UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Posgrado en Ciencia e Ingeniería de la Computación
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Campo de Inteligencia Artificial

Colonia de hormigas con múltiples indicadores para el problema de enrutamiento de vehículos multi-objetivo bajo incertidumbre

T E S I S

que para optar por el grado de

Maestro en Ciencia e Ingeniería de la Computación

PRESENTA:
Rodrigo Fernando Velázquez Cruz

Tutor Principal:
Dr. Carlos Ignacio Hernández Castellanos
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

México, CDMX, Noviembre 2024

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y

HONESTIDAD ACADÉMICA Y PROFESIONAL

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado "Colonia de hormigas con múltiples indicadores para el problema de enrutamiento de vehículos multi-objetivo bajo incertidumbre", que presenté para obtener el grado de Maestro en Ciencia e Ingeniería de la Computación, es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Programa de Posgrado, citando las fuentes, ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación/graduación

Atentamente

Rodrigo Fernando Velázquez Cruz, 315254565

Resumen

En optimización multi-objetivo muchas veces no se considera el impacto de factores externos como la incertidumbre en el desempeño de las soluciones construidas. Sin embargo, esto es muy importante cuando se considera revolver problemas reales, donde el entorno donde se desempeñan las soluciones es poco conocido o muy cambiante. Por estar razón, es importante encontrar soluciones que se mantengan con un buen desempeño a pesar de distintos escenarios que existen y que tengan un desempeño garantizado en escenarios muy bien conocidos. En el área de cómputo evolutivo se han propuesto bastantes algoritmos multi-objetivo basados en indicadores de calidad que son competitivos para problemas de optimización muy conocidos. En particular la optimización mediante colonia de hormigas (ACO), que ha sido utilizada con existo para problemas de enrutamiento de vehículos. Principalmente porque las hormigas pueden construir soluciones a través de realizar recorridos en una gráfica. Sin embargo, existen pocos algoritmos en la literatura que usan indicadores de calidad dentro de colonia de hormigas y que obtengan soluciones robustas para problemas bajo incertidumbre.

En esta tesis se propone un nuevo algoritmo multi-objetivo cooperativo basado en múltiples indicadores de calidad para colonia de hormigas (cMIBACO). Este algoritmo se enfoca en mejorar el trabajo colectivo de las hormigas artificiales a través del intercambio de conocimiento entre distintas colonias. El algoritmo se prueba en el problema de enrutamiento de vehículos multi-objetivo generalizado consistente bajo incertidumbre (MOGenConVRP) donde la incertidumbre se encuentra en los clientes. Y se busca minimizar tres objetivos: el tiempo total de los recorridos, el número de distintos vehículos que visitan un cliente y la diferencia entre los tiempos de llegada.

Para el nuevo algoritmo se realizó un análisis sobre el impacto de los distintos componentes que tiene respecto a un algoritmo tradicional. También se evaluó su desempeño contra los algoritmos de colonia de hormigas de un solo indicador y suma ponderara, obteniendo resultados competitivos en soluciones aproximadas. Finalmente se aplicó para obtener soluciones ligeramente robustas, donde las soluciones obtenidas muestran una pérdida de calidad aceptable respecto a soluciones aproximadas.

Abstract

In multi-objective optimization the impact of external factors such as uncertainty on the performance of the built solutions is often not considered. However, this is very important when considering real-world problems, because the environment where the solution performs is little known or unchanging. For this reason, it is important to find solutions that maintain good performance despite different scenarios. In evolutionary computation, well-defined algorithms based on quality indicators have been proposed that are competitive for well-known optimization problems. In particular, the ant colony optimization (ACO) metaheuristic has been successfully used for vehicle routing problems. This is because the ants can build solutions using tours on a given graph. However, there are few algorithms in the literature that use quality indicators within ACO and find robust solutions for problems under uncertainty.

In this thesis, we propose a new cooperative multi-indicator based ant colony optimization algorithm (cMIBACO). This algorithm focuses on improving the collective work of artificial ants through the exchange of knowledge between different colonies. The algorithm is tested on the multi-objective generalized consistent vehicle routing problem under uncertainty (MOGenConVRP), where the uncertainty lies on the customers. The objectives to minimize are the total travel time, the number of different vehicles that serve a customer, and the difference between arrival times.

For the new algorithm, an analysis was done about the impact of the different components it has compared to a traditional algorithm. Its performance was also evaluated against single-indicator and weighted-sum ant colony algorithms, obtaining competitive results in approximate solutions. Finally, it was applied to obtain lightly robust solutions, where the solutions obtained show an acceptable loss of quality.

Agradecimientos

Comienzo agradeciendo a la Universidad Nacional Autónoma de México por la oportunidad de realizar la maestría en Ciencia e Ingeniería de la Computación. Gracias a este posgrado tengo una visión más amplia del campo en el que me quiero desempeñar y que mejoró significativamente mis conocimientos.

Quiero agradecer a todos los investigadores del Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas que logré conocer durante sus cursos y que sin duda fueron clases de gran nivel. En particular a Carlos Hernández que es tutor de esta tesis, gracias a su dirección y enseñanza se logró terminar este trabajo y un artículo de congreso. Muchas gracias por impulsarme a mejorar en este campo de conocimiento. A mis revisores de tesis, Katya, Gibran, Wendy y Jesús que dedicaron su tiempo para hacer revisiones que mejoraron este trabajo.

Gracias a mi familia que ha logrado apoyarme desde el comienzo de mis estudios y que ha sido fundamental para que pueda mejorar en todos los aspectos.

También agradezco al CONAHCYT por la beca otorgada durante toda la maestría y a la UNAM por el apoyo en el proyecto PAPIIT IA202923.

Índice general

Acrónimos por sus siglas en inglés

Se listan los acrónimos usados más importantes.

ACO	Optimización mediante colonia de hormigas. 33
cMIBACO	Algoritmo cooperativo de colonia de hormigas basada en múltiples indicadores. 67
EA	Algoritmo evolutivo. 20
GA	Algoritmo genético. 20
IB-MOEA	Algoritmo evolutivo multi-objetivo basado en indicadores. 31
IBACO	Colonia de hormigas basada en un indicador. 41
MOACO	Optimización mediante colonia de hormigas multi-objetivo. 36
MOEA	Algoritmo evolutivo multi-objetivo. 20
MOGenConVRP	Problema de enrutamiento de vehículos multi-objetivo generalizado consistente. 45
MOP	Problema de optimización multi-objetivo. 17
VRP	Problema de enrutamiento de vehículos. 42

Índice de figuras

Índice de tablas

Índice de algoritmos

Capítulo 1

Introducción

Los problemas de optimización se encuentran presentes en todas las áreas de conocimiento donde se busque mejorar algún costo, tales como en la ingeniería, economía, biología entre otras. Este tipo de problemas suelen tener más de un objetivo a optimizar y la mejora de uno puede afectar a algún otro, lo que lleva a tratar con problemas de optimización multi-objetivo (MOPs, por sus siglas en inglés[1]). En la optimización multi-objetivo, se presentan retos importantes tales como el manejo de incertidumbre, diseño de nuevas metaheurísticas para problemas con espacio de búsqueda complejos. Además de tener que encontrar una solución al problema, también está el cómo evaluar qué tan viable es la solución al problema o cómo escoger una solución de entre varias posibles. Puede ser que encontrar las soluciones óptimas al problema sea tan complicado que es preferible encontrar soluciones aproximadas o soluciones en las que se tenga alguna pérdida de calidad de acuerdo con los intereses del tomador de decisiones.

En la vida real durante la ejecución de algún algoritmo para el problema de optimización a tratar pueden surgir factores físicos, sociales entre otros que pueden afectar el desempeño de las soluciones. Por lo que es necesario tomar en cuenta la incertidumbre dentro de los algoritmos multi-objetivo, como tener soluciones que puedan ser robustas frente a cualquier escenario, de forma que su desempeño ante cualquier escenario o evento imprevisto no se vea reducido significativamente.

Dentro de los posibles algoritmos multi-objetivo a emplear, están los algoritmos bio-inspirados y los que pertenecen al cómputo evolutivo. Estos algoritmos se basan en principios biológicos de la naturaleza como la evolución y el comportamiento colectivo de animales. Los algoritmos basados en población pueden proveernos de alguna población de soluciones de entre la cual el tomador de decisiones escoja a las mejores soluciones de acuerdo con sus necesidades.

Particularmente los algoritmos bio-inspirados como la optimización por colonia de hormigas (ACO, por sus siglas en inglés[2]) que simulan el comportamiento de hormigas, donde el trabajo de una sola hormiga puede no ser tan relevante, pero todo su trabajo colectivo y el intercambio de información pueden aportar lo suficiente para construir alguna solución eficiente. La metaheurística ACO funciona bastantemente bien para problemas donde las soluciones se puedan construir a partir de trayectorias. Como pueden ser los problemas de enrutamiento de vehículos (VRP, por sus siglas en inglés[3]). Donde se tiene que visitar clientes que requieren algún servicio por parte de un

[1] Multi-objective Optimization Problems
[2] Ant Colony Optimization
[3] Vehicle Routing Problem

vehículo sujeto a ciertos restricciones y el objetivo más común a optimizar es la distancia de todos los recorridos. Como este tipo de problemas se puede modelar con una gráfica donde los nodos representan clientes y los arcos representan trayectos de un cliente a otro, se puede resolver con un algoritmo basado en colonia de hormigas.

