Artículo Original

Biblioteca para resolver problemas de asignación de clientes a depósitos en la planificación de rutas de vehículos

Library for solving the customer-to-depot assignment in vehicle routing problems

Eric Ramos Aragón[1] *, Isis Torres Pérez, Alejandro Rosete Suárez, Ananda de la Caridad Morales Morales

[1]Facultad de Ingeniería Informática. Universidad Tecnológica de La Habana "José Antonio Echeverría" (CUJAE). Calle 114 No 11901 entre 119 y 127, Marianao, La Habana, Cuba.
*Autor de correspondencia: ceramos@ceis.cujae.edu.cu

Resumen

Los problemas de optimización asociados a la planificación de rutas de vehículos son muy frecuentes en la vida moderna, donde no solo es importante en la logística de las cadenas de suministros, sino también la gestión eficiente de recursos para servicios de transporte público, recolección de residuos sólidos y la distribución urbana de mercancías. Un caso concreto es el Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos (MDVRP), que requiere en su resolución dividir el problema en dos partes: la asignación de clientes a depósitos y la construcción de las rutas. La asignación óptima de clientes a depósitos es un problema complejo en sí mismo, que considera restricciones de capacidad y acceso, a la vez que intenta reducir los costos de las rutas creadas. En este trabajo se presenta BHAVRP, una biblioteca ofrecida como software libre, que implementa en Java y Python un conjunto de algoritmos para la asignación de clientes a depósitos y se integra fácilmente en soluciones académicas y empresariales debido a su flexibilidad. A través de un conjunto de experimentos computacionales realizados sobre 33 instancias clásicas del MDVRP, se evaluó el desempeño de los algoritmos implementados, destacándose la adaptación de UPGMC con los mejores resultados en términos de calidad de la solución.

Palabras clave: Problema Planificación de Rutas de Vehículos con Múltiples Depósitos, asignación de clientes a depósitos, algoritmos heurísticos, software de código abierto.

Abstract

Optimization problems related to vehicle routing planning are very common in modern life, where not only is essential for the logistics of supply chains, but also the efficient management of resources for public transportation services, solid waste collection, and urban goods distribution. A specific case is the Multi-Depot Vehicle Routing Problem (MDVRP), which requires splitting the problem into two parts: assigning customers to depots and constructing the routes. The optimal assignment of customers to depots is a complex problem in itself, considering capacity and access constraints while aiming to minimize the cost of the resulting routes. This work presents BHAVRP, an open-source library implemented in Java and Python, which offers a set of algorithms for customer-to-depot assignment and can be easily integrated into academic and business solutions due to its flexibility. Through a set of computational experiments conducted on 33 classical MDVRP instances, the performance of the implemented algorithms was evaluated, with the adaptation of UPGMC standing out by achieving the best results in terms of solution quality.

Keywords: Multiple Depot Vehicle Routing Problem, customer-to-depot assignment, heuristic algorithms, open-source software.

1. Introducción

Los Problemas de Planificación de Rutas de Vehículos (*Vehicle Routing Problem*, VRP) son problemas de optimización combinatoria que tienen como objetivo minimizar los costos siempre que se cumplan con las restricciones existentes. Además, cuentan con una flota de vehículos y buscan satisfacer las demandas de clientes dispersos geográficamente [1]. Este tipo de problemas se clasifican como NP-duros [2], por lo que los métodos de optimización exactos resultan poco prácticos para instancias de gran tamaño, debido a sus altos tiempos de procesamiento. En su lugar, se emplean métodos aproximados, como los algoritmos heurísticos y metaheurísticos, que permiten encontrar soluciones de buena calidad en tiempos razonables [3].

Una de las variantes de VRP más estudiadas es el Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos (*Multi Depot Vehicle Routing Problem*, MDVRP) [4], que constituye una extensión del VRP clásico (*Capacitated Vehicle Routing Problem*, CVRP) [5], donde cada vehículo comienza y termina su ruta en el mismo depósito y limita el tamaño de la flota de vehículos. La diferencia entre estos problemas radica en que la variante MDVRP incorpora varios depósitos con ubicaciones diferentes, y los clientes deben ser servidos con los vehículos pertenecientes al depósito que les fue asignado [6].

Debido a la complejidad de estos problemas, los investigadores han concluido que encontrar una solución óptima es extremadamente costosa en cuanto a tiempo. Por tanto, es usual el uso de heurísticas y metaheurísticas como alternativa de solución [7]. Esta investigación se enfoca en las heurísticas que trabajan en dos fases de acuerdo a la estrategia "Agrupar primero, planificar después" (*Cluster First – Route Second*, CFRS) [8], ya que son las que mayormente se emplean en este contexto, según la literatura [7] [9]. Esta estrategia consiste en realizar la fase de asignación de clientes a depósitos para luego planificar las rutas que van a realizar los vehículos de cada depósito.

Para cumplir con la primera etapa de esta estrategia existen diferentes algoritmos, también denominados heurísticas de asignación. Según [10] [11] [12], estas heurísticas se clasifican en tres categorías principales: asignación a través de urgencias (Asignación en paralelo, Asignación simplificada y Asignación en barrido), asignación cíclica (Algoritmo de asignación cíclica) y asignación por clústeres (Coeficiente de propagación y Agrupamiento por tres criterios). Además de estas heurísticas de asignación, en la literatura existen diversos trabajos donde se utilizan algoritmos de agrupamiento entre los que destacan *K-means* y *K-medoids* [13] [14].

