- Predominan las metaheurísticas GTS y GD en combinación con las heurísticas SV, SW y PMCA.

Rendimiento en instancias de tamaño 400 (10 instancias):

- Se destacan variadas combinaciones, como las heurísticas LCCI, FUMV, PCI, PCA y las metaheurísticas GD, GLS, GTS. Gran variedad de heurísticas resultaron eficientes para este grupo de instancias.

Rendimiento en instancias de tamaño 600 (3 instancias):

- Se destaca la metaheurística TS junto a las heurísticas LCCI y FUMV. Excepto en la instancia `C1_6_1`, donde resalta la combinación de BI como heurística y GLS como metaheurística.

En general, las combinaciones más efectivas consisten en utilizar las heurísticas PMCA, BI, LCCI, SW, SV y FUMV junto a las metaheurísticas GLS, GTS, GD y TS.

Tabla 15: Mejores resultados de las instancias VRPTW utilizando la distancia manhattan.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
C1_2_1	CH	SA	3 844	20	5.6651
C1_2_2	SCI	TS	4 036	21	8.0696
C1_2_3	ES	GLS	4 006	20	8.9046
C1_2_4	AU	TS	3 754	19	14.8680
C1_2_5	SCI	TS	3 760	21	7.6596
C1_2_6	PMCA	GD	4 058	22	6.6393
C1_2_7	SV	GD	3 988	22	7.0812
C1_2_8	LCI	GD	3 802	20	10.4968

C1_2_9	PMCA	GD	3 834	19	7.2653
C1_2_10	PCA	GTS	3 826	19	11.0672
C1_4_1	GCA	TS	9 784	42	55.2870
C1_4_2	BI	SA	10 164	43	52.5815
C1_4_3	BI	SA	10 124	39	50.3343
C1_4_4	LCI	GTS	9 576	37	55.3368
C1_4_5	PCA	SA	10 044	43	38.3173
C1_4_6	SCI	TS	10 304	43	44.6753
C1_4_7	BI	GLS	10 116	43	55.2357
C1_4_8	GCA	GLS	10 038	41	46.3659
C1_4_9	LCI	TS	9 926	39	38.3050
C1_4_10	ES	GLS	9 716	38	44.8220
C1_6_1	LCI	GTS	20 492	63	60.2175
C1_6_2	LCA	GD	21 500	65	60.0049
C1_6_3	LCCI	GD	21 164	60	60.0042

Rendimiento en instancias de tamaño 200 (10 instancias):

- Predominan las metaheurísticas TS y GD en combinación con las heurísticas SCI y PMCA, respectivamente.
- En las instancias `C1_2_1`, `C1_2_2` y `C1_2_10` se destacaron las heurísticas CH, SCI y PCA junto a las metaheurísticas SA, TS, GTS.

Rendimiento en instancias de tamaño 400 (10 instancias):

- Se destacan variadas combinaciones, como las metaheurísticas TS, GLS y SA junto a las heurísticas BI, GCA, LCI.
- En la instancia `C1_4_4` se destacó la heurística LCI junto a la metaheurística GTS.

Rendimiento en instancias de tamaño 600 (3 instancias):

- Se destaca la metaheurística GD junto a las heurísticas LCCI y LCA. Excepto en la instancia `C1_6_1`, donde resalta la combinación de LCI como heurística y GTS como metaheurística.

En general, las combinaciones más efectivas consisten en utilizar las heurísticas SCI, LCI, BI junto a las metaheurísticas GLS, GD y TS.

2.4.5. MDVRP

En esta sección se presentan los resultados del experimento para la variante MDVRP. En la Tabla 16 se describen las 8 instancias del problema utilizadas para el experimento recopiladas de [53], no se hace ninguna modificación. Como parámetros fijos configurados por defecto se tiene:

- **slack_max:** define la cantidad máxima de tiempo de espera (o slack) permitida en la ruta de un vehículo = (0).
- **fix_start_cumul_to_zero:** indica si el valor acumulado de la distancia para cada vehículo comenzará en cero = (True).
- **capacity:** máxima distancia que deben recorrer los vehículos = (500).

Tabla 16: Descripción de las instancias MDVRP.

Instancia	Tamaño	Rango de coordenadas X [mínimo, máximo]	Rango de coordenadas Y [mínimo, máximo]	Cantidad de vehículos	Cantidad de depósitos
p31	10	[-64.709, 72.095]	[-72.894, 66.498]	2	2
p15	48	[-91.943, 57.404]	[-55.811, 99.341]	4	4
p1	50	[5, 63]	[6, 69]	4	4
p2	50	[5, 63]	[6, 69]	4	4
p3	75	[6, 75]	[5, 76]	5	5
p5	100	[2, 100]	[2, 77]	2	2
p6	100	[2, 100]	[3, 77]	4	4
p19	144	[-91.669, 82.794]	[-88.501, 84.296]	6	6

En la Tabla 17 se presentan los mejores resultados obtenidos al realizar el experimento con las instancias descritas, utilizando la distancia euclideana.

Mientras que en la Tabla 18 se muestran los resultados utilizando la distancia manhattan.

Tabla 17: Mejores resultados de las instancias MDVRP utilizando la distancia euclideana.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
p31	PMCA	GLS	540	2	0.0108
p15	SW	SA	994	4	1.0145
p1	ES	GTS	500	4	2.1560
p2	LCI	GTS	500	4	2.6098
p3	LCCI	GLS	576	5	3.4102
p5	PCI	SA	708	2	10.2772
p6	PMCA	SA	745	4	7.0317
p19	PCI	GLS	1 609	6	12.2308

Rendimiento en instancias de tamaño entre 10 y 75 (5 instancias):

- Las combinaciones más efectivas fueron la metaheurística GTS con las heurísticas ES y LCI y la metaheurística GLS con las heurísticas PMCA y LCCI. Excepto en la instancia `p15` donde la combinación de la heurística SW y la metaheurística SA logró el mejor resultado.

Rendimiento en instancias de tamaño entre 100 y 216 (3 instancias):

- La metaheurística que resaltó fue SA con la combinación de PCI y PMCA como heurísticas. Mientras que la heurística PCI destaca igualmente con la combinación de GLS.

En general, los resultados de las mejores técnicas son muy variados ya que hay diferentes combinaciones que lograron buenos resultados. Sin embargo, las más destacadas son las heurísticas LCI, PCI, PMCA y las metaheurísticas GLS, SA y GTS.

Tabla 18: Mejores resultados de las instancias MDVRP utilizando la distancia manhattan.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
p31	PCA	TS	788	2	0.0094
p15	PCA	SA	1 134	4	1.1194
p1	PCA	GD	672	4	1.3901
p2	PCA	GD	672	4	1.3803
p3	BI	GD	756	5	5.1132
p5	PCA	GLS	866	2	23.0106
p6	PCA	GD	942	4	14.4528
p19	SCI	SA	2 244	6	5.1480

Rendimiento en instancias de tamaño entre 10 y 75 (5 instancias):

- La combinación más efectiva fue la heurística PCA junto a las metaheurísticas GD, SA y TS.
- En la instancia `p3` se destaca BI como heurística y GD como metaheurística.