1.1. Motivación y justificación

Muchos de los problemas de optimización combinatoria actuales como el problema de enrutamiento de vehículos siguen siendo de mucho interés debido a objetivos tales como emisiones de carbono [7], satisfacción del cliente [95], reducción en la diferencia de los tiempos de entrega [65], entre otros. Así como manejar la incertidumbre ante escenarios probables o poco probables como cambios en los tiempos de las rutas de transito [121] o cambios en la demanda de los clientes [11]. Muchas de las variantes de VRP existentes han sido tratados con bastantes algoritmos de colonia de hormigas multi-objetivo, sin embargo, son pocos los que se emplean indicadores de calidad. En cuanto a la incertidumbre en problemas de enrutamiento, se tienen bastantes algoritmos que tratan la incertidumbre con números difusos o bien, distribuciones de probabilidad, muchos agregando la incertidumbre como pesos en la función objetivo o restricciones. Aunque existen trabajos que usan escenarios discretos para modelar incertidumbre, aún falta explorar la robustez ligera mediante soluciones aproximadas en este tipo de problemas.

En la literatura existente se tienen muchos algoritmos multi-objetivo basados en colonia de hormigas, tales como basados en descomposición [24], basados en múltiples colonias y matrices de feromonas [77], entre otros. Sin embargo, para el caso de indicadores de calidad, en este tipo de algoritmos para problemas de optimización combinatoria existen pocos desde que se propuso el primer algoritmo basado en indicadores de calidad para colonia de hormigas [83]. Por lo que falta investigación en cuanto a las ventajas de usar múltiples indicadores de calidad en algoritmos multi-objetivo basados en colonia de hormigas para problemas combinatorios. Además de que existen pocos algoritmos bio-inspirados que tratan de buscar soluciones aproximadas [119].

1.2. Problema y propuesta de solución

Después de realizar la optimización multi-objetivo y obtener un conjunto de soluciones, pueden suceder escenarios no contemplados que afecten el costo real de las soluciones, por lo que una alternativa es realizar una optimización robusta. Donde se buscan las soluciones tales que en la mayoría de los escenarios tengan un buen desempeño. Esto se puede conseguir, por ejemplo optimizando los peores escenarios [34] o a partir de un escenario nominal [2]. En particular, las soluciones ligeramente robustas, que a partir de un conjunto de soluciones aproximadas considera sus peores casos y toma aquellas soluciones tal que no existe una alguna que sea mejor en todos los escenarios.

Por otra parte, este trabajo se enfoca en el problema de enrutamiento de vehículos generalizado consistente multi-objetivo (MOGenConVRP, por sus siglas en inglés[4]) [66] que tiene tres objetivos; minimizar el tiempo de las rutas, minimizar la cantidad de distintos vehículos que visitan a cada cliente y minimizar la diferencia entre los tiempos de llegada a los clientes. Este problema aún no ha sido tratado para el caso con incertidumbre en los clientes ni con algún algoritmo basado en colonia de hormigas.

[4]Multi-objective Generalized Consistent Vehicle Routing Problem

En este proyecto se propone un nuevo algoritmo de optimización por colonia de hormigas basado en la cooperación de múltiples indicadores de calidad que tiene como objetivo mejorar el intercambio de información entre colonias de hormigas y la exploración del frente de soluciones óptimas. Se presenta también un análisis de los componentes de este nuevo algoritmo y su desempeño en soluciones aproximadas y ligeramente robustas. Además de aplicar este algoritmo en el MOGenConVRP bajo incertidumbre, donde la incertidumbre está dada por los clientes y se buscan soluciones ligeramente robustas a través de archivos externos [105].

1.3. Objetivos

En esta sección se presentan los objetivos de esta tesis.

1.3.1. Objetivo general

En este trabajo se propone combinar múltiples indicadores de calidad para mejorar la exploración y diversidad de las soluciones creadas por las hormigas artificiales. Mediante distintas colonias de hormigas que de forma cooperativa pueden obtener soluciones aproximadas para un problema particular de enrutamiento de vehículos multi-objetivo. Este conjunto de soluciones aproximadas se utiliza como escenario nominal para tratar la incertidumbre en este problema y obtener soluciones ligeramente robustas que tienen una pérdida de calidad aceptable.

1.3.2. Objetivos particulares

- Diseñar un algoritmo para colonia de hormigas que use indicadores de calidad de forma cooperativa. Donde las hormigas dadas las reglas de probabilidad, puedan construir soluciones para el problema de enrutamiento de vehículos multi-objetivo.
- Analizar el efecto de usar operadores de cruza, mutación y búsqueda local en el nuevo algoritmo, esto sirve para comprobar si existe alguna mejora a solo usar colonia de hormigas en su forma base.
- Optimizar los parámetros adecuados para cada algoritmo que se utilice en los experimentos y evaluarlos con las métricas adecuadas. Estos parámetros son los que se utilizan en el cálculo de probabilidades de las hormigas cada vez que construyen caminos.
- Realizar un estudio comparativo de los algoritmos basados en indicadores de colonia de hormigas con la nueva propuesta tanto para soluciones aproximadas como soluciones ligeramente robustas con el fin de conocer las ventajas de múltiples indicadores en colonia de hormigas.

1.4. Hipótesis

Se quiere conocer si el comportamiento cooperativo de múltiples colonias de hormigas donde cada colonia usa un indicador de calidad para evaluar soluciones y actualizar la matriz de feromonas puede tener un mejor desempeño que usar los indicadores de forma individual. Además de probar un algoritmo de colonia de hormigas basado en múltiples indicadores para obtener soluciones aproximadas en problemas de optimización combinatoria. Este nuevo algoritmo se puede extender para

obtener soluciones ligeramente robustas en problemas de enrutamiento de vehículos multi-objetivo bajo incertidumbre. Finalmente, probar si las soluciones robustas tienen una pérdida de calidad aceptable respecto a soluciones óptimas de acuerdo con los intereses del tomador de decisiones.

1.5. Preguntas de investigación

Las preguntas de investigación de este trabajo son:

- ¿Existe alguna mejora al combinar diversos indicadores de desempeño en colonia de hormigas?
- ¿Cuál es el efecto de agregar componentes de cruza, mutación y búsqueda local en la nueva propuesta?
- ¿Existe alguna diferencia en usar los indicadores de forma individual, suma ponderada y cooperativa en la colonia de hormigas?
- ¿Cómo se desempeña el algoritmo para soluciones ligeramente robustas en MOGenConVRP bajo incertidumbre?
- ¿Cúal es la pérdida de calidad de las nuevas soluciones robustas frente a soluciones aproximadas?

1.6. Contribuciones

Se listan las contribuciones de esta tesis.

- Un nuevo algoritmo multi-objetivo basado en colonia de hormigas con múltiples indicadores de calidad para obtener soluciones aproximadas y ligeramente robustas.
- Aplicación del algoritmo para problemas de enrutamiento de vehículos multi-objetivo bajo incertidumbre.
- Comparación del nuevo algoritmo frente a los algoritmos basados en un indicador y suma ponderada en colonia de hormigas.

Se listan los productos de esta tesis:

- Un póster aceptado en el congreso *The Genetic and Evolutionary Computation Conference (GECCO 2024)* [118].
- Un póster aceptado en el congreso *Numerical Evolutionary Optimization (NEO 2024)*.
- Un artículo para revista en proceso.
- Paquete de software científico.

1.7. Estructura de tesis

El documento está organizado como sigue. En el Capítulo 2, se presentan los antecedentes y trabajo relacionado sobre optimización multi-objetivo, algoritmos de optimización por colonia de hormigas y problemas de enrutamiento de vehículos multi-objetivo. En el Capítulo 3, se describe la propuesta del nuevo algoritmo de colonia de hormigas con múltiples indicadores y su aplicación a MOGenConVRP bajo incertidumbre, primero para soluciones aproximadas en escenario nominal y después su extensión para robustez ligera. En el Capítulo 4, se presentan los resultados de distintas variantes del nuevo algoritmo para analizar el efecto de cada uno de sus componentes, además de la comparación entre usar los indicadores de forma individual, suma ponderada y cooperativa dentro de la optimización robusta. Finalmente, en el Capítulo 5 las conclusiones y el trabajo futuro. En el apéndice A se muestra una ejecución paso a paso del cálculo de probabilidades de las hormigas para una instancia del problema. En el apéndice B se muestran todos los resultados numéricos obtenidos en todos los experimentos.

Capítulo 2

Antecedentes y trabajo relacionado

En este capítulo se presentan los antecedentes y el trabajo relacionado necesarios de esta investigación. En la Sección 2.1, se introducen los conceptos básicos relacionados a optimización multi-objetivo. En la Sección 2.2, se describe la optimización con colonia de hormigas multi-objetivo. En la Sección 2.3, lo referente a problemas de enrutamiento de vehículos multi-objetivo.