Para facilitar el desarrollo y uso de estas técnicas, han surgido diversas bibliotecas y herramientas que brindan una integración eficaz de estos algoritmos en sistemas complejos. Algunos ejemplos de componentes de software desarrollados en Python y C++ son Google OR-Tools [15], VROOM [16] y Scikit-learn [17], y otros desarrollados en Java como JSprit [18] y OptaPlanner [19], ofrecen soluciones especializadas para el MDVRP, resolviendo la fase de asignación mediante el uso de técnicas de optimización combinatoria y algoritmos de agrupamiento. Sin embargo, estas herramientas presentan limitaciones cuando se trata de proporcionar soluciones flexibles y modulares para resolver la fase de asignación de clientes a depósitos [20]. La Tabla 1 presenta una comparación entre las herramientas antes mencionadas y otras desarrolladas en el contexto académico cubano como CaSVRP [21] y BHCVRP [22]. Las filas corresponden a cada componente de software analizado, mientras que las columnas detallan aspectos como las estrategias que utilizan para resolver el MDVRP: heurísticas clásicas de asignación (HCA), algoritmos de agrupamiento (AA) o heurísticas de construcción (HC), evaluación de resultados, lenguajes soportados, interoperabilidad y si es de código abierto.

Tabla 1. Comparación entre herramientas que resuelven MDVRP

Herramientas	Estrategia que utilizan	Evaluación de resultados	Lenguajes soportados	Interoperabilidad	Código abierto
JSprit	HC	No	Java	Limitada	Sí
Google OR-Tools	HCA	No	C++, Python, Java, C#	Alta	Sí
VROOM	HCA	No	C++	Alta	No
Scikit-learn	AA	Sí	Python	Alta	Sí
OptaPlanner	HCA	No	Java	Limitada	Sí
CaSVRP	HCA	No	Java	Baja	Sí
BHCVRP	HCA	No	Java, Python	Baja	Sí

Dado este panorama, existe una necesidad de disponer de una herramienta que combine flexibilidad, modularidad y facilidad de uso, además de ofrecer soporte para las estrategias clásicas de asignación y los algoritmos de agrupamiento. Esta herramienta debe garantizar interoperabilidad con sistemas externos y estar disponible como software de código abierto, facilitando su integración en entornos académicos y empresariales diversos.

Como respuesta a esta necesidad, se propone la Biblioteca de Heurísticas de Asignación para Problemas de Planificación de Rutas de Vehículos con Múltiples Depósitos (BHAVRP). Esta biblioteca está diseñada para resolver la fase de asignación de clientes a depósitos en los MDVRP. BHAVRP integra un conjunto de heurísticas clásicas de asignación y algoritmos de agrupamiento. La biblioteca propuesta se presenta en dos lenguajes de programación, Java y Python y su diseño modular fomenta la experimentación y personalización durante la resolución de los MDVRP.

El objetivo de este trabajo es evaluar el desempeño y la equivalencia funcional de las implementaciones en Java y Python de BHAVRP, así como analizar el comportamiento de los algoritmos de asignación desde diferentes criterios. Con este propósito, el artículo está estructurado de la siguiente manera: en la Sección 2 se describe BHAVRP, detallando los algoritmos de asignación disponibles y la arquitectura de la biblioteca en sus versiones Java y Python. En la Sección 3 se presentan los experimentos realizados para evaluar el desempeño de los algoritmos de asignación en ambas implementaciones, describiendo las instancias utilizadas, el entorno de prueba y los análisis estadísticos aplicados para comparar los resultados obtenidos. Por último, la Sección 4 presenta las conclusiones y se resaltan las implicaciones prácticas de los resultados alcanzados con BHAVRP.

2. Materiales y Métodos

En BHAVRP están disponibles 18 algoritmos de asignación, de las cuales seis son heurísticas de asignación conocidas en la literatura, mientras que cinco son adaptaciones de algoritmos de agrupamiento empleados en la minería de datos [23]. Se puede apreciar en la Tabla 2, Tabla 3 y Tabla 4, una breve descripción de estos algoritmos organizados en: heurísticas clásicas de asignación (basadas en urgencias, ciclos y clústeres), algoritmos de agrupamiento (particionales y jerárquicos), además de otros enfoques (aleatorios, estrategias fundamentadas en distancias y otras inspiradas en la asignación cíclica). Todos los algoritmos implementados consideran las restricciones propias del problema, como la capacidad de los depósitos y la demanda de los clientes.

Tabla 2. Heurísticas clásicas de asignación implementados en BHAVRP

Subcategoría	Algoritmo	Descripción
Asignación a través de urgencias	Parallel (P)	Asigna considerando las distancias a todos los depósitos disponibles [24].
	Simplified (S)	Asigna teniendo en cuenta los dos depósitos más cercanos a cada clientes [24].
	Sweep (SW)	Asigna teniendo en cuenta el depósito con mayor demanda insatisfecha y el depósito más cercano a cada cliente [24].
Asignación cíclica	Cyclic Assignment (CA)	Asigna de forma cíclica el cliente más cercano al último cliente asignado [24].
Asignación por clústeres	Coefficient Propagation (CP)	Asigna los clientes a los depósitos según la distancia escalada [24].
	Three Criteria Clustering (TCC)	Asigna de acuerdo a tres criterios: distancia promedio a los clústeres, varianza de las distancias promedio, y distancia al cliente más cercano asignado en los clústeres [24].

Tabla 3. Adaptaciones de algoritmos de agrupamiento implementados en BHAVRP

Subcategoría	Algoritmo	Descripción
Agrupamiento particional	K-means (KM)	Asigna seleccionando los clientes con la menor distancia a un depósito [25].
	Partitional Around Medoids (PAM)	Realiza diferentes asignaciones de los clientes a los depósitos con el objetivo de encontrar la asignación que minimice el costo [25].
	Farthest-First (FF)	Asigna seleccionando como centroides los elementos con la mayor distancia entre ellos [26].
	Clustering Large Applications (CLARA)	Es una extensión de la adaptación de PAM para grandes conjuntos de datos [26].
Agrupamiento jerárquico	Unweighted Pair Group Method with Centroids (UPGMC)	Agrupa los clústeres más cercanos hasta que el número de clústeres sea equivalente al número de depósitos [25].

