**Universidad Tecnológica de la Habana**

**“José Antonio Echeverría”**

**Facultad de Ingeniería Informática**



Optimización de Rutas de Vehículos: Análisis Comparativo de Algoritmos en OR-Tools

**Autor**

Dayana Vázquez Rodríguez

**Tutores**

Dr. C. Alejandro Rosete Suárez

Dr. C. Isis Torres Pérez

La Habana, junio 2025

**Introducción**

La optimización es una disciplina fundamental, en la investigación operativa y la inteligencia artificial, utilizada para encontrar soluciones óptimas en una amplia variedad de problemas. En el contexto logístico y del transporte, la optimización de rutas de vehículos juega un papel crucial en la reducción de costos, la mejora de los tiempos de entrega y la eficiencia operativa. Este tipo de problemas se presenta en diversos sectores, desde la distribución de mercancías hasta el servicio de transporte de pasajeros.

Uno de los problemas más estudiados en este ámbito es el Problema del Viajante de Comercio (TSP, por sus siglas en inglés, *Traveling Salesman Problem*), que consiste en encontrar la ruta más corta que un vendedor ambulante debe seguir para visitar un conjunto de ciudades una sola vez y regresar al punto de partida. Derivado del TSP surge el Problema de Planificación de Rutas de Vehículos (VRP, *Vehicle Routing Problem*), cuyo objetivo es determinar las rutas óptimas para un conjunto de vehículos que deben atender a un grupo de clientes con demandas específicas. A partir de esta formulación básica han surgido diversas variantes que incorporan restricciones adicionales:

* Problema de Planificación de Rutas de Vehículos con Capacidad (CVRP, *Capacitated Vehicle Routing Problem*): establece restricciones de capacidad en los vehículos. Es ampliamente utilizado en la distribución de mercancías, optimizando el uso de la flota al considerar la capacidad de los vehículos.
* Problema de Planificación de Rutas de Vehículos con Ventanas de Tiempo (VRPTW, *Vehicle Routing Problem with Time Windows*): agrega restricciones de tiempo para las entregas. Es esencial en servicios de entrega y transporte, garantizando que las operaciones cumplan con horarios establecidos, como en el reparto de productos perecederos o la gestión de transporte público.
* Problema de Planificación de Rutas de Vehículos con Recogida y Entrega (VRPPD, *Vehicle Routing Problem with Pickup and Delivery*): involucra la recogida y entrega de cargas en distintas ubicaciones. Se aplica en sistemas de transporte de pasajeros y en la logística de última milla, donde se requiere recoger y entregar paquetes en distintos puntos
* Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos (MDVRP, *Multi-Depot Vehicle Routing Problem*): considera múltiples depósitos para la distribución. Es crucial en redes de distribución con múltiples centros de almacenamiento, optimizando rutas para reducir costos y tiempos de entrega en sectores como el comercio electrónico y la logística de cadenas de suministro.

Para resolver estos problemas, se han desarrollado diferentes enfoques de solución que pueden agruparse en tres grandes categorías: métodos exactos, heurísticas y metaheurísticas. Los métodos exactos buscan garantizar la obtención de la solución óptima a través de técnicas matemáticas y algoritmos de optimización combinatoria. Aunque estos enfoques garantizan la mejor solución, su principal limitación radica en el crecimiento exponencial del tiempo de cómputo a medida que aumenta el tamaño del problema. En problemas de planificación de rutas con muchas restricciones y grandes volúmenes de datos, los métodos exactos pueden volverse ineficientes, lo que hace necesario recurrir a enfoques más flexibles como las heurísticas y las metaheurísticas.

Las heurísticas son estrategias diseñadas para encontrar soluciones aproximadas en tiempos computacionales reducidos. En lugar de buscar la solución óptima, las heurísticas aplican reglas específicas o criterios de construcción para generar soluciones viables de manera eficiente. Estas estrategias pueden proporcionar resultados aceptables con tiempos de ejecución considerablemente menores en comparación con los métodos exactos, aunque sin la garantía de encontrar la mejor solución. Las heurísticas suelen ser utilizadas cuando se requiere rapidez en la toma de decisiones, especialmente en problemas de gran escala donde los métodos exactos son impracticables.

Por otro lado, las metaheurísticas son algoritmos de optimización de alto nivel que mejoran las soluciones generadas por heurísticas mediante técnicas avanzadas de búsqueda en el espacio de soluciones. A diferencia de las heurísticas, las metaheurísticas incorporan mecanismos que permiten escapar de óptimos locales y explorar de manera efectiva el conjunto de posibles soluciones. Su flexibilidad las hace aplicables a una amplia variedad de problemas y, aunque no garantizan la optimalidad, pueden ofrecer soluciones cercanas al óptimo en tiempos razonables.

El desarrollo de estos métodos ha dado lugar a múltiples herramientas computacionales especializadas en la resolución de problemas de optimización. En este estudio, se ha utilizado OR-Tools, una biblioteca de código abierto desarrollada por Google para resolver problemas de optimización combinatoria. OR-Tools proporciona una amplia gama de algoritmos, como búsqueda local, programación lineal y entera mixta, así como métodos especializados para VRP. Su flexibilidad y eficiencia la convierten en una herramienta ideal para experimentar con diferentes enfoques de optimización de rutas.

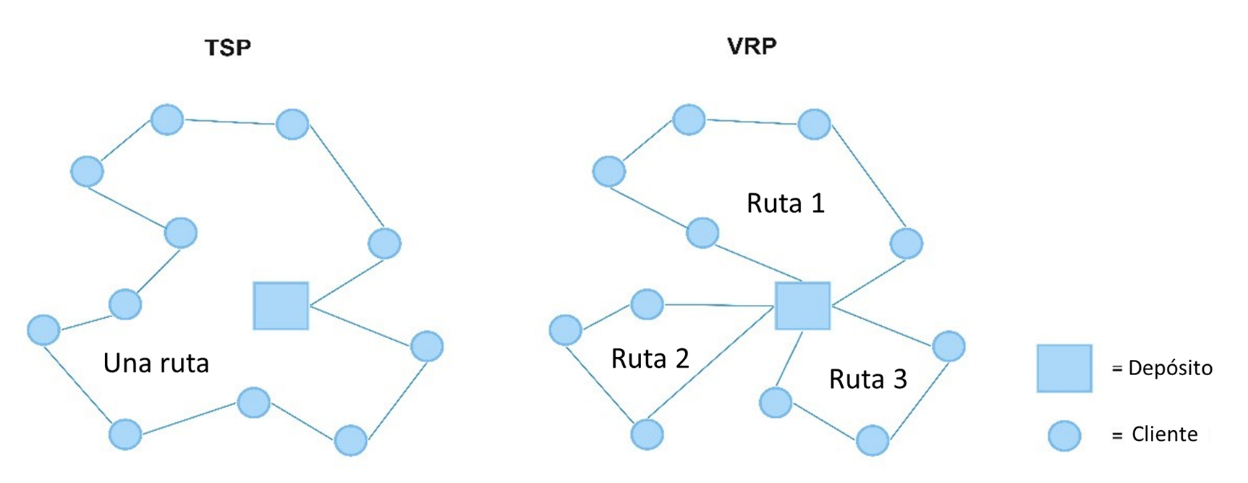
El objetivo de esta monografía es analizar el rendimiento de diversas heurísticas y metaheurísticas proporcionadas por OR-Tools en la resolución de estos problemas. Para ello, se realizan 10 iteraciones con cada una de las instancias seleccionadas de cada problema y se evalúan los resultados para determinar las mejores soluciones. Para el cálculo de la distancia entre los nodos, se utilizan los tipos de distancia Euclidiana y Manhattan, siendo esta última la más utilizada en aplicaciones reales. Además, se utiliza la generación de soluciones a partir de una solución inicial dada que brinda OR-Tools, es decir, rutas iniciales. Sin embargo, estas soluciones iniciales no son obligatorias y no es necesario pasarlas como parámetros. Si se proporciona una ruta inicial, esta se tomará como punto de partida y se aplicará una heurística seguida de una metaheurística para mejorar la solución. En caso de no proporcionar una ruta inicial, el algoritmo simplemente ejecutará la heurística desde cero y luego aplicará la metaheurística sin depender de una ruta predefinida. Adicionalmente, se entrenan árboles de decisión utilizando como dataset los resultados obtenidos en las iteraciones realizadas para cada problema. Estos modelos permiten predecir la combinación de heurísticas y metaheurísticas más eficiente según ciertas características de la instancia, como la cantidad de nodos, la capacidad de los vehículos, entre otros.

Este estudio busca contribuir a la selección de métodos adecuados para distintos escenarios de optimización de rutas, ofreciendo un análisis comparativo de estrategias y brindando recomendaciones basadas en el desempeño de los algoritmos en distintas condiciones. Con ello, se espera proporcionar una guía útil para la toma de decisiones en aplicaciones reales de logística y transporte. Se aborda, en primer lugar, el estudio de los problemas de optimización de rutas de vehículos (VRP), iniciando con una revisión exhaustiva de las técnicas de optimización, tanto exactas como aproximadas, junto con una descripción detallada de las herramientas que implementan dichas metodologías. Posteriormente, se profundiza en el análisis de OR-Tools, examinando sus parámetros de configuración, el manejo de rutas iniciales, así como las heurísticas y metaheurísticas que emplea para la resolución de problemas. A continuación, se presenta un experimento aplicado a las variantes de VRP mencionadas anteriormente, cuyos resultados son evaluados mediante la prueba de Friedman para determinar el rendimiento relativo de los algoritmos. En una fase posterior, se realiza una comparación entre OR-Tools y otras herramientas de optimización, utilizando la prueba de Wilcoxon con el objetivo de identificar diferencias estadísticamente significativas en su desempeño. Finalmente, se exploran diversas técnicas de predicción y análisis de datos, tales como regresión lineal, redes neuronales, árboles de decisión y bosques de decisión, entre otras. En particular, se aplican bosques de decisión con el propósito de predecir los algoritmos más adecuados de la biblioteca para cada problema, en función de las características específicas de las instancias analizadas.

**1. Problemas de optimización de rutas de vehículos**

Los VRP son un conjunto de problemas combinatorios fundamentales en la investigación operativa y la logística, que buscan determinar las rutas más eficientes para un grupo de vehículos que deben satisfacer las demandas de un conjunto de clientes, respetando diversas restricciones. Existen variantes de este problema que se enfocan en distintos aspectos logísticos y operativos. A continuación se detalla cada una de las variantes más importantes:

* Problema del Viajante de Comercio (TSP): este es el caso más básico de los problemas de rutas, donde se debe encontrar la ruta más corta que un vendedor debe seguir para visitar una lista de ciudades, pasando solo una vez por cada una y regresando al punto de partida. Aunque es un problema clásico, es de alta complejidad computacional, especialmente cuando el número de ciudades aumenta, ya que su solución requiere explorar todas las combinaciones posibles. A diferencia del resto de variantes, solo incluye un vehículo.
* Problema de Planificación de Rutas de Vehículos con Capacidad (CVRP): en este problema, además de optimizar las rutas, se debe asegurar que los vehículos no excedan su capacidad de carga. Esto es relevante en problemas logísticos donde se debe distribuir mercancía a varios clientes, y cada vehículo tiene un límite de carga que no puede ser superado. El desafío radica en encontrar una asignación eficiente de las cargas a los vehículos sin incumplir esta restricción.
* Problema de Planificación de Rutas de Vehículos con Ventanas de Tiempo (VRPTW): en este caso, los vehículos deben realizar las entregas dentro de ciertos períodos de tiempo establecidos por los clientes. Estas ventanas de tiempo añaden una capa de complejidad, ya que es necesario que las rutas no solo sean eficientes en términos de distancia o tiempo, sino que también cumplan con los plazos de entrega específicos.
* Problema de Planificación de Rutas de Vehículos con Recogida y Entrega (VRPPD): esta variante involucra la recogida y entrega de cargas en distintos puntos geográficos, donde las entregas de un cliente deben realizarse solo después de que sus artículos hayan sido recogidos en otro punto. Esta situación es común en la logística de transporte de mercancías, y la complejidad se incrementa al considerar los tiempos y rutas tanto de recogida como de entrega.
* Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos (MDVRP): en este caso, los vehículos operan desde varios depósitos en lugar de uno solo. El desafío adicional es asignar vehículos a los depósitos adecuados y optimizar las rutas partiendo desde múltiples puntos de origen. Esto es útil en grandes operaciones logísticas donde se dispone de varios centros de distribución.
* Problema de Rutas de Autobuses Escolares (*School Bus Routing Problem,* SBRP): 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 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): 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): 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).



**Ilustración 1**: Gráfico de comparación entre los problemas TSP (una ruta) y variantes VRP (varias rutas).

Estas variantes del VRP representan problemas reales que las empresas de logística y transporte deben resolver para minimizar los costos operativos y mejorar la eficiencia de sus operaciones. Resolverlos de manera eficiente es crucial para optimizar el uso de los recursos, como los vehículos, el combustible y el tiempo de los conductores, lo cual repercute directamente en la reducción de costos y la mejora del servicio al cliente. La complejidad de estos problemas ha llevado al desarrollo de algoritmos heurísticos y metaheurísticos, que buscan obtener soluciones aproximadas en tiempos razonables, ya que encontrar la solución óptima es computacionalmente costoso para problemas de gran escala. 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*): 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 (x1,y1) y (x2,y2) en un plano se calcula utilizando el teorema de Pitágoras. La Ecuación 1 muestra la fórmula para este cálculo.

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

Donde:

(x1,y1) => punto de partida en el eje de coordenadas (x,y)

(x2,y2) => punto de llegada en el eje de coordenadas (x,y)

* Distancia Manhattan: 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 (x1,y1) y (x2,y2) se calcula como se muestra en la Ecuación 2.

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

Donde:

(x1,y1) => punto de partida en el eje de coordenadas (x,y)

(x2,y2) => punto de llegada en el eje de coordenadas (x,y)

* Distancia de Chebyshev: 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.

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

Donde:

p = (p1, p2,…,pn) => es un vector que representa las coordenadas del punto p en el espacio n-dimensional. Cada pi​ (donde i va de 1 a n) es una coordenada del punto p.

q = (q1, q2,…,qn) => es un vector que representa las coordenadas del punto q en el espacio n-dimensional. Cada qi​ (donde i va de 1 a n) es una coordenada del punto q.

* Distancia Haversine: 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.

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

Donde:

r => radio de la esfera (por ejemplo, el radio promedio de la Tierra)

Δφ => diferencia de latitudes entre los dos puntos en radianes

Δλ => diferencia de longitudes entre los dos puntos en radianes

φ₁, φ₂ => latitudes de los dos puntos en radianes

**2. Técnicas de optimización**

La optimización de problemas de rutas de vehículos ha sido un área activa de investigación durante varias décadas. Debido a la complejidad combinatoria y la variedad de restricciones involucradas, se han desarrollado diversas técnicas, algoritmos y métodos para resolver estos problemas de manera eficiente. Estas técnicas se pueden clasificar en enfoques exactos y aproximados, siendo los enfoques aproximados los más utilizados en problemas de gran escala debido a las limitaciones de tiempo y recursos computacionales.

**2.1. Enfoques exactos**

Los enfoques exactos buscan encontrar la solución óptima, es decir, la mejor solución posible, sin aproximaciones. Sin embargo, estos enfoques pueden ser muy costosos computacionalmente, especialmente cuando el tamaño del problema aumenta, ya que los algoritmos exactos suelen tener complejidad exponencial.

* Programación Lineal (LP, *Linear Programming*, por sus siglas en inglés) y Programación Lineal Entera (ILP, *Integer Linear Programming*) son técnicas que se utilizan para modelar y resolver problemas de optimización mediante la formulación matemática de restricciones y objetivos. En el caso de los problemas de rutas de vehículos, el objetivo es minimizar la distancia recorrida, los costos o el tiempo, sujeto a una serie de restricciones, como la capacidad de los vehículos o las ventanas de tiempo de los clientes. La ILP es particularmente útil cuando las decisiones son discretas, como la asignación de vehículos a rutas, ya que las variables pueden tomar solo valores enteros (por ejemplo, un vehículo puede estar asignado a una ruta o no). Estos métodos pueden garantizar encontrar la solución óptima, pero son muy lentos para problemas grandes debido al tamaño del espacio de búsqueda.
* Ramificación y Acotación (*Branch and Bound*): es una técnica exacta que se utiliza para resolver problemas de optimización combinatoria, como el VRP. Se basa en explorar el espacio de soluciones de manera estructurada. El método divide el problema en subproblemas más pequeños (ramificación) y calcula límites (acotación) para descartar soluciones no viables o subóptimas, lo que reduce el espacio de búsqueda. Este enfoque es efectivo cuando se combinan técnicas de programación lineal con heurísticas para acotar las soluciones y acelerar el proceso.

**2.2. Enfoques aproximados**

Los enfoques aproximados no garantizan la solución óptima, pero pueden encontrar buenas soluciones en un tiempo razonable, lo que los hace más adecuados para problemas grandes. Estas técnicas incluyen heurísticas y metaheurísticas.

* Heurísticas: son métodos de búsqueda que proporcionan soluciones en un tiempo relativamente rápido, pero no garantizan que sean óptimas. Son útiles cuando se necesita una solución razonablemente buena sin la necesidad de calcular la mejor posible. Algunos ejemplos son:
  + Vecino más Cercano (*Nearest Neighbor*): este es un enfoque simple en el cual el vehículo comienza desde un punto de inicio y, en cada paso, elige el cliente más cercano que aún no ha sido atendido. Este proceso continúa hasta que se cubren todos los clientes. Aunque es rápido, a menudo no produce soluciones cercanas a la óptima.
  + Algoritmo de Clarke-Wright: Este es un enfoque de inserción que se basa en el ahorro de costos de las rutas. Comienza con una ruta para cada vehículo y luego va combinando rutas basándose en el ahorro de distancia que se produce al unir dos clientes en una sola ruta. Este algoritmo es especialmente útil cuando se tienen múltiples vehículos y un alto número de clientes.
* Metaheurísticas: las metaheurísticas son técnicas más sofisticadas que guían la búsqueda hacia mejores soluciones a través de un proceso iterativo, en el cual se exploran múltiples soluciones sin evaluar exhaustivamente todo el espacio de búsqueda. Estas técnicas pueden proporcionar soluciones de alta calidad en menos tiempo que los enfoques exactos y son muy populares en la optimización de rutas de vehículos.
  + Algoritmos Genéticos: son una metaheurística inspirada en el proceso de evolución natural. Se utilizan para encontrar soluciones aproximadas mediante la simulación de un proceso evolutivo en el que las soluciones (individuos) se cruzan, mutan y compiten por encontrar la mejor solución. En el contexto del VRP, un cromosoma puede representar una ruta de vehículo, y los operadores de cruce y mutación permiten explorar nuevas rutas potenciales.
  + Búsqueda Tabú: esta técnica mantiene una lista de soluciones anteriores para evitar que el algoritmo explore el mismo conjunto de soluciones repetidamente. Utiliza un enfoque de búsqueda local para explorar el espacio de soluciones y encontrar el mejor recorrido. La búsqueda tabú es especialmente útil en problemas de gran escala, ya que evita caer en óptimos locales, permitiendo explorar nuevas áreas del espacio de búsqueda.

En resumen, para resolver problemas de optimización de rutas de vehículos, los enfoques aproximados como las heurísticas y metaheurísticas son los más utilizados, ya que permiten obtener soluciones efectivas en tiempos razonables, lo cual es clave para enfrentar los desafíos logísticos de gran escala. Las herramientas como OR-Tools proporcionan una plataforma robusta para experimentar con diferentes algoritmos y enfoques, lo que facilita la implementación de soluciones para problemas reales de optimización.

**3.** **Herramientas de optimización**

Las herramientas de optimización son fundamentales para resolver problemas complejos en diversas disciplinas, proporcionando soluciones eficientes mediante el uso de algoritmos avanzados. Estas herramientas ofrecen desde técnicas clásicas como la programación lineal y entera, hasta métodos más avanzados como heurísticas y metaheurísticas, que ayudan a encontrar soluciones cercanas a la óptima cuando los métodos exactos no son viables. A continuación, se detallan algunas herramientas destacadas en el campo de la optimización.

**3.1. Xpress**

Es una herramienta poderosa y versátil utilizada en diversos sectores industriales y académicos. Desarrollada por FICO, está diseñada para abordar una amplia variedad de problemas de optimización, incluyendo problemas complejos de programación lineal, entera y cuadrática. Xpress se integra bien con varios lenguajes de programación como Python, C++, Java y otros, lo que la hace ideal para entornos de desarrollo variados. Es especialmente eficaz para resolver variantes del Vehicle Routing Problem (VRP), como CVRP, VRPTW, y MDVRP, entre otros. Sin embargo, su alto costo y la curva de aprendizaje compleja pueden ser un obstáculo, especialmente para usuarios sin experiencia previa en optimización o en el uso de esta herramienta. A pesar de ser una solución robusta, es menos accesible debido a su modelo comercial.