Rendimiento en instancias de tamaño entre 100 y 216 (3 instancias):

- La heurística que resaltó fue PCA con la combinación de GLS y GD como metaheurísticas. Mientras que la heurística SCI destaca igualmente con la combinación de SA en la instancia `p19`.

En general, la heurística más destacada fue PCA junto a una variedad de metaheurísticas como GD, SA, TS y GLS, resaltando GD entre el resto.

2.4.6. VRPPD

En esta sección se presentan los resultados del experimento para la variante VRPPD. En la Tabla 19 se describen las 10 instancias del problema utilizadas para el experimento. Donde se reutilizaron las instancias de otros problemas

añadiendo los pares de entrega y recogida de forma aleatoria. Todos los nodos son utilizados en alguno de los objetivos (entrega o recogida) siempre que el tamaño de la instancia sea par, de lo contrario un nodo queda sin emparejar y se elimina.

Como parámetros fijos configurados por defecto se tiene:

- **slack_max:** define la cantidad máxima de tiempo de espera (o slack) permitida en la ruta de un vehículo = (0).
- **fix_start_cumul_to_zero:** indica si el valor acumulado de la distancia para cada vehículo comenzará en cero = (True).
- **capacity:** máxima distancia que deben recorrer los vehículos = (500).

Tabla 19: Descripción de las instancias VRPPD.

Instancia	Tamaño	Rango de coordenadas X [mínimo, máximo]	Rango de coordenadas Y [mínimo, máximo]	Cantidad de vehículos
p31	10	[-64.709, 72.095]	[-72.894, 66.498]	2
CVRP_100	15	[5.0, 52.0]	[16.0, 64.0]	15
CVRP_1	50	[5.0, 63.0]	[6.0, 69.0]	5
CVRP_2	50	[5.0, 63.0]	[6.0, 69.0]	5
CVRP_3	50	[5.0, 63.0]	[6.0, 69.0]	5
p1	50	[5, 63]	[6, 69]	4
p2	50	[5, 63]	[6, 69]	4
CVRP_4	75	[6.0, 70.0]	[4.0, 76.0]	9
CVRP_5	75	[6.0, 70.0]	[4.0, 76.0]	9
p3	75	[6, 75]	[5, 76]	5

En la Tabla 20 se presentan los mejores resultados obtenidos al realizar el experimento con las instancias descritas, utilizando la distancia euclideana. Mientras que en la Tabla 21 se muestran los resultados utilizando la distancia manhattan.

Tabla 20: Mejores resultados de las instancias VRPPD utilizando la distancia euclideana.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
p31	PCA	GD	634	2	0.0402
CVRP_100	FUMV	GD	308	3	0.0865
CVRP_1	SW	GLS	573	1	19.9996
CVRP_2	PMCA	SA	546	1	19.9997
CVRP_3	ES	TS	533	1	19.9997
p1	GCA	GD	618	1	19.9997
p2	ES	GLS	594	1	19.9998
CVRP_4	PMCA	SA	959	1	19.9999
CVRP_5	ES	GD	988	1	19.9999
p3	PCA	GLS	1 013	1	19.9999

Rendimiento en instancias de tamaño entre 10 y 80 (10 instancias):

- Las metaheurísticas que mejores soluciones brindaron fueron GD y GLS con la combinación de las heurísticas PCA y ES como las más predominantes.
- La heurística PMCA en combinación con la metaheurística SA logra la mejor solución en las instancias `CVRP_2` y `CVRP_4`.

En general, la combinación de la heurística PCA junto a la metaheurística GD logró obtener los mejores resultados en la mayoría de las instancias evaluadas.

Tabla 21: Mejores resultados de las instancias VRPPD utilizando la distancia manhattan.

Instancia	Heurística	Metaheurística	Distancia total	Total de rutas	Tiempo de ejecución
p31	PCA	GLS	806	2	0.2635
CVRP_100	FUMV	GTS	394	3	0.1277
CVRP_1	LCCI	SA	966	4	34.6045
CVRP_2	PCA	GTS	842	3	2.5260
CVRP_3	PMCA	GLS	852	4	8.8085

p1	GLA	GLS	930	4	3.6264
p2	AU	TS	948	4	11.9820
CVRP_4	AU	SA	1 196	5	34.6045
CVRP_5	SV	GLS	1 190	5	42.7558
p3	ES	SA	1 110	5	39.1691

Rendimiento en instancias de tamaño entre 10 y 80 (10 instancias):

- Las metaheurísticas que mejores soluciones brindaron fueron SA y GLS con la combinación de variedad de heurísticas como PCA, ES, AU, LCCI, entre otras.
- En las instancias `CVRP_2` y `CVRP_100` predomina la metaheurística GTS junto a las heurísticas FUMV y PCA, respectivamente.
- En la instancia `p2` se destaca la heurística AU junto a la metaheurística SA.

En general, la combinación de las heurística PCA y AU junto a las metaheurísticas GLS y SA obtuvieron los mejores resultados en la mayoría de las instancias evaluadas.

2.4.7. Comparación general

En la resolución de problemas de planificación de rutas de vehículos, la elección de heurísticas y metaheurísticas es crucial y depende en gran medida del tamaño de las instancias y el problema a resolver. En términos generales:

- Se destaca la combinación diversa de metaheurísticas como GD, TS y GLS con heurísticas específicas como SV, PCA, LCI y PMCA.
- La heurística PCA resulta ser la más destacada en todos los problemas tratados, aportando buenas soluciones a un grupo considerable de instancias.
- Aunque todas las metaheurísticas, en combinación con distintas heurísticas, ofrecieron buenos resultados, GD y GLS sobresalieron en todos los problemas.

- La combinación de la mayoría de heurísticas y de todas las metaheurísticas lograron buenos resultados para diferentes instancias de todos los problemas, adaptándose bien a las diferentes características de cada uno.

2.5. Conclusiones parciales

En este capítulo se presentó la propuesta de solución utilizando la biblioteca ORTools en Python para resolver problemas de optimización de rutas de vehículos, específicamente TSP y las variantes del VRP.

- Se logró la conformación de una estructura y arquitectura del proyecto sólida implementando patrones de diseño para mejorar la mantenibilidad y escalabilidad.
- Se consiguieron resolver los problemas VRP con restricción de distancia, VRP con ventanas de tiempo, VRP con capacidad, VRP con entrega y recogida, VRP con múltiples depósitos y TSP.
- ORTools demostró ser robusta y adaptable gracias a su capacidad de manejar combinaciones de restricciones y distintas instancias de los problemas.
- GLS y GD demostraron ser las metaheurísticas más efectivas para resolver los problemas, mientras que las heurísticas PCA, LCI, PMCA y SV mostraron una eficiencia constante en la resolución de instancias de diferentes tamaños, destacando especialmente PCA.