Tabla 4. Algoritmos con otros enfoques implementados en BHAVRP

Subcategoría	Algoritmo	Descripción
Basadas en distancia	Best Nearest (BN)	Asigna teniendo en cuenta la mejor distancia entre clientes y depósitos [10].
	Nearest by Customer (NC)	Asigna el cliente más cercano al depósito seleccionado [10].
	Nearest by Depot (ND)	Asigna a un depósito sus clientes más cercanos [20].
Enfoque aleatorio	Random by Element (RE)	Asigna aleatoriamente clientes a depósitos [24].
Basadas en cíclicas	Best Cyclic Assignment (BCA)	Asigna de forma cíclica el cliente más cercano al depósito, tomando como referencia el último cliente asignado [20].
	Sequential Cyclic (SC)	Asigna el cliente más cercano al último cliente asignado, teniendo en cuenta un depósito a la vez [20].
	Random Sequential Cyclic (RSC)	Introduce aleatoriedad en el algoritmo SC [20].

El desarrollo de BHAVRP se basa en la implementación dual en Java y Python, aprovechando las fortalezas inherentes de ambos lenguajes para abordar diferentes necesidades. Java, como lenguaje compilado, destaca por su rendimiento optimizado gracias a las optimizaciones directas en el código antes de su ejecución. Por otro lado, Python ofrece flexibilidad y facilidad de uso, lo que facilita el prototipado rápido y la integración con bibliotecas científicas ampliamente utilizadas. Esta dualidad permite adaptar BHAVRP a diversos contextos, utilizando Java cuando el rendimiento computacional es prioritario y Python para investigaciones académicas o desarrollos ágiles. Así, la elección del lenguaje se alinea con las necesidades específicas del proyecto, destacando la versatilidad de la biblioteca.

Para soportar esta variedad de algoritmos y garantizar su correcta ejecución, BHAVRP presenta una arquitectura n-capas con un enfoque basado en reutilización, promoviendo una consistencia con los lenguajes de desarrollo seleccionados. Esta arquitectura se presenta en la Figura 1, que ofrece una vista detallada de la organización en capas de BHAVRP.

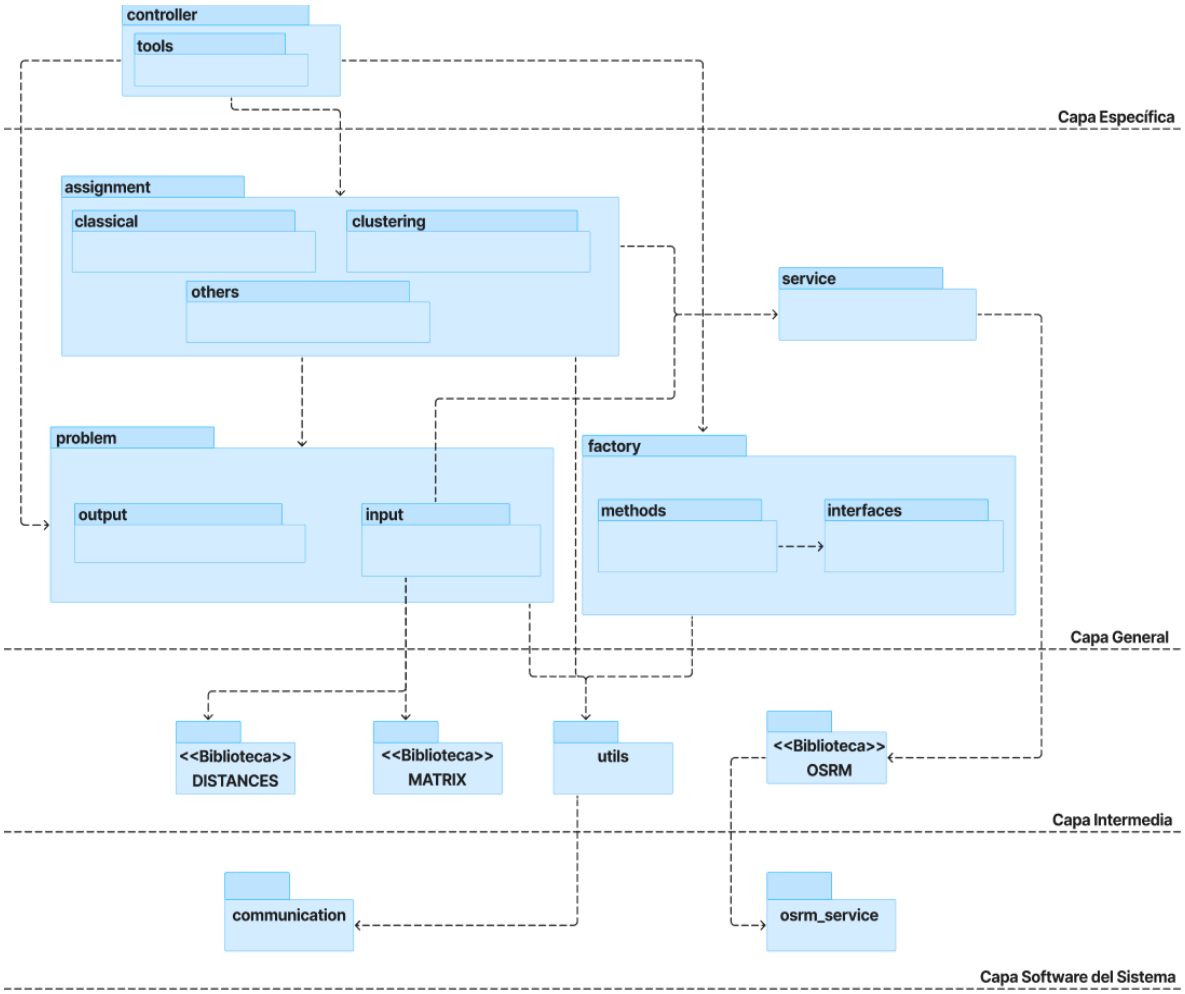

Fig.1 Vista de la arquitectura de BHAVRP

La arquitectura de BHAVRP mantiene un diseño similar independientemente del lenguaje utilizado, con cambios principalmente en la Capa Intermedia y en la Capa Software del Sistema. En la Capa General, que representa el núcleo de la biblioteca, se conservan los mismos principios de diseño, algoritmos y estructuras de datos en ambas implementaciones. Esta capa es responsable de la definición del problema, la modelación de las instancias y la ejecución de los algoritmos de asignación.