**3.2. PuLP**

Es una biblioteca de optimización de código abierto para Python que permite resolver problemas de optimización lineales y enteros de manera sencilla y flexible. PuLP es especialmente popular entre los científicos de datos y los ingenieros debido a su facilidad de uso y su integración fluida con el lenguaje Python. Los usuarios pueden formular problemas de optimización de manera clara y concisa, y PuLP se puede combinar con solvers de alto rendimiento como Xpress o CBC para mejorar su capacidad de resolución. Aunque es gratuita, su rendimiento puede ser inferior en problemas grandes y complejos en comparación con herramientas comerciales, lo que puede limitar su eficacia en algunos escenarios.

**3.2. SciPy**

Aunque SciPy es una biblioteca general de cálculo científico y matemático en Python, incluye un módulo de optimización que puede ser útil para problemas menos complejos. Aunque no está especializada en problemas complejos como el VRP, ofrece una variedad de algoritmos de optimización no lineales, minimización de funciones y ajuste de curvas. Además, cuenta con módulos para cálculos de distancias, como Euclidean, Manhattan y Chebyshev, lo que la convierte en una opción interesante para aplicaciones de optimización más simples. Sin embargo, el rendimiento de SciPy puede no ser tan eficiente como el de herramientas especializadas, especialmente cuando se trata de problemas más complejos o grandes. Además, la gestión de las dependencias, como la instalación de NumPy, puede ser un desafío para usuarios novatos.

**3.3. BHCVRP**

Es una biblioteca especializada en la resolución de variantes del VRP mediante el uso de heurísticas de construcción. Desarrollada inicialmente en Java y con una versión en Python en desarrollo, esta herramienta implementa siete heurísticas clásicas para resolver problemas como CVRP, HFVRP y TTRP. Entre las heurísticas disponibles se encuentran el Algoritmo de Ahorros, el Algoritmo de Barrido, y la Heurística del Vecino más Cercano, entre otras. Además, permite utilizar diferentes tipos de distancia, como Euclidean, Manhattan, Haversine y Chebyshev. Sin embargo, BHCVRP tiene limitaciones: no implementa metaheurísticas, lo que reduce su capacidad para mejorar soluciones iniciales y evitar caer en óptimos locales. Tampoco cuenta con una comunidad de soporte activa ni con documentación detallada, lo que puede hacer su uso y personalización más complicado para algunos usuarios.

**3.4. OR-Tools**

Es una suite de software de código abierto desarrollada por Google, diseñada para resolver problemas de optimización como rutas de vehículos (VRP), programación lineal, programación entera y problemas con restricciones. Es una herramienta versátil y fácil de usar, destacada en aplicaciones de logística, planificación de rutas y optimización de recursos. Entre sus ventajas se encuentran su naturaleza gratuita y de código abierto, su compatibilidad multiplataforma (Windows, Linux, macOS) y con lenguajes como Python, C++, Java y C#, así como su API intuitiva y bien documentada. Aunque no es tan rápido como solvers comerciales (por ejemplo, Gurobi o CPLEX), ofrece un alto rendimiento para problemas de mediana y gran escala, flexibilidad para abordar diversos problemas de optimización, y la posibilidad de integrarse con solvers externos para mejorar su eficiencia. Además, cuenta con una comunidad activa y soporte oficial de Google, lo que facilita su adopción y aprendizaje.

En aplicaciones reales, OR-Tools se utiliza en la planificación de rutas de vehículos (por ejemplo, flotas de reparto como Amazon o UPS, transporte público y logística de última milla), asignación de tareas (fabricación y programación de horarios), optimización de recursos (construcción y gestión de inventarios), secuenciación (líneas de producción y aerolíneas) y resolución de problemas con restricciones (puzzles y planificación de eventos). Su versatilidad y eficiencia lo convierten en una herramienta esencial para la optimización en diversos sectores.

En resumen, cada una de estas herramientas ofrece ventajas y limitaciones según el tipo de problema y el entorno de trabajo. Herramientas como Xpress y PuLP son muy útiles para resolver problemas complejos, pero pueden tener diferencias significativas en rendimiento y costos. SciPy es más adecuada para problemas más sencillos o como complemento a otras herramientas especializadas, mientras que BHCVRP está específicamente orientada a problemas de rutas de vehículos. OR-Tools destaca como una solución versátil y gratuita para optimización, especialmente en logística y planificación de rutas. Es fácil de usar, compatible con varios lenguajes y eficiente en problemas de mediana y gran escala. La elección final dependerá del problema, el presupuesto y la experiencia del usuario.

**4. OR-Tools**

OR-Tools (*Optimization Tools)* es una biblioteca de optimización de código abierto desarrollada por Google, diseñada para resolver una amplia gama de problemas de optimización combinatoria, como problemas de rutas de vehículos, problemas de programación lineal y otros tipos de problemas complejos. Es una herramienta muy popular debido a su eficiencia, flexibilidad y facilidad de uso, siendo ideal tanto para investigadores como para profesionales que necesitan resolver problemas de optimización en aplicaciones del mundo real.

Está diseñado específicamente para resolver problemas complejos de optimización combinatoria que no pueden ser abordados de manera eficiente mediante métodos exactos. La biblioteca es capaz de abordar problemas con miles de nodos y restricciones en tiempos razonables, lo que la convierte en una herramienta poderosa para la investigación operativa y la inteligencia artificial aplicada a la logística y otros campos.

Está disponible en varios lenguajes de programación, lo que permite su integración en diferentes aplicaciones. Además, puede incorporar y gestionar restricciones complejas en los problemas de optimización lo que es especialmente útil en problemas como el VRP con ventanas de tiempo (VRPTW), donde se deben manejar las restricciones de tiempo, o el VRP con capacidad (CVRP), donde los vehículos tienen límites de capacidad.

**4.1. Parámetros**

OR-Tools utiliza una variedad de parámetros para configurar y resolver problemas de optimización. Algunos de los más comunes incluyen:

1. Parámetros del solver:
   * time\_limit: establece un límite de tiempo para la ejecución del solver.
   * solution\_limit: limita el número de soluciones que el solver puede encontrar.
   * log\_search: habilita o deshabilita la impresión de logs durante la búsqueda.
2. Parámetros de búsqueda:
   * first\_solution\_strategy: define la estrategia para encontrar una solución inicial utilizando una heurística.
   * local\_search\_metaheuristic: especifica la metaheurística para mejorar la solución inicial.
3. Parámetros de rutas:
   * initial\_routes: permite especificar rutas iniciales para guiar la búsqueda.
   * vehicle\_capacities: define la capacidad de los vehículos en problemas de CVRP.
4. Parámetros de costos:
   * distance\_matrix: define una matriz de distancias personalizada.
   * cost\_callback: permite definir una función personalizada para calcular los costos entre nodos.

### **4.2. Manejo de rutas iniciales**

El parámetro initial\_routes en OR-Tools permite especificar rutas iniciales para guiar la búsqueda de soluciones. Esto es útil cuando se tiene conocimiento previo sobre cómo deberían ser las rutas (por ejemplo, basado en datos históricos o restricciones operativas). Los pasos para su ejecución son:

1. Definición de rutas iniciales: se proporciona una lista de rutas iniciales, donde cada ruta es una secuencia de nodos (clientes) que un vehículo debe visitar.
2. Uso en el solver: OR-Tools utiliza estas rutas como punto de partida para la búsqueda de soluciones, lo que puede acelerar la convergencia hacia una solución óptima o cercana a la óptima.
3. Flexibilidad: las rutas iniciales no tienen que ser óptimas; el solver las mejorará durante el proceso de optimización.

La biblioteca es una herramienta poderosa y flexible para resolver problemas de optimización, especialmente en aplicaciones de logística y planificación de rutas. Sus ventajas incluyen su facilidad de uso, su naturaleza de código abierto y su capacidad para manejar una amplia gama de problemas. Con parámetros como *initial\_routes*, los usuarios pueden guiar la búsqueda de soluciones para mejorar la eficiencia y la calidad de los resultados. Es una excelente opción tanto para proyectos académicos como para aplicaciones industriales.

**4.3. Heurísticas**

La biblioteca implementa diversas heurísticas que sirven como puntos de partida para mejorar la eficiencia en la búsqueda de soluciones. A continuación, se describen algunas de estas heurísticas, junto con sus siglas correspondientes, para facilitar su referencia en el resto del informe:

* **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 PATH\_CHEAPEST\_ARC, 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, OR-Tools 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 OR-Tools 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 BEST\_INSERTION, 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 BEST\_INSERTION 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 PARALLEL\_CHEAPEST\_INSERTION.
* **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 LOCAL\_CHEAPEST\_INSERTION, 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.

**4.4. Metaheurísticas**

Después de aplicar las técnicas iniciales (heurísticas), OR-Tools propone optimizar las soluciones mediante el uso de metaheurísticas robustas. Estas técnicas permiten explorar el espacio de búsqueda de manera eficiente, evitando caer en óptimos locales y mejorando la calidad de las soluciones. A continuación, se presentan las metaheurísticas implementadas, junto con sus siglas correspondientes para facilitar su referencia en el informe:

* 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.

**5. Descripción del experimento**

Para evaluar el desempeño de las heurísticas y metaheurísticas implementadas en OR-Tools, se realizaron experimentos sobre un conjunto de 67 instancias con distintas configuraciones y tamaños, utilizando las distancias Euclideana, Manhattan, Haversine y Chebyshev. Estas instancias corresponden a diferentes problemas de optimización combinatoria, incluyendo TSP, CVRP, VRPTW, VRPPD y MDVRP.

Cada instancia fue procesada aplicando diversas combinaciones entre las 15 heurísticas y las 5 metaheurísticas de la biblioteca, con el objetivo de analizar su eficiencia en la resolución de los problemas mencionados. Las heurísticas fueron utilizadas para generar soluciones iniciales, que posteriormente fueron mejoradas mediante la aplicación de una metaheurística.

Los experimentos fueron diseñados para responder a la siguiente pregunta clave: ¿Cuáles combinaciones de heurísticas y metaheurísticas logran las mejores soluciones?

Para responder a esto, cada combinación fue evaluada en términos de dos métricas principales:

* Distancia total recorrida: se busca minimizar la distancia recorrida por las rutas generadas, asegurando una mejor utilización de los recursos disponibles.
* Tiempo de ejecución: se mide en segundos y permite evaluar la eficiencia computacional de cada algoritmo.

Como parámetros generales, se estableció un tiempo máximo de ejecución de 50 segundos por instancia. Este límite se fijó tras observar en pruebas preliminares que la mayoría de los algoritmos de OR-Tools encontraban soluciones óptimas o cercanas en menos tiempo.

Para el cálculo de distancias, se consideraron cuatro tipos: Haversine, Chebyshev, Euclidean y Manhattan. En caso de no seleccionar ninguna, se utiliza Manhattan por defecto, ya que es la más empleada en aplicaciones reales.

Para asegurar una evaluación robusta, cada instancia se ejecutó 10 veces con cada combinación de heurística y metaheurística, sumando un total de 75 combinaciones (15 heurísticas × 5 metaheurísticas). Se registraron los valores de distancia recorrida y tiempo de ejecución, lo que permitió realizar un análisis comparativo detallado.

Adicionalmente, todas las ejecuciones, soluciones y datos utilizados en los experimentos están disponibles en el siguiente enlace de Google Drive:

<https://drive.google.com/drive/folders/1cmIz3nMJjpyfxUnMpvWiXmQz2h43qHmG?usp=drive_link>

**5.1. TSP**

Este problema recibe como parámetro un diccionario compuesto por:

* “locations”: una lista de ubicaciones representadas por tuplas con coordenadas cartesianas (X, Y).
* “num\_locations”: un valor entero que indica la cantidad de nodos en la instancia.
* “num\_vehicles”: un valor entero que representa la cantidad de vehículos; en este caso, siempre es 1.
* “depot”: un valor entero que indica el índice del depósito, generalmente 0.

Se utilizaron un total de 13 instancias, con un mínimo de 16 y un máximo de 1000 nodos. Estas instancias fueron seleccionadas de TSPLIB, una biblioteca estándar de instancias de referencia para el TSP, la cual contiene tanto problemas sintéticos como representaciones de problemas reales. La Tabla 1 muestra una descripción detallada de cada una, incluyendo el número de nodos y los rangos de coordenadas en los ejes X e Y.

**Tabla 1:** Descripción de las instancias TSP.

|  |  |  |  |
| --- | --- | --- | --- |
| **Instancia** | **Cantidad de nodos** | **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] |

* Instancias "pr" (pr76, pr107, pr124): provienen de problemas industriales reales relacionados con planificación logística y transporte.
* Instancias "ulysses" (ulysses16, ulysses22): basadas en coordenadas geográficas reales.
* Instancias "att" y "st" (att48, st70): se utilizan en problemas de telecomunicaciones y redes.
* Instancias "d" y "rd" (d493, d657, rd100): derivadas de aplicaciones en distribución de bienes y planificación de rutas.

La dificultad del problema no solo depende del número de nodos, sino también de la distribución espacial de los puntos.

* Instancias con coordenadas más compactas (por ejemplo, st70, ch150) pueden generar rutas más optimizadas, ya que los nodos están cerca entre sí.
* Instancias con coordenadas dispersas (como pr76, pr124, dsj1000) pueden dificultar la optimización, ya que las distancias entre nodos son más variadas y pueden generar caminos más largos.
* Instancias con grandes diferencias en coordenadas (como d493, d657, dsj1000) pueden representar problemas donde la variabilidad en la distribución de nodos hace que algunas rutas sean mucho más largas que otras.

En la Tabla 2 y la Tabla 3 se presentan los mejores resultados obtenidos para cada instancia, junto con la metaheurística y heurística utilizada, el tipo de distancia empleado y el tiempo de ejecución.

#### Rendimiento en instancias menores a 50 nodos (3 instancias):

🔹 Distancia Euclidiana

* La metaheurística GLS mostró el mejor desempeño al combinarse con las heurísticas PCA y ES.
* La heurística PCA también destacó en combinación con la metaheurística GD.

🔹 Distancia Manhattan

* La metaheurística TS obtuvo los mejores resultados en combinación con las heurísticas LCI y SCI.
* La heurística PMCA, junto con la metaheurística GTS, destacó en la instancia ulysses16.

#### Rendimiento en instancias entre 51 y 200 nodos (6 instancias):

🔹 Distancia Euclidiana

* Las metaheurísticas GTS y GD sobresalieron al combinarse con la heurística PCA.
* En las instancias pr76 y pr124, las mejores soluciones se obtuvieron con las heurísticas PCI y BI, respectivamente, en combinación con la metaheurística GTS.

🔹 Distancia Manhattan

* Las metaheurísticas TS, SA y GD predominaron al combinarse con la heurística PCA.
* En las instancias st70 y pr76, las combinaciones más eficientes fueron:
  + SCI + TS en st70
  + LCA + GLS en pr76

#### Rendimiento en instancias mayores a 200 nodos (4 instancias):

🔹 Distancia Euclidiana

* La metaheurística GD se destacó en combinación con la heurística PCA.
* En las instancias d493 y dsj1000, las mejores soluciones se lograron con:
  + TS + PCA en d493
  + GTS + GCA en dsj1000

🔹 Distancia Manhattan

* En este grupo, las mejores soluciones se lograron con:
  + GLS + SV
  + GD + FUMV
  + TS + GLA
  + GTS + PCI

En general, la heurística PCA mostró ser más efectiva al combinarse con las metaheurísticas GTS y GD. En ciertos casos, las metaheurísticas GLS y TS también lograron proporcionar soluciones competitivas.

**Tabla 2:** 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 |

Tabla 3: 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 |

**5.2. CVRP**

Este problema recibe como parámetro un diccionario compuesto por los siguientes elementos:

* "distance\_matrix": matriz que representa las distancias entre las coordenadas cartesianas (X, Y).
* "num\_locations": valor entero que indica la cantidad de nodos en la instancia.
* "num\_vehicles": valor entero que representa la cantidad de vehículos disponibles.
* "depot": valor entero que indica el índice del depósito, generalmente 0.
* "vehicle\_capacities": lista con los valores enteros de las capacidades de cada vehículo.
* “demands”: lista de valores enteros que representan las demandas de los clientes para cada nodo.

Además, como parámetros fijos configurados por defecto, se tienen:

* “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).

Se utilizaron un total de 20 instancias. De estas, 10 originalmente fueron utilizadas para el VRPTW. En este experimento, se ignoran las ventanas de tiempo para poder utilizar las instancias en CVRP. Las otras 10 instancias presentan ligeras transformaciones. En este caso, para cada instancia se mantienen los clientes, la cantidad de vehículos y sus capacidades, pero se adaptan para trabajar con un único depósito.

En el caso de las instancias HFVRP (*Heterogeneous Fleet Vehicle Routing Problem*), se mantienen los clientes y la cantidad de vehículos con sus datos originales, mientras que las 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 diferentes, según su capacidad.

Las instancias contienen un mínimo de 50 y un máximo de 600 nodos, y entre 3 y 150 vehículos. Así como capacidades homogéneas (primeras 15 instancias) y heterogéneas (últimas 5 instancias).

La Tabla 7 muestra una descripción detallada de cada una de estas instancias, incluyendo el número de nodos, la cantidad de vehículos, sus capacidades y los rangos de coordenadas en los ejes X e Y, así como el rango de demandas.

* Instancias CVRP:
  + CVRP\_1 (50 nodos, 5 vehículos) tiene coordenadas compactas, demandas entre [3, 41], y capacidad de vehículos de 200.
  + CVRP\_4 (75 nodos, 9 vehículos) con coordenadas más amplias y demandas de [1, 37], tiene vehículos con capacidad de 250.
  + CVRP\_p3 (75 nodos, 3 vehículos) similar a CVRP\_4, pero con menor cantidad de vehículos y capacidad de 480.
  + CVRP\_p5 (100 nodos, 5 vehículos) tiene una distribución más extensa de coordenadas y capacidades variadas (400).
  + CVRP\_p14 (360 nodos, 5 vehículos) presenta una distribución aún más dispersa y demandas más pequeñas entre [1, 12], con capacidad de vehículos de 500.

**Tabla 4:** Descripción de las instancias CVRP.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instancia** | **Cantidad de nodos** | **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 |

* Instancias C1\_2 y C1\_4:
  + Estas instancias varían entre 200 y 400 nodos y entre 50 y 100 vehículos. Tienen un rango de coordenadas moderado y demandas de [10, 40] con capacidad de vehículos constante en 200.
* Instancias C1\_6:
  + C1\_6\_1 a C1\_6\_3 (600 nodos, 150 vehículos) tienen coordenadas dispersas y capacidades de vehículos de 200, lo que aumenta la complejidad del problema.
* Instancias HFVRP:
  + Son versiones modificadas de las instancias CVRP con capacidades de vehículos más variadas (de 50 a 650). Esto permite mayor flexibilidad pero también complica la optimización de rutas.

Las instancias como CVRP\_p14, C1\_4\_1, C1\_4\_4 y las instancias de HFVRP con mayores rangos son más dispersas. Mientras que las instancias CVRP\_1, C1\_2\_1 a C1\_2\_8 y las versiones C1\_6 tienen distribuciones más compactas.

En la Tabla 8 y la Tabla 9 se presentan los mejores resultados obtenidos para cada instancia, junto con la metaheurística y heurística utilizada, el tipo de distancia empleado y el tiempo de ejecución.

Rendimiento en instancias con capacidades homogéneas (15 instancias):

🔹 Distancia Euclidiana

* 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.

🔹 Distancia Manhattan

* 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):

🔹 Distancia Euclidiana

* 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.

🔹 Distancia Manhattan

* 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.

**Tabla 5**: 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 |

En general, las heurísticas más destacadas, como PCA, PMCA y SV, son eficaces para generar soluciones iniciales sólidas al optimizar la asignación de rutas de manera rápida. Por otro lado, las metaheurísticas como GLS, TS y SA juegan un papel crucial al refinar estas soluciones, explorando el espacio de búsqueda de manera más exhaustiva para evitar óptimos locales y mejorar la calidad de las rutas finales.

**Tabla 6**: 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 |

**5.3. VRPTW**

Este problema recibe como parámetro un diccionario compuesto por los siguientes elementos:

* "locations": lista de ubicaciones representadas por tuplas con coordenadas cartesianas (X, Y).
* "num\_locations": valor entero que indica la cantidad de nodos en la instancia.
* "num\_vehicles": valor entero que representa la cantidad de vehículos disponibles.
* "depot": valor entero que indica el índice del depósito, generalmente 0.
* "vehicle\_capacity": valor entero que representa la capacidad de los vehículos.
* "demands": lista de valores enteros que representan las demandas de los clientes en cada nodo.
* "time\_windows": lista de tuplas que contienen los intervalos de tiempo en los que el vehículo debe llegar a cada nodo.
* "service\_time": valor entero que representa el tiempo que el vehículo debe permanecer en un nodo antes de continuar al siguiente.

Además, como parámetros fijos configurados por defecto, se tienen:

* "vehicle\_max\_distance": valor entero que representa la distancia máxima que pueden recorrer los vehículos = (1000).
* "vehicle\_max\_time": valor entero que representa el tiempo máximo permitido para que un vehículo complete su ruta = (1500).
* "vehicle\_speed": valor decimal que representa la velocidad de los vehículos = (83.33 km/h).

Se utilizaron un total de 23 instancias, con un mínimo de 200 y un máximo de 600 nodos, así como entre 50 y 150 vehículos.