2.1. Optimización multi-objetivo

En los problemas del mundo real es de interés optimizar más de un objetivo, sin embargo, es muy común que las funciones objetivo entren en conflicto. Por lo que diseñar algoritmos que puedan encontrar soluciones que sean aceptables para el tomador de decisiones es un aspecto importante. En esta sección se presenta lo referente a conceptos básicos de optimización multi-objetivo, algunos algoritmos representativos, optimización robusta y archivos externos.

2.1.1. Conceptos básicos

Un espacio de búsqueda o decisión Q contiene las soluciones al problema que pueden ser combinaciones de valores discretos tales como permutaciones en caso de que el problema a resolver sea discreto. Por otro lado, si el problema es continuo, el conjunto $Q \in \mathbb{R}^n$ contiene vectores de dimensión n. Un problema de optimización multi-objetivo (MOP), donde se tienen k funciones objetivo a minimizar o maximizar y éstas se encuentran en conflicto se define a continuación.

Definición 2.1.1 (Problema de optimización multi-objetivo [25])
Un problema de **optimización multi-objetivo (MOP)** donde se busca minimizar los objetivos, sin pérdida de generalidad se puede definir formalmente como:

$$\min_{x \in Q} F(x), \tag{2.1}$$

sujeto a:

$$g_i(x) \leq 0,\ i = 1, .., p, \tag{2.2}$$

$$h_j(x) = 0,\ j = 1, .., q, \tag{2.3}$$

donde $g_i : Q \to \mathbb{R}$ y $h_i : Q \to \mathbb{R}$ son restricciones que debe cumplir una solución $x \in Q$ para ser factible, el vector de funciones objetivo $F : Q \to \mathbb{R}^k$ con los objetivos $f_i : Q \to \mathbb{R}$ para toda $i = 1, .., k$.

El hecho de tener múltiples objetivos en conflicto en un MOP hace necesario definir una forma de comparar cualquier par de soluciones, para esto se utiliza la dominancia de Pareto [97].

Definición 2.1.2 (Dominancia de Pareto [97])
Dado un MOP con espacio de decisión Q y k objetivos, se define:
(a) Sean $v, w \in \mathbb{R}^k$ dos vectores en espacio de los objetivos. El vector v es menor que w ($v <_p w$), si $v_i < w_i,\ \forall i \in \{1, .., k\}$.
(b) El vector v es menor o igual que w ($v \leq_p w$), si $v_i \leq w_i,\ \forall i \in \{1, .., k\}$.
(c) Una solución $y \in Q$ es dominada por una solución $x \in Q$ ($x \prec y$) si se cumple que:

$$F(x) \leq_p F(y) \text{ con } F(x) \neq F(y),$$

en otro caso, y es *no dominada* por x.

Definición 2.1.3 (Dominancia débil [97]) Una solución $y \in Q$ es débilmente dominada por una solución $x \in Q$ ($x \preceq y$) si se cumple que $F(x) \leq_p F(y)$.

Definición 2.1.4 (Solución de Pareto [97]) Una solución $x \in Q$ se llama solución de Pareto de un MOP si no existe alguna $y \in Q$ que domine a x.

Definición 2.1.5 (Solución débil de Pareto [97]) Una solución $x \in Q$ es una solución débil de Pareto si no existe una $y \in Q$ tal que $F(y) <_p F(x)$.

Definición 2.1.6 (Conjunto de Pareto [97]) El conjunto con todas las soluciones óptimas de Pareto se define como:

$$P_Q = \{x \in Q : \nexists y \in Q : y \prec x\}. \tag{2.4}$$

Definición 2.1.7 (Frente de Pareto [97]) La imagen $F(P_Q)$ de P_Q es llamado el frente de Pareto.

2.1.2. Algoritmos básicos de optimización multi-objetivo

Para resolver un MOP existen algoritmos y categorías muy variables como algoritmos basados en búsqueda local o población [113], así como algoritmos basados en trayectorias o descontinuación [15]. Este trabajo se enfoca en la clasificación propuesta por Coello et al. [25]. Esta clasificación se divide principalmente en dos clases, algoritmos *deterministas* y *estocásticos*, que a su vez comprenden más clases de algoritmos como lo son:

- **Deterministas**:
  - Enumerativo.
  - Algoritmos glotones.

- Ascenso de colinas.
- Algoritmos de ramificación y acotamiento.
- Algoritmos sobre gráficas.
- Algoritmos primero el mejor.
- Algoritmos basados en cálculo.

- **Estocásticos**:
  - Búsqueda aleatoria.
  - Recocido simulado.
  - Monte Carlo.
  - Búsqueda Tabú.
  - Algoritmos bio-inspirados.

Algoritmos deterministas

Comenzando con los algoritmos enumerativos, son los más sencillos de analizar porque se enfocan en revisar todas las posibles soluciones en el espacio de decisiones. Para problemas con espacios de soluciones con un mayor tamaño son ineficientes, por lo que no son de interés en este trabajo.

Empezando con los algoritmos glotones [18], éstos van construyendo la solución tomando las decisiones que al momento obtengan un mejor costo, con la desventaja de obtener una solución que sea un óptimo local. De forma similar, un algoritmo clásico para búsqueda de soluciones es el ascenso de colinas [103], éste siempre va tomando soluciones con dirección ascendente, sin embargo, es común que se quede atrapada en un óptimo local tempranamente.

Los algoritmos de ramificación y acotamiento [72] tratan de encontrar soluciones de acuerdo con la búsqueda en un árbol. Realizan un acotamiento de las soluciones a explorar, evalúan las posibles soluciones y expanden aquellas que son más promisorias. La forma en que generan y evalúan soluciones dependen de las heurísticas para el problema.

Cuando es viable representar el espacio de decisiones en una gráfica, se puede aplicar los métodos ya muy conocidos para iterar sobre éstas como la búsqueda en anchura (BFS, por sus siglas en inglés[1]) o búsqueda en profundidad (DFS, por sus siglas en inglés[2]). En general, este tipo de algoritmos sobre gráficas tienden a tener complejidad polinomial sujeto al tamaño de la gráfica.

Para los métodos de primero el mejor, uno de sus mayores representantes es A^* [55] que considera heurísticas para evaluar la cercanía desde el estado actual al inicio y la cercanía desde el actual al estado meta. Bajo ciertas condiciones A^* puede encontrar el óptimo, pero el costo en memoria perjudica bastante al algoritmo, ya que tiene que guardar y actualizar los estados que va encontrando durante la búsqueda. Los algoritmos basados en cálculos [92] requieren que el dominio de las soluciones sean continuas y aplican métodos basados en programación lineal, cálculo de derivadas entre otros.

[1] Breadth First Search
[2] Depth First Search

Algoritmos estocásticos

A pesar de que los algoritmos deterministas pueden tener un buen desempeño en ciertos problemas no tan complejos como problemas donde su complejidad es polinomial, cuando se trata de resolver problemas NP-Completos, estos algoritmos son muy poco útiles. Debido a que el espacio de búsqueda es altamente complejo o no se tiene suficiente información del problema, por lo que una mejor opción es usar un algoritmo estocástico. Los algoritmos estocásticos buscan sobre el espacio de soluciones a través de decisiones aleatorias, sin embargo, no se garantiza encontrar una solución óptima.

La búsqueda aleatoria es el algoritmo más simple que va tomando soluciones dada una probabilidad y que termina divergiendo o quedando estancado en óptimos locales, no se recomienda para los MOP. El recocido simulado [21] es una mejora en la búsqueda por ascenso de colinas, ya que trata de salir de óptimos locales aceptando peores soluciones. Usa como parámetro a la temperatura y de acuerdo con ésta puede aceptar una mala solución. Cuando la temperatura es alta se puede aceptar cualquier solución con alta probabilidad y a medida que desciende, acepta peores soluciones con menor probabilidad. La búsqueda tabú [51] usa un mecanismo de memoria para evitar repetir caminos, asociando a soluciones visitadas como tabú mientras realiza una heurística. La dificultad de este algoritmo está en cómo manejar la memoria de soluciones prohibidas ante espacios de búsqueda de tamaño exponencial.

Dentro de los algoritmos estocásticos se encuentran los algoritmos basados en trayectoria, que son aquellos algoritmos donde se va tomando solo una solución en cada tiempo. En este tipo de algoritmos el más sobresaliente es la búsqueda local y todas sus variantes. Iniciando con la búsqueda de vecindario variable (VNS, por sus siglas en inglés[3]) [86] se compone de tres pasos importantes. En el primero, seleccionar un operador de entre un conjunto de operadores que mapean una solución a un vecindario. Segundo, con la nueva solución después de aplicar el operador aplicar mejoras con algún método de búsqueda local hasta llegar a un óptimo local. Finalmente, bajo algún criterio decidir si se toma la nueva solución y su vecindario para la siguiente iteración.

La búsqueda local iterativa (ILS, por sus siglas en inglés[4]) [50] tiene como objetivo aplicar búsqueda local en distintos vecindarios de soluciones. Dada una solución inicial, de forma iterativa aplica una perturbación a la solución actual y luego aplica búsqueda local. Después se compara la nueva solución obtenida con la actual y se actualiza la solución actual en caso de que esta sea mejor. Esto se repite hasta satisfacer un criterio de término. También se tiene la búsqueda larga en vecindarios (LNS, por sus siglas en inglés[5]) [99] que mediante operadores de construcción y destrucción trata de modificar lo suficiente la solución como para atravesar bastantes vecindarios y escapar de un óptimo local.