La biblioteca trabaja con una estructura de datos que representan los elementos clave del MDVRP: clientes, depósitos, y sus respectivas restricciones y parámetros. BHAVRP permite cargar instancias de problemas en formatos estándar como *.data* o *.txt*. La estructura básica para utilizar la biblioteca incluye un encabezado con el número de depósitos, clientes y vehículos disponibles por depósito, seguido de las capacidades máximas de los vehículos. Los clientes y los depósitos están definidos por un identificador y sus coordenadas (x, y). Además, cada cliente tiene una demanda asociada, mientras que cada depósito cuenta con una capacidad equivalente a la suma de las capacidades de los vehículos en su flota.

Otro aspecto clave en los problemas de planificación de rutas de vehículos es el manejo de distancias. En este sentido, BHAVRP permite trabajar tanto con distancias reales como aproximadas (*Euclidean*, *Manhattan*, *Chebychev* y *Haversine*). El cálculo de distancias es un aspecto fundamental en los problemas de planificación de rutas de vehículos. Las distancias aproximadas, aunque simples y de bajo costo computacional, no siempre reflejan la realidad debido a la falta de consideración de factores como la infraestructura vial o las condiciones de tráfico. Para abordar esta limitación, BHAVRP integra herramientas como OSRM (*Open Source Routing Machine*) [27], que calcula distancias y tiempos de viaje basados en mapas reales, mejorando significativamente la precisión de las soluciones, especialmente en escenarios urbanos complejos.

Por otra parte, en la Capa Intermedia, la versión en Java emplea bibliotecas desarrolladas específicamente para BHAVRP, mientras que la versión en Python adopta un enfoque basado en bibliotecas de código abierto, como *numpy* [28] y *scipy* [29]. Además, los protocolos de comunicación también son específicos de cada lenguaje: Java utiliza la JVM (*Java Virtual Machine*), mientras que Python emplea *buffers*. La biblioteca en ambos lenguajes se encuentra disponible en el repositorio de GitHub: https://github.com/vrpsolutions/BHAVRP.

Por último, la arquitectura de BHAVRP implementa los principios SOLID [30] para garantizar un diseño bien organizado. Entre estos principios, destacan especialmente el Principio de Segregación de Interfaces (ISP) y el Principio de Inversión de Dependencias (DIP), que promueven modularidad, bajo acoplamiento y alta cohesión en el sistema. Estos principios se complementan con los patrones de diseño implementados en la biblioteca, reforzando la asignación adecuada de responsabilidades y mejorando la flexibilidad del código. En cuanto a los patrones de diseño GoF [31], se aplican principalmente patrones creacionales para gestionar la creación de objetos y promover un diseño flexible y reusable dentro del sistema. Tal es el caso del patrón *Factory Method* [31], que facilita la selección dinámica de los algoritmos, mejorando la modularidad y permitiendo futuras extensiones. Otro de los patrones adoptados es el *Template Method* [30] que define una estructura común para los 18 algoritmos, asegurando consistencia mientras se permiten variaciones específicas. Finalmente, se incorporan los patrones GRASP [32], como Bajo Acoplamiento y Alta Cohesión, que complementan los principios SOLID y los patrones GoF.

3. Resultados y Discusión

Para evaluar la equivalencia entre las implementaciones de BHAVRP en los lenguajes Java y Python, se lleva a cabo un análisis experimental dividido en tres etapas principales. En primer lugar, se aplica la prueba de Wilcoxon [33] para determinar si existen diferencias significativas en los costos de las soluciones y en los tiempos de ejecución obtenidos en ambas versiones. Posteriormente, se emplea la prueba estadística no paramétrica de Friedman $1 \times N$ para comparar el rendimiento de los algoritmos de asignación, seguido de un análisis *post-hoc* para identificar si existen diferencias específicas entre ellos [34].

Estos análisis se realizan considerando que la calidad de la asignación de clientes a depósitos se evalúa indirectamente a través de los costos de las rutas generadas, tal como se discute en trabajos previos [10] [35]. En estos estudios, se evidencia que una mala asignación inicial puede incrementar sustancialmente los costos de las rutas [36], lo que subraya la importancia de garantizar una buena

asignación como paso crítico para resolver el MDVRP. Por tanto, todos los análisis realizados se basan en el resultado que se obtiene con la construcción de las rutas.

Para el estudio experimental se utilizaron las 33 instancias de problemas propuestas por [37] en sus investigaciones sobre el MDVRP. Estas instancias son reconocidas en la comunidad científica de la optimización de rutas y fueron seleccionadas para evaluar el desempeño de los algoritmos implementados en BHAVRP. En la Tabla 5 se presenta para cada instancia del problema el total de clientes, el total de depósitos, las cantidades de vehículos por depósito y las capacidades de los mismos. Es importante señalar que las instancias que comparten un mismo identificador en la celda de 'ID' son equivalentes en términos de la cantidad de clientes, depósitos, vehículos por depósito y capacidad de los vehículos.