La Tabla 10 presenta una descripción detallada de cada instancia, proporcionando información clave como el número de nodos, la cantidad de vehículos disponibles y los rangos de coordenadas en los ejes X e Y. Además, se incluyen los valores de demanda y las ventanas de tiempo, permitiendo una visión completa de las características de cada escenario analizado.

Las instancias presentadas corresponden a conjuntos de datos utilizados para evaluar el desempeño de algoritmos en problemas de ruteo de vehículos con restricciones de capacidad y ventanas de tiempo. Se organizan en tres grupos principales según su tamaño:

* Instancias C1\_2\_X (200 nodos, 50 vehículos)
  + Estas instancias cuentan con 200 nodos distribuidos en un espacio de coordenadas relativamente compacto ([0, 139] en X y [0, 139] en Y). Las demandas de los clientes oscilan entre 10 y 40 unidades, mientras que la capacidad de los vehículos es de 200. El número de vehículos es de 50, y las ventanas de tiempo varían significativamente entre las instancias, con rangos que van desde intervalos cortos hasta valores cercanos a 1600.

**Tabla 7:** Descripción de las instancias VRPTW.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instancia** | **Cantidad de nodos** | **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] |

* Instancias C1\_4\_X (400 nodos, 100 vehículos)
  + Estas instancias aumentan la complejidad al incluir 400 nodos en un área más amplia ([0, 200] en X y Y en la mayoría de los casos). Aunque las demandas por nodo siguen en el rango de 10 a 50 unidades, la mayor cantidad de nodos y vehículos (100) introduce una mayor variabilidad en la planificación de rutas. Algunas instancias tienen ventanas de tiempo cortas (por ejemplo, [1, 10]), lo que restringe fuertemente la flexibilidad de las rutas.
* Instancias C1\_6\_X (600 nodos, 150 vehículos)
  + Estas instancias representan los escenarios más grandes, con 600 nodos distribuidos en un área aún mayor ([0, 287] en X y [0, 294] en Y). Se mantiene la misma capacidad de los vehículos (200) y el mismo rango de demandas (10 a 40 unidades). Sin embargo, la cantidad de vehículos aumenta a 150, y las restricciones de ventanas de tiempo varían ampliamente, alcanzando hasta 1403 en algunas instancias.

La complejidad del problema no solo depende del número de nodos y vehículos, sino también de la distribución de los puntos en el espacio y la variabilidad de las restricciones de tiempo. Instancias con ventanas de tiempo más ajustadas y una distribución más dispersa tienden a ser más difíciles de resolver de manera óptima.

En la Tabla 11 y la Tabla 12 se presentan los mejores resultados obtenidos para cada instancia, junto con la metaheurística y heurística utilizada, el tipo de distancia empleado y el tiempo de ejecución.

Rendimiento en instancias de tamaño 200 (10 instancias):

🔹 Distancia Euclidiana

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

🔹 Distancia Manhattan

* 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):

🔹 Distancia Euclidiana

* 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.

🔹 Distancia Manhattan

* 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.

**Tabla 8:** 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 600 (3 instancias):

🔹 Distancia Euclidiana

* 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.

🔹 Distancia Manhattan

* 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.

Tabla 9: 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 |

En general, heurísticas como SCI, LCI, BI, PMCA, LCCI, SW, SV y FUMV optimizan la solución inicial y mejoran la asignación de rutas, mientras que metaheurísticas como GLS, GD, GTS y TS refinan estas soluciones explorando el espacio de búsqueda de manera eficiente. La combinación de estos enfoques logra un equilibrio entre exploración y explotación, resultando en soluciones de alta calidad.

**5.4. MDVRP**

Este problema recibe como parámetro un diccionario compuesto por los siguientes elementos:

* "distance\_matrix": una matriz que representa la distancia entre las coordenadas cartesianas (X, Y).
* "num\_locations": valor entero que indica la cantidad de nodos en la instancia.
* "num\_vehicles": valor entero que representa la cantidad de vehículos disponibles.
* "starts": lista de enteros conformada por los índices de los depósitos de salida para cada vehículo.
* "ends": lista de enteros conformada por los índices de los depósitos de entrada para cada vehículo.

Además, como parámetros fijos configurados por defecto, se tienen:

* “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).
* “vehicle\_maximum\_travel\_distance”: valor entero que representa la distancia máxima a recorrer por los vehículos = (500).

Se utilizaron un total de 8 instancias, con un mínimo de 10 y un máximo de 144 nodos, así como entre 2 y 6 vehículos.

La Tabla 13 presenta una descripción detallada de cada instancia, proporcionando información clave como el número de nodos, la cantidad de depósitos, la cantidad de vehículos disponibles y los rangos de coordenadas en los ejes X e Y.

* Dificultad del problema:
  + Instancias con coordenadas compactas (como p31, p1, p2) facilitan la optimización de rutas, ya que los nodos están relativamente cerca.
  + Instancias con coordenadas dispersas (como p5, p6 y p19) pueden generar rutas más largas y, por lo tanto, son más difíciles de optimizar debido a la variabilidad en las distancias entre los nodos.

**Tabla 10:** Descripción de las instancias MDVRP.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instancia** | **Cantidad de nodos** | **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 14 y la Tabla 15 se presentan los mejores resultados obtenidos para cada instancia, junto con la metaheurística y heurística utilizada, el tipo de distancia empleado y el tiempo de ejecución.

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

🔹 Distancia Euclidiana

* 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.

🔹 Distancia Manhattan

* 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):

🔹 Distancia Euclidiana

* 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.

🔹 Distancia Manhattan

* 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`.

**Tabla 11:** Mejores resultados de las instancias MDVRP utlizando 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 |

Tabla 12: Mejores resultados de las instancias MDVRP utlizando 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 |

En general, los mejores resultados provienen de diversas combinaciones de técnicas. Las heurísticas más destacadas fueron LCI, PCI, PMCA y PCA, junto con las metaheurísticas GLS, SA, GTS y GD, siendo GD la más sobresaliente entre ellas.

**5.5. VRPPD**

Este problema recibe como parámetro un diccionario compuesto por los siguientes elementos:

* "distance\_matrix": una matriz que representa la distancia entre las coordenadas cartesianas (X, Y).
* "num\_locations": valor entero que indica la cantidad de nodos en la instancia.
* "num\_vehicles": valor entero que representa la cantidad de vehículos disponibles.
* "depot": valor entero que indica el índice del depósito, generalmente 0.
* “pickups\_deliveries”: una matriz compuesta por los índices de los nodos donde se recoge y entrega la mercancía.

Además, como parámetros fijos configurados por defecto, se tienen:

* “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).
* “vehicle\_maximum\_travel\_distance”: valor entero que representa la distancia máxima a recorrer por los vehículos = (500).

Se utilizaron un total de 10 instancias, con un mínimo de 10 y un máximo de 75 nodos, así como entre 2 y 15 vehículos. Las instancias provienen de los problemas CVRP y MDVRP, manteniendo las características comunes con el VRPPD, y añadiendo de forma aleatoria los índices de recogida y entrega.

La Tabla 16 presenta una descripción detallada de cada instancia, proporcionando información clave como el número de nodos, la cantidad de vehículos disponibles y los rangos de coordenadas en los ejes X e Y.

* Dificultad del problema:
  + Las instancias donde los puntos de recogida y entrega están más cercanos entre sí suelen ser más fáciles de optimizar, ya que los vehículos pueden recorrer distancias más cortas.
  + Sin embargo, como estos puntos se generan de forma aleatoria, en algunas instancias los puntos de recogida y entrega pueden quedar muy alejados, lo que dificulta la optimización, ya que se generan rutas más largas y complejas.

**Tabla 13:** Descripción de las instancias VRPPD.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Instancia** | **Cantidad de nodos** | **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 17 y la Tabla 18 se presentan los mejores resultados obtenidos para cada instancia, junto con la metaheurística y heurística utilizada, el tipo de distancia empleado y el tiempo de ejecución.

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

🔹 Distancia Euclidiana

* 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`.

🔹 Distancia Manhattan

* 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 heuristicas FUMV y PCA, respectivamente.
* En la instancia `p2` se destaca la heurística AU junto a la metaheurística SA.

En general, la combinación de la heurística PCA con las metaheurísticas GD, GLS y SA, así como la combinación de las heurísticas PCA y AU con las metaheurísticas GLS y SA, lograron obtener los mejores resultados en la mayoría de las instancias evaluadas.

**Tabla 14:** 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 |

Tabla 15: 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 |

5.6. Conclusiones del experimento

* En las diferentes variantes del VRP, las heurísticas más destacadas fueron PCA, PMCA, SV, LCI, y FUMV. Las combinaciones más exitosas fueron aquellas que integraron estas heurísticas con diferentes metaheurísticas, logrando un equilibrio adecuado.
* GD y GLS fueron las metaheurísticas más destacadas en la mayoría de las instancias, con GTS y TS también obteniendo buenos resultados en ciertas categorías. SA, aunque efectiva, no tuvo el mismo nivel de consistencia que las otras.
* Dependiendo de la métrica de distancia (Euclidiana o Manhattan), las combinaciones óptimas variaron, pero en general, las combinaciones de heurísticas como PCA y PMCA con metaheurísticas como GD, GLS, SA y GTS ofrecieron buenos resultados. En algunas instancias específicas, la heurística y la metaheurística más efectiva variaron, como en las instancias CVRP\_2 y p2, donde ciertas combinaciones se destacaron. En CVRP\_2, la combinación PCA + GD y LCI + GD fueron las más efectivas. En p2, PMCA + GTS y LCI +SA se destacaron, dependiendo de la complejidad y el tamaño de las instancias.
* Para instancias de mayor tamaño (como las que superan los 200 nodos), las combinaciones más efectivas fueron las que integraron metaheurísticas como GD, GLS, y GTS con heurísticas como PCA, PMCA y SV. La complejidad de las instancias más grandes requirió una mayor exploración del espacio de soluciones, y las metaheurísticas como SA y TS también jugaron un papel importante en refinar las soluciones.
* En varias instancias, se observó que las combinaciones más efectivas eran variadas, lo que resalta la importancia de una estrategia flexible que permita adaptarse a diferentes características de las instancias. Por ejemplo, en el problema del VRPTW y el CVRP, las combinaciones de heurísticas y metaheurísticas variaban según el tipo de capacidad de los vehículos. (homogénea o heterogénea).
* Para el problema del TSP, la heurística PCA combinada con las metaheurísticas GTS y GD mostró resultados sobresalientes en la mayoría de las instancias. La metaheurística GTS junto con PCA fue particularmente eficaz en las instancias más grandes.

**6. Prueba de Friedman**

La prueba de Friedman es una técnica estadística no paramétrica utilizada para evaluar si existen diferencias significativas entre tres o más grupos relacionados. Este test de comparación de rangos es especialmente útil en estudios donde los datos son dependientes o emparejados, es decir, cuando las observaciones de los grupos están interrelacionadas.

En su funcionamiento, se asignan rangos a las observaciones dentro de cada grupo o bloque, y luego se comparan las sumas de estos rangos entre los grupos. Si las diferencias entre los rangos son lo suficientemente grandes, se rechaza la hipótesis nula, indicando que existen diferencias estadísticamente significativas entre los grupos.

Esta prueba es apropiada cuando se comparan más de dos grupos relacionados, como diferentes algoritmos o tratamientos aplicados sobre un mismo conjunto de datos, y cuando no se puede asumir que los datos siguen una distribución normal.

En este epígrafe se aplica la prueba de Friedman a todos los problemas analizados previamente utilizando el software Keel de Java. Para cada instancia de cada problema, se generan soluciones en 10 iteraciones (las mejores soluciones se reflejaron en el epígrafe 5), y a partir de estas se calcula el promedio de cada algoritmo. Este procedimiento es necesario para realizar la prueba estadística, dado que los algoritmos evaluados son de naturaleza aproximada.

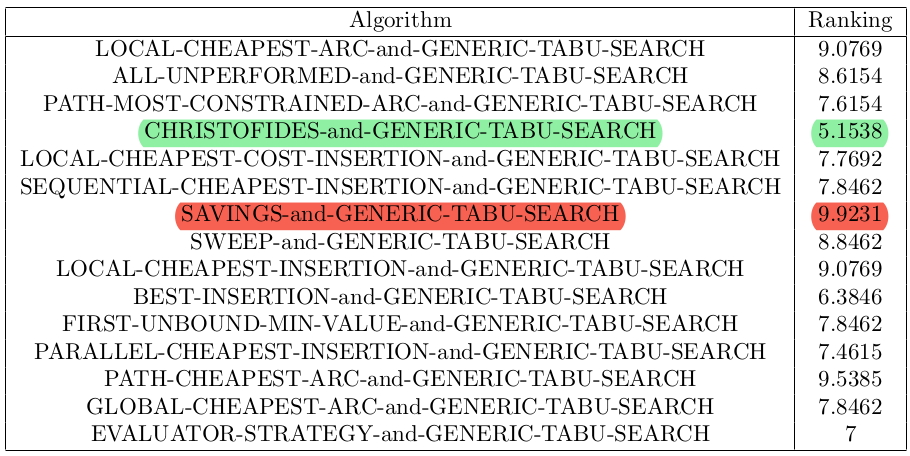
El objetivo principal es determinar si existen diferencias significativas entre los algoritmos y establecer un ranking de desempeño, tanto en calidad de las soluciones como en tiempo de ejecución. Inicialmente, la prueba se aplica para identificar la mejor heurística para cada metaheurística, obteniendo así cinco combinaciones óptimas. Posteriormente, estas combinaciones son sometidas nuevamente a la prueba de Friedman para generar un ranking final y seleccionar la mejor configuración global.

**6.1. Calidad de las soluciones**

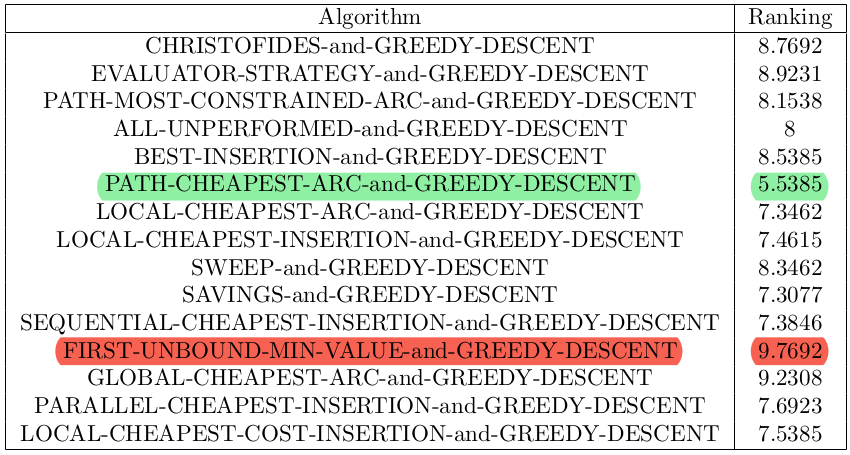
En este apartado se aplica la prueba de Friedman para evaluar la calidad de las soluciones en cada uno de los problemas analizados previamente. Se presentan imágenes con los resultados del ranking, inicialmente para cada metaheurística y, posteriormente, el ranking general. Esto permite identificar el mejor algoritmo según cada variante del VRP y el método de cálculo de distancia utilizado, así como la heurística óptima para cada metaheurística. En todas las ilustraciones el color rojo refleja el peor algoritmo y el verde el mejor.

**6.1.1. Resultados para TSP**

Los resultados obtenidos mediante el uso de las distancias mencionadas en epígrafes anteriores para las cinco metaheurísticas proporcionadas por OR-Tools se presentan en las ilustraciones siguientes. Los análisis estadísticos indican que en todos los casos el p-valor obtenido fue superior a 0.05, lo que sugiere que no existen diferencias estadísticamente significativas entre los algoritmos evaluados. Por último, luego de los rankings por cada metaheurística, se muestra el ranking final al aplicar la prueba por segunda vez a los mejores algoritmos obtenidos en los resultados anteriores.



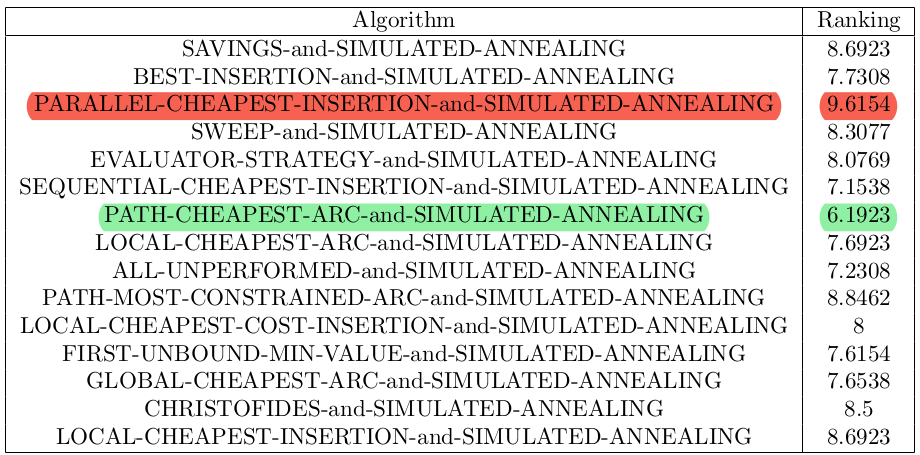
**Ilustración 2**: Ranking de mejores heurísticas para la metaheurística GTS en cuanto a calidad de las soluciones utilizando Euclidean en el problema TSP.



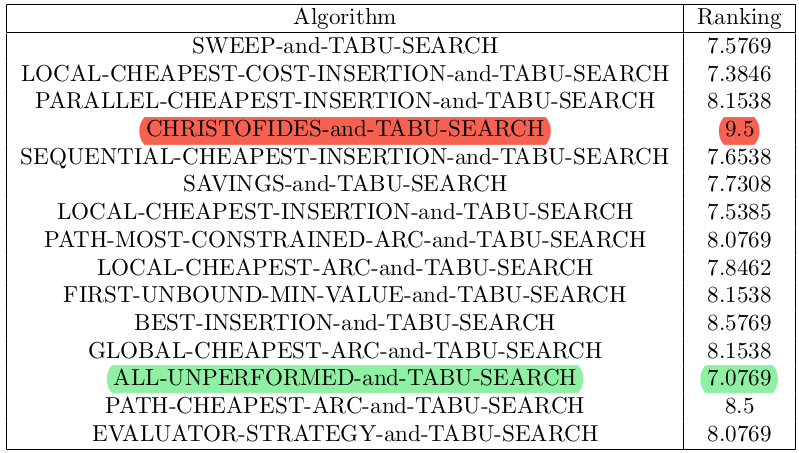
**Ilustración 3**: Ranking de mejores heurísticas para la metaheurística GD en cuanto a calidad de las soluciones utilizando Euclidean en el problema TSP.



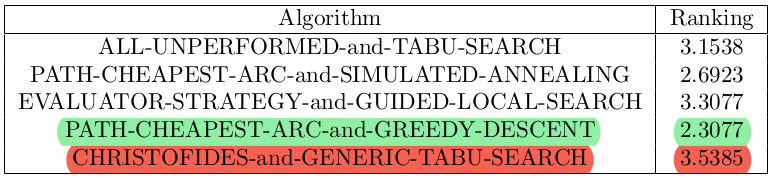
**Ilustración 4**: Ranking de mejores heurísticas para la metaheurística GLS en cuanto a calidad de las soluciones utilizando Euclidean en el problema TSP.

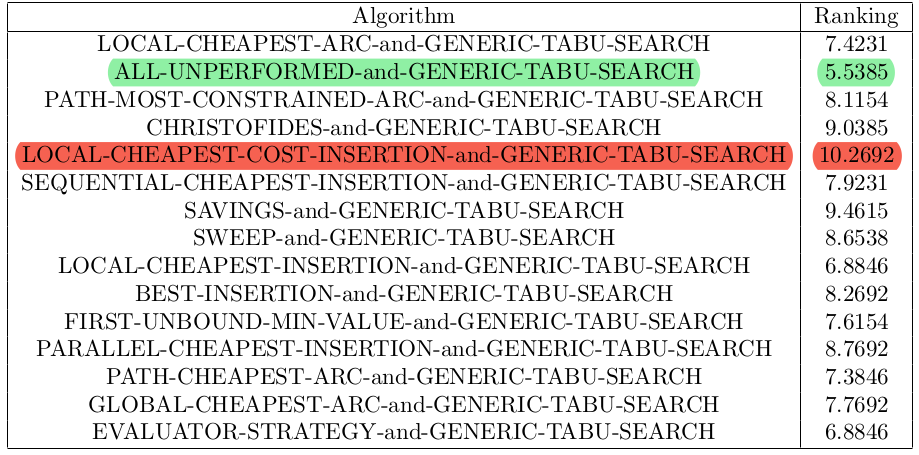


**Ilustración 5**: Ranking de mejores heurísticas para la metaheurística SA en cuanto a calidad de las soluciones utilizando Euclidean en el problema TSP.