2.1.3. Algoritmos evolutivos multi-objetivo

El cómputo evolutivo [46] engloba a los algoritmos que se basan en el proceso natural de la evolución, iniciando con los algoritmos genéticos (GA, por sus siglas en inglés[6]) [59] y programación genética (GP, por sus siglas en inglés[7]) [67]. También se tienen las estrategias evolutivas (ES,

[3]Variable Neighborhood Search
[4]Iterated Local Search
[5]Large Neighborhood Search
[6]Genetic Algorithms
[7]Genetic Programming

por sus siglas en inglés[8]) [101], todos estos conocidos como algoritmos evolutivos (EAs, por sus siglas en inglés[9]) [47]. Para tratar problemas multi-objetivo, los algoritmos evolutivos multi-objetivo (MOEAs, por sus siglas en inglés[10]) [25] son de los más populares [27], debido a que no necesitan extensa información del problema, son relativamente fáciles de implementar y son flexibles porque se les pueden modificar continuamente algunos mecanismos como archivos externos. Estos algoritmos se basan principalmente en poblaciones de soluciones. Los algoritmos modernos se pueden clasificar en tres: **basados en dominancia, descomposición y basados en indicadores**.

MOEAs basados en dominancia

Los MOEAs basados en dominancia se enfocan en dividir las soluciones en subconjuntos de acuerdo con el rango de dominancia que tienen, dando mayor prioridad a soluciones que dominan a más soluciones. Un algoritmo clásico que se basa en dominancia es el algoritmo genético multi-objetivo (MOGA, por sus siglas en inglés[11]) [90], que para cada solución calcula su rango de acuerdo con la dominancia de Pareto. Este rango se utiliza para calcular la aptitud de cada solución mediante alguna función lineal. Después se promedia la aptitud de las soluciones con el mismo rango, esto con el fin de que soluciones similares tengan las mismas posibilidades de ser escogidas. Con la aptitud calculada, se realizan los mecanismos de selección, cruza y mutación específicos del problema.

Un algoritmo que usa un archivo externo como memoria es (PAES, por sus siglas en inglés[12])[64], que en cada iteración para cada solución padre genera una solución hija que le aplica mutación, calcula su aptitud y luego evalúa cuál de las dos soluciones se agrega al archivo. Además de que toma en cuenta la diversidad que aporta la solución candidata al agregarla en el archivo externo.

Otro algoritmo representativo es (NSGA-II, por sus siglas en inglés[13]) [29], que se enfoca en ordenar las soluciones mediante el rango de dominancia que tienen para luego seleccionar como soluciones de la siguiente población a los conjuntos con mejores rangos de mayor a menor. Las soluciones restantes por añadir a la siguiente población se seleccionan considerando para cada solución qué tan distribuida se encuentra.

Otro algoritmo representativo que usa un archivo externo es SPEA [5], que mantiene un archivo externo y en cada iteración guarda en él las soluciones no dominadas de la población actual. Este algoritmo calcula la aptitud de cada solución mediante el rango de dominancia de cada solución. Para luego aplicar operadores de cruza y mutación con lo que obtiene la siguiente población.

MOEAs basados en descomposición

Los algoritmos basados en descomposición se enfocan en descomponer el problema multi-objetivo en varios subproblemas mono-objetivo para luego tomar aquellas soluciones que mejor puedan guiar la búsqueda. El algoritmo más representativo en esta clase es el algoritmo multi-objetivo evolutivo basado en descomposición (MOEA/D, por sus siglas en inglés[14]) [123], que usa vectores de pesos

[8] Evolution Strategy
[9] Evolutionary Algorithms
[10] Evolutionary Multi-objective Optimization Algorithms
[11] Multi-objective Genetic Algorithm
[12] Pareto Archived Evolution Strategy
[13] Non-dominated Sorted Genetic Algorithm II
[14] Multi-objective evolutionary algorithm based on decomposition

y una función de utilidad, con los cuales transforma el problema multi-objetivo a un solo objetivo usando la suma ponderada con la métrica de Tchebycheff [49]. De esta forma genera soluciones por cada vector y a su vez asigna un conjunto de vecinos para cada vector. Posteriormente genera nuevas soluciones con operadores genéticos que podrán sustituir una solución vecina solo si mejoran respecto a un vector de pesos.

Otro algoritmo relevante es (NSGA-III, por sus siglas en inglés[15]) [28], que mejora a NSGA-II usando vectores de dirección. Donde las soluciones son seleccionadas de acuerdo con el rango de dominancia como en la versión anterior pero ahora la diversidad de las soluciones depende de la cercanía a los vectores de dirección.

MOEAs basados en indicadores

Para los MOEAs basados en indicadores, estos suelen usar algún indicador para evaluar la aptitud de las soluciones o guiar la búsqueda en espacio de soluciones. Uno de los más representativos es SMS-EMOA [14], este algoritmo calcula el rango de dominancia de las soluciones en la población al igual que NSGA-II. De las soluciones con el rango de dominancia más bajo se eliminan aquellas que tiene menor contribución de acuerdo con el indicador de hipervolumen.

En IBEA [124], se emplean los indicadores binarios de hipervolumen y ϵ^+ comparando uno contra todos a cada solución. Removiendo al individuo menos apto hasta que se alcance un tamaño de población. Una variante representativa de IBEA es R2-IBEA [98] que escoge las soluciones con el indicador R2, usando la métrica de Tchebycheff con pesos uniformemente distribuidos.

2.1.4. Indicadores de calidad

Durante la evaluación de desempeño de los MOEAs, no basta con solo realizar observaciones cualitativas de las aproximaciones al frente de Pareto [39]. Cuando el número de objetivos crece, es difícil evaluar la calidad de estas soluciones con solo la visualización de las aproximaciones al frente de Pareto. Por lo que se tienen distintas funciones que pueden realizar una evaluación de estos conjuntos de soluciones, conocidas como indicadores de calidad. Los indicadores de calidad, dado un conjunto de soluciones como entrada, evalúan qué tan buenas son las soluciones. Dependiendo del indicador de calidad, éstos pueden evaluar qué tan cercanas son las soluciones respecto a un punto ideal o un conjunto de puntos, o bien, pueden evaluar la distribución de un conjunto de puntos. Así mismo, pueden mejorar la búsqueda de un algoritmo multi-objetivo como tan bien transformar un problema multi-objetivo a mono-objetivo usando los indicadores como funciones objetivo. A continuación las definiciones sobre indicador de calidad, Pareto compatibilidad y las relacionadas a los indicadores en su forma binaria.

Definición 2.1.8 (Indicador de calidad [126]) Un **indicador de calidad** $I : \Omega \to \mathbb{R}$ es una función que mapea un conjunto de soluciones $\Omega \subset Q$ a un número real.

Definición 2.1.9 (Pareto compatibilidad [126]) Dados dos conjuntos de aproximación A, B, un indicador unario I es Pareto compatible si $A \lhd B \Rightarrow I(A) > I(B)$.

Definición 2.1.10 (Pareto compatibilidad débil [126]) Dados dos conjuntos de aproximación A, B, un indicador unario I es débil Pareto compatible si $A \lhd B \Rightarrow I(A) \geq I(B)$.

[15] Non-dominated Sorted Genetic Algorithm III

Los indicadores binarios de calidad son una función $I : \Omega \times \Omega \to \mathbb{R}$ que mapea un par de conjuntos de soluciones a un número real y preservan una relación de dominancia como se enuncia en el Teorema 2.1.1.

Teorema 2.1.1 (Preservación de dominancia [124]) Un indicador binario de calidad preserva la dominancia si se cumple:

- $x_2 \prec x_1 \Rightarrow I(x_1, x_2) < I(x_2, x_1)$,
- $x_2 \prec x_1 \Rightarrow I(x_3, x_1) \geq I(x_3, x_2)\ \forall x_1, x_2, x_3 \in Q$.