Capítulo 3: Comparación de los resultados

3.1. Introducción

En este capítulo, se realiza un análisis comparativo entre las soluciones generadas por varias bibliotecas y los resultados obtenidos en el capítulo anterior con el uso de los algoritmos proporcionados por la biblioteca ORTools. El enfoque de este análisis está en evaluar la eficiencia de la biblioteca ORTools en la resolución de variantes de Problemas de Planificación de Rutas de Vehículos (VRP).

Se tienen en cuenta factores como el tiempo que demora ejecutar el algoritmo y devolver la solución, así como la distancia total recorrida por todas las rutas y la cantidad de rutas generadas. Estos aspectos se evalúan utilizando diferentes instancias de cada problema, lo que permite concluir cuál biblioteca ofrece mejores resultados para problemas e instancias específicas. Por último se realiza la prueba de Wilcoxon para analizar las diferencias entre los elementos a comparar.

3.2. Comparación con la biblioteca BHCVRP [49]

En este epígrafe, se comparan los resultados obtenidos utilizando ORTools y la versión en Python de BHCVRP. Se emplean 20 instancias de la variante CVRP donde 5 de ellas presentan una flota heterogénea (aquellas cuyo nombre comienza con HFVRP) y 10 una flota homogénea. Se utilizan las distancias euclidean y manhattan como métricas.

Estas instancias se derivan del artículo de Homberger y Gehring del año 1999 [33], mencionado en el capítulo 1. Aunque originalmente se utilizaron para la variante VRPTW, en el experimento realizado se han ignorado los datos de ventanas de tiempo para resolver la variante CVRP con estas instancias. Los

resultados mostrados para ORTools son los mismos obtenidos en el experimento del capítulo 2 en la sección de la variante CVRP.

Para llevar a cabo esta comparación, se consideraron varios factores clave que son determinantes en la evaluación de los algoritmos brindados por las bibliotecas. En primer lugar, se analiza el tiempo de ejecución requerido para encontrar y devolver una solución viable. Este aspecto es crucial en aplicaciones del mundo real, donde la rapidez de respuesta puede ser un factor decisivo. Además, se examina la calidad de las soluciones proporcionadas, específicamente en términos de la distancia total recorrida por todas las rutas y la cantidad de rutas que se requieren en la solución. Este factor es fundamental, ya que una menor distancia recorrida se traduce directamente en una mayor eficiencia operativa y una reducción en los costos asociados.

A continuación, la Tabla 22 refleja la comparación de los resultados utilizando la distancia euclideana, mientras que la Tabla 23 muestra los resultados utilizando la distancia manhattan.

Tabla 22: Comparación entre BHCVRP y ORTools para la variante CVRP utilizando la distancia euclideana.

	BHCVRP			**ORTools**		
Instancia	**Distancia total**	**Total de rutas**	**Tiempo de ejecución (seg)**	**Distancia total**	**Total de rutas**	**Tiempo de ejecución (seg)**
CVRP_1	**558.89**	4	0.54	2 152.00	4	**0.13**
CVRP_4	**787.61**	6	1.25	3 916.00	6	**0.33**
CVRP_p3	**666.86**	3	15.75	4 133.00	3	**0.27**
CVRP_p5	**814.49**	4	36.10	4 534.00	4	**0.55**
CVRP_p14	**6 134.72**	4	94.80	9 300.00	4	**14.44**
HFVRP_1	**550.34**	4	0.48	3 225.00	4	**0.14**
HFVRP_4	**799.87**	6	9.36	3 730.00	4	**0.30**
HFVRP_p3	**666.311**	3	1.35	4 030.00	3	**0.21**
HFVRP_p5	**768.28**	3	347	1 512.00	3	**0.69**
HFVRP_p14	**6 125.32**	4	5226	9 325.00	4	**12.70**
C1_2_1	**2 391.66**	18	9.71	2 793.00	18	**1.52**
C1_2_4	**2 449.29**	18	4.83	2 793.00	18	**1.54**

C1_2_5	**2 338.88**	18	4.98	2 793.00	18	**1.53**
C1_2_6	**2 449.94**	18	4.93	2 793.00	18	**1.64**
C1_2_8	**2 417.94**	18	5.04	2 793.00	18	**1.53**
C1_4_1	**6 239.59**	36	136.57	7 245.00	36	**8.69**
C1_4_4	**6 394.63**	36	57.28	7 245.00	36	**8.99**
C1_6_1	**13 386.91**	56	62.29	14 148.00	56	**19.09**
C1_6_2	**13 222.27**	56	81.74	14 148.00	56	**19.18**
C1_6_3	**12 744.52**	56	79.73	14 148.00	56	**19.09**

Tabla 23: Comparación entre BHCVRP y ORTools para la variante CVRP utilizando la distancia manhattan.

	BHCVRP			**ORTools**		
Instancia	**Distancia total**	**Total de rutas**	**Tiempo de ejecución (seg)**	**Distancia total**	**Total de rutas**	**Tiempo de ejecución (seg)**
CVRP_1	**756.00**	4	0.48	2 972.00	4	**0.11**
CVRP_4	**934.00**	6	2.88	5 526.00	6	**0.13**
CVRP_p3	**897.00**	3	16.39	5 726.00	3	**0.27**
CVRP_p5	**1 029.00**	4	5.92	6 322.00	4	**0.48**
CVRP_p14	**6 460.00**	5	232.76	12 580.00	4	**6.14**
HFVRP_1	**640.00**	2	1.18	3 676.00	4	**0.20**
HFVRP_4	**892.00**	3	16.58	5 086.00	5	**0.37**
HFVRP_p3	**835.00**	3	4.79	5 612.00	3	**0.24**
HFVRP_p5	**1 010.00**	3	2.44	1 938.00	3	**0.49**
HFVRP_p14	**7 800.00**	4	95.02	12 660.00	4	**6.38**
C1_2_1	**3 118.00**	18	19.57	3 684.00	18	**1.58**
C1_2_4	**3 365.00**	18	19.90	3 684.00	18	**1.58**
C1_2_5	**3 189.00**	18	18.18	3 684.00	18	**1.58**
C1_2_6	**3 140.00**	18	19.93	3 684.00	18	**1.59**
C1_2_8	**3 234.00**	18	18.40	3 684.00	18	**1.58**
C1_4_1	**8 053.00**	36	240.53	9 112.00	36	**7.09**
C1_4_4	**8 342.00**	36	240.75	9 112.00	36	**7.14**
C1_6_1	**16 410.00**	56	1 254.25	18 760.00	56	**17.72**
C1_6_2	**16 740.58**	56	1 582.21	18 760.00	56	**17.65**
C1_6_3	**16 360.97**	56	1 358.92	18 760.00	56	**17.57**

Los resultados presentados en la tabla comparativa revelan que la biblioteca BHCVRP demostró un desempeño superior a la biblioteca ORTools en todas las instancias analizadas, en términos de calidad de las soluciones. Sin embargo, ORTools destacó por su menor tiempo de ejecución.