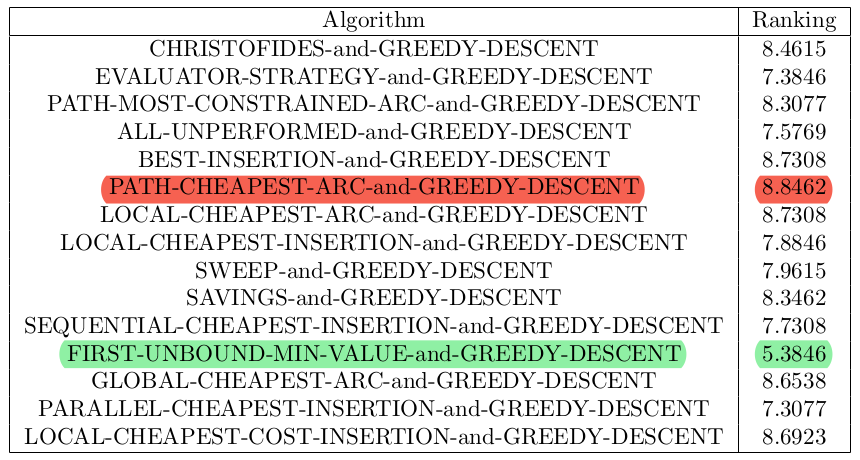


**Ilustración 6**: Ranking de mejores heurísticas para la metaheurística TS en cuanto a calidad de las soluciones utilizando Euclidean en el problema TSP.

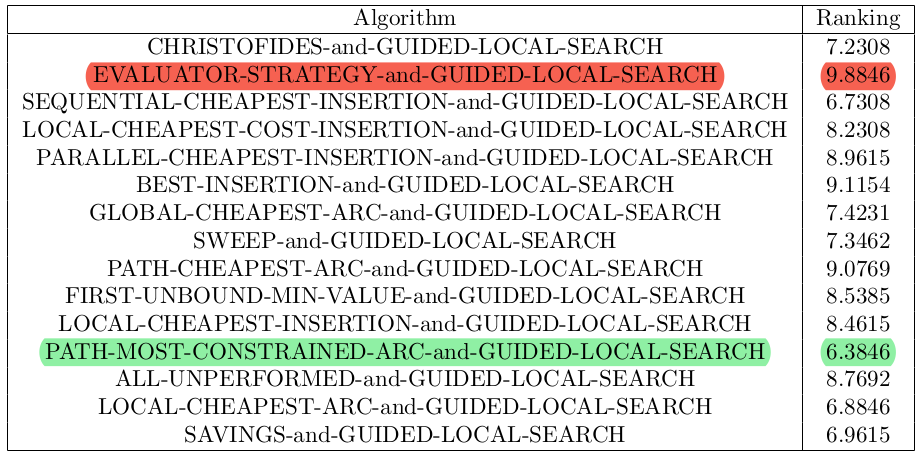
**Il****ustración 7**: Ranking de mejores algoritmos para TSP utilizando Euclidean.



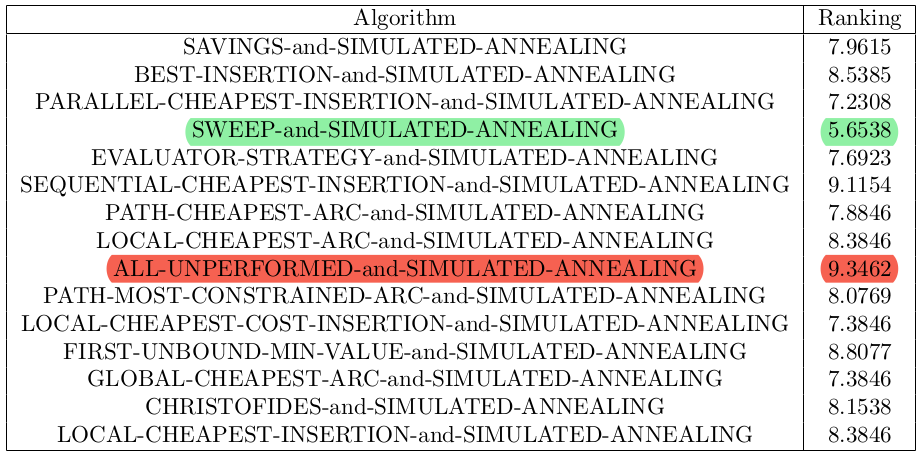
**Ilustración 8**: Ranking de mejores heurísticas para la metaheurística GTS en cuanto a calidad de las soluciones utilizando Manhattan en el problema TSP.



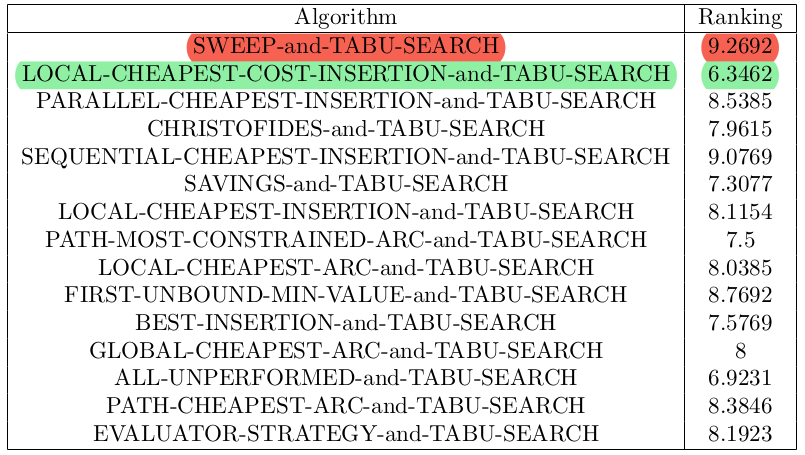
**Ilustración 9**: Ranking de mejores heurísticas para la metaheurística GD en cuanto a calidad de las soluciones utilizando Manhattan en el problema TSP.



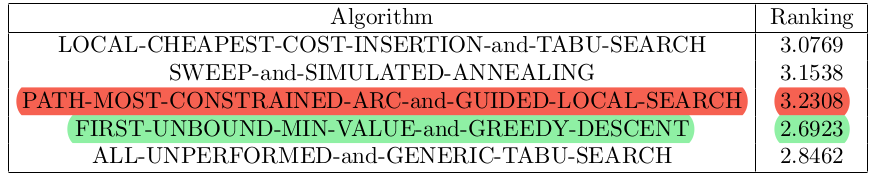
**Ilustración 10**: Ranking de mejores heurísticas para la metaheurística GLS en cuanto a calidad de las soluciones utilizando Manhattan en el problema TSP.



**Ilustración 11**: Ranking de mejores heurísticas para la metaheurística SA en cuanto a calidad de las soluciones utilizando Manhattan en el problema TSP.



**Ilustración 12**: Ranking de mejores heurísticas para la metaheurística TS en cuanto a calidad de las soluciones utilizando Manhattan en el problema TSP.



**Ilustración 13**: Ranking de mejores algoritmos para TSP utilizando Manhattan.

**6.2. Resultados para CVRP**

Los resultados obtenidos mediante el uso de ambas distancias revelaron un p-value extremadamente bajo (0.0), lo que indica que existen diferencias estadísticamente significativas entre los distintos pares de algoritmos analizados. Estas diferencias sugieren que el rendimiento de los algoritmos varía considerablemente en función de las combinaciones de estrategias utilizadas.

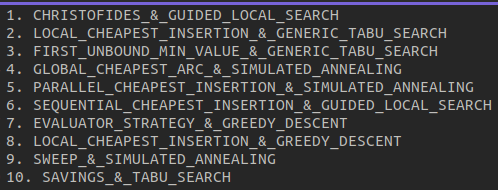
En particular, algunas combinaciones de algoritmos y estrategias demostraron un rendimiento superior en la resolución de los problemas planteados, destacándose las siguientes como las más efectivas:

* Euclidean:
  + CHRISTOFIDES\_&\_GUIDED\_LOCAL\_SEARCH
  + LOCAL\_CHEAPEST\_INSERTION\_&\_GENERIC\_TABU\_SEARCH
* Manhattan:
  + PATH\_CHEAPEST\_ARC\_&\_GUIDED\_LOCAL\_SEARCH
  + PATH\_CHEAPEST\_ARC\_&\_SIMULATED\_ANNEALING

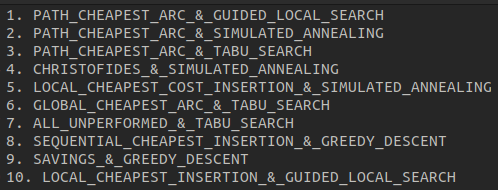
Por otro lado, otras combinaciones mostraron un desempeño inferior y podrían considerarse menos eficientes, como es el caso de:

* Euclidean:
  + PATH\_CHEAPEST\_ARC\_&\_GREEDY\_DESCENT
* Manhattan:
  + SWEEP\_&\_GENERIC\_TABU\_SEARCH

Este análisis permite generar un *ranking* de los algoritmos mostrado en la Ilustración 4 y la Ilustración 5 (que presenta los primeros 10 algoritmos) para la distancia Euclidean y Manhattan, respectivamente.



**Ilustración 14**: Ranking de mejores algoritmos para CVRP utilizando Euclidean.



**Ilustración 15**: Ranking de mejores algoritmos para CVRP utilizando Manhattan.

**6.3. Resultados para VRPTW**

Los resultados obtenidos mediante el uso de ambas distancias revelaron un p-value extremadamente bajo (0.0) al igual que en el epígrafe anterior, lo que indica que existen diferencias estadísticamente significativas entre los distintos pares de algoritmos analizados.

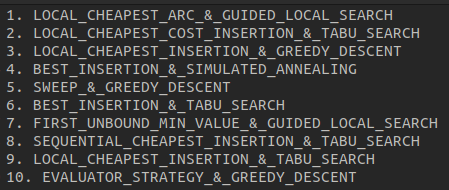
En particular, algunas combinaciones de algoritmos y estrategias demostraron un rendimiento superior en la resolución de los problemas planteados, destacándose las siguientes como las más efectivas:

* Euclidean:
  + LOCAL\_CHEAPEST\_ARC\_&\_GUIDED\_LOCAL\_SEARCH
  + LOCAL\_CHEAPEST\_COST\_INSERTION\_&\_TABU\_SEARCH
* Manhattan:
  + PARALLEL\_CHEAPEST\_INSERTION\_&\_GENERIC\_TABU\_SEARCH
  + GLOBAL\_CHEAPEST\_ARC\_&\_GENERIC\_TABU\_SEARCH

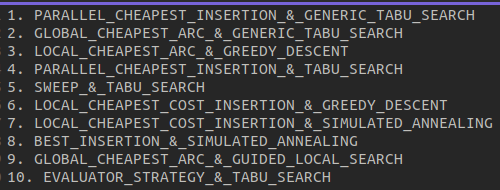
Por otro lado, otras combinaciones mostraron un desempeño inferior en ambas distancias y podrían considerarse menos eficientes, como es el caso de:

* PATH\_CHEAPEST\_ARC\_&\_GREEDY\_DESCENT

Este análisis permite generar un *ranking* de los algoritmos mostrado en la Ilustración 6 y la Ilustración 7 (que presenta los primeros 10 algoritmos) para la distancia Euclidean y Manhattan, respectivamente.



**Ilustración 16**: Ranking de mejores algoritmos para VRPTW utilizando Euclidean.



**Ilustración 17**: Ranking de mejores algoritmos para VRPTW utilizando Manhattan.

**6.4. Resultados para MDVRP**

El análisis realizado con la distancia Euclidiana mostró un p-value extremadamente bajo (8.667990682398551e-137), mientras que con la distancia Manhattan se obtuvo un valor igualmente reducido (1.624231102415977e-107). Esto evidencia la existencia de diferencias estadísticamente significativas entre los distintos pares de algoritmos evaluados. Tales diferencias indican que el desempeño de los algoritmos varía notablemente según las combinaciones de estrategias empleadas.

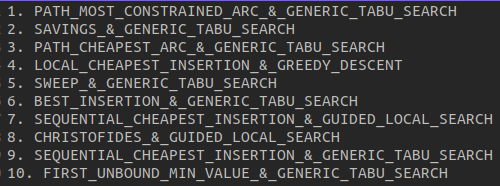
En particular, ciertas combinaciones de algoritmos y estrategias lograron un desempeño sobresaliente en la resolución de los problemas planteados, destacándose las siguientes como las más eficientes:

* Euclidean:
  + PATH\_MOST\_CONSTRAINED\_ARC\_&\_GENERIC\_TABU\_SEARCH
  + SAVINGS\_&\_GENERIC\_TABU\_SEARCH
* Manhattan:
  + PATH\_CHEAPEST\_ARC\_&\_GREEDY\_DESCENT
  + BEST\_INSERTION\_&\_GREEDY\_DESCENT

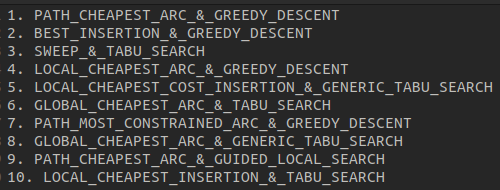
Por otro lado, otras combinaciones mostraron un desempeño inferior y podrían considerarse menos eficientes, como es el caso de:

* Euclidean:
  + PATH\_CHEAPEST\_ARC\_&\_GREEDY\_DESCENT
* Manhattan:
  + PARALLEL\_CHEAPEST\_INSERTION\_&\_SIMULATED\_ANNEALING

Este estudio permite elaborar un *ranking* de los algoritmos, representado en la Ilustración 8 y la Ilustración 9, donde se muestran los 10 algoritmos mejor posicionados al emplear las distancias Euclidiana y Manhattan, respectivamente. Esto proporciona una guía para seleccionar los algoritmos más apropiados en la resolución de problemas similares, optimizando así los recursos computacionales y el desarrollo de estrategias más eficientes para abordar este tipo de desafíos.

****

**Ilustración 18**: Ranking de mejores algoritmos para MDVRP utilizando Euclidean.

****

**Ilustración 19**: Ranking de mejores algoritmos para MDVRP utilizando Manhattan.

**6.5. Resultados para VRPPD**

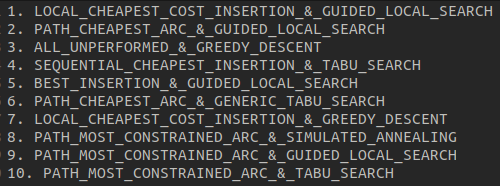
El análisis realizado utilizando la distancia Euclidiana arrojó un p-value de 0.0, mientras que con la distancia Manhattan se obtuvo un valor igualmente bajo (4.652793683894027e-133). Estos resultados evidencian diferencias estadísticamente significativas entre los distintos pares de algoritmos evaluados, lo que sugiere que el desempeño de cada algoritmo varía sustancialmente en función de las combinaciones de estrategias empleadas. En particular, ciertas combinaciones de algoritmos y estrategias destacaron por su desempeño superior en la resolución de los problemas analizados, identificándose las siguientes como las más eficientes:

* Euclidean:
  + LOCAL\_CHEAPEST\_COST\_INSERTION & GUIDED\_LOCAL\_SEARCH
  + PATH\_CHEAPEST\_ARC & GUIDED\_LOCAL\_SEARCH
* Manhattan:
  + EVALUATOR\_STRATEGY & SIMULATED\_ANNEALING
  + PARALLEL\_CHEAPEST\_INSERTION & SIMULATED\_ANNEALING

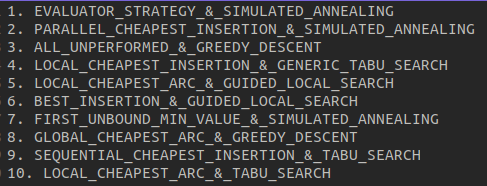
Por otro lado, algunas combinaciones presentaron un rendimiento inferior, lo que sugiere que podrían ser menos eficientes en la resolución de estos problemas:

* Euclidean:
  + FIRST\_UNBOUND\_MIN\_VALUE & GUIDED\_LOCAL\_SEARCH
* Manhattan:
  + SAVINGS & SIMULATED\_ANNEALING

Este estudio permite establecer un ranking de los algoritmos, representado en la Ilustración 10 y la Ilustración 11, donde se muestran los 10 algoritmos con mejor desempeño según las distancias Euclidiana y Manhattan, respectivamente.

****

**Ilustración 20**: Ranking de mejores algoritmos para VRPPD utilizando Euclidean.



**Ilustración 21**: Ranking de mejores algoritmos para VRPPD utilizando Manhattan.

**7. Comparación con otras herramientas**

En el contexto de los VRP, existen diversas bibliotecas que ofrecen algoritmos para resolver variantes del problema, y es fundamental evaluar su rendimiento bajo diferentes condiciones. Este análisis comparativo tiene como objetivo medir la eficiencia de OR-Tools, una biblioteca ampliamente utilizada para la resolución de VRP, en comparación con otras herramientas especializadas en optimización.

El enfoque de este análisis radica en evaluar no solo los resultados obtenidos, sino también la eficiencia con la que cada herramienta ejecuta los algoritmos y devuelve una solución. Factores clave como el tiempo de ejecución, la distancia total recorrida por todas las rutas y la cantidad de rutas generadas son esenciales para evaluar el rendimiento de cada herramienta. Al utilizar diferentes instancias de los problemas VRP, se busca determinar qué herramienta ofrece los mejores resultados en términos de precisión y rapidez, así como qué herramienta se adapta mejor a problemas con características particulares.

Además, para profundizar en la validez estadística de los resultados obtenidos, se realiza una prueba de Wilcoxon, que permite hallar las diferencias significativas entre las soluciones generadas por las distintas bibliotecas. Esta prueba es útil para identificar si existen variaciones importantes en los resultados, proporcionando una evaluación no paramétrica que no asume distribuciones específicas en los datos.

Por otro lado, la prueba de Spearman también se utiliza en este análisis para evaluar la correlación entre los resultados obtenidos de las diferentes bibliotecas. A diferencia de la prueba de Wilcoxon, que compara las diferencias directas, la prueba de Spearman mide el grado de concordancia entre las posiciones relativas de las soluciones generadas por cada biblioteca, sin asumir una relación lineal entre los valores. Esto es particularmente importante cuando se busca entender si las bibliotecas presentan un comportamiento similar en cuanto a la eficiencia de las soluciones a medida que varían las instancias del problema.

**7.1. BHCVRP vs OR-Tools**

En este apartado, se presenta un análisis comparativo de los resultados obtenidos mediante el uso de OR-Tools y la implementación en Python de la biblioteca BHCVRP, mencionada previamente en el epígrafe 3. Para este estudio, se emplearon 20 instancias de la variante CVRP, de las cuales 5 corresponden a una flota heterogénea y 15 a una flota homogénea. Estas instancias son las mismas que se detallaron en el epígrafe 5.2, donde se describen sus características principales y se presentan los mejores resultados alcanzados utilizando los algoritmos de OR-Tools.

El análisis se llevó a cabo considerando dos tipos de distancias: Euclidiana y Manhattan. Estos enfoques permiten evaluar el comportamiento de los algoritmos en diferentes contextos geométricos, lo que resulta relevante para aplicaciones prácticas donde la métrica de distancia puede variar según el escenario. Además, se evaluaron varios factores clave que influyen en la efectividad de las soluciones propuestas:

1. Tiempo de ejecución: se midió el tiempo requerido por cada herramienta para encontrar y devolver una solución viable. Este aspecto es de vital importancia en aplicaciones del mundo real, donde la capacidad de proporcionar respuestas rápidas puede ser un factor determinante para la toma de decisiones operativas.
2. Calidad de las soluciones: se analizó la calidad de las soluciones en términos de dos métricas principales:
   * Distancia total recorrida: se evaluó la eficiencia de las rutas calculadas midiendo la distancia total recorrida por todos los vehículos. Una menor distancia implica una mayor eficiencia operativa y una reducción directa en los costos asociados al combustible, el mantenimiento y el tiempo de viaje.
   * Número de rutas requeridas: se examinó la cantidad de rutas necesarias para satisfacer la demanda de todas las ubicaciones. Un menor número de rutas puede traducirse en una optimización de los recursos disponibles, como vehículos y conductores.

Este análisis exhaustivo permite no solo comparar el rendimiento de ambas herramientas, sino también identificar sus fortalezas y debilidades en diferentes escenarios. Los resultados obtenidos proporcionan elementos valiosos para la selección de la herramienta más adecuada según los requisitos específicos de cada aplicación, ya sea en términos de velocidad, eficiencia o calidad de las soluciones.

A continuación, la Tabla 19 y la Tabla 20 reflejan la comparación de los resultados utilizando la distancia euclideana y la distancia manhattan, respectivamente.