Con los indicadores de calidad binarios se puede usar la función de aptitud en la Ecuación (2.5), que compara una solución contra la mayoría de la población y que se busca maximizar. Esta función asigna mayores valores a las soluciones dominantes sobre las dominadas. Y su buen funcionamiento depende de una constante $k > 0$ y además, esta función puede consumir bastante tiempo para poblaciones grandes.

$$Fit(x_1) = \sum_{x_2 \in P \setminus \{x_1\}} -e^{-I(x_2,x_1)/k}. \tag{2.5}$$

Indicador de hipervolumen

El indicador de hipervolumen [125] calcula el hiperespacio que domina un conjunto de soluciones A en espacio de los objetivos dado algún punto de referencia r^* y una medida de Lebesgue $\mathcal{L}(\cdot)$. La elección del punto de referencia también es un problema a resolver, algunas propuestas básicas son; tomar el vector ideal con los mejores valores de los objetivos, tomar el vector con los peores valores objetivos o los peores valores del frente de soluciones óptimas. En la Figura 2.1 se muestra un ejemplo gráfico de este indicador. Maximizando el indicador de hipervolumen se encuentra el frente de Pareto y se define formalmente como:

$$HV(\mathcal{A}, r^*) = \mathcal{L}(\cup_{a \in \mathcal{A}} \{b | a \prec b \prec r^*\}). \tag{2.6}$$

Su forma binaria se define como:

$$I_{HV}(x, y) = \begin{cases} I_H(y) - I_H(x) & y \prec x \\ I_H(x + y) - I_H(x) & y \not\prec x \end{cases}. \tag{2.7}$$

Indicador R2

La familia de indicadores R [53] evalúan la proximidad de soluciones al frente de Pareto mediante funciones de utilidad que mapean un vector a un número real. En la Figura 2.2 se muestra un ejemplo gráfico de este indicador. Dado un conjunto discreto U con distribución uniforme de probabilidad p sobre U, el indicador $R2$ [54] se define como:

$$R2(R, A, U) = \frac{1}{|U|} \sum_{u \in U} (\max_{r \in R}\{u(r)\} - \max_{a \in A}\{u(a)\}). \tag{2.8}$$

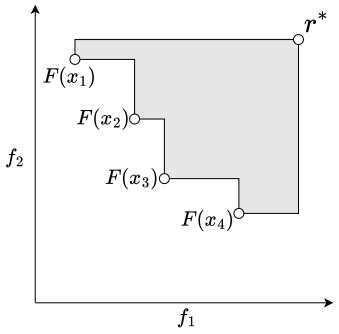


Figura 2.1: Hipervolumen (gris) de un conjunto de soluciones dado un punto de referencia r^*.

Usando como función de utilidad a la métrica ponderada de Tchebycheff con un conjunto de pesos W y un punto de referencia z^* se define formalmente como:

$$R2(A : W, z^*) = \frac{1}{|W|} \sum_{\vec{w} \in W} \min_{a \in A} \{ \max_{i \in \{1,..,k\}} \vec{w}_i |a_i - z_i^*| \}. \tag{2.9}$$

Su forma binaria se define como:

$$I_{R2}(x, y) = R2(\{x\} : W, z^*) - R2(\{x \cup y\} : W, z^*). \tag{2.10}$$

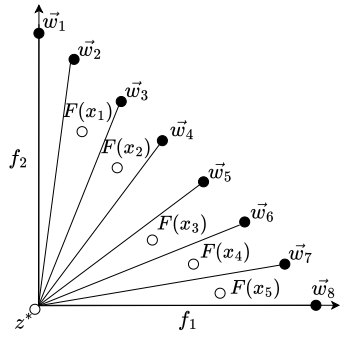


Figura 2.2: Vectores $W = \{\vec{w_1}, \vec{w_2}, .., \vec{w_8}\}$ y punto de referencia z^* de acuerdo con el indicador de R2.

Indicador ϵ^+

El indicador binario ϵ^+ [126] es la distancia mínima necesaria que se debe trasladar un conjunto de puntos para dominar débilmente a otro conjunto. En la Figura 2.3 se muestra un ejemplo gráfico

de este indicador. Se define como:

$$I_{\epsilon^+}(A,B) = \min_{\varepsilon\in\mathbb{R}}\{\forall x_2 \in B\ \exists x_1 \in A : f_i(x_1) - \varepsilon \leq f_i(x_2)\ \forall i \in \{1,2,..,k\}\}\cdot \tag{2.11}$$

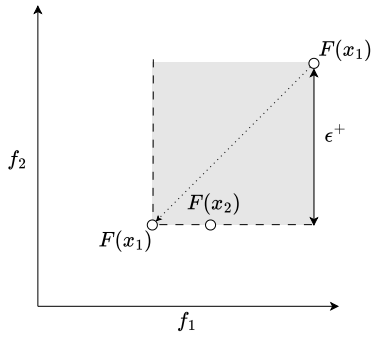


Figura 2.3: Solución $F(x_1)$ trasladada ϵ unidades para dominar débilmente a $F(x_2)$.

Indicador Riesz s-energy

El indicador de Riesz s-energy (E_s) [43] mide la diversidad en un conjunto de soluciones sumando el inverso de las distancias con $s > 0$ como parámetro que controla la uniformidad.

$$E_s(\mathcal{A}) = \sum_{i\neq j} ||a_i - a_j||^{-s}. \tag{2.12}$$

2.1.5. Optimización multi-objetivo robusta

Dentro de la optimización multi-objetivo pueden existir múltiples escenarios que afecten de forma positiva o negativa la evaluación de las funciones objetivo. Para esto se buscan soluciones que puedan tener un desempeño relativamente bueno a pesar de cualquier escenario, esto significa que su costo no sea tan reducido de acuerdo con alguna tolerancia. En este caso se usa la optimización robusta [34], que se aplica a problemas donde las soluciones tienen que afrontar diversos escenarios.

Tomando como ejemplo los problemas de enrutamiento de vehículos donde la incertidumbre se puede encontrar en las rutas como se muestra en la Figura 2.4, los objetivos son minimizar el tiempo y el costo de tomar cualquiera de las rutas.

Se tienen tres posibles rutas x_1, x_2, x_3 y dos escenarios ξ_1, ξ_2. El escenario ξ_1 se refiere a cuando sucede un fenómeno meteorológico (como una tormenta) en medio del recorrido del vehículo y el escenario ξ_2 cuando éste no sucede. La ruta x_1 es un camino visiblemente más largo y con más curvas que los otros dos, teniendo que cuando sucede ξ_1 tanto el costo y tiempo aumentan 1.9 y 2.1 veces respectivamente. Por otro lado, la ruta x_2 parece más corta y lo más cercano a una línea recta. Sin embargo, su costo en ambos escenarios es mayor que los costos de la ruta x_1. La ruta x_3 tiene costos menores en ambos escenarios respecto a las rutas anteriores, pero los tiempos por

escenario son mayores. De esta forma nos interesa conocer cual de las rutas es lo suficientemente buena a pesar de lo que ocurra cuando sucede alguno de los dos escenarios.

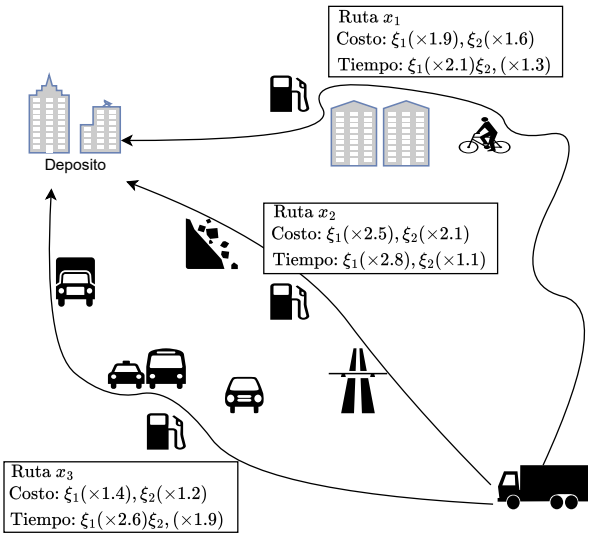


Figura 2.4: Ejemplo del efecto de dos escenarios en un problema con tres rutas posibles x_1, x_2, x_3.

Definición 2.1.11 (Problema de optimización bajo incertidumbre multi-objetivo [34]) Un problema de optimización multi-objetivo bajo incertidumbre $\mathcal{P}(\mathcal{U}) = \{\mathcal{P}(\xi), \xi \in \mathcal{U}\}$ se define como una familia de problemas parametrizados:

$$\mathcal{P}(\xi) = \min_{x \in Q} F(x, \xi), \tag{2.13}$$

donde $F(x, \xi)$ es la evaluación de x en el escenario ξ.

De la definición anterior, no es claro cuál de todos los problemas de optimización resolver, ni que solución satisface todos los problemas. Ehrgott et al. [34] extienden diferentes formas de definir a una solución como robusta para un problema de optimización bajo incertidumbre. Tales conceptos como considerar una solución robusta como aquella solución tal que en al menos un escenario no es dominada o de forma contraria, como aquella solución que en todos los escenarios no es dominada. En este trabajo se utiliza la definición de robustez Min-Max, que se enfoca en minimizar el peor escenario para las soluciones.

Definición 2.1.12 (Robustez Min-Max para problemas mono-objetivo [34]) Dado un problema de optimización bajo incertidumbre $\mathcal{P}(\mathcal{U})$ mono-objetivo. Una solución $x \in Q$ es óptima robusta Min-Max para $\mathcal{P}(\mathcal{U})$ si es una solución óptima para:

$$\min_{x \in Q} \sup_{\xi \in \mathcal{U}} f(x, \xi). \tag{2.14}$$

En la anterior definición, el problema se describe para un solo objetivo y para trasladar esto a multi-objetivo es necesario evaluar cada solución en sus escenarios. Para cada solución x y un conjunto de escenarios $\mathcal{U}$, el conjunto de valores objetivo es:

$$F_{\mathcal{U}}(x) = \{F(x, \xi) : \xi \in \mathcal{U}\}. \tag{2.15}$$

Con esto se puede definir la eficiencia robusta Min-Max basada en conjuntos.