Cabe recalcar que en instancias pequeñas (de 50 a 100 nodos) BHCVRP superó a ORTools por una gran diferencia en el costo total, mientras que para las instancias más grandes (de 200 a 600 nodos) la diferencia entre ambas resultó ser mucho más pequeña.

En la Ilustración 6 se muestra un gráfico para ilustrar lo antes mencionado, comparando las soluciones obtenidas en términos de distancia total recorrida para ambas bibliotecas. Mientras que en la Ilustración 7 se refleja la diferencia en términos de tiempo de ejecución para ambas bibliotecas.

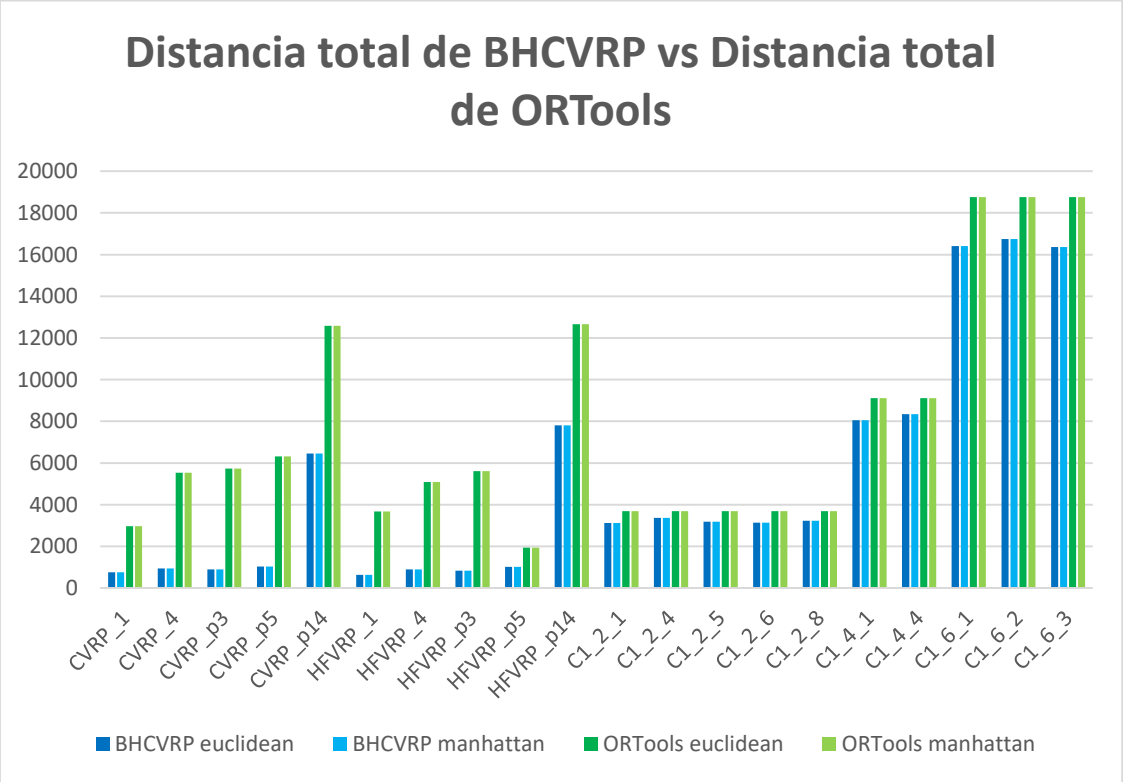

Ilustración 6: Gráfico de comparación entre las distancias totales de BHCVRP y ORTools.

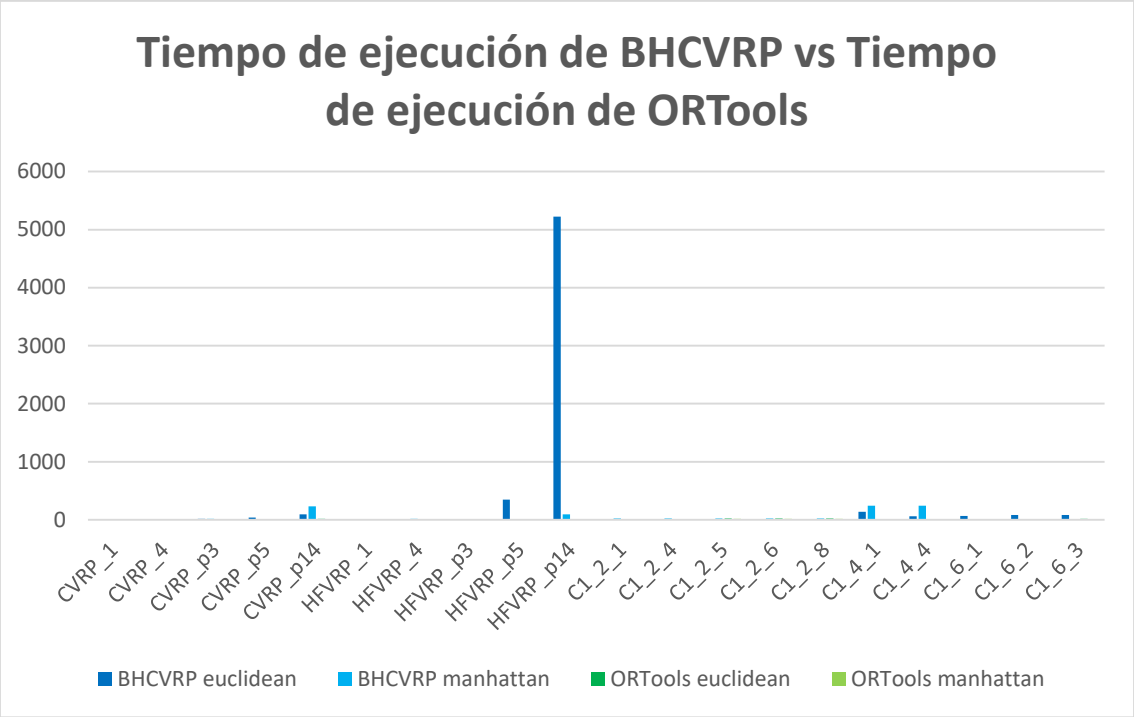

Ilustración 7: Gráfico de comparación entre los tiempos de ejecución de BHCVRP y ORTools.