**Tabla 16:** Comparación entre BHCVRP vs OR-Tools 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 17:** Comparación entre BHCVRP vs OR-Tools 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 | 0.4817 | 2 972.00 | 4 | **0.11** |
| **CVRP\_4** | **934.00** | 6 | 2.8849 | 5 526.00 | 6 | **0.13** |
| **CVRP\_p3** | **897.00** | 3 | 16.3997 | 5 726.00 | 3 | **0.27** |
| **CVRP\_p5** | **1 029.00** | 4 | 5.9259 | 6 322.00 | 4 | **0.48** |
| **CVRP\_p14** | **6 460.00** | 5 | 232.7621 | 12 580.00 | 4 | **6.14** |
| **HFVRP\_1** | **640.00** | 2 | 1.1869 | 3 676.00 | 4 | **0.20** |
| **HFVRP\_4** | **892.00** | 3 | 16.5868 | 5 086.00 | 5 | **0.37** |
| **HFVRP\_p3** | **835.00** | 3 | 4.7913 | 5 612.00 | 3 | **0.24** |
| **HFVRP\_p5** | **1 010.00** | 3 | 2.4405 | 1 938.00 | 3 | **0.49** |
| **HFVRP\_p14** | **7 800.00** | 4 | 95.0211 | 12 660.00 | 4 | **6.38** |
| **C1\_2\_1** | **3 118.00** | 18 | 19.5793 | 3 684.00 | 18 | **1.58** |
| **C1\_2\_4** | **3 365.00** | 18 | 19.9045 | 3 684.00 | 18 | **1.58** |
| **C1\_2\_5** | **3 189.00** | 18 | 18.1829 | 3 684.00 | 18 | **1.58** |
| **C1\_2\_6** | **3 140.00** | 18 | 19.9371 | 3 684.00 | 18 | **1.59** |
| **C1\_2\_8** | **3 234.00** | 18 | 18.4007 | 3 684.00 | 18 | **1.58** |
| **C1\_4\_1** | **8 053.00** | 36 | 240.5319 | 9 112.00 | 36 | **7.09** |
| **C1\_4\_4** | **8 342.00** | 36 | 240.7538 | 9 112.00 | 36 | **7.14** |
| **C1\_6\_1** | **16 410.00** | 56 | 1 254.2598 | 18 760.00 | 56 | **17.72** |
| **C1\_6\_2** | **16 740.58** | 56 | 1 582.2116 | 18 760.00 | 56 | **17.65** |
| **C1\_6\_3** | **16 360.97** | 56 | 1 358.9239 | 18 760.00 | 56 | **17.57** |

Por un lado, BHCVRP logró soluciones de menor costo en todas las instancias evaluadas, lo que indica que esta biblioteca es más eficiente en la optimización del recorrido total. Por ejemplo, en la instancia CVRP\_1, BHCVRP obtuvo una distancia total de 558.89, mientras que OR-Tools presentó una solución con 2152.00, lo que representa una diferencia significativa. Esta tendencia se mantiene en otras instancias, como HFVRP\_1 (550.34 vs. 3225.00), lo que refuerza la superioridad de BHCVRP en términos de calidad de las rutas generadas.

Sin embargo, esta ventaja en la calidad de la solución tiene un costo considerable en términos de tiempo de ejecución. OR-Tools es consistentemente más rápido en todas las instancias evaluadas. En la instancia CVRP\_p3, BHCVRP tardó 15.75 segundos, mientras que OR-Tools solo 0.27 segundos, lo que implica que esta última encuentra soluciones de manera más eficiente. Esta diferencia es aún más pronunciada en instancias grandes, como HFVRP\_p14, donde BHCVRP requirió 5226 segundos para resolver el problema, mientras que OR’Tools lo hizo en apenas 12.70 segundos.

En instancias pequeñas, la diferencia de calidad de solución entre ambos algoritmos es significativa a favor de BHCVRP, lo que lo hace una mejor opción si la prioridad es optimizar costos en problemas con pocos nodos. No obstante, a medida que el tamaño del problema aumenta, la diferencia en distancia total entre ambas herramientas se reduce. Por ejemplo, en la instancia C1\_6\_3, BHCVRP obtuvo 12744.52, mientras que OR-Tools llegó a 14148.00, una diferencia mucho menor que en instancias pequeñas como CVRP\_1.

En conclusión:

* Si se busca una mejor calidad en la solución y se cuenta con tiempo suficiente para el cálculo, BHCVRP es la opción ideal.
* Si el tiempo de ejecución es un factor crítico y se requiere una solución rápida, aunque no óptima, ORTools es la mejor alternativa.
* En problemas más pequeños, la ventaja de BHCVRP en calidad es considerable, pero en problemas más grandes, la diferencia se reduce, haciendo que la rapidez de OR-Tools sea una ventaja más relevante.

En la Ilustración 2 se muestra un gráfico para apreciar e ilustrar mejor lo antes mencionado, comparando las soluciones obtenidas en términos de distancia total recorrida para ambas bibliotecas. Mientras que en la Ilustración 3 se refleja la diferencia en términos de tiempo de ejecución.

**Ilustración 22**: Gráfico de comparación entre las distancias totales de BHCVRP vs ORTools.

**Ilustración 23**: Gráfico de comparación entre los tiempos de ejecución de BHCVRP vs ORTools

**7.1.1. Prueba de Wilcoxon**

La prueba de Wilcoxon es una herramienta estadística ampliamente utilizada en análisis no paramétricos para evaluar diferencias entre dos grupos de datos, ya sea en escenarios de muestras relacionadas o independientes. Su importancia radica en su capacidad para analizar datos sin requerir supuestos estrictos sobre su distribución, como la normalidad, lo que la hace útil para datos ordinales, sesgados o con una cantidad limitada de observaciones.

Para analizar de manera estadística la diferencia entre BHCVRP y OR-Tools, se aplica la prueba de rangos con signo de Wilcoxon (*Wilcoxon signed-rank test*).Esta prueba es adecuada para este tipo de comparación porque los datos son emparejados o dependientes, es decir, cada instancia del problema tiene dos valores asociados (uno para BHCVRP y otro para OR-Tools). Evalúa si las diferencias entre estas soluciones tienen una mediana distinta de cero, lo que permite determinar si hay una diferencia sistemática entre los dos enfoques y si esta es estadísticamente significativa. En la Tabla 21 y la Tabla 22 se muestra el resultado de la prueba utilizando el software Keel para evaluar las variables de distancia total y tiempo de ejecución, respectivamente.

Tabla 18: Resultados de la prueba de Wilcoxon entre ORTools vs BHCVRP para distancia total.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 |

Tabla 19: Resultados de la prueba de Wilcoxon entre ORTools vs BHCVRP para tiempo de ejecución.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | 0.000076 | [-5213.3,-0.34] | 0 |

Interpretación de los resultados:

* El valor P asintótico de 0.000082 (< 0.05) y 0.000076 (< 0.05) indica que la diferencia entre ambas herramientas no es producto del azar, lo que respalda estadísticamente la superioridad de BHCVRP en calidad de solución y de OR-Tools en tiempo de ejecución.
* El intervalo de confianza para α = 0.90 y α = 0.95 es [-3719.51, -143.06], lo que confirma que, con un 90% y 95% de confianza, la mediana de las diferencias en distancia total entre OR-Tools y BHCVRP se encuentra en un rango negativo, es decir, BHCVRP genera soluciones más cortas en términos de distancia. Mientras que para el tiempo de ejecución con un intervalo de [-5213.3,-0.34], lo que indica que en este caso OR-Tools supera a BHCVRP.
* Sin embargo, el valor P exacto (≥ 0.2) sugiere que, en ciertos casos, la diferencia puede no ser tan marcada, especialmente en instancias más grandes donde las soluciones de ambas herramientas son más cercanas.

Los resultados sugieren que BHCVRP es estadísticamente superior a OR-Tools en términos de calidad de solución, ya que el valor P asintótico respalda esta diferencia significativa y el intervalo de confianza muestra que BHCVRP genera distancias más cortas. Sin embargo, el valor P exacto indica que en instancias más grandes las diferencias entre las dos herramientas pueden ser menos pronunciadas, lo que sugiere que la ventaja de BHCVRP podría disminuir en problemas de mayor escala.

**7.2. Homberger y Gehring vs OR-Tools**

En este apartado se analiza la comparación entre las soluciones presentadas en el estudio de Homberger y Gehring y los resultados obtenidos con la biblioteca OR-Tools para la variante VRPTW. Esta variante VRP introduce restricciones de tiempo en las entregas, lo que añade una mayor complejidad a la optimización de las rutas.

Para llevar a cabo esta comparación, se evaluaron 23 instancias de distintos tamaños, con escenarios que abarcan desde 200 hasta 600 nodos. Se utilizó la distancia euclidiana como métrica de referencia, en coherencia con la metodología aplicada en el artículo original, garantizando así una base común para la evaluación del desempeño de ambas estrategias.

Los resultados obtenidos con OR-Tools corresponden a los mismos reportados en el experimento descrito en el epígrafe 5.3, dentro de la sección dedicada a la variante VRPTW donde se detallan las características de las instancias.

Para determinar la calidad de las soluciones generadas, se han considerado dos factores clave:

* Distancia total recorrida: un menor valor implica costos operativos más bajos, mayor eficiencia en el uso del combustible y una reducción en el impacto ambiental.
* Número de rutas empleadas: hace referencia a la cantidad de vehículos requeridos para completar la distribución, lo que influye en la planificación logística, costos de operación y disponibilidad de la flota.

La comparación de estos factores permite evaluar qué enfoque logra un mejor balance entre eficiencia y costos. En un contexto donde la optimización de rutas es esencial para mejorar la rentabilidad y sostenibilidad del transporte, este análisis proporciona una visión clara de las ventajas y limitaciones de cada metodología. No se tiene en cuenta el tiempo de ejecución ya que no es un elemento brindado en el artículo a comparar.

A continuación, la Tabla 22 refleja la comparación de los resultados utilizando la distancia euclideana.

**Tabla 20:** Comparación entre el artículo de Homberger y Gehring vs 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 evidencian que el método propuesto por Homberger y Gehring supera a OR-Tools tanto en la eficiencia de las rutas utilizadas como en la calidad de las soluciones generadas. En instancias de menor tamaño (200 nodos), ambas metodologías producen soluciones relativamente similares en términos de distancia total y número de rutas. Sin embargo, a medida que el tamaño de las instancias aumenta (400 y 600 nodos), la diferencia entre los dos algoritmos se hace más notoria.

Los datos muestran que Homberger y Gehring logra sistemáticamente soluciones con menor distancia total recorrida y, en muchos casos, con un menor número de rutas utilizadas. Por ejemplo, en la instancia C1\_6\_2 (600 nodos), el método obtiene una distancia total de 13,948.3 con 58 rutas, mientras que OR-Tools genera una distancia de 16,759.0 con 63 rutas, evidenciando una diferencia significativa en la optimización de los recorridos.

Este comportamiento sugiere que, aunque OR-Tools puede generar soluciones competitivas en problemas de menor escala, su rendimiento se ve comprometido en escenarios de mayor complejidad. En particular, para instancias más grandes, el método de Homberger y Gehring muestra una mayor robustez y eficiencia en la variante VRPTW, optimizando tanto la distancia total como el número de rutas necesarias.

Para una mejor comprensión de estos resultados, la Ilustración 4 presenta la comparación de las distancias totales obtenidas por OR-Tools y el método de Homberger y Gehring, reflejando visualmente las diferencias observadas y reforzando las conclusiones expuestas.

**Ilustración 24:** Gráfico de comparación entre las distancias totales del artículo de Homberger y Gehring vs ORTools.

**7.2.1. Prueba de Wilcoxon**

En esta sección se realiza una comparación entre el algoritmo propuesto por Homberger y Gehring (1999) y la biblioteca OR-Tools, con el objetivo de evaluar si existen diferencias significativas en la calidad de las soluciones obtenidas por ambos métodos. Dado que las soluciones generadas son datos emparejados y dependientes (es decir, cada instancia tiene una solución obtenida por ambos algoritmos), se ha aplicado la prueba de rangos con signo de Wilcoxon para analizar estadísticamente estas diferencias.

Los resultados obtenidos, presentados en la Tabla 23, ofrecen evidencia cuantitativa sobre la existencia de diferencias significativas entre ambos métodos. En particular, se observa que el valor p asintótico es 0.000082, lo que indica que las diferencias encontradas no son producto del azar y que hay una ventaja estadísticamente significativa a favor del método propuesto en [33]. Además, los intervalos de confianza muestran que, con un nivel de confianza del 90% y 95%, la mediana de las diferencias es negativa, lo que confirma que Homberger y Gehring obtiene mejores resultados en términos de optimización de rutas en comparación con OR-Tools.

**Tabla 21:** 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:

* El valor p exacto obtenido es (≥ 0.2), lo que indicaría en un principio que no hay una diferencia significativa entre los dos algoritmos. Sin embargo, el valor p asintótico es de 0.000013, que es significativamente menor que 0.05. Este valor p asintótico tan bajo sugiere que las diferencias observadas entre los dos algoritmos no son producto del azar, sino que son una muestra de una verdadera disparidad en el rendimiento de los métodos.
* El intervalo de confianza para ambos niveles de confianza (90% y 95%) es de [-12,213.4, 14.6], lo que indica un rango amplio de diferencias en las soluciones generadas por los algoritmos. Aunque la diferencia central se encuentra cerca de cero, el hecho de que el intervalo incluya valores negativos y positivos sugiere que las diferencias entre los algoritmos pueden ser de magnitudes variadas, reflejando una alta variabilidad en las diferencias observadas. Esto implica que, aunque se encuentra una diferencia significativa entre ambos métodos, el rendimiento relativo de los algoritmos podría variar dependiendo de las características específicas de las instancias analizadas. Por tanto, no se puede generalizar completamente que uno de los algoritmos sea consistentemente superior al otro en todas las situaciones, ya que las diferencias dependen en parte del contexto.

Estos resultados confirman que existe una diferencia estadísticamente significativa entre los algoritmos comparados, pero la variabilidad de los resultados resalta la importancia de considerar las características particulares de cada instancia y el contexto en el que se implementan los algoritmos. Las conclusiones indican que, aunque el algoritmo propuesto en por Homberger y Gehring tiene un rendimiento superior en general, la magnitud de esta superioridad puede fluctuar dependiendo de las condiciones específicas de cada caso de estudio.

**8. Predicción y análisis de datos**

La predicción es el proceso de utilizar datos actuales o pasados para estimar el valor de una variable desconocida en el futuro. Este proceso es fundamental en una amplia gama de campos, desde la estadística hasta el aprendizaje automático y la inteligencia artificial. En términos generales, la predicción permite tomar decisiones informadas, basadas en patrones y tendencias detectadas en los datos, anticipando así resultados futuros.

Consiste en desarrollar modelos matemáticos o computacionales que, a partir de un conjunto de datos de entrada, generen estimaciones sobre eventos o resultados que aún no han ocurrido. Los datos históricos sirven como base para identificar patrones o relaciones entre variables, lo que permite extrapolar estos patrones a situaciones futuras.

En el ámbito de la ciencia de datos y el análisis de negocios, la predicción es utilizada para anticipar el comportamiento de consumidores, estimar demandas de productos, prever el rendimiento financiero de empresas, o incluso para tareas más complejas como la predicción del clima o el pronóstico de enfermedades. Los modelos de predicción emplean algoritmos y técnicas de análisis estadístico o de *machine learning*, para entrenar sobre un conjunto de datos (conjunto de entrenamiento) y realizar predicciones sobre nuevos datos (conjunto de prueba).

Existen diversas herramientas y técnicas que permiten hacer predicciones con base en diferentes tipos de datos y problemas. Algunas de las más conocidas se muestran a continuación.

**8.1. Regresión Lineal**

La regresión lineal es una técnica de predicción que asume una relación lineal entre las variables independientes (predictores) y la variable dependiente (objetivo). Se utiliza principalmente para predecir valores continuos, como el precio de un bien, la demanda de un producto o tendencias económicas. Es una herramienta sencilla y eficiente para problemas donde la relación entre las variables es clara y lineal. Sin embargo, su principal limitación es que no es adecuada para modelar relaciones no lineales o datos complejos, lo que reduce su aplicabilidad en problemas más avanzados.

**8.2. Redes Neuronales**

Las redes neuronales, especialmente en su forma más profunda (Deep Learning), son modelos inspirados en el funcionamiento del cerebro humano. Pueden modelar relaciones no lineales y complejas en grandes volúmenes de datos, lo que las hace ideales para tareas como predicción de series temporales, procesamiento de imágenes, reconocimiento de voz y análisis de texto. Su principal ventaja es su flexibilidad y capacidad para aprender patrones complejos. No obstante, requieren grandes cantidades de datos y recursos computacionales, y su naturaleza de "caja negra" las hace difíciles de interpretar.

**8.3. Redes bayesianas**

Las redes bayesianas son modelos probabilísticos que representan relaciones condicionales entre variables mediante grafos. Utilizan el teorema de Bayes para realizar predicciones en situaciones con incertidumbre, lo que las hace útiles en aplicaciones como diagnósticos médicos, predicción de fallos en sistemas y análisis de riesgos. Su principal ventaja es que manejan incertidumbre y relaciones complejas entre variables de manera interpretable. Sin embargo, requieren conocimiento previo para definir la estructura del grafo y pueden ser computacionalmente costosas para grandes volúmenes de datos.

**8.4. Árboles de decisión**

El funcionamiento de los árboles de decisión se basa en la división recursiva de los datos en subconjuntos más pequeños, de forma que las decisiones tomadas en cada nodo maximizan la pureza de los subconjuntos resultantes. Esta técnica se conoce como "división" o "splitting", y tiene como objetivo que los nodos terminales o hojas del árbol contengan ejemplos lo más homogéneos posibles, es decir, que todos los ejemplos en una hoja pertenezcan a la misma clase o tengan valores similares en una variable de salida continua.

Los árboles de decisión son útiles principalmente para problemas de clasificación y regresión. En clasificación, se utilizan para predecir una clase o categoría de una variable objetivo, mientras que en regresión se usan para predecir un valor numérico continuo. Esta capacidad de adaptarse tanto a tareas de clasificación como a problemas de regresión los hace muy versátiles y aplicables a una amplia variedad de problemas.

Se utilizan en una gran variedad de aplicaciones, desde sistemas de recomendación hasta diagnóstico médico, pasando por la predicción de tendencias de mercado, análisis de riesgos y optimización de procesos. En el contexto de la predicción, los árboles de decisión permiten hacer recomendaciones automáticas sobre las mejores acciones a seguir según las características observadas. De manera general, son herramientas valiosas para la toma de decisiones informada, ya que no solo proporcionan un resultado, sino que también explican cómo se llegó a esa conclusión.

Una de las ventajas clave de los árboles de decisión es su interpretabilidad. A diferencia de otros modelos más complejos como las redes neuronales, los árboles de decisión pueden ser entendidos fácilmente por los humanos, lo que facilita la interpretación de los resultados y la justificación de las decisiones que se toman con ellos.

El proceso de predicción con árboles de decisión consiste en seguir las ramas del árbol desde el nodo raíz hasta una hoja, tomando decisiones en cada nodo en función de las características de los datos de entrada. En cada etapa, se evalúa una condición sobre un atributo de los datos y se elige la rama correspondiente. Este proceso continúa hasta llegar a una hoja, que contiene la predicción final. La capacidad de predecir es tan eficiente como las decisiones tomadas en los nodos, y la precisión del árbol depende de cómo se construye el árbol y qué criterios se usan para dividir los datos.

**8.5. Bosques de decisión**

Los bosques de decisión, también conocidos como *Random Forest*, son un grupo de árboles de decisión que trabajan en conjunto para mejorar la precisión y la estabilidad de las predicciones. A diferencia de un único árbol de decisión, que puede ser propenso al sobreajuste, un bosque de decisión combina múltiples árboles para obtener un modelo más robusto y generalizable.

El principio detrás de los bosques de decisión es la agregación de múltiples árboles de decisión entrenados con diferentes subconjuntos de datos. Esto se logra mediante dos técnicas principales:

* Agrupación por remuestreo (*Bootstrap Aggregating* (*Bagging*)): cada árbol en el bosque se entrena con una muestra aleatoria con reemplazo del conjunto de datos original. Esto ayuda a reducir la varianza y hacer el modelo más estable frente a nuevas muestras.
* Selección aleatoria de características: en cada nodo de los árboles, en lugar de evaluar todas las características disponibles, se selecciona aleatoriamente un subconjunto de ellas. Esto evita la dominancia de ciertas variables y promueve la diversidad entre los árboles.

Una vez entrenado el bosque, la predicción final se obtiene mediante:

* Votación mayoritaria (para problemas de clasificación): Cada árbol emite un voto y la clase con más votos es la predicción final.
* Promedio de predicciones (para problemas de regresión): Se calcula el promedio de las predicciones de todos los árboles.

Los bosques de decisión presentan varias ventajas sobre los árboles individuales:

* Mayor precisión y menor sobreajuste: al combinar múltiples modelos, se reducen errores y sesgos individuales.
* Manejo de datos faltantes y ruido: son menos sensibles a datos atípicos o valores perdidos.
* Versatilidad: se pueden aplicar a problemas de clasificación y regresión en diversas áreas, como predicción de mercado, diagnóstico médico, procesamiento de imágenes y optimización de rutas.

**8.6. Predicción con bosques de decisión**

En este epígrafe, se explora el uso de bosques de decisión para predecir las mejores heurísticas y metaheurísticas para la resolución de problemas VRP. La elección de la técnica más adecuada depende de las características específicas del problema, lo que hace que el proceso de selección sea desafiante. En este sentido, los bosques de decisión se presentan como una herramienta poderosa para modelar y predecir cuál es la estrategia más efectiva, basándose en datos previos y características de las instancias del problema. Este enfoque no solo facilita la toma de decisiones, sino que también mejora la eficiencia al reducir la necesidad de prueba y error al seleccionar la heurística o metaheurística más apropiada.