Definición 2.1.13 (Robustez Min-Max basada en conjuntos [34]) Dado un problema de optimización multi-objetivo bajo incertidumbre $\mathcal{P}(\mathcal{U})$ y el conjunto $\mathbb{R}^k_{\preceq} = \{z \in \mathbb{R}^k : 0 \preceq z\}$. Una solución factible $x \in Q$ es definida como eficiente Min-Max robusta basada en conjuntos si no existe otra solución $x' \in Q \backslash \{x\}$ tal que:

$$F_{\mathcal{U}}(x') \subseteq F_{\mathcal{U}}(x) - \mathbb{R}^k_{\preceq}. \tag{2.16}$$

En la Figura 2.5 se muestra un ejemplo de algunas soluciones evaluadas en tres escenarios ξ_1, ξ_2, ξ_3 en espacio de los objetivos. Donde solo x_1 y x_3 son soluciones robustas Min-Max basadas en conjuntos, pues el conjunto $F_{\mathcal{U}}(x_3)$ está contenido en $F_{\mathcal{U}}(x_2) - \mathbb{R}^k_{\preceq}$. Además de que no existe alguna solución tal que su conjunto de puntos esté contenida en $F_{\mathcal{U}}(x_3) - \mathbb{R}^k_{\preceq}$. Para x_1, el conjunto $F_{\mathcal{U}}(x_3)$ no está del todo contenido en $F_{\mathcal{U}}(x_1) - \mathbb{R}^k_{\preceq}$ debido a el punto $F(x_3, \xi_1)$. Siendo así que x_2 es la única solución que no es robusta Min-Max.

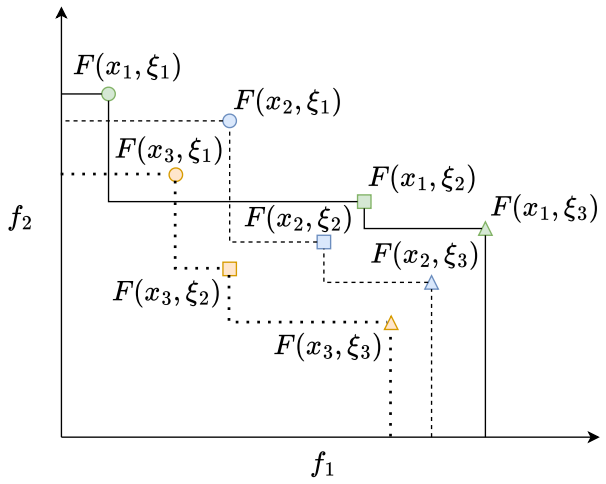


Figura 2.5: Ejemplo de robustez Min-Max basada en conjuntos. Solo x_1 y x_3 cumplen con este tipo de robustez ya que $F_{\mathcal{U}}(x_3) \subset F_{\mathcal{U}}(x_2) - \mathbb{R}^k_{\preceq}$.

Soluciones aproximadas

En ocasiones es factible que dentro del problema de optimización multi-objetivo se permita alguna pérdida de calidad en las soluciones o una tolerancia sobre el tipo de soluciones que se

aceptan, esto tiene que ver con la cercanía entre soluciones. Como lo es con alguna constante epsilon dada para obtener un conjunto de soluciones aproximadas.

Definición 2.1.14 (ϵ-dominancia [119]) Sea $\epsilon = (\epsilon_1, .., \epsilon_k)^\intercal \in \mathbb{R}^k_+$ con $x, y \in Q$. Se dice que x ϵ-domina a y, denotado como $(x \prec_\epsilon y)$ si:

$$F(x) - \epsilon \ \leq \ F(y) \text{ y } F(x) - \epsilon \neq F(y). \tag{2.17}$$

Por otro lado, una definición similar es la de $-\epsilon$-dominancia, definida como sigue:

Definición 2.1.15 ($-\epsilon$-dominancia [119]) Sea $\epsilon = (\epsilon_1, .., \epsilon_k)^\intercal \in \mathbb{R}^k_+$ con $x, y \in Q$. Se dice que x $-\epsilon$-domina a y, denotado como $(x \prec_{-\epsilon} y)$ si:

$$F(x) + \epsilon \ \leq \ F(y) \text{ y } F(x) + \epsilon \neq F(y). \tag{2.18}$$

La razón para definir ambas aproximaciones es para obtener conjuntos de soluciones aproximadas, que son aquellas soluciones que pueden estar cercanas entre si en espacio objetivo por una tolerancia epsilon. Además de que este conjunto contiene soluciones que pueden estar cercanas al frente de soluciones óptimas y que tienen una pérdida de calidad aceptable.

Definición 2.1.16 (Soluciones aproximadas $P_{Q,\epsilon}$ [119]) Sea $P_{Q,\epsilon} \subset Q$ el conjunto de soluciones que no son $-\epsilon$-dominadas por alguna otra solución en Q, definido como:

$$P_{Q,\epsilon} := \{x \in Q | \nexists y \in Q : y \prec_{-\epsilon} x\}. \tag{2.19}$$

Robustez ligera

En la robustez ligera [2] se toma un escenario considerado nominal, que es aquel que se puede considerar como más probable para el problema o aquel que tiene la menor cantidad de perturbación. La razón de buscar un escenario así es porque puede servir como punto de búsqueda en el espacio de soluciones y porque en la práctica es muy posible que suceda. A diferencia de usar solamente la robustez Min-Max basada en conjuntos que se enfoca en considerar lo mejor de entre peores escenarios. La robustez ligera considera los peores escenarios de las mejores soluciones en el escenario nominal.

Dado un problema de optimización multi-objetivo bajo incertidumbre y un escenario nominal $\xi' \in \mathcal{U}$, sea $Q_{\xi'}$ el conjunto de soluciones eficientes para el problema $\mathcal{P}(\xi')$. Para cada solución eficiente $x' \in Q_{\xi'}$ y dado $0 \leq \epsilon \in \mathbb{R}^k$ se define el problema $LR(x', \epsilon, \mathcal{U}) := (LR(x', \epsilon, \xi), \xi \in \mathcal{U})$, como la familia de problemas de optimización multi-objetivo:

$$\begin{aligned} LR(x', \epsilon, \xi) &:= \min_{x \in Q} F(x, \xi), \\ \text{sujeto a } \ F(x, \xi) &\leq \min_{x' \in P_Q} F(x', \xi') + \epsilon. \end{aligned}$$

Con esto se puede definir robustez ligera.

Definición 2.1.17 (Eficiencia ligera robusta [34]) Dado un problema de optimización multi-objetivo bajo incertidumbre $\mathcal{P}(\mathcal{U})$ con un escenario nominal $\xi' \in \mathcal{U}$ y alguna $\epsilon \in \mathbb{R}^k$. Se tiene que una solución $x' \in Q$ es eficiente ligeramente robusta para el problema $\mathcal{P}(\mathcal{U})$ si es eficiente Min-Max robusta basada en conjuntos para el problema $LR(x', \epsilon, \mathcal{U})$ para alguna solución $x' \in Q_{\xi'}$.

Con la definición de robustez ligera para un problema, se puede definir la contraparte robusta para múltiples problemas multi-objetivo como:

$$\min_{x \in P_{Q,\epsilon}} \sup_{\xi \in \mathcal{U}} F(x, \xi). \tag{2.20}$$

Donde se buscar tomar los peores escenarios del conjunto de soluciones aproximadas y filtrar aquellas que sean Min-Max robustas basadas en conjuntos.

2.1.6. Archivos externos

Un mecanismo de memoria usado principalmente en algoritmos evolutivos durante su búsqueda de soluciones son los archivos externos. Dado algún conjunto de soluciones guarda en él aquellas soluciones que cumplen un criterio de dominancia. El archivo externo más simple y más usado es solo guardar soluciones no dominadas.

En este trabajo son de interés tres cosas: obtener el conjunto de soluciones aproximadas $P_{Q,\epsilon}$, que sean soluciones diversas en espacio de los objetivos y que sean robustas. Los archivos que se muestran a continuación son detallados por Hernández y Schütze [105].

El en el Algoritmo 1 se muestra como encontrar el conjunto de soluciones aproximadas $P_{Q,\epsilon}$. Dada una población P, la población A contenida en el archivo y un vector $\epsilon \in \mathbb{R}^k_+$, primero agrega las soluciones en P que no sean $-\epsilon$-dominadas por alguna solución del archivo. Posteriormente elimina del archivo a las soluciones que sean $-\epsilon$-dominadas por alguna del conjunto P.

Algoritmo 1 *ArchiveUpdate*$P_{Q,\epsilon}$

Input: population P, current archive A_0, dominance vector $\epsilon \in \mathbb{R}^k_+$
Output: updated archive A

```
A := A_0
for all p ∈ P do
    if ∄a ∈ A : a ≺_{-ε} p then
        A := A ∪ {p}
    end if
    for all a ∈ A do
        if p ≺_{-ε} a then
            A := A\{a}
        end if
    end for
end for
return A
```

Es deseable para el tomador de decisiones poder delimitar qué tan cercanas son las soluciones en espacio de los objetivos o en decisión o en ambos. El Algoritmo 2, sigue el comportamiento de la función *ArchiveUpdate*$P_{Q,\epsilon}$ para encontrar soluciones aproximadas pero realizando la discretización en espacio de decisión. Por lo que aquí solo se agregan soluciones al archivo si además de cumplir la restricción del *ArchiveUpdate*$P_{Q,\epsilon}$ también se cumple que no existe alguna otra solución que esté lo suficientemente cerca de acuerdo con un umbral Δ.