3.2.1. Prueba de Wilcoxon entre BHCVRP y ORTools

La prueba de Wilcoxon es una prueba no paramétrica utilizada para comparar dos muestras emparejadas o relacionadas. En este caso, se utiliza para comparar dos bibliotecas, ORTools y BHCVRP en cuanto a calidad de las soluciones obtenidas. Los resultados se muestran en la Tabla 24.

Tabla 24: Resultados de la prueba de Wilcoxon entre BHCVRP y ORTools.

Biblioteca	R+(Suma de rangos positivos)	R−(Suma de rangos negativos)	Valor P exacto	Valor P asintótico	Intervalo de confianza para α = 0.90 y α = 0.95	Confianza exacta
ORTools vs BHCVRP	210.0	0.0	≥ 0.2	0.000082	[-3719.51,-143.06]	0

Interpretación de los resultados:

- Valor P exacto y Valor P asintótico: el valor P exacto es mayor o igual a 0.2 y el valor P asintótico es 0.000082 < 0.05, lo que indica que hay una diferencia significativa entre ORTools y BHCVRP.
- Intervalos de confianza: el intervalo de confianza [-3719.51,-143.06] para α = 0.90 y α = 0.95 indica que, con un 90% y un 95% de confianza, respectivamente, la mediana de las diferencias entre ORTools y BHCVRP se encuentra dentro de este rango negativo.

3.3. Comparación con el artículo de Homberger y Gehring del año 1999 [33]

En este epígrafe se presenta la comparación de las soluciones de [33], con los resultados obtenidos utilizando la biblioteca ORTools, en la variante VRPTW. Se han empleado 23 instancias de distintos tamaños (de 200 a 600 nodos), utilizando la distancia euclidiana como medida ya que es la empleada por el artículo a comparar. Los resultados mostrados para ORTools son los mismos obtenidos en el experimento del capítulo 2 en la sección de la variante VRPTW.

Para realizar esta comparación, se han considerado factores como la distancia total recorrida y la cantidad de rutas empleadas. Una menor distancia total implica una reducción de los costos y un aumento en la eficiencia, lo cual es fundamental para determinar la mejor solución entre las dos metodologías comparadas.

En la Tabla 25 se muestra la comparación entre los resultados obtenidos en [33] y ORTools teniendo en cuenta lo mencionado anteriormente.

Tabla 25: Comparación entre el artículo de Homberger y Gehring del año 1999 y ORTools para la variante VRPTW.

	Homberger y Gehring del año 1999		ORTools	
Instancia	**Distancia total**	**Total de rutas**	**Distancia total**	**Total de rutas**
C1_2_1	**2 698.6**	20	2 990.0	21
C1_2_2	**2 694.3**	20	2 761.0	20
C1_2_3	**2 675.8**	20	2 828.0	19
C1_2_4	**2 625.6**	19	2 815.0	20
C1_2_5	**2 694.9**	20	3 105.0	21
C1_2_6	**2 694.9**	20	2 859.0	21
C1_2_7	**2 694.9**	20	2 811.0	21
C1_2_8	**2 684.0**	20	2 708.0	20
C1_2_9	**2 639.6**	19	2 963.0	20
C1_2_10	**2 624.7**	19	2 861.0	20
C1_4_1	**7 138.8**	40	7 242.0	40
C1_4_2	**7 113.3**	40	7 427.0	40
C1_4_3	**6 929.9**	38	7 711.0	41
C1_4_4	**6 777.7**	37	7 491.0	38
C1_4_5	**7 138.8**	40	7 440.0	41
C1_4_6	**7 140.1**	40	7 478.0	42
C1_4_7	**7 136.2**	40	7 477.0	40
C1_4_8	**7 083.0**	39	7 557.0	42
C1_4_9	**6 927.8**	37	7 521.0	39
C1_4_10	**6 825.4**	37	7 343.0	38
C1_6_1	**14 076.6**	60	15 364.0	64
C1_6_2	**13 948.3**	58	16 759.0	63
C1_6_3	**13 756.5**	57	15 778.0	59

Los resultados obtenidos indican que ORTools fue superado por el método propuesto por [33] tanto en la eficiencia de las rutas utilizadas como en la calidad de las soluciones generadas. Es importante señalar que, en instancias de menor tamaño, las soluciones y cantidad de rutas utilizadas son bastante similares. Sin embargo, a medida que aumenta el tamaño de las instancias, las diferencias entre los dos algoritmos se vuelven más pronunciadas.

Esto sugiere que, aunque ORTools puede generar buenas soluciones en problemas de menor escala, su rendimiento se ve comprometido en escenarios de mayor complejidad, donde el método propuesto en [33] demuestra una mayor robustez y eficiencia en la variante VRPTW. Para un mejor entendimiento la Ilustración 8 expone las distancias totales obtenidas por la biblioteca ORTools y [33], donde se reflejan las conclusiones mencionadas anteriormente.

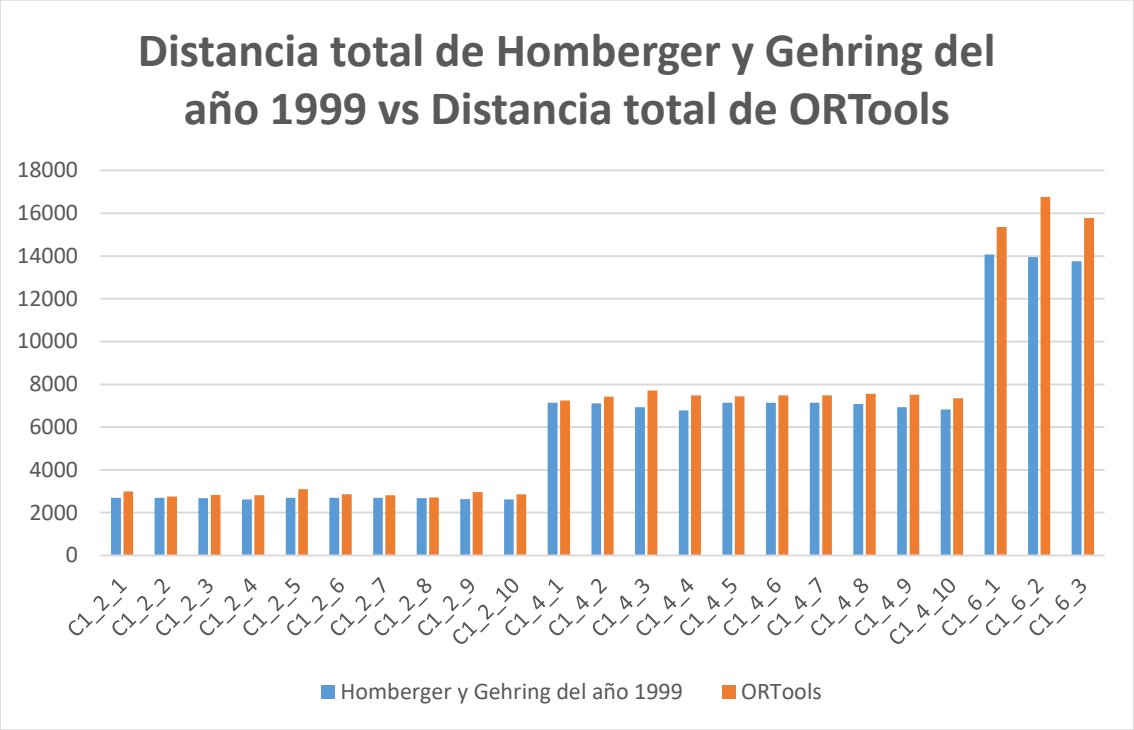

Ilustración 8: Gráfico de comparación entre las distancias totales del artículo de Homberger y Gehring del año 1999 y ORTools.