Tabla 5. Descripción de las instancias MDVRP

ID	Total de clientes	Total de depósitos	Total de vehículos por depósito	Capacidad de los vehículos
p01	50	4	4	80
p02	50	4	2	160
p03	75	5	3	140
p04	100	2	8	100
p05	100	2	5	200
p06	100	3	6	100
p07	100	4	4	100
p08	249	2	14	500
p09	249	3	12	500
p10	249	4	8	500
p11	249	5	6	500
p12, p13, p14	80	2	5	60
p15, p16, p17	160	4	5	60
p18, p19, p20	240	6	5	60
p21, p22, p23	360	9	5	60
pr1	48	4	1	200
pr2	96	4	2	195
pr3	144	4	3	190
pr4	192	4	4	185
pr5	240	4	5	180
pr6	288	4	6	175
pr7	72	1	6	200
pr8	144	2	6	190
pr9	216	6	3	180
pr10	288	6	4	170

Todos los experimentos se llevan a cabo en un entorno controlado, utilizando un ordenador con las siguientes especificaciones: procesador Intel CORE i5-6400 de 2.7 GHz, 8 GB de memoria RAM y sistema operativo Windows 11 Pro de 64 bits. Cada algoritmo se ejecuta 20 veces por instancia y para la planificación de rutas, se emplea la heurística *NearestNeighbourWithRLC* con un valor de RLC igual a 1, para garantizar que tenga un comportamiento determinista. Esta heurística de construcción es ejecutada desde la Biblioteca de Heurísticas de Construcción para Problemas de Planificación de Rutas de Vehículos (BHCVRP) [22].

En la Figura 2 se presentan los diagramas de cajas y bigotes con las distribuciones de costos por algoritmos en ambas implementaciones de BHAVRP, destacando que la mayoría de los algoritmos presentan una distribución compacta. Al comparar cada par de algoritmos, se observa consistencia en la calidad de las soluciones obtenidas en ambas implementaciones. Este resultado se debe a la

naturaleza determinista de la mayoría de los algoritmos de BHAVRP, ya que las distribuciones de costos son similares, y los valores atípicos, principalmente en algoritmos como RE, son consistentes en ambos lenguajes, respaldando la equivalencia funcional en términos de calidad de las soluciones generadas.

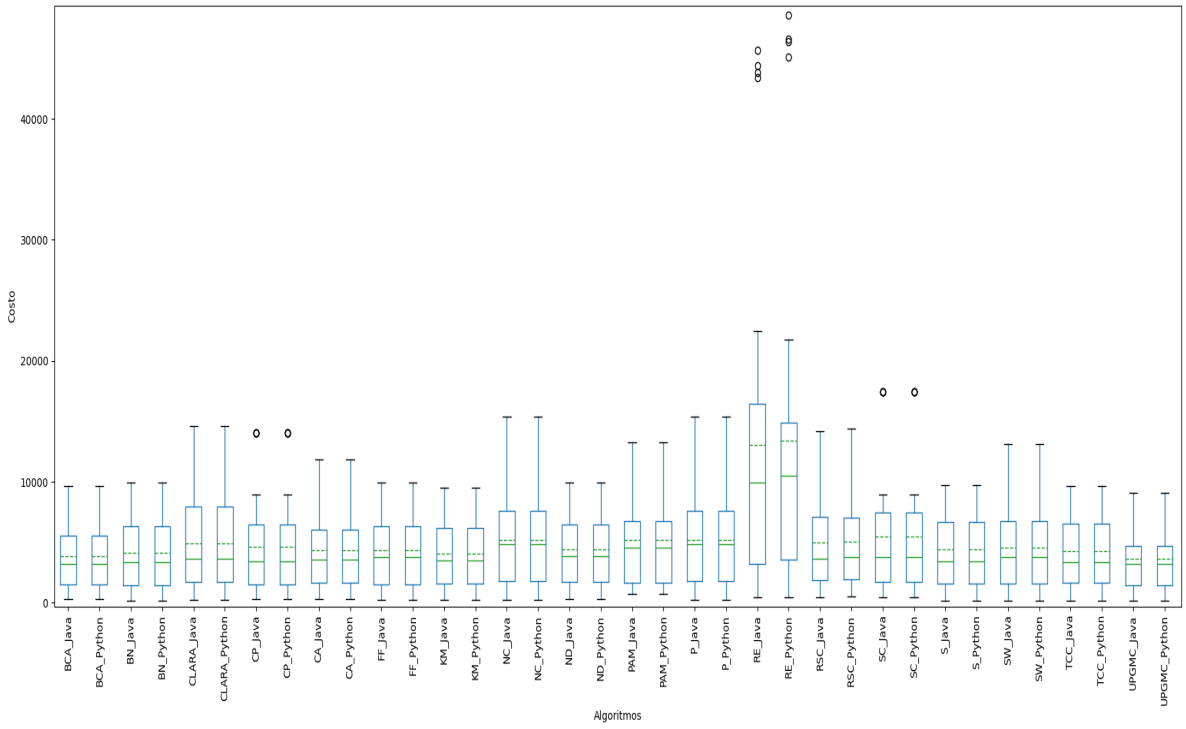

Fig.2 Distribución de costos por algoritmo en ambos lenguajes

Sin embargo, para los algoritmos estocásticos con su inherente variabilidad en los resultados, se procede a aplicar la prueba estadística no paramétrica de Wilcoxon para comprobar si existen diferencias estadísticamente significativas entre las distribuciones de costos generadas por ambas implementaciones. En la Tabla 6 se muestran los resultados obtenidos con Wilcoxon para los algoritmos RE y RSC, comparando sus implementaciones en Java y Python. Al trabajar con un nivel de significancia de $\propto = 0.05$, los resultados indican que no existen diferencias significativas entre las implementaciones para ninguno de los dos algoritmos. Esto sugiere que ambas versiones son funcionalmente equivalentes en términos de la calidad de las soluciones generadas.

Tabla 6. Resultados del test de Wilcoxon en cuanto a la calidad de las soluciones con los algoritmos estocásticos en Java y Python

Algoritmos	R^+	R^-	*p-value*
RE	381.0	180.0	0.07372
RSC	329.0	232.0	0.38129

En la Figura 3 se presentan los diagramas de cajas y bigotes con las distribuciones de tiempos por algoritmos en ambas implementaciones de BHAVRP, destacando que se aprecian diferencias entre las implementaciones.