En cuanto a la discretización en espacio de los objetivos, en el Algoritmo 3 se muestra el *ArchiveUpdate*$P_{Q,\epsilon}D_y$. Donde añade una solución si además de pertenecer a $P_{Q,\epsilon}$ se encuentra a una

Algoritmo 2 *ArchiveUpdate*$P_{Q,\epsilon}D_x$

Input: population P, archive A_0, dominance vector $\epsilon \in \mathbb{R}_+^k$, threshold $\Delta \in \mathbb{R}_+, \Delta^* \in (0, \Delta)$
Output: updated archive A

```
1:  A := A_0
2:  for all p ∈ P do
3:      if ∄a_1 ∈ A : a ≺_{-ε} p and ∄a_2 ∈ A : d_H(a_2, p) ≤ Δ* then
4:          A := A ∪ {p}
5:          for all a ∈ A do
6:              if p ≺_{-ε} a then
7:                  A := A\{a}
8:              end if
9:          end for
10:     end if
11: end for
12: return A
```

distancia en espacio objetivo mayor a Δ para algún objetivo. Y finalmente el *ArchiveUpdate*$P_{Q,\epsilon}D_{xy}$ mostrado en el Algoritmo 4 discretiza en espacio de los objetivos y de decisión.

Algoritmo 3 *ArchiveUpdate*$P_{Q,\epsilon}D_y$

Input: population P, archive A_0, dominance vector $\epsilon \in \mathbb{R}_+^k$, threshold $\Delta \in \mathbb{R}_+, \Delta^* \in (0, \Delta)$
Output: updated archive A

```
1:  A := A_0
2:  for all p ∈ P do
3:      if ∄a_1 ∈ A : a ≺_{-ε} p and ∄a_2 ∈ A : d_H(F(a_2), F(p)) ≤ Δ* then
4:          A := A ∪ {p}
5:          for all a ∈ A do
6:              if p ≺_{-(ε+1Δ)} a then
7:                  A := A\{a}
8:              end if
9:          end for
10:     end if
11: end for
12: return A
```

Para ejemplificar el uso de archivos externos, en la Figura 2.6 se muestran soluciones en el espacio de los objetivos usando el *ArchiveUpdate*$P_{Q,\epsilon}D_y$ para una algunas instancias específicas del problema de enrutamiento de vehículos. Donde las soluciones obtenidas con este archivo externo se muestran mejor distribuidas y que sirven como posibles alternativas para el tomador de decisiones.

En cuanto a los archivos externos dentro de la optimización robusta, teniendo múltiples escenarios en los que se puede evaluar una solución, hace necesario agregar soluciones que sean robustas. En este caso se utiliza el *ArchiveUpdate*$\mathcal{R}$ [58], que agrega soluciones al archivo externo siempre que cumplan con la robustez Min-Max basada en conjuntos. Como se muestra en el Algoritmo 5 donde agrega nuevas soluciones que cumplan esta robustez para luego eliminar aquellas soluciones

Algoritmo 4 *ArchiveUpdate*$P_{Q,\epsilon}D_{xy}$

Input: population P, archive A_0, dominance vector $\epsilon \in \mathbb{R}_+^k$, threshold in decision space $\Delta_x \in \mathbb{R}_+$, threshold in objective space $\Delta_y \in \mathbb{R}_+, \Delta_x^* \in (0, \Delta_x), \Delta_y^* \in (0, \Delta_y)$
Output: updated archive A

$A := A_0$
for all $p \in P$ **do**
  if $\nexists a_1 \in A : a \prec_{-\epsilon} p$ and $\nexists a_2 \in A : d_H(F(a_2), F(p)) \leq \Delta_y^*$ and $\nexists a_2 \in A : d_H(a_2, p) \leq \Delta_x^*$ **then**
    $A \leftarrow A \cup \{p\}$
    $A' = \{a_1 \in A | \nexists a_2 \in A : a_2 \prec_{-(\epsilon+\Delta_y)} a_1\}$
    for all $a \in A \backslash A'$ **do**
      if $p \prec_{-(\epsilon+\Delta_y)} a$ and $dist(a, A') \geq 2\Delta_x$ **then**
        $A \leftarrow A \backslash \{a\}$
      end if
    end for
  end if
end for
return A

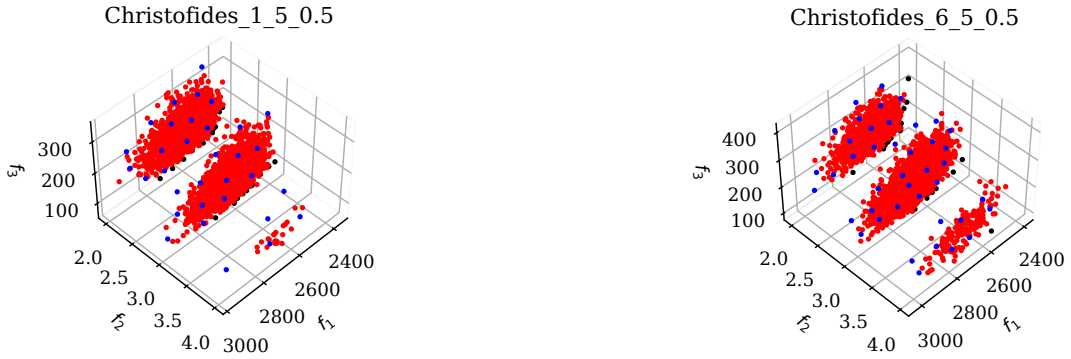


Figura 2.6: Diferentes soluciones en espacio de los objetivos. (azul) Soluciones aproximadas. (negro) Soluciones no dominadas. (rojo) Soluciones encontradas durante la búsqueda.

que dejan de ser robustas.

2.1.7. Trabajo relacionado

En esta sección se presentan los trabajos más recientes sobre algoritmos evolutivos multi-objetivo basados en múltiples indicadores y aquellos que obtienen soluciones aproximadas. Comenzando con el trabajo propuesto por Coello et al. [42], los autores definen el algoritmo evolutivo multi-objetivo

Algoritmo 5 *ArchiveUpdate*$\mathcal{R}$

Input: population P, archive A_0
Output: updated archive A

```
1: A := A_0
2: for all p ∈ P do
3:     if ∄a ∈ A : f_U(a) ⊆ f_U(p) − ℝ^k_≼ then
4:         A := A ∪ {p}
5:     end if
6:     for all a ∈ A do
7:         if f_U(p) ⊆ f_U(a) − ℝ^k_≼ then
8:             A := A\{a}
9:         end if
10:    end for
11: end for
12: return A
```

cooperativo basado en múltiples indicadores (cMIB-MOEA) que es el algoritmo en el cual se basa la propuesta de este trabajo. Esta propuesta se muestra en el Algoritmo 6, usa cinco indicadores de calidad ($HV, R2$,IGD$^+, \varepsilon^+, \Delta_p$) en un modelo basado en islas, donde cada IB-MOEA [37] de cada indicador explora distintas zonas del espacio de soluciones. Luego, se guardan todas las soluciones no dominadas en un archivo externo. Después, intercambia las soluciones entre poblaciones a través de un proceso de migración para añadirlas a la población de la siguiente iteración. Este algoritmo muestra las ventajas de la cooperación de múltiples algoritmos basados en indicadores y el intercambio de información.

Otra alternativa para combinar indicadores es utilizarlos como los objetivos a optimizar como lo proponen Coello et al. [41]. Aquí los autores proponen resolver un problema usando los indicadores de IGD^+ y Riesz s-energy. El algoritmo escalariza los indicadores con la suma ponderada de Tchebycheff y usa un algoritmo similar a SMS-EMOA.

También es posible que los indicadores de calidad compitan como lo proponen Falcón-Cardona y Coello [36], con el MIHPS que se usa múltiples IB-DEs [39] basado en los indicadores de R2, IGD$^+$,ϵ^+,Δ_p, seleccionando un algoritmo específico por un número de iteraciones. Mantiene un registro de convergencia usando el indicador $R2$, si el comportamiento del algoritmo actual es bueno, entonces se le puede incrementar la probabilidad de seleccionarlo en posteriores iteraciones. Por otro lado, si muestra poca convergencia, la probabilidad del algoritmo actual disminuye. Esta hiperheurística muestra el comportamiento competitivo de los algoritmos por ser seleccionado.

Ahora nos enfocamos en los algoritmos recientes que incorporan archivos externos para soluciones aproximadas. Comenzando con $P_{Q,\epsilon}$-NSGA-II propuesto por Schütze et al. [106], que usa el esquema básico de NSGA-II pero añade el archivo externo para soluciones aproximadas y dominancia epsilon. Este algoritmo mantiene los mecanismos de selección y mutación de NSGA-II en un paso para luego añadir la nueva población en el archivo externo.