3.3.1. Prueba de Wilcoxon entre Homberger y Gehring del año 1999 y ORTools

En esta sección se compara el algoritmo presentado en [33] con la biblioteca ORTools, con el propósito de determinar si existen diferencias significativas entre ellos. Para ello, se ha aplicado la prueba de Wilcoxon, cuyos resultados se presentan en la Tabla 26.

Tabla 26: Resultados de la prueba de Wilcoxon entre Homberger y Gehring del año 1999 y ORTools.

Biblioteca	R+(Suma de rangos positivos)	R−(Suma de rangos negativos)	Valor P exacto	Valor P asintótico	Intervalo de confianza para α = 0.90 y α = 0.95	Confianza exacta
ORTools vs Homberger y Gehring del año 1999	324.0	1.0	≥ 0.2	0.000013	[-12,213.4 , 14.6]	0

Interpretación de los resultados:

- Valor P exacto y Valor P asintótico: un valor p exacto ≥ 0.2 y un valor p asintótico de 0.000013 < 0.05 indican que hay una baja probabilidad de que las diferencias observadas sean por azar, sugiriendo una diferencia significativa entre los algoritmos.
- Intervalos de confianza: el intervalo de confianza es de [-12,213.4, 14.6] para ambos niveles de confianza (90% y 95%), lo que sugiere que hay una alta variabilidad en las diferencias observadas, ya que es un intervalo amplio.

3.4. Conclusiones parciales

En este capítulo se ejecutaron las comparaciones de los resultados obtenidos en el capítulo 2 utilizando la biblioteca ORTools en la resolución de problemas de optimización de rutas.

- Se llevaron a cabo pruebas detalladas para evaluar el desempeño de la biblioteca en la resolución de los problemas, como pruebas de rendimiento (tiempo de ejecución) y calidad de las soluciones.
- Se validaron los resultados con un incremento gradual de la complejidad, así como con instancias pequeñas e instancias más complejas para verificar la consistencia y estabilidad de las soluciones.

- Los resultados de la comparación arrojaron que para las instancias utilizadas del problema CVRP la biblioteca BHCVRP aporta mejores soluciones que ORTools. Mientras que ORTools obtiene mejores tiempos de ejecución.
- Los resultados de la comparación arrojaron que para las instancias utilizadas del problema VRPTW los resultados del artículo de Homberger y Gehring del año 1999 brindaron mejores soluciones que ORTools.

Conclusiones

Este trabajo se centró en la experimentación con la biblioteca ORTools para resolver problemas de optimización de rutas de vehículos, específicamente TSP y las variantes VRP. La experimentación se llevó a cabo en varias fases, desde una revisión exhaustiva del estado del arte y los conceptos teóricos hasta la implementación práctica de soluciones usando la biblioteca para evaluar sus fortalezas y debilidades en este ámbito.

- Existen en el estado del arte diversos trabajos que dan solución a los problemas de optimización de rutas de vehículos donde se implementan técnicas heurísticas y metaheurísticas.
- En la búsqueda realizada se identificaron diversas herramientas de optimización, cuyas ventajas y desventajas fueron resumidas para su comparación. Entre estas herramientas, ORTools demostró ser robusta, versátil y capaz de implementar una amplia variedad de técnicas esenciales en la optimización combinatoria. Sin embargo, no presenta una guía donde se evalúe las ventajas y desventajas de la biblioteca en la optimización de rutas.
- Se plantearon soluciones para resolver los problemas TSP, DVRP, CVRP, VRPTW, VRPPD y MDVRP, evaluando los resultados obtenidos en diferentes tipos de instancias, teniendo en cuenta los tiempos de ejecución y la calidad de las soluciones.
- Las metaheurísticas recomendadas para la resolución de los problemas son GLS y GD, mientras que las heurísticas son PCA, LCI, PMCA y SV. Estas técnicas demostraron ser eficientes para diferentes tamaños de instancias, aunque el resto también logró buenos resultados para ciertas instancias.
- Los resultados obtenidos en la comparación de ORTools con la biblioteca BHCVRP y el algoritmo de Homberger y Gehring del año 1999 mostraron que ORTools fue superado en la calidad de las soluciones por ambas. Sin embargo, se destacó en términos de rendimiento en el tiempo de ejecución.

- La biblioteca demostró ser una herramienta eficaz para la optimización de rutas de vehículos, capaz de devolver buenas soluciones en un corto tiempo, a pesar de ser superada por otras herramientas.

Recomendaciones

Después de identificar las fortalezas y debilidades de la biblioteca ORTools para abordar los problemas presentados en este informe (CVRP, DVRP, VRPTW, MDVRP, TSP, VRPPD), se plantea lo siguiente para el seguimiento de la tesis:

- Explorar el Subproblema de Selección de Paradas (*Bus Stop Selection*, BSS) y el Problema de Rutas de Autobuses Escolares (*School Bus Routing Problem*, SBRP) utilizando tanto los módulos de la biblioteca como algoritmos desarrollados internamente, proporcionando las limitaciones y ventajas de la biblioteca en estos problemas.
- Experimentar con problemas del tipo VRPTW con límites de tiempo, descansos de los vehículos y restricciones de distancia, con el propósito de evaluar el desempeño de la biblioteca en estas variantes.
- Realizar demos de uso de las funcionalidades de ORTools en el ámbito de optimización de rutas de vehículos para los problemas mencionados en los puntos anteriores.
- Incorporar la funcionalidad de reoptimización en tiempo real que ofrece la biblioteca a partir de soluciones iniciales.
- Analizar el Principio de Substitución de Liskov (*Liskov Substitution Principle)* que implementa ORTools para combinarla con BHCVRP y el componente BSS.
- Emplear técnicas de Minería de Datos para obtener reglas que predigan cuál heurística o metaheurística es mejor usar entre las que ofrece ORTools y predecir cuándo es mejor ORTools u otra alternativa.
- Debido a que resultaron ser buenas las soluciones de las heurísticas implementadas en BHCVRP, investigar la posibilidad de combinarlas con las metaheurísticas que ofrece la biblioteca.