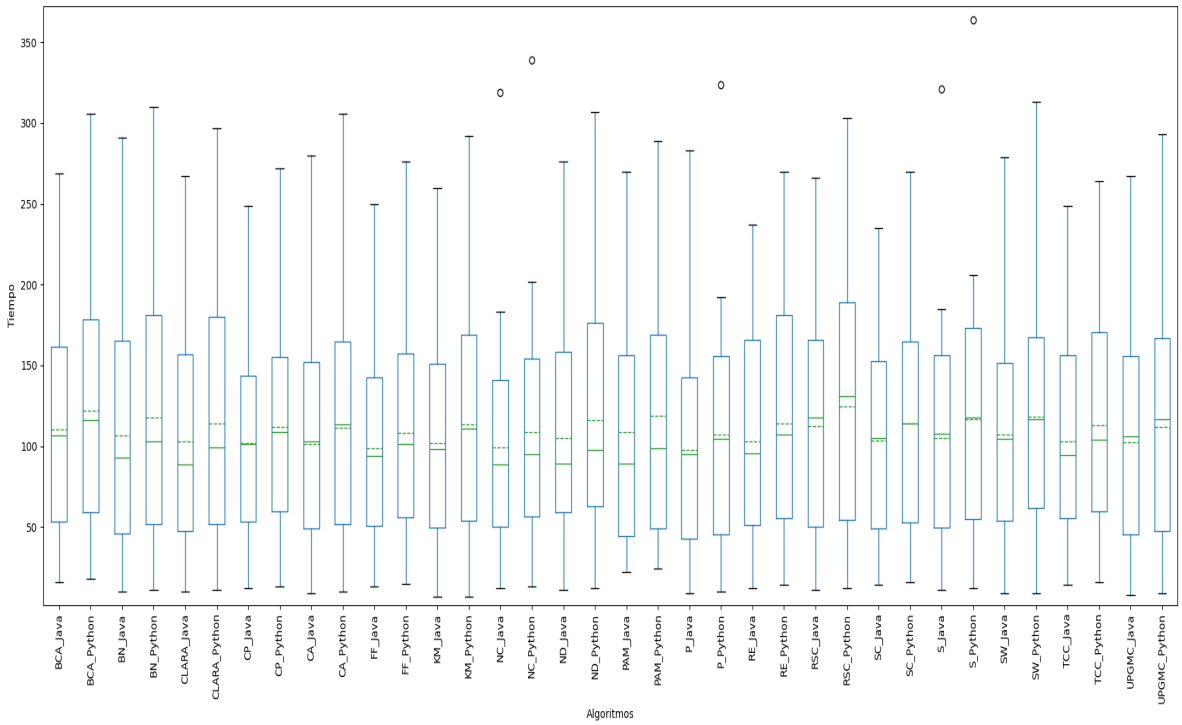

Fig.3 Distribución de tiempos por algoritmo en ambos lenguajes

Para evaluar estas diferencias, se aplica también la prueba de Wilcoxon a los tiempos promedios de ejecución de todos los algoritmos en ambas implementaciones de la biblioteca para todas las instancias analizadas. Los resultados indican que existen diferencias estadísticamente significativas entre las dos implementaciones, con un valor p extremadamente pequeño de 1.32×10^{-10}, lo que confirma que los tiempos de ejecución en Python son notablemente mayores que en Java.

Los resultados obtenidos eran esperables dado que Java es un lenguaje compilado, lo que permite optimizar el código directamente a nivel de máquina antes de su ejecución. En contraste, Python es un lenguaje interpretado, lo que introduce sobrecarga adicional durante la ejecución debido a la interpretación dinámica del código. Como resultado, aunque ambas implementaciones producen soluciones equivalentes en términos de calidad, la versión en Java supera a la versión en Python en términos de velocidad de ejecución.

Para evaluar el desempeño de los 18 algoritmos implementados en BHAVRP en términos de la calidad de las soluciones generadas (costo total de las rutas construidas por BHCVRP), se realiza un análisis estadístico utilizando la prueba estadística no paramétrica de Friedman $1 \times N$. Esta prueba arrojó un valor de 278.49 con un *p-valor* de 3.03×10^{-49}, lo que indica que existen diferencias significativas entre al menos dos de los algoritmos evaluados. Este resultado justifica la realización de un análisis *post-hoc* para caracterizar estas diferencias.

En la Tabla 7 se presentan los rangos promedio obtenidos por cada algoritmo. Los algoritmos con menor rango promedio son considerados los de mejor rendimiento. Al observar los rangos promedio, destaca la adaptación de UPGMC, que obtuvo el menor rango promedio (2.5), seguido de los algoritmos BCA y KM, lo que sugiere que estos métodos presentan los mejores resultados en términos de calidad de solución.

Tabla 7. Resultados de los rangos promedio por algoritmo

Algoritmos	Rangos promedio
UPGMC	**2.5**
BCA	4.54
KM	5.44
BN	6.31
CP	7.28
S	7.5
TCC	8.29
SW	8.68
CA	8.82
FF	9.29
ND	10.57
CLARA	11.41
RSC	11.74
SC	12.41
P	12.71
NC	12.71
PAM	12.88
RE	17.91

Estos resultados preliminares motivan la aplicación de análisis *post-hoc* para determinar con precisión si existen diferencias estadísticamente significativas en el desempeño de la adaptación de UPGMC, algoritmo de referencia, y el resto de los algoritmos. A continuación, se describen los resultados obtenidos mediante los métodos *post-hoc* de Finner y Li [38] [39].