El enfoque que se propone consiste en predecir, a partir de las características específicas de cada instancia en los problemas (TSP, CVRP, VRPTW, MDVRP, VRPPD), el mejor algoritmo compuesto por una heurística como solución inicial, complementada con una metaheurística que optimice esa solución. Las características que se consideran en la predicción incluyen la cantidad de nodos, la cantidad de vehículos disponibles, la capacidad de los vehículos y la cantidad promedio de demandas de los nodos. Estos factores juegan un papel fundamental en la selección del algoritmo adecuado, ya que influyen tanto en la complejidad del problema como en la eficiencia de la búsqueda de soluciones.

Al emplear bosques de decisión, se busca mejorar la capacidad predictiva al combinar múltiples árboles de decisión, reduciendo el sobreajuste y aumentando la robustez del modelo. Este enfoque permite identificar patrones y correlaciones entre estas características y los algoritmos que ofrecen las mejores soluciones en cada caso, optimizando así la selección de técnicas y mejorando la calidad de las soluciones obtenidas para los problemas analizados.

**8.6.1. Configuración del modelo**

El dataset utilizado para entrenar y evaluar el modelo fue obtenido de las 10 iteraciones realizadas para cada uno de los problemas mencionados. Estos datos fueron recopilados para representar diferentes instancias ya evaluadas anteriormente. Para acceder a todos los resultados de las iteraciones, se puede consultar el enlace de Google Drive proporcionado en el epígrafe 5.

Con un total de 67 instancias y 75 algoritmos (15 heurísticas combinadas con 5 metaheurísticas) se entrena un modelo por cada problema (TSP, CVRP, VRPTW, MDVRP, VRPPD) y cada tipo de distancia (Euclidean, Manhattan, Haversine, Chebyshev). Las características seleccionadas a tomar en cuenta son:

* La cantidad de nodos
* La capacidad de los vehículos
* La cantidad de vehículos
* El promedio de demandas

Para la predicción del método óptimo en cada instancia, se implementó un modelo de clasificación basado en *Random Forest*. Se utilizó Python como lenguaje de programación y la biblioteca *Scikit-learn* para la construcción y evaluación del modelo. A continuación, se describen los pasos seguidos en la ejecución:

1. Carga y preprocesamiento de datos: se organiza la información en un *DataFrame* de la biblioteca de Python *Pandas*. Posterioremente, se calcula un puntaje ("*Score*") según la Ecuación 3 ponderada y se determina el método óptimo por instancia seleccionando aquel con el menor puntaje.

**Score = Objective \* 0.6 + Time \* 0.2 + Routes \* 0.2**

Ecuación 5: Ecuación para el cálculo de la puntuación de la predicción del mejor método.

Donde:

* + Objective: distancia recorrida por todos los vehículos
  + Time: tiempo de ejecución del algoritmo
  + Routes: cantidad de rutas utilizadas

1. Manejo del desbalanceo de clases: se emplea la técnica de sobremuestreo SMOTE (*Synthetic Minority Over-sampling Technique*) para balancear la distribución de clases en el conjunto de entrenamiento, mitigando el problema de clases subrepresentadas.
2. División del conjunto de datos: se separan los datos en un conjunto de entrenamiento (70%) y otro de prueba (30%).
3. Selección de características: para reducir la dimensionalidad y mejorar el rendimiento del modelo, se aplica *SelectKBest* para seleccionar las mejores características en función de su relevancia estadística. Además, se emplea PCA (Análisis de Componentes Principales) para retener el 95% de la varianza de los datos.
4. Entrenamiento del Modelo:
   * Clasificación con *RandomForestClassifier*, utilizando una estrategia de ponderación de clases de forma balanceada.
   * Para la optimización de hiperparámetros se evaluaron los siguientes:
     + n\_estimators: cantidad de árboles en el bosque (100 a 1000).
     + max\_depth: profundidad máxima del árbol (valores predefinidos entre 5 y 100 o sin límite).
     + min\_samples\_split: mínimo de muestras requeridas para dividir un nodo (2 a 20).
     + min\_samples\_leaf: mínimo de muestras en una hoja (1 a 10).
     + bootstrap: uso de muestreo con reemplazo (True o False).
     + max\_features: método para seleccionar número de características (sqrt, log2).
     + criterion: métrica para evaluar divisiones (gini, entropy).
5. Evaluación del modelo: se evalúa el modelo en el conjunto de prueba con las siguientes métricas:
   * Precisión (*accuracy\_score*): es la proporción de predicciones correctas entre el total de predicciones realizadas. Mide qué tan bien el modelo clasifica correctamente todas las instancias del conjunto de datos. Un valor de 1.0 (100%) indica que el modelo ha clasificado correctamente todas las muestras sin errores.
   * Puntuación F1 ponderada (*f1\_score*): es la media armónica entre la precisión (precision) y el recall (sensibilidad), proporcionando un equilibrio entre ambas.
     + Precisión: proporción de verdaderos positivos sobre todas las predicciones positivas.
     + Recall: proporción de verdaderos positivos sobre el total de muestras realmente positivas.
   * Informe de clasificación detallado (*classification\_report*): Es una tabla generada en modelos de clasificación que muestra métricas clave para evaluar el rendimiento del modelo en cada categoría de la variable objetivo.
     + *precision*: mide la exactitud de las predicciones positivas. Indica qué tan confiable es el modelo al clasificar una instancia en una categoría específica.
     + *recall*: mide la capacidad del modelo para detectar correctamente todas las instancias de una clase dada. Indica qué proporción de los casos reales de una categoría fueron correctamente identificados.
     + *f1-score*: es la media armónica entre la precisión y el recall, proporcionando un equilibrio entre ambos. Es útil cuando hay un desequilibrio entre las clases, ya que combina ambas métricas en una sola.
     + *support*: representa el número total de instancias en el conjunto de datos que pertenecen a esa clase. No es una métrica de rendimiento, sino una referencia del tamaño de cada categoría en los datos.

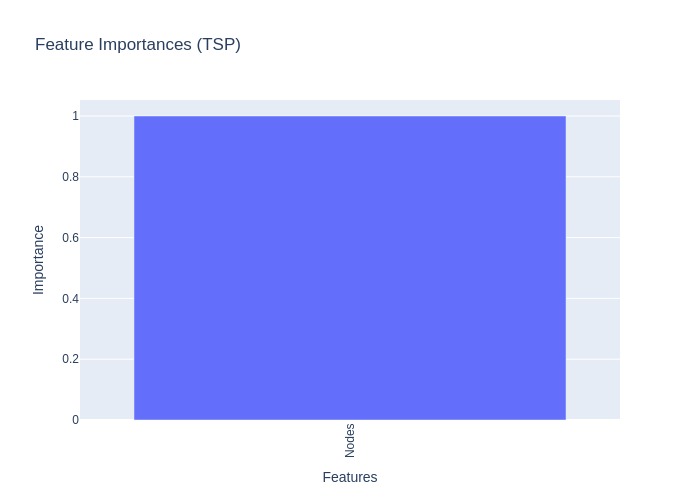
### Filas especiales:

* + - * accuracy: muestra la precisión global del modelo, es decir, el porcentaje de predicciones correctas en todas las clases.
      * macro avg: promedio simple de las métricas (precision, recall, f1-score) para todas las clases, sin considerar el número de instancias por clase.
      * weighted avg: promedio ponderado de las métricas, considerando el número de instancias de cada clase. Es más representativo cuando las clases están desbalanceadas.
  + Validación cruzada estratificada de 10 pliegues (*StratifiedKFold):* divide los datos en varios subconjuntos, llamados "*folds*" (pliegues), y el proceso de validación se repite varias veces para asegurar que el modelo sea probado en diferentes partes de los datos.

Este procedimiento permite optimizar la selección del método óptimo, garantizando un modelo robusto y bien calibrado para la clasificación de instancias según las características disponibles.

**8.6.2. Resultados para TSP**

En este epígrafe se presentan los resultados del modelo de predicción para el problema TSP, utilizando los resultados de las instancias descritas en el epígrafe 5.1. En la Ilustración 5 se reflejan las características más relevantes para la predicción, ya que, mediante la técnica SelectKBest, mencionada en el epígrafe anterior, se descartaron aquellas características que no son relevantes.



**Ilustración 25:** Gráfico de las características relevantes de la predicción de TSP.

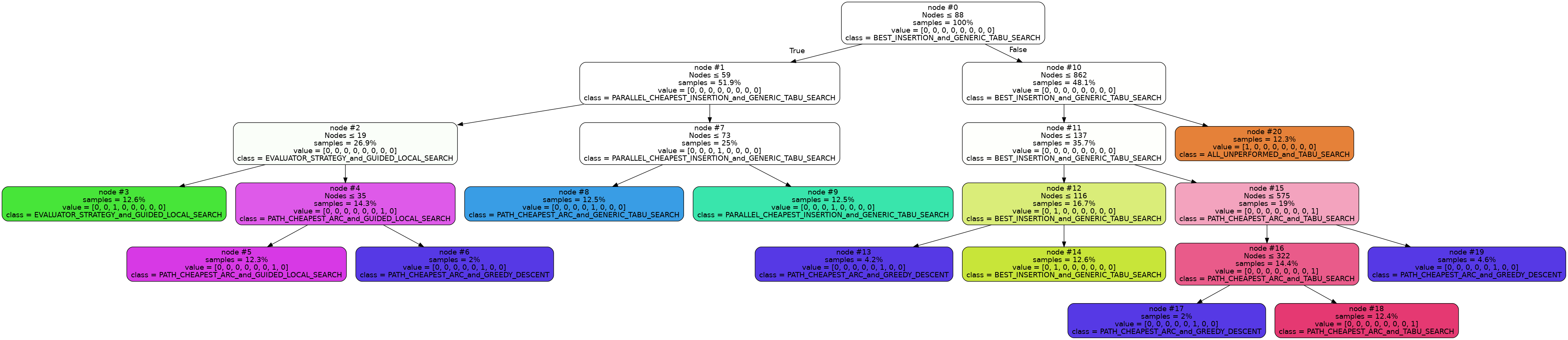
Como se observa, la característica más destacada es la cantidad de nodos del problema, la cual presenta una relevancia máxima de 1, lo que indica su impacto significativo en la predicción. Este resultado se mantuvo constante para todos los tipos de distancia evaluados.

A continuación, en la Ilustración 6, se muestra uno de los árboles generados en el bosque de decisión utilizando las soluciones obtenidas con la distancia Euclidean, mientras que la Ilustración 7 presenta el árbol generado con la distancia Manhattan.

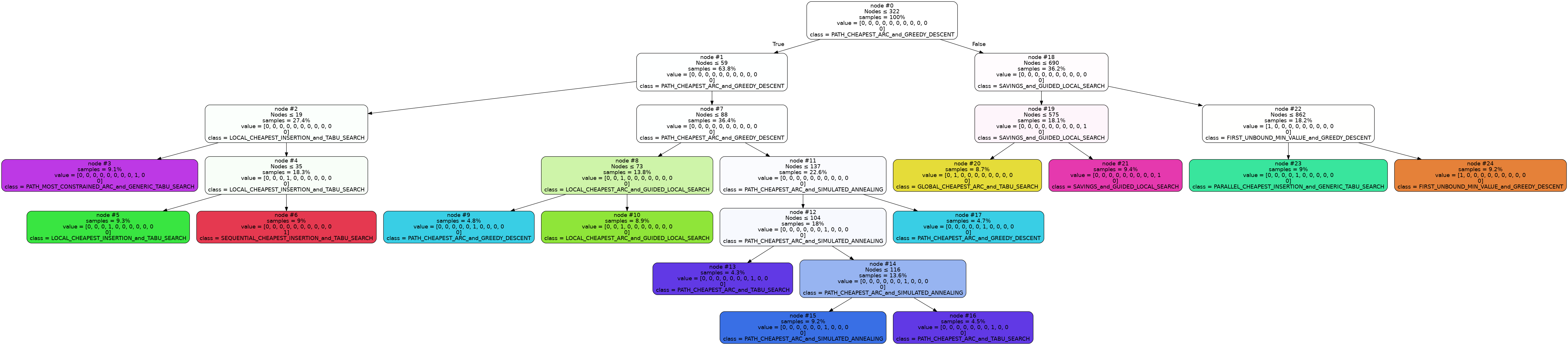
De todos los parámetros probados, los que ofrecieron el mejor desempeño fueron:  
{'bootstrap': True, 'criterion': 'entropy', 'max\_depth': 30, 'max\_features': 'sqrt', 'min\_samples\_leaf': 8, 'min\_samples\_split': 8, 'n\_estimtors': 221}.

Las métricas analizadas reflejaron resultados satisfactorios en ambos casos, como se detalla a continuación:

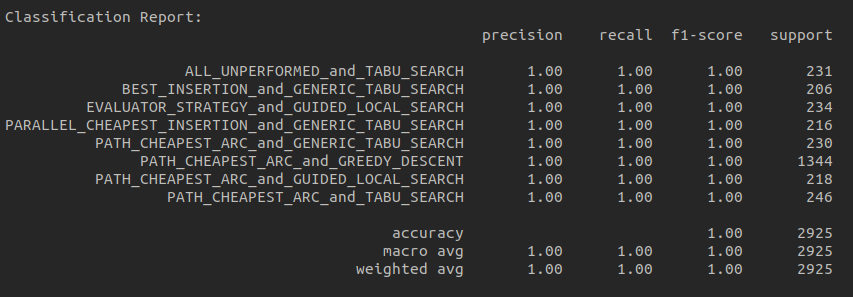
* Precisión (*Accuracy*) = 1.0
* Puntuación F1 ponderada (*F1 Score*) = 1.0
* Validación cruzada (*Cross Validation Score*) = [1.0, 1.0, 1.0, 1.0]
* Informe de clasificación detallado (*Classification Report*): la Ilustración 8 y la Ilustración 9 muestran el reporte para las distancias Euclidean y Manhattan, respectivamente.



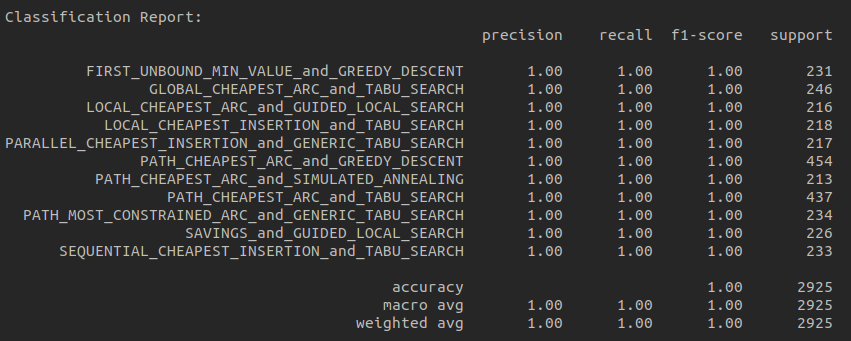
**Ilustración 26:** Árbol de predicción para TSP utilizando Euclidean.



**Ilustración 27:** Árbol de predicción para TSP utilizando Manhattan.



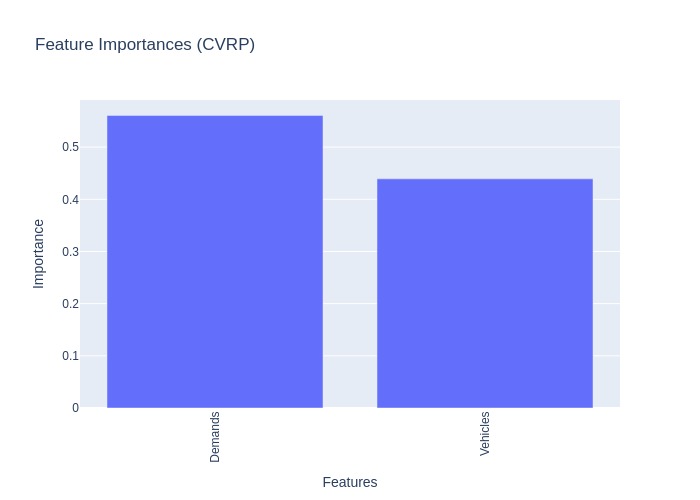
**Ilustración 28:** Reporte de clasificación para TSP utilizando Euclidean.



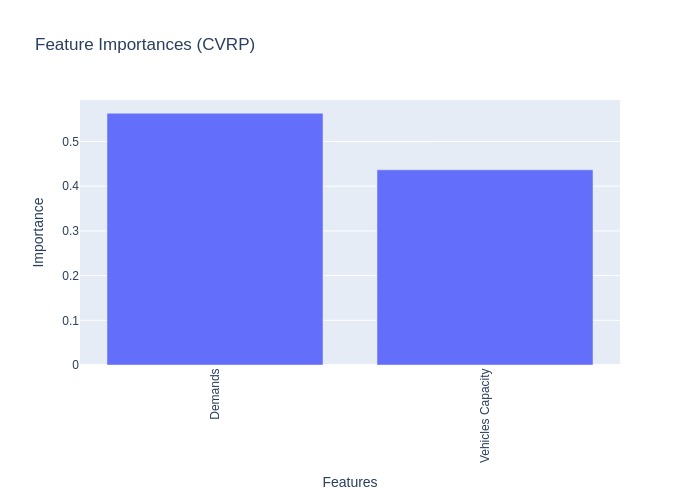
**Ilustración 29:** Reporte de clasificación para TSP utilizando Manhattan.

**8.6.3. Resultados para CVRP**

En este epígrafe se presentan los resultados del modelo de predicción aplicado al problema CVRP, utilizando los datos de las instancias descritas en el epígrafe 5.2. En la Ilustración 10, se muestran las características más significativas para la predicción utilizando la distancia Euclidean, destacando el promedio de demandas y la cantidad de vehículos. Por otro lado, en la Ilustración 11, que representa el caso con la distancia Manhattan, el promedio de demandas sigue siendo la característica más relevante, pero la segunda característica más importante cambia a la capacidad de los vehículos.



**Ilustración 30:** Gráfico de las características relevantes de la predicción de CVRP utilizando Euclidean.



**Ilustración 31:** Gráfico de las características relevantes de la predicción de CVRP utilizando Manhattan.

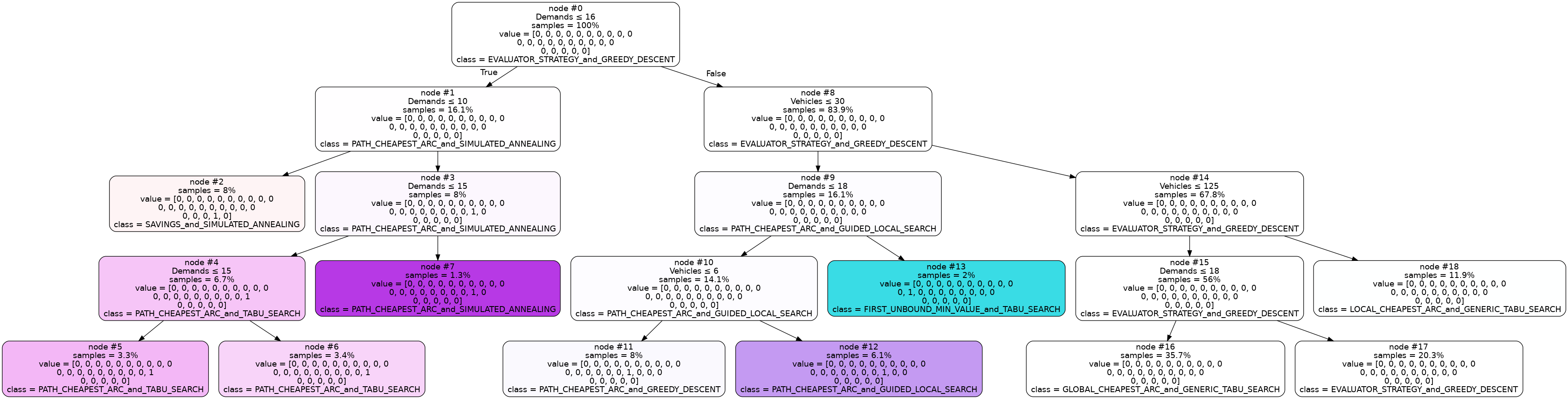
Adicionalmente, la Ilustración 12 presenta uno de los árboles generados en el bosque de decisión utilizando la distancia Euclidean, mientras que la Ilustración 13 muestra el caso correspondiente a la distancia Manhattan.