Referencias bibliográficas

[1] A. Ramos, P. Sánchez, J. M. Ferrer, J. Barquín, and P. Linares, "Modelos matemáticos de optimización," *Publicación Técnica,* vol. 1, 2010.

[2] A. Aranda and J. Jiménez de Vega, "Optimización de rutas de transporte," *Proyecto de Sistemas Informáticos. Facultad de Informática, Universidad Complutense de Madrid.,* 2013.

[3] S. Gupta, A. Rana, and V. Kansal, "Comparison of Heuristic techniques: A case of TSP," in *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2020: IEEE, pp. 172-177.

[4] G. Kim, Y.-S. Ong, C. K. Heng, P. S. Tan, and N. A. Zhang, "City vehicle routing problem (city VRP): A review," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, no. 4, pp. 1654-1666, 2015.

[5] R. A. J. Pirabán, "Métodos Aproximados para la Solución del Problema de Enrutamiento de Vehículos (Dic 2008)," *Universidad Nacional de Colombia,* 2018.

[6] M. Á. Morales Hernández, J. M. González Camacho, H. Robles Vásquez, D. H. d. Valle Paniagua, and J. R. Durán Moreno, "Machine Learning Algorithms for Predicting of Academic Achievement," *RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo,* vol. 12, no. 24, 2022.

[7] C. Henríquez-Rivas, C. Guerrero-Ortiz, and A. Ávila Barrera, "Trabajo matemático de profesores universitarios: Heurísticas de solución de una tarea," *Educación matemática,* vol. 33, no. 3, pp. 233-262, 2021.

[8] G. Rodríguez Canal, "Estudio de métodos exactos y aproximados para la resolución del problema de localización sin capacidades," *Universidad de Valladolid. Facultad de CienciasAutoridad UVA,* 2020. [Online]. Available: http://uvadoc.uva.es/handle/10324/43821

[9] T. Tetzlaff *et al.*, "Metaheurísticas, búsqueda estocástica y cómputo eficiente en optimización aplicada," in *XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja)*, 2021.

[10] S. Heipcke and Y. Colombani, "Xpress mosel: modeling and programming features for optimization projects," in *Operations Research Proceedings 2019: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Dresden, Germany, September 4-6, 2019*, 2020: Springer, pp. 677-683.

[11] J. P. García Sabater, "Programación Matemática en Python con PULP," *RIUNET Repositorio UPV,* p. 49, 2021. [Online]. Available: http://hdl.handle.net/10251/158416.

[12] P. Virtanen *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature methods,* vol. 17, no. 3, pp. 261-272, 2020.

[13] C. Thibaut, "OR-Tools' Vehicle Routing Solver: A Generic Constraint-Programming Solver with Heuristic Search for Routing Problems (VRPs)," *Cadena González R.A,* February 23 2023. [Online]. Available: https://hal.science/hal-04015496v1/file/ROADEF_2023_ORTools_slides.pdf.

[14] P. Offermann, S. Blom, M. Schönherr, and U. Bub, "Artifact types in information systems design science–a literature review," in *Global Perspectives on Design Science Research: 5th International Conference, DESRIST 2010, St. Gallen, Switzerland, June 4-5, 2010. Proceedings. 5*, 2010: Springer, pp. 77-92.

[15] J. Castillo Ungar, "Problemas NP-completos en grafos," *Universidad de los Andes, Departamento de Matemáticas.,* Mayo 20 2023.

[16] N. Göbel, "The Vehicle Routing Problem Explained," ed. Mathematics & algorithms, 2022.

[17] A. C. P. Pérez, E. S. Ansola, and A. Rosete, "A metaheuristic solution for the school bus routing problem with homogeneous fleet and bus stop selection," *Ingeniería,* vol. 26, no. 2, pp. 233-253, 2021.

[18] D. Ospina-Toro, E. M. Toro-Ocampo, and R. A. Gallego-Rendón, "Solución del MDVRP usando el algoritmo de búsqueda local iterada," *REVISTA COLOMBIANA DE TECNOLOGIAS DE AVANZADA (RCTA),* vol. 1, no. 31, pp. 120-127, 2018.

[19] P. P. Ballesteros Silva and A. Escobar Zuluaga, "Revisión del estado del arte del problema de ruteo de vehículos con recogida y entrega (VRPPD)," *Ingeniería y Desarrollo,* vol. 34, no. 2, pp. 463-482, 2016.

[20] R. M. Cunquero, "Algoritmos heurísticos en optimización combinatoria," *Valencia: Universodad de Valencia. Retrieved,* vol. 11, no. 01, p. 2012, 2003.

[21] E. C. Gonzalez, W. Adarme-Jaimes, and J. A. Orjuela-Castro, "Stochastic mathematical model for vehicle routing problem in collecting perishable products," *Dyna,* vol. 82, no. 189, pp. 199-206, 2015.

[22] C. Archetti and M. G. Speranza, "The split delivery vehicle routing problem: A survey," *The vehicle routing problem: Latest advances and new challenges,* pp. 103-122, 2008.

[23] M. Jafari-Eskandari, A. Aliahmadi, and G. Khaleghi, "A robust optimisation approach for the milk run problem with time windows with inventory uncertainty: an auto industry supply chain case study," *International Journal of Rapid Manufacturing,* vol. 1, no. 3, pp. 334-347, 2010.

[24] E. J. Guachamin Sánchez, "Modelos integrados de optimización de transporte público: heurísticas para el problema integrado de calendarización y enrutamiento de pasajeros," 2023.

[25] S. Boyd and J. Mattingley, "Branch and bound methods," *Notes for EE364b, Stanford University,* vol. 2006, p. 07, 2007.

[26] J. D. Mantilla Mejía, "Uso del operador swap genera soluciones eficientes computacionales en un caso de enrutamiento de vehículos con enfoque de ventanas de tiempo," *Computer and Electronic Sciences: Theory and Applications,* vol. 2, no. 1, pp. 51-60, 2021.

[27] D. G. Reina, A. T. Córdoba, and Á. R. del Nozal, *Algoritmos Genéticos con Python: Un enfoque practico para resolver problemas de ingeniería*. Marcombo, 2020.

[28] J. A. J. Builes, R. E. A. Sanchez, and L. D. J. Pinzón, "Métodos de búsqueda usando los algoritmos de enjambre de partículas y genético," *Lámpsakos,* no. 16, pp. 52-60, 2016.

[29] D. V. A. ORTEGA, "Implementación de un algoritmo heurístico de Recocido Simulado para el problema de enrutamiento de vehículos con capacidades homogéneas," Tesis de maestría, Universidad Autónoma del Estado de México, 2023. [Online]. Available: http://riaa.uaem.mx/handle/20.500.12055/3913

[30] E. López, Ó. Salas, and Á. Murillo, "El problema del agente viajero: un algoritmo determinístico usando búsqueda tabú," *Revista de Matemática: teoría y aplicaciones,* vol. 21, no. 1, pp. 127-144, 2014.