Incorporar soluciones aproximadas en un esquema básico de MOEA fue propuesto por Schütze et al. [107] con $P_{Q,\epsilon}$-MOEA, que mantiene un archivo externo de soluciones aproximadas. Inicialmente agrega cualquier solución al archivo y en siguientes iteraciones toma soluciones del archivo para cruzarlas, aplicar mutación y agregar la nueva solución al archivo. Otro algoritmo con soluciones aproximadas es el algoritmo evolutivo multi-objetivo para soluciones epsilon cercanas (nevMOGA)

Algoritmo 6 cMIB-MOEA

Input: Number $nmig$ of solutions to migrate; migration frequency $fmig$; population size μ; set of indicators $\mathcal{I} = \{I_1, .., I_k\}$
Output: Pareto front approximation

```
Set archive A as empty
for j = 1 to k do do
    Randomly initialize subpopulation P_j of size μ\k
    Initialize the j^th IB-MOEA
end for
A ← Nondominated(⋃_{j=1}^{k} P_j)
while stopping criterion is not fulfilled do
    for j = 1 to k do do
        P_j ←IB-MOEA(P_j, I_j, fmig)
    end for
    A ← A ∪ {⋃_{j=1}^{k} P_j}
    A ← Nondominated(A)
    Obtain z⃗* and z⃗^nad from A to normalize A
    while |A| > μ do
        a⃗_worst ← argmax_{a⃗∈A} C_{E_s}(a⃗, A)
        A ← A\{a⃗_worst}
    end while
    {P_1, .., P_k} ← Migration(nmig, {P_1, .., P_k}, {I_1, .., I_k})
end while
return A
```

[93], que mantiene dos archivos, uno para soluciones no dominadas y otro con soluciones aproximadas. El algoritmo tiene cuatro poblaciones dentro del algoritmo, una población para soluciones aproximadas, otra para soluciones no dominadas, una para soluciones aproximadas no dominadas y la población con las soluciones que se van generando durante la búsqueda.

Los algoritmos anteriores a pesar de buscar las soluciones aproximadas, tienen la desventaja de no considerar la distribución de las soluciones en espacio de los objetivos y de decisión. El algoritmo de NϵSGA propuesto por Hernández et al. [57] logra mejorar las dificultades que los algoritmos anteriores, pues utiliza parte de los principios de NSGA-II solo que ahora realiza el ordenamiento de rango de acuerdo con la $-\epsilon$-dominancia. Tiene dos poblaciones, una población en la que trata de acercarse al frente de Pareto y otra población de soluciones aproximadas para las que en algún momento realiza un proceso de migración. También realiza discretización en ambos espacios y con todos estos elementos el algoritmo logra mejorar la diversidad de soluciones.

2.2. Optimización con colonia de hormigas

La metaheurística de optimización mediante colonia de hormigas (ACO, por sus siglas en inglés[16]) propuesta por Dorigo et al. [31], es una estrategia en la cual una colonia artificial de hormigas trabajan en conjunto para construir soluciones en problemas de optimización. Debido a

[16] Ant Colony Optimization

que las hormigas por su naturaleza pueden presentar una organización estructurada y como resultado, lograr tareas que para una sola hormiga resultaría complicado.

Las hormigas se comunican indirectamente a través del ambiente, como puede ser con los rastros de feromonas en el suelo. Una hormiga en esta metaheurística se comporta como un agente que iterativamente va construyendo una solución usando el conocimiento que puede adquirir del ambiente y que comparte de los otros agentes. Logrando que el trabajo individual de una hormiga se vuelve importante cuando múltiples hormigas exploran el ambiente.

Las hormigas cada vez que pasan por algún camino, pueden dejar un rastro de feromona, que sirve como un mecanismo de memoria a largo plazo y va a depender de la hormiga la cantidad de feromona que se deposite en cada camino que recorra. La manera en la que una hormiga puede decidir que camino tomar depende del rastro de feromona asociado a cada camino, mientras más grande sea la cantidad de feromona en un camino más grande es la probabilidad de escogerlo. Una de las maneras de evitar que los caminos tomen grandes cantidades de feromonas y como consecuencia, las hormigas prematuramente tomen siempre el mismo camino es usando mecanismos de evaporación de feromona.

2.2.1. Descripción básica

Para utilizar la metaheurística en problemas de optimización combinatoria es necesario definir los elementos necesarios para que las hormigas puedan construir las soluciones. Iniciando con la representación del problema, donde se modelan todas las variables de decisión y estructuras de datos usados por las hormigas. También es importante definir las reglas de probabilidad respecto a las cantidades de feromona y la información heurística. Además de definir como actualizar la matriz de feromonas y su evaporación. A continuación se muestra una descripción básica de estos elementos.

Representación del problema

Una hormiga artificial se encarga de ir construyendo de forma iterativa una solución a partir una parcial [31], por lo que inicialmente se tiene que definir el cómo representar una solución del problema a resolver.

Definición 2.2.1 (Problema de optimización combinatoria [31]) Un **problema de optimización combinatoria** se define como una tercia (S, f, Ω) donde:

S es el conjunto de soluciones candidatas al problema,
f es la función objetivo $f : S \to \mathbb{R}$ que asigna un costo $f(s)$ para cada solución $s \in S$,
Ω es el conjunto de restricciones para f.

Para aplicar una metaheurística en un problema de optimización, éste debe tener bien definidos los siguientes elementos:

- Un conjunto finito de componentes $C = \{c_1, c_2, .., c_n\}$ con N_c el número de componentes.
- Los estados del problema se definen mediante conjuntos de componentes x=$\{c_i, c_j, .., c_k\}$ de tamaño finito y el conjunto con todos los estados se define como Q.
- El conjunto de soluciones candidatas S es un subconjunto de los estados Q.

- Existe un conjunto de soluciones óptimas $S^* \subset S$.

Dado el modelo definido del problema, las hormigas artificiales construyen soluciones mediante decisiones estocásticas en una gráfica de construcción $G_c = (C, L)$ con C el conjunto de componentes ya definido y L denotado como el conjunto de conexiones entre componentes. En cada arco de la gráfica $l_{ij} \in L$ se tiene asociado un rastro de feromona τ_{ij} y un valor heurístico η_{ij}. Estos valores son usados en las reglas de probabilidad de las hormigas.

Metaheurística

A continuación se explican los mecanismos más importantes de ACO en los problemas de optimización, como lo son la construcción de soluciones, la actualización de feromonas y la evaporación de feromonas.

Construcción de soluciones

El hecho de usar ACO en un problema de optimización significa que el problema se puede modelar con una gráfica. De esta forma, las soluciones para el problema a tratar se pueden construir tomando el recorrido que realizan las hormigas sobre la gráfica. En cuanto a la manera en la que las hormigas pueden ir recorriendo la gráfica, se pueden basar en tomar decisiones de una manera probabilista. Las hormigas cada vez que visitan un nodo y teniendo la información de las cantidades de feromona que se tiene en los nodos adyacentes, debe tomar una decisión de cual es el siguiente nodo a visitar. Dada una probabilidad p_{ij}^k, con k la hormiga actual, (i, j) el arco con feromona τ_{ij}^α y N_i^k como los nodos adyacentes a i que son alcanzables como se muestra en la Ecuación (2.21). Una vez tomada la decisión, la hormiga deja un rastro de feromona en el arco que eligió.

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha}{\sum_{l \in N_i^k} \tau_{il}^\alpha} & \text{si } j \in N_i^k \\ 0 & \text{si } j \notin N_i^k \end{cases}. \tag{2.21}$$

Actualización de feromonas

En esta etapa las feromonas de la gráfica pueden ser modificadas, ya sea para aumentar sus valores gracias al número de veces que una hormiga toma un camino o evaporar un porcentaje de la feromona. Las hormigas pueden evitar caminos que previamente ya habían sido explorados, una de las complicaciones de usar ACO es la convergencia temprana hacia óptimos locales. Por esto, es importante el manejo de la feromona que una hormiga agrega. Ya sea en las actualizaciones locales donde las hormigas depositan una cantidad de feromona después de escoger un camino. Y las actualizaciones globales, donde una vez que se construyeron soluciones basándose en alguna condición se actualizan algunos arcos de la gráfica. Una manera común de actualizar la feromona τ_{ij} en el arco (i, j) es añadiendo una cantidad Δ^k por la hormiga k y teniendo un factor de evaporación $\rho \in (0, 1]$ como se muestra en la Ecuación (2.22).

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta^k. \tag{2.22}$$

Acciones adicionales

Cuando las hormigas terminan de construir sus soluciones, se pueden usar estrategias que no realizan las hormigas por si solas, como pueden ser utilizar estrategias de búsqueda local a partir de algunas soluciones obtenidas. También se puede agregar ciertas cantidades adicionales de feromona en arcos de soluciones que satisfacen algún criterio, o bien almacenar en memoria algunas soluciones.

En el Algoritmo 7 se muestra el planteamiento general de la metaheurística de ACO. Donde primero se inicializa la matriz de feromonas de acuerdo con los parámetros del problema. Mientras no se cumpla algún criterio de término, las hormigas construyen soluciones, aplican algún algoritmo de búsqueda local y actualizan la matriz de feromonas.

Algoritmo 7 Ant Colony Optimization

```
1: Set parameters, initialize pheromone trails
2: while termination condition not met do
3:     Construct Ant Solutions
4:     ApplyLocalSearch (optional)
5:     UpdatePheromones
6: end while
```

Variantes clásicas de ACO mono-objetivo

Se menciona breve