Los parámetros que mejor desempeño tuvieron fueron:

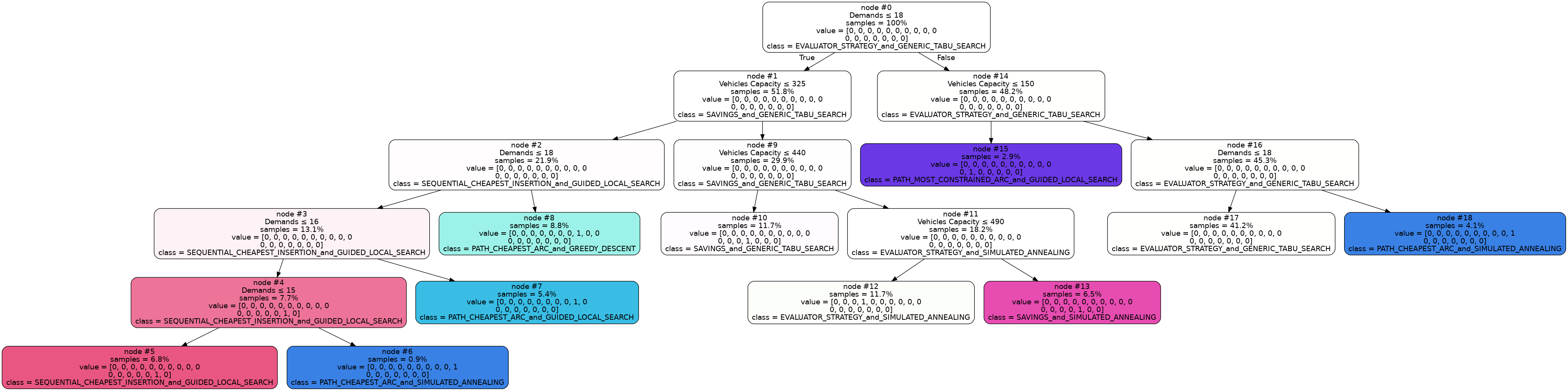
* Euclidean:  
  {'bootstrap': True, 'criterion': 'gini', 'max\_depth': 100, 'max\_features': 'log2', 'min\_samples\_leaf': 8, 'min\_samples\_split': 6, 'n\_estimators': 983}.
* Manhattan:  
  {'bootstrap': True, 'criterion': 'entropy', 'max\_depth': 30, 'max\_features': 'sqrt', 'min\_samples\_leaf': 8, 'min\_samples\_split': 8, 'n\_estimators': 221}.

Las métricas analizadas reflejaron resultados no tan satisfactorios como en el problema anterior, como se detalla a continuación:

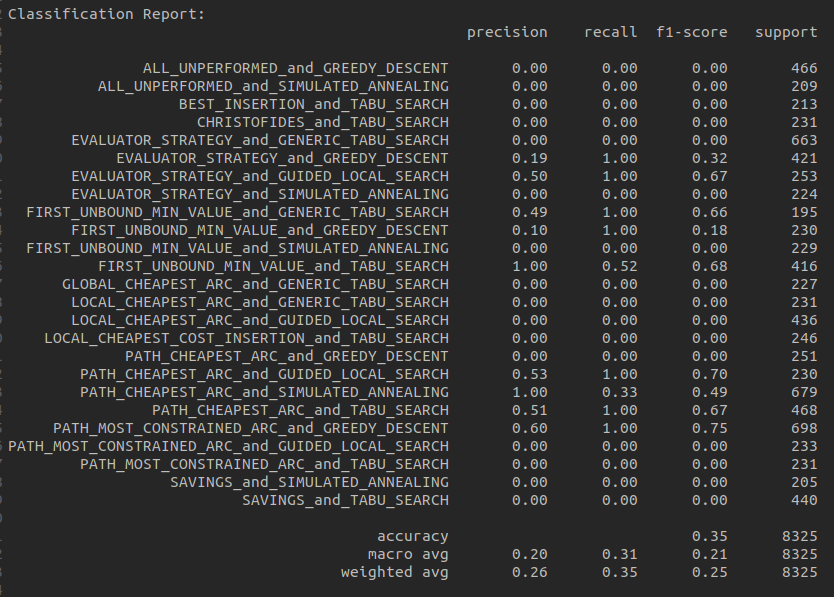
* Euclidean:
  + Precisión (*Accuracy*) = 0.3523
  + Puntuación F1 ponderada (*F1 Score*) = 0.2514
  + Validación cruzada (*Cross Validation Score*) = [0.35387387, 0.27063063, 0.29315315, 0.35225225, 0.29765766]
  + Informe de clasificación detallado (*Classification Report*): la Ilustración 14 muestra el reporte correspondiente. Refleja que muchos valores de precisión, recall y F1-score son 0.00, lo que indica que el modelo no está identificando correctamente las instancias de muchas clases. La precisión global es 0.35, lo que indica que el modelo solo acierta el 35% de las predicciones. Algunas clases tienen un alto número de instancias, como por ejemplo 663 instancias, lo que podría generar un desbalance entre clases. Si las clases más representadas son las que el modelo está clasificando mejor, el desbalance puede estar afectando las clases menos representadas. Las clases con buen desempeño resultaron ser:
    - EVALUATOR\_STRATEGY\_and\_GUIDED\_LOCAL\_SEARCH
    - FIRST\_UNBOUND\_MIN\_VALUE\_and\_TABU\_SEARCH
    - PATH\_CHEAPEST\_ARC\_and\_SIMULATED\_ANNEALING
    - PATH\_CHEAPEST\_ARC\_and\_TABU\_SEARCH
    - PATH\_MOST\_CONSTRAINED\_ARC\_and\_GREEDY\_DESCENT
* Manhattan:
  + Precisión (*Accuracy*) = 0.6559
  + Puntuación F1 ponderada (*F1 Score*) = 0.6069
  + Validación cruzada (*Cross Validation Score*) = [0.6218018, 0.65189189, 0.62198198, 0.61927928, 0.64720721]
  + Informe de clasificación detallado (*Classification Report*): la Ilustración 15 muestra el reporte correspondiente. La precisión global es 0.66, lo que indica que el modelo acierta en aproximadamente el 66% de las predicciones, lo cual es una mejora significativa en comparación con Euclidean. Algunas clases tienen valores de precisión, recall y F1-score relativamente buenos, especialmente para las clases con un recall de 1.00, lo que significa que el modelo está identificando correctamente todas las instancias de estas clases. Mientras que con otras no lo está haciendo. Existen clases con un número significativamente mayor de instancias, como 2465, esto puede explicar por qué las clases más representadas tienen un buen desempeño. El modelo puede estar sesgado hacia las clases con más datos. Las clases con un buen rendimiento son:
  + PATH\_CHEAPEST\_ARC\_and\_GUIDED\_LOCAL\_SEARCH
  + PATH\_CHEAPEST\_ARC\_and\_SIMULATED\_ANNEALING
  + PATH\_MOST\_CONSTRAINED\_ARC\_and\_GUIDED\_LOCAL\_SEARCH
  + clases asociadas con SAVINGS

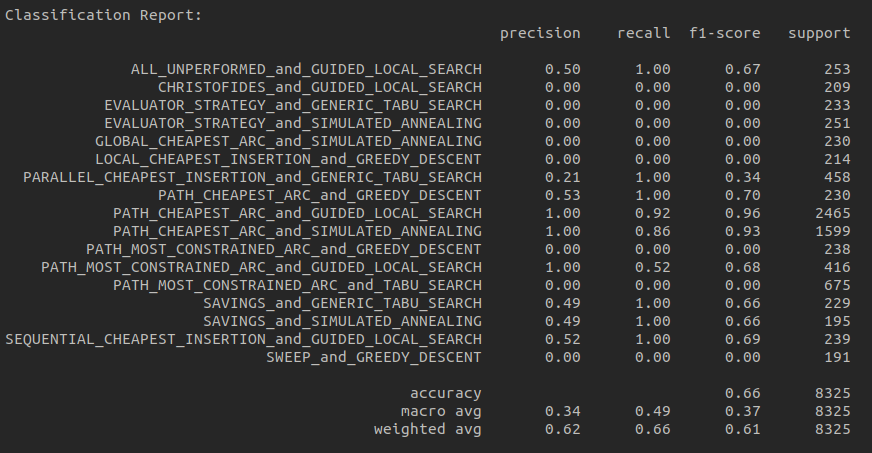


**Ilustración 32:** Árbol de predicción para CVRP utilizando Euclidean.



**Ilustración 33:** Árbol de predicción para CVRP utilizando Manhattan.

**Ilustración 34:** Reporte de clasificación para CVRP utilizando Euclidean.

**Ilustración 35:** Reporte de clasificación para CVRP utilizando Manhattan.

**8.6.4. Resultados para VRPTW**

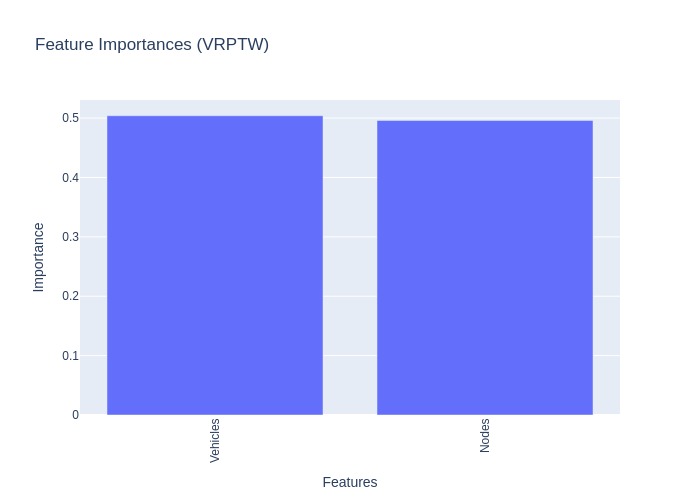
En este epígrafe se presentan los resultados del modelo de predicción aplicado al problema VRPTW, utilizando los resultados de las instancias descritas en el epígrafe 5.3. En la Ilustración 16 se muestran las características más significativas para la predicción utilizando la distancia Euclidiana. En la Ilustración 17 se presenta el caso utilizando la distancia de Manhattan.

Se puede observar que ambas distancias comparten una característica común: la cantidad de nodos. Sin embargo, en el caso de la distancia Euclidiana, la cantidad de vehículos fue un factor más relevante para la predicción, mientras que en la distancia Manhattan, lo que tuvo mayor relevancia fue el promedio de demandas.

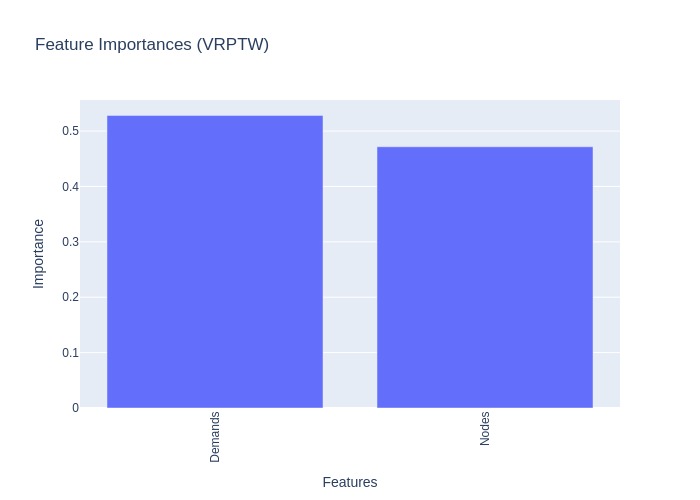
Adicionalmente, la Ilustración 18 presenta uno de los árboles generados en el bosque de decisión utilizando la distancia Euclidean, mientras que la Ilustración 19 muestra el caso correspondiente a la distancia Manhattan.

Los parámetros que mejor desempeño tuvieron fueron:

* Euclidean:  
  {'bootstrap': True, 'criterion': 'entropy', 'max\_depth': 30, 'max\_features': 'sqrt', 'min\_samples\_leaf': 8, 'min\_samples\_split': 8, 'n\_estimators': 221}



**Ilustración 36:** Gráfico de las características relevantes de la predicción de VRPTW utilizando Euclidean.

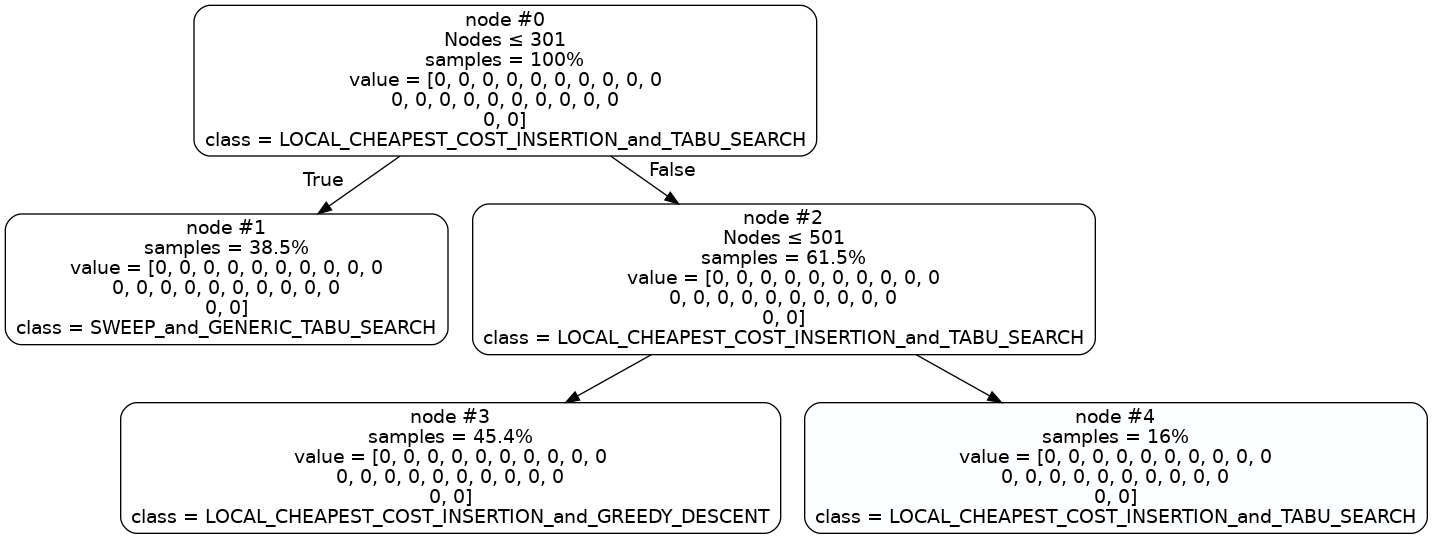


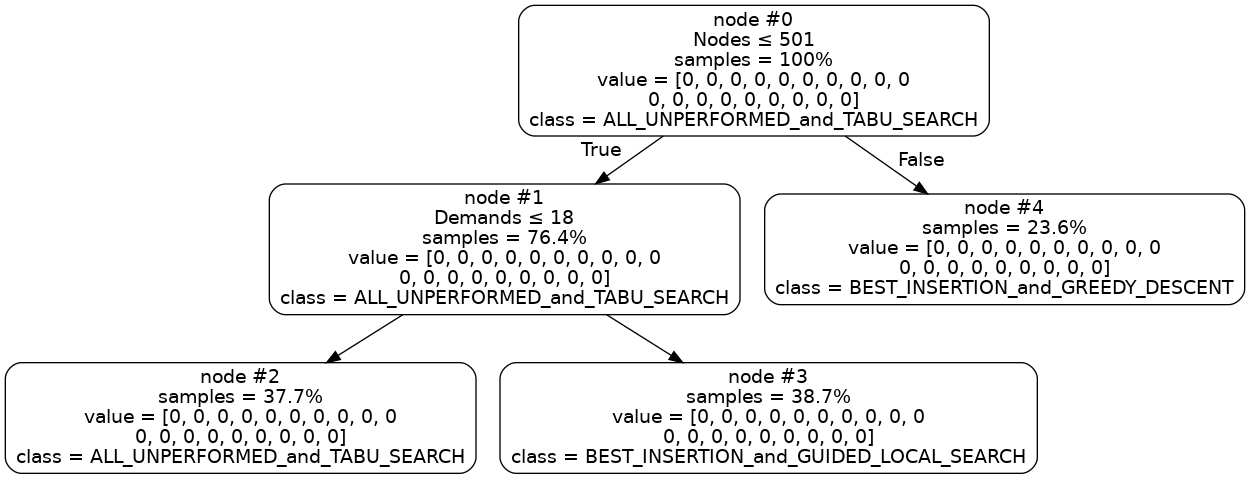
**Ilustración 37:** Gráfico de las características relevantes de la predicción de VRPTW utilizando Manhattan.

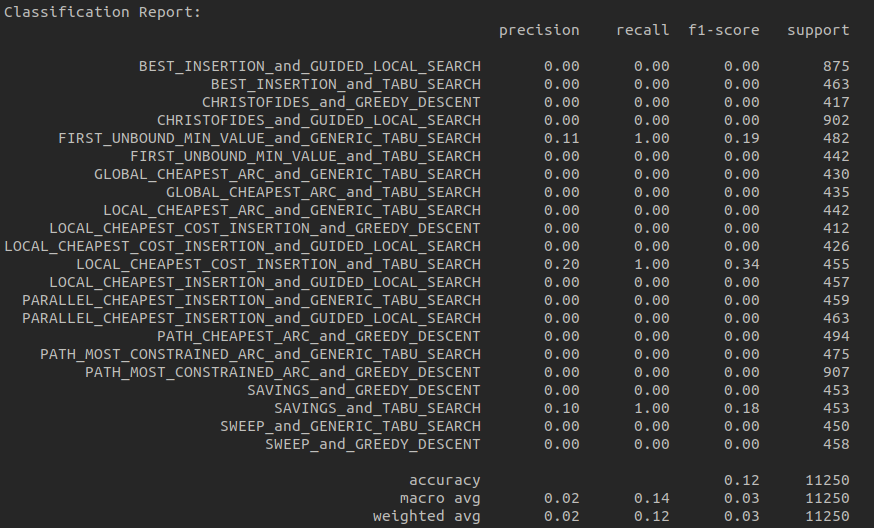
* Manhattan:  
  {'bootstrap': False, 'criterion': 'gini', 'max\_depth': 50, 'max\_features': 'sqrt', 'min\_samples\_leaf': 2, 'min\_samples\_split': 13, 'n\_estimators': 513}

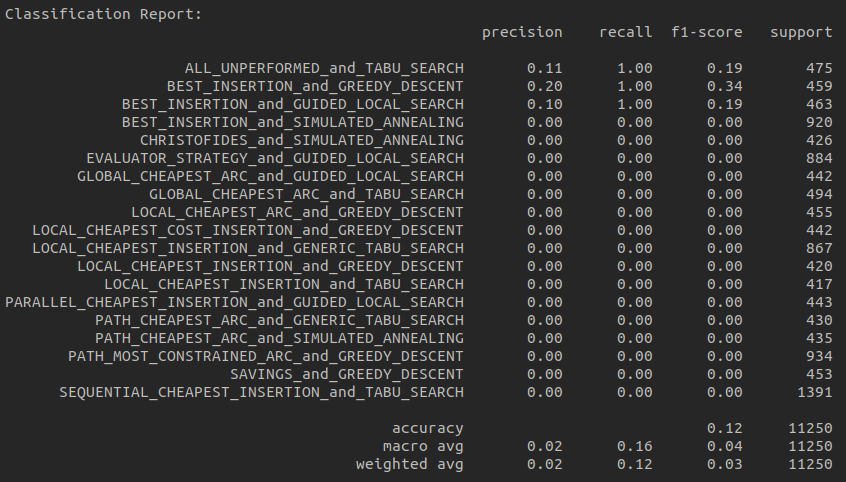
Las métricas analizadas reflejaron resultados no tan satisfactorios como se detalla a continuación:

* Euclidean:
  + Precisión (*Accuracy*) = 0.1236
  + Puntuación F1 ponderada (*F1 Score*) = 0.0292
  + Validación cruzada (*Cross Validation Score*) = [0.16, 0.16, 0.12, 0.16, 0.2 ]
  + Informe de clasificación detallado (*Classification Report*): la Ilustración 20 muestra el reporte correspondiente. Las métricas de precisión, recall y f1-score son muy bajas, especialmente para la mayoría de las clases, indicando un bajo desempeño del modelo para clasificar correctamente el problema. La clase con mejor desempeño es:
    - FIRST\_UNBOUND\_MIN\_VALUE\_and\_GENERIC\_TABU\_SEARCH, con una precisión de 0.11 y un recall de 1.00, aunque la puntuación f1 sigue siendo baja.
* Manhattan:
  + Precisión (*Accuracy*) = 0.1242
  + Puntuación F1 ponderada (*F1 Score*) = 0.0296
  + Validación cruzada (*Cross Validation Score*) = [0.12 0.12 0.12 0.12 0.12]
  + Informe de clasificación detallado (*Classification Report*): la Ilustración 21 muestra el reporte correspondiente. Aunque el desempeño no fue mucho mejor que en el caso Euclidiano, se observa un pequeño incremento en las métricas de precisión y recall, con una precisión promedio de 0.02 y un f1-score de 0.03. Las clases con el mejor desempeño fueron:
    - ALL\_UNPERFORMED\_and\_TABU\_SEARCH
    - BEST\_INSERTION\_and\_GREEDY\_DESCENT

**Ilustración 38:** Árbol de predicción para VRPTW utilizando Euclidean.

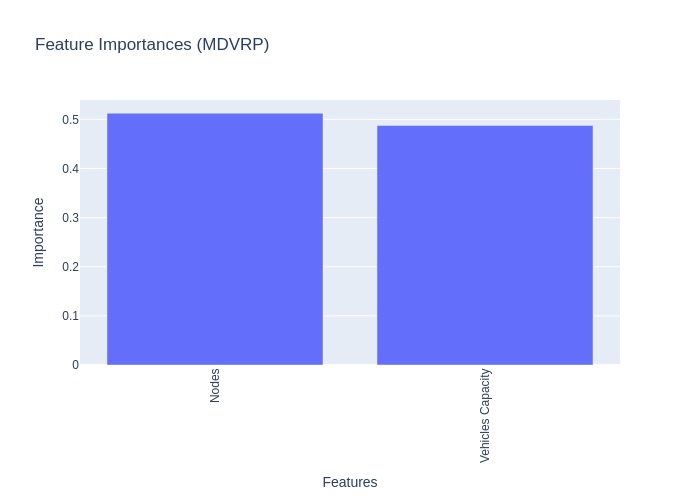
**Ilustración 39:** Árbol de predicción para VRPTW utilizando Manhattan.