[31] F. Obando-Vidal, N. Díaz-Mariño, and E. Martínez-Flor, "Algoritmo de optimización de colonia de hormigas aplicado a TSP, una revisión sistemática," *Revista Ibérica de Sistemas e Tecnologías de Información,* no. E38, pp. 404-417, 2020.

[32] J. Ruiz-Meza, "Problema de ruteo de vehículos multi-objetivo con entregas y recogidas simultáneas y minimización de emisiones," *Ingeniare. Revista chilena de ingeniería,* vol. 29, no. 3, pp. 435-449, 2021.

[33] J. Homberger and H. Gehring, "Two evolutionary metaheuristics for the vehicle routing problem with time windows," *INFOR: Information Systems and Operational Research,* vol. 37, no. 3, pp. 297-318, 1999.

[34] J. B. COLIN, "Diseño de un algoritmo metaheurístico para la solución del Problema de Ruteo de Vehículos MultiDepósito," *RIAA UAEM. Biblioteca Central Universitaria.,* 2024. [Online]. Available: http://riaa.uaem.mx/handle/20.500.12055/4518.

[35] J. A. M. Viscaya and M. A. C. Liera, "Una implementacion eficiente de algoritmo memético para TSP," *Instituto Tecnológico de La Paz, División de estudios de posgrado e investigación, La Paz, Baja California Sur, México*.

[36] J. Sui, S. Ding, R. Liu, L. Xu, and D. Bu, "Learning 3-opt heuristics for traveling salesman problem via deep reinforcement learning," in *Asian Conference on Machine Learning*, 2021: PMLR, pp. 1301-1316.

[37] Q. Wang, C. Zhang, and C. Tang, "Discovering Lin-Kernighan-Helsgaun heuristic for routing optimization using self-supervised reinforcement learning," *Journal of King Saud University-Computer and Information Sciences,* vol. 35, no. 8, p. 101723, 2023.

[38] P. D. R. Vaca, D. V. R. Perez, M. P. O. Díaz, and S. A. D. Pazmiño, "Herramientas tecnológicas utilizadas para la optimización de la gestión de transporte," *Polo del Conocimiento: Revista científico-profesional,* vol. 7, no. 4, p. 6, 2022.

[39] M. A. Herrera Montoya and C. A. Ramírez Beltrán, "La transformada de Fourier fraccional y algunas aplicaciones," Tesis de Diploma, 2022. [Online]. Available: http://hdl.handle.net/11349/34374

[40] E. Bressert, "SciPy and NumPy: an overview for developers," *Sebastopol, CA: O'Reilly Media, Inc.,* 2012. [Online]. Available: http://oreilly.com/catalog/errata.csp?isbn=9781449305468.

[41] H. G. Salas, *Programación lineal aplicada-3da edición*. Bogotá, Colombia: Ecoe Ediciones, 2022, p. 389.

[42] D. Molina Pérez and E. Cabrera Estupiñán, "Programación entera para modelos lineales," *Ingeniería Hidráulica y Ambiental,* vol. 35, no. 1, pp. 62-76, 2014.

[43] Y. B. Colina, "Aplicaciones de programación lineal, entera y mixta," *Ingeniería Industrial. Actualidad y Nuevas Tendencias,* vol. 2, no. 7, pp. 85-104, 2011.

[44] K. O'Brien, "ResearchGate," *Journal of the Medical Library Association: JMLA,* vol. 107, no. 2, p. 284, 2019. [Online]. Available: https://www.researchgate.net/

[45] R. Vine, "Google scholar," *Journal of the Medical Library Association,* vol. 94, no. 1, p. 97, 2006. [Online]. Available: https://scholar.google.com/scholar.

[46] A. L. Packer *et al.*, "SciELO: uma metodologia para publicação eletrônica," *Ciência da informação,* vol. 27, pp. nd-nd, 1998. [Online]. Available: https://scielo.org/es/.

[47] R. S. Estrada, "Pensar y diseñar en plural. Los siete principios del diseño universal," *Revista digital universitaria,* vol. 18, no. 4, 2017.

[48] O. D. G. Alvarez, N. P. L. Larrea, and M. V. R. Valencia, "Análisis comparativo de Patrones de Diseño de Software," *Polo del Conocimiento: Revista científico-profesional,* vol. 7, no. 7, pp. 2146-2165, 2022.

[49] L. Díaz, "Nueva versión de la Biblioteca de Heurísticas de Construcción para Problemas de Planificación de Rutas de Vehículos," Tesis de Diploma, Instituto Superior Politécnico José Antonio Echeverría, 2016.

[50] L. B. R. Medina, "Una revisión al estado del arte del problema de ruteo de vehículos: Evolución histórica y métodos de solución," *Ingeniería,* Artículo de Revisión vol. 16, pp. 35-55, 2011.

[51] I. L. S. González, "Solución del problema de planificación de rutas de vehículos capacitados en el contexto del comercio electrónico en Cuba," Departamento de Inteligencia Artificial e Infraestructura de Sistemas Informáticos, Instituto Superior Politécnico José Antonio Echeverría, 2022.

[52] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA journal on computing,* vol. 3, no. 4, pp. 376-384, 1991. [Online]. Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[53] E. Alba., "Networking and emerging optimiztion.," *Universisdad de Malaga,* Enero 2014. [Online]. Available: http://neo.lcc.uma.es/vrp/vrp-instances/.

[54] E. Maria, E. Budiman, and M. Taruk, "Measure distance locating nearest public facilities using Haversine and Euclidean Methods," in *Journal of Physics: Conference Series*, 2020, vol. 1450, no. 1: IOP Publishing, p. 012080.

[55] M. Malkauthekar, "Analysis of euclidean distance and manhattan distance measure in face recognition," in *Third International Conference on Computational Intelligence and Information Technology (CIIT 2013)*, 2013: IET, pp. 503-507.

[56] É. O. Rodrigues, "Combining Minkowski and Chebyshev: New distance proposal and survey of distance metrics using k-nearest neighbours classifier," *Pattern Recognition Letters,* vol. 110, pp. 66-71, 2018.

[57] A. Z. Maciel, R. R. Andrade, C. R. M. Valenzuela, and F. Pivot, "Evaluación de receptores GPS de bajo costo de alta sensibilidad para trabajos geodésicos. Caso de estudio: línea base geodésica," *CIENCIA ergo-sum,* vol. 27, no. 1, p. 6, 2020.

[58] J. Cuzick, "A Wilcoxon-type test for trend," *Statistics in medicine,* vol. 4, no. 1, pp. 87-90, 1985.