Con base en los resultados obtenidos, se puede concluir que existen diferencias significativas en el desempeño de varios algoritmos en términos de costos en la distancia euclidiana. Según el método de Finner, UPGMC muestra un desempeño superior en comparación con los 17 algoritmos restantes, destacándose costos inferiores con un nivel de significancia de 0.05. Estos hallazgos son respaldados por el método de Li, que refuerza la superioridad estadística de UPGMC frente a alternativas menos eficientes.

Para las 33 instancias utilizadas en el estudio, la adaptación de UPGMC obtuvo el menor costo de solución en 15 de ellas, seguido por BCA con 11 instancias y KM con 4. Otros algoritmos como CA, ND y FF fueron los mejores en solo una instancia.

Para determinar qué algoritmo es más aconsejable en cada una de las instancias del MDVRP, se decide construir un árbol de decisión utilizando en la herramienta KNIME [40], el nodo *Decision Tree Learner*, que implementa una variante del algoritmo C4.5 (J48) [41]. El árbol de decisión obtuvo una precisión global de 75.76% y una cobertura del 100%, en función de parámetros clave, como la cantidad de depósitos, la demanda total de los clientes, la capacidad promedio por vehículo y el número de clientes. Las reglas obtenidas son:

1. Si la cantidad de depósitos es menor o igual a 3 y la capacidad de los vehículos es menor o igual a 195, entonces el algoritmo recomendado es UPGMC.
2. Si la cantidad de depósitos es menor o igual a 3 y la capacidad de los vehículos es mayor a 195, entonces el algoritmo recomendado es BCA.
3. Si la cantidad de depósitos está entre 4 y 5, y la capacidad total de los vehículos es menor o igual a 395, entonces el algoritmo recomendado es BCA.
4. Si la cantidad de depósitos está entre 4 y 5, la capacidad total de los vehículos es mayor a 395, y la capacidad de cada vehículo es menor o igual a 182, entonces el algoritmo recomendado es CA.

5. Si la cantidad de depósitos está entre 4 y 5, la capacidad total de los vehículos es mayor a 395, y la capacidad de cada vehículo es mayor a 182, entonces el algoritmo recomendado es BCA.
6. Si la cantidad de depósitos supera 5, entonces el algoritmo recomendado es UPGMC, independientemente de otras características.

4. Conclusiones

El presente estudio introduce la biblioteca de heurísticas BHAVRP. La biblioteca representa una herramienta modular que aborda la fase de asignación de clientes a depósitos en el contexto del MDVRP en los lenguajes Java y Python.
Los experimentos realizados demuestran que no hay diferencia en términos de calidad de las soluciones generadas. Sin embargo, la versión en Java presenta tiempos de ejecución significativamente menores debido a su naturaleza compilada, lo que la hace más adecuada para aplicaciones donde el rendimiento computacional es crítico. El análisis estadístico mediante la prueba estadística no paramétrica de Friedman y los procedimientos *post-hoc* (Finner y Li) reveló que la adaptación del algoritmo UPGMC presenta los mejores resultados en términos de costo para la mayoría de las instancias evaluadas. Adicionalmente, el árbol de decisión generado proporcionó un conjunto de reglas claras y estructuradas para seleccionar el algoritmo más adecuado en función de las características de las instancias; siendo UPGMC recomendado en la mayoría de los casos.
Como parte de las líneas de investigación que continúan se distingue la incorporación de métricas en BHAVRP para evaluar la calidad de las asignaciones y validar su desempeño frente a otras herramientas.

Referencias

[1] T. Vidal, G. Laporte and P. Matl, "A concise guide to existing and emerging vehicle routing problem variants.," *European Journal of Operational Research,* vol. 286, no. 3, pp. 791-802, 2020.

[2] M. R. Garey, Computers and intractability: a guide to the theory of NP-completeness., San Francisco: W. H. Freeman, 1979.

[3] M. A. Masmoudi, M. Hosny, K. Braekers and A. Dammark, "A hybrid genetic algorithm for the multi-depot vehicle routing problem with time windows.," *Computers & Industrial Engineering,* vol. 144, 2020.

[4] M. Rabbani, S. M. T. Fatemi-Ghomi and N. Kazemi, "A novel hybrid algorithm for the multi-depot vehicle routing problem with time windows.," *Journal of Cleaner Production,* vol. 278, 2021.

[5] I. Malheiros, R. Ramalho, B. Passeti, T. Bulhões and A. Subramanian, "A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem.," *Computers & Operations Research,* vol. 129, 2021.

[6] P. Toth and D. Vigo, The Vehicle Routing Problem., SIAM, 2002.

[7] Y. Zhang, Z. Li, F. Wang and Y. Zhang, "A hybrid metaheuristic algorithm for the capacitated vehicle routing problem.," *Applied Soft Computing,* vol. 97, 2020.

[8] G. W. Webb, "A comparative evaluation of heuristic methods for a two-stage vehicle routing problem.," *Computers & Operations Research,* vol. 8, no. 3, pp. 193-203, 1981.

[9] M. Gendrau, G. Laporte and R. Séguin, "A tabu search heuristic for the vehicle routing problem with time windows.," *Transportation Science,* vol. 29, no. 1, pp. 2-14, 1995.

[10] I. Torres, A. Rosete, G. Sosa-Gomez and O. Rojas, "New heuristics for assigning in the Multi-Depot Vehicle Routing Problem," *Science Direct,* vol. 55, no. 10, pp. 2228-2233, 2022.

[11] J.-J. Salazar-González, "Heuristics for the Multi-Depot Vehicle Routing Problem.," *Computers & Operations Research,* vol. 37, no. 4, pp. 722-733, 2010.

[12] N. Christofides, "Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem.," *Operations Research Forum,* vol. 3, no. 20, 2022.

[13] J. C. Molina, I. Eguia, J. Racero and F. Guerrero, "A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem.," *Computers & Operations Research,* vol. 124, 2020.

[14] Y. Zhang, Z. Li and X. Wang, "A survey on clustering algorithms for the vehicle routing problem.," *Artificial Intelligence Review,* vol. 54, no. 5, pp. 3471-3500, 2021.

[15] G. LLC, "Google OR-Tools Documentation," [Online]. Available: https://developers.google.com/optimization. [Accessed 24 01 2025].

[16] V. Project, "VROOM GitHub Repository," [Online]. Available: https://github.com/VROOM-Project/vroom. [Accessed 24 01 2025].

[17] F. Pedregosa, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[18] J. Project, "JSprit: A Java-based library for solving vehicle routing problems," [Online]. Available: https://jsprit.readthedocs.io/en/latest/. [Accessed 24 01 2025].

[19] O. Project, "OptaPlanner, The Business Process Optimization Solver," [Online]. Available: https://www.optaplanner.org/. [Accessed 24 01 2025].

[20] E. Ramos, I. Torres and A. Rosete, "Nueva versión de la Biblioteca de Heurísticas de Asignación para Problemas de Rutas de Vehículos con Múltiples Depósitos.," Universidad Tecnológica de la Habana "José Antonio Echeverría", Facultad de Ingeniería Informática, La Habana, Cuba, 2025.

[21] G. Yañez and I. Torres, "Desarrollo de nuevos modelos para el Problema del Transporte Obrero en CaSVRP.," Universidad Tecnológica de la Habana "José Antonio Echeverría", Facultad de Ingeniería Informática, La Habana, Cuba, 2021.

[22] A. Morales, I. Torres and A. Rosete, "Nueva versión de la biblioteca de heurísticas de construcción para problemas de planificación de rutas de vehículos.," Universidad Tecnológica de la Habana "José Antonio Echeverría", Facultad de Ingeniería Informática, La Habana, Cuba, 2025.

[23] J. Han, M. Kamber and J. Pei, Data Mining: Concepts and Techniques, San Francisco, CA, USA: Morgan Kaufmann, 2020.

[24] M. Machado, I. Torres and A. Rosete, "Biblioteca de heuristicas de asignación para el problema de planificación de rutas de vehículos con múltiples depósitos.," Universidad Tecnológica de la Habana "José Antonio Echeverría", Facultad de Ingeniería Informática, La Habana, Cuba, 2019.

[25] N. Pérez and I. Torres, "Incorporación de algoritmos de agrupamiento en la Biblioteca de Heurísticas de Asignación para Problemas de Planificación de Rutas de Vehículos con Múltiples Depósitos," Universidad Tecnologica de La Habana "Jose Antonio Echeverria", Facultad de Ingeniería Informática , La Habana, Cuba, 2022.

[26] E. Ramos and I. Torres, "Adaptación de algoritmos de agrupamiento para la asignación de clientes en BHAVRP.," Universidad Tecnológica de la Habana "José Antonio Echeverría", Facultad de Ingeniería Informática , La Habana, Cuba, 2022.

[27] D. Vetter and L. C., "Real-time Routing with OpenStreetMap Data," *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* pp. 513-516, 2011.

[28] C. R. Harris, "Array programming with NumPy," *Nature,* vol. 585, no. 7825, pp. 357-362, 2020.

[29] P. Virtanen, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods,* vol. 17, no. 261-272, pp. 261-272, 2020.

[30] R. C. Martin, Agile Software Development: Principles, Patterns, and Practices., Pearson Education, 2003.

[31] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software., Addison-Wesley, 1994.

[32] C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development., Pearson Education, 2002.

[33] J. D. Gibbons and S. Chakraborti, "Nonparametric Statistical Inference," CRC Press, 2020.

[34] S. Chakraborti and J. D. Gibbons, "Nonparametric Statistical Inference, 5th ed. Boca Raton.," CRC Press, FL, 2011.

[35] L. Tansini, M. Urquhart and O. Viera, "A Comparative Study of Heuristic Assignment Methods for the Multi-Depot Vehicle Routing Problem.," *Proceedings of the International Conference on Operations Research and Logistics,* pp. 123-130, 2002.

[36] L. Tansini and O. V. Libertad, "Adapted Clustering Algorithms for the Assignment Problem in the MDVRPTW," 2004.

[37] J. F. Cordeau, J. Desrosiers and M. M. Solomon, "The VRP with Multiple Depots: State-of-the-art and New Insights.," *Transportation Science,* vol. 36, no. 4, pp. 278-292, 2002.

[38] G. Finner, "On a monotonicity problem in step-down multiple test procedures," *Journal of the American Statistical Association,* vol. 88, no. 423, pp. 920-923, 1993.

[39] J. Li, "A two-step rejection procedure for testing multiple hypotheses," *Journal of Statistical Planning and Inference,* vol. 138, no. 6, pp. 1521-1527, 2008.

[40] P. A. B. van der Lans, "KNIME Analytics Platform: A Comprehensive Tool for Data Science and Machine Learnig," *Journal of Open Source Software,* vol. 6, no. 58, pp. 1-5, 2021.

[41] I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, "Data Mining: Practical Machine Learning Tools and Techniques," in *The Morgan Kaufmann Series in Data Management Systems*, San Francisco, CA, USA, Morgan Kaufmann, 2020, pp. 123-145.

Conflicto de Intereses

No hay conflictos de intereses que declarar.

Contribución de los autores

Eric Ramos Aragón. ORCID: https://orcid.org/0009-0000-9828-5558
Participó en la curación de datos, programación, investigación, metodología, validación, redacción y edición.
Isis Torres Pérez. ORCID: https://orcid.org/0000-0002-1079-8285
Participó en conceptualización, investigación, metodología, supervisión y validación.
Alejandro Rosete Suárez. ORCID: https://orcid.org/0000-0002-4579-3556
Participó en conceptualización. metodología y supervisión.
Ananda de la Caridad Morales Morales. ORCID: https://orcid.org/0009-0003-9208-8029
Participó en recursos y supervisión.