**Ilustración 40:** Reporte de clasificación para VRPTW utilizando Euclidean.

**Ilustración 41:** Reporte de clasificación para VRPTW utilizando Manhattan.

**8.6.5. Resultados para MDVRP**

En este epígrafe se presentan los resultados obtenidos del modelo de predicción aplicado al problema MDVRP, utilizando las instancias descritas en el epígrafe 5.4. En la Ilustración 22 se destacan las características más significativas para la predicción empleando la distancia Euclidiana, mientras que la Ilustración 23 ilustra el caso utilizando la distancia Manhattan.



**Ilustración 42:** Gráfico de las características relevantes de la predicción de MDVRP utilizando Euclidean.



**Ilustración 43:** Gráfico de las características relevantes de la predicción de MDVRP utilizando Manhattan.

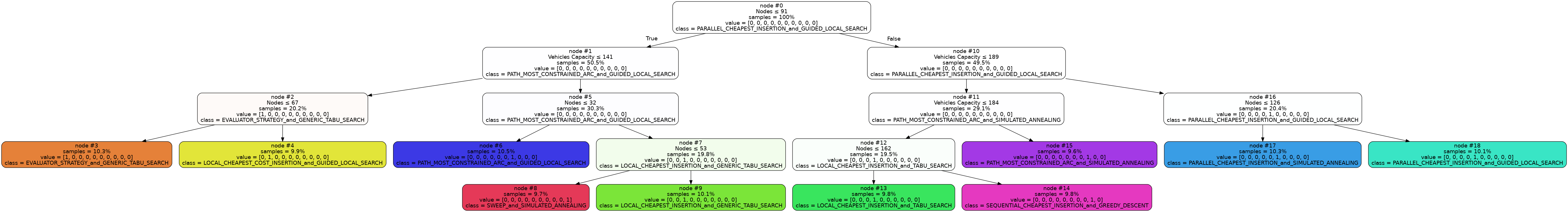
Los resultados indican que, en el caso de la distancia Euclidiana, las características más relevantes fueron la cantidad de nodos y la capacidad de los vehículos. Por otro lado, para la distancia Manhattan, el promedio de las demandas y la cantidad de vehículos resultaron ser los factores de mayor influencia.

Además, la Ilustración 24 presenta uno de los árboles generados dentro del bosque de decisión utilizando la distancia Euclidiana, mientras que la Ilustración 25 muestra el árbol correspondiente al caso de la distancia Manhattan.

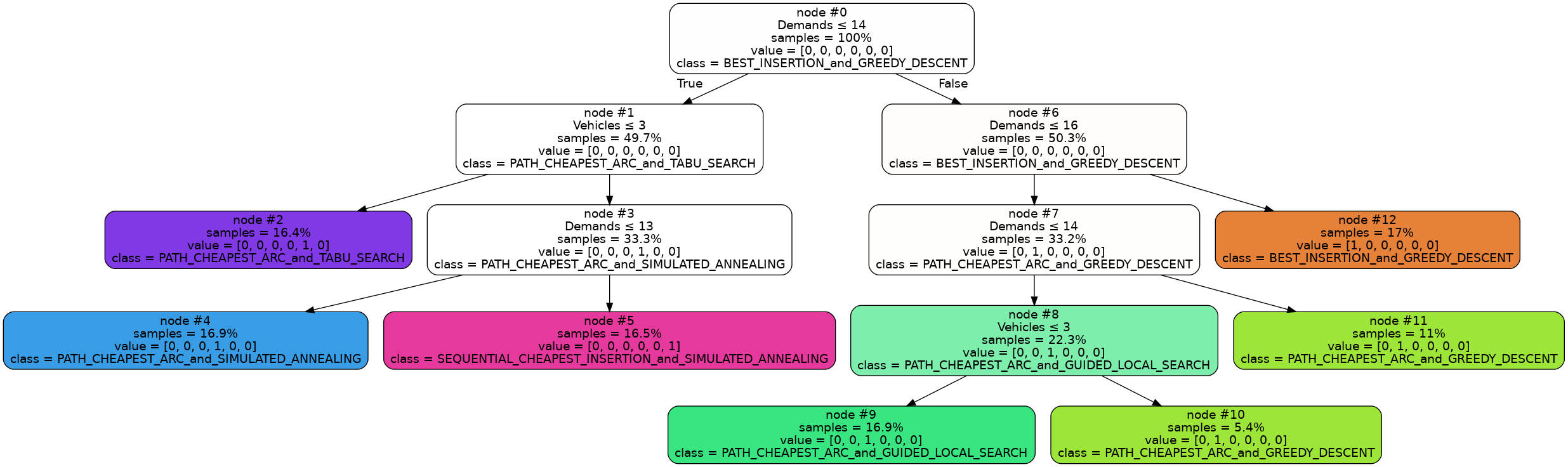
Los parámetros que mejor desempeño tuvieron para ambos casos fueron: {'bootstrap': True, 'criterion': 'entropy', 'max\_depth': 30, 'max\_features': 'sqrt', 'min\_samples\_leaf': 8, 'min\_samples\_split': 8, 'n\_estimators': 221}.

Las métricas analizadas reflejaron resultados satisfactorios como se detalla a continuación:

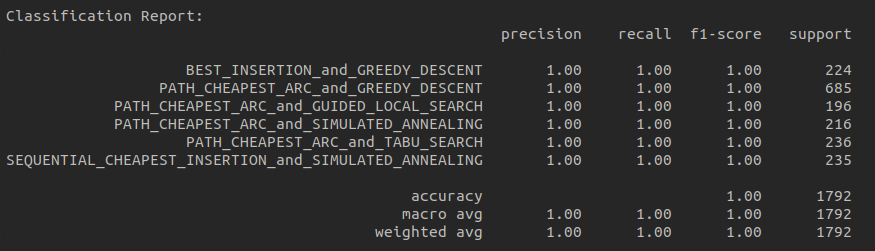
* Precisión (*Accuracy*) = 1.0
* Puntuación F1 ponderada (*F1 Score*) = 1.0
* Validación cruzada (*Cross Validation Score*) = [1.0, 1.0, 1.0, 1.0, 1.0]
* Informe de clasificación detallado (*Classification Report*): la Ilustración 24 y la Ilustración 25 muestran el reporte para las distancias Euclidean y Manhattan, respectivamente.

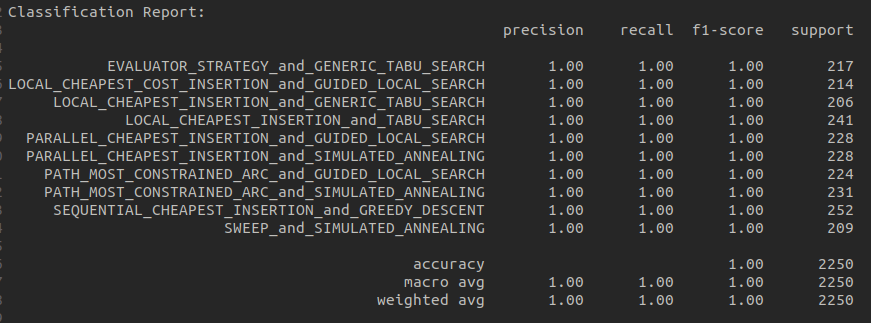


**Ilustración 44:** Árbol de predicción para MDVRP utilizando Euclidean.



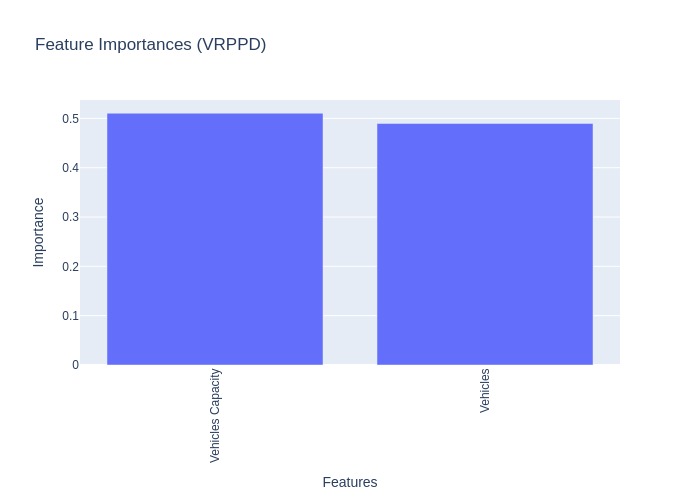
**Ilustración 45:** Árbol de predicción para MDVRP utilizando Manhattan.

**Ilustración 46:** Reporte de clasificación para MDVRP utilizando Euclidean.

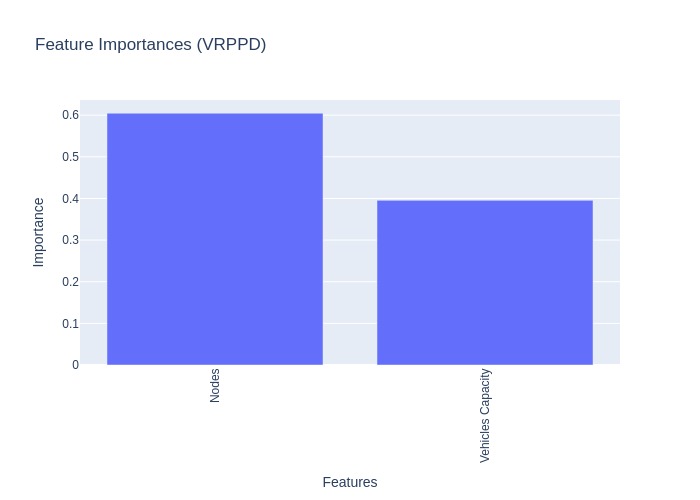
**Ilustración 47:** Reporte de clasificación para MDVRP utilizando Manhattan.

**8.6.6. Resultados para VRPPD**

En este epígrafe se presentan los resultados del modelo de predicción aplicado al problema VRPPD, basándose en las instancias descritas en el epígrafe 5.5. La Ilustración 28 muestra las características más relevantes para la predicción utilizando la distancia euclidiana, mientras que en la Ilustración 29 se presentan otros aspectos clave del análisis.



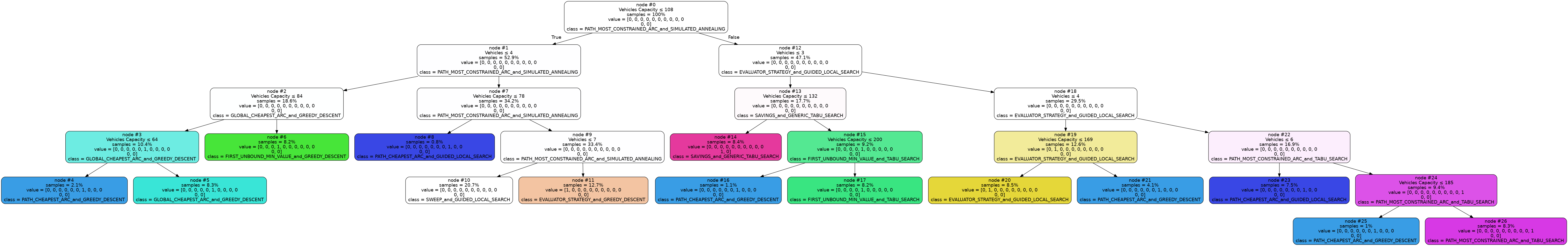
**Ilustración 48:** Gráfico de las características relevantes de la predicción de VRPPD utilizando Euclidean.

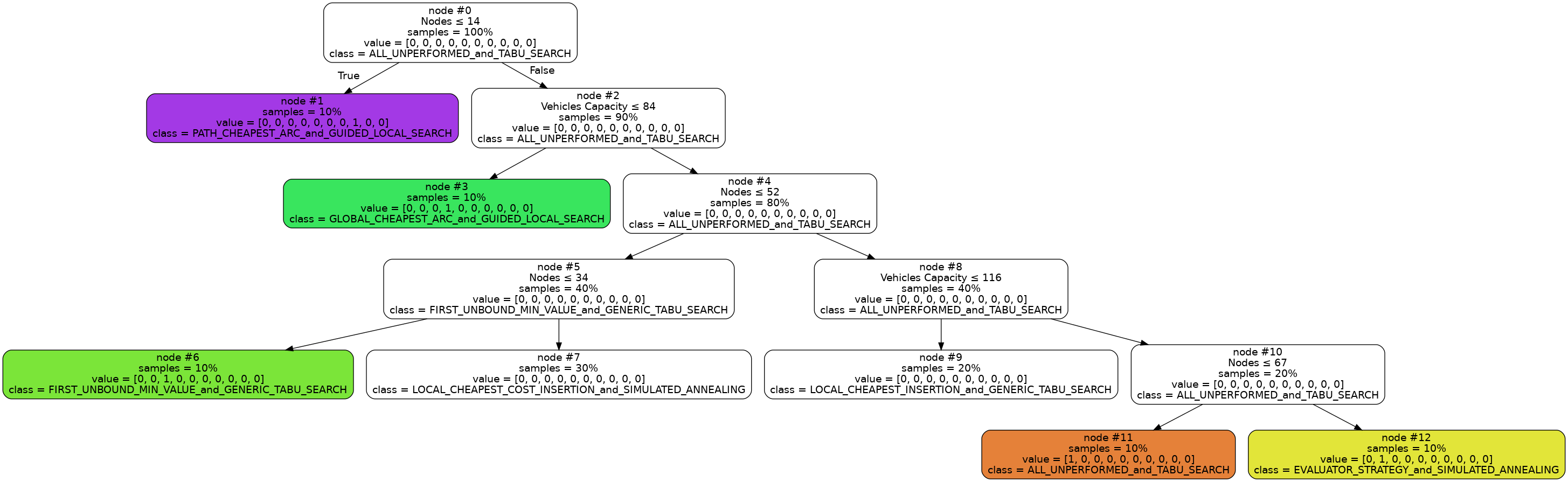


**Ilustración 49:** Gráfico de las características relevantes de la predicción de VRPPD utilizando Manhattan.

Como se observa en la primera imagen, la capacidad de los vehículos fue una característica clave en el caso de la distancia Euclidiana. Para la distancia Manhattan, aunque también tuvo relevancia, la cantidad de nodos desempeñó un papel aún más significativo.

A continuación, la Ilustración 30 muestra uno de los árboles generados dentro del bosque de decisión utilizando la distancia Euclidiana, mientras que la Ilustración 31 presenta el árbol correspondiente al caso de la distancia Manhattan.

**Ilustración 50:** Árbol de predicción para VRPPD utilizando Euclidean.

**Ilustración 51:** Árbol de predicción para VRPPD utilizando Manhattan.

Los parámetros que mejor desempeño tuvieron fueron:

* Euclidean:  
  {'bootstrap': True, 'criterion': 'entropy', 'max\_depth': 30, 'max\_features': 'sqrt', 'min\_samples\_leaf': 8, 'min\_samples\_split': 8, 'n\_estimators': 221}
* Manhattan:

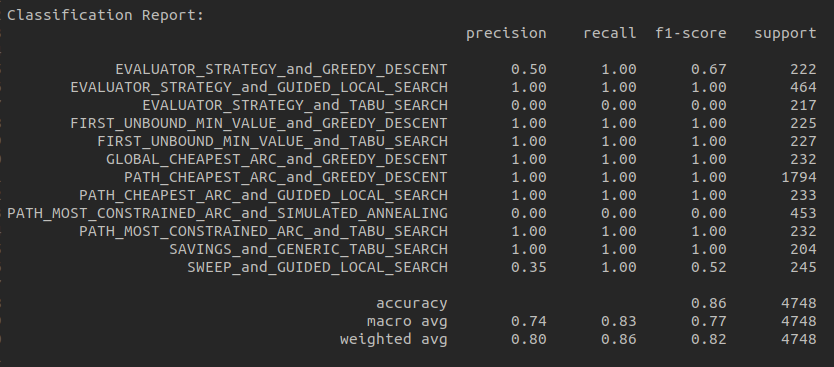
{'bootstrap': False, 'criterion': 'gini', 'max\_depth': 50, 'max\_features': 'sqrt', 'min\_samples\_leaf': 2, 'min\_samples\_split': 13, 'n\_estimators': 513}

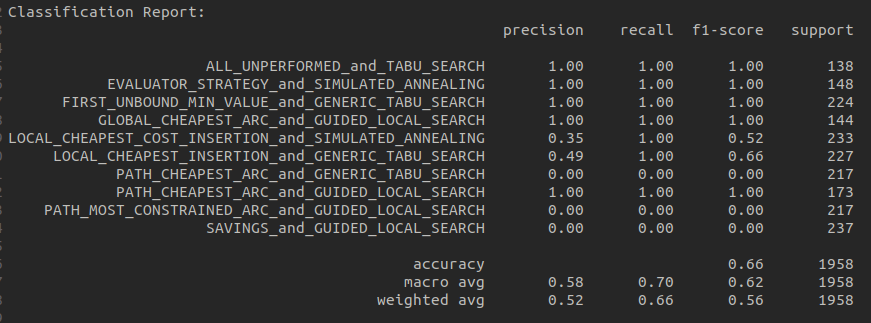
Las métricas analizadas reflejaron resultados satisfactorios como se detalla a continuación:

* Euclidean:
  + Precisión (*Accuracy*) = 0.8589
  + Puntuación F1 ponderada (*F1 Score*) = 0.8187
  + Validación cruzada (*Cross Validation Score*) = [0.85781991, 0.85781991, 0.85781991, 0.85781991, 0.85781991]
  + Informe de clasificación detallado (*Classification Report*): la Ilustración 32 muestra el reporte correspondiente. El promedio macro de F1-score fue 0.77, indicando un rendimiento aceptable en términos generales, aunque con variabilidad entre clases. Las mejores estrategias que lograron una precisión y recall del 100%. resultaron ser:
    - GUIDED\_LOCAL\_SEARCH
    - TABU\_SEARCH
    - GREEDY\_DESCENT

en combinación con:

* + - FIRST\_UNBOUND\_MIN\_VALUE
    - GLOBAL\_CHEAPEST\_ARC
    - PATH\_CHEAPEST\_ARC
* Manhattan:
  + Precisión (*Accuracy*) = 0.6573
  + Puntuación F1 ponderada (*F1 Score*) = 0.5602
  + Validación cruzada (*Cross Validation Score*) = [0.65517241, 0.65517241, 0.65517241, 0.65517241, 0.65517241]
  + Informe de clasificación detallado (*Classification Report*): la Ilustración 33 muestra el reporte correspondiente. El modelo alcanzó un 66% de precisión, clasificando correctamente varias estrategias, pero con dificultades en algunas combinaciones. Las estrategias que lograron una precisión y recall del 100% fueron:
    - ALL\_UNPERFORMED\_and\_TABU\_SEARCH
    - EVALUATOR\_STRATEGY\_and\_SIMULATED\_ANNEALING
    - FIRST\_UNBOUND\_MIN\_VALUE\_and\_GENERIC\_TABU\_SEARCH
    - GLOBAL\_CHEAPEST\_ARC\_and\_GUIDED\_LOCAL\_SEARCH
    - PATH\_CHEAPEST\_ARC\_and\_GUIDED\_LOCAL\_SEARCH

**Ilustración 52:** Reporte de clasificación para VRPPD utilizando Euclidean.

**Ilustración 53:** Reporte de clasificación para VRPPD utilizando Manhattan.

**8.6.7. Conclusiones generales**

* El desempeño varió significativamente según el problema y la métrica de distancia utilizada. En general, la distancia Euclidiana obtuvo mejores resultados que Manhattan en varios escenarios, reflejando mayor precisión y mejor desempeño en clasificación. Sin embargo, en el caso de CVRP, la distancia Manhattan superó a Euclidean, alcanzando un 66% de precisión frente a un 35%.
* Para VRPPD, las estrategias GUIDED\_LOCAL\_SEARCH, TABU\_SEARCH y GREEDY\_DESCENT combinadas con heurísticas como GLOBAL\_CHEAPEST\_ARC y PATH\_CHEAPEST\_ARC lograron los mejores resultados. Mientras que para CVRP y VRPTW, ciertas combinaciones de estrategias lograron buenos resultados, aunque el desempeño general fue menor.
* MDVRP y TSP mostraron el mejor desempeño, alcanzando una precisión y F1-score de 1.0, indicando que el modelo clasificó correctamente todas las instancias.
* VRPTW tuvo el peor desempeño, con precisiones inferiores al 13%, lo que sugiere que el modelo no fue capaz de generalizar bien para este problema.
* En CVRP, el desempeño fue moderado, con un fuerte impacto del desbalance de clases en los resultados.
* En varios casos, se observó que las clases con mayor cantidad de instancias fueron mejor clasificadas, mientras que las clases menos representadas tuvieron peores métricas. Esto indica que el modelo podría estar sesgado hacia las clases con mayor frecuencia, afectando el rendimiento en escenarios con menor representación de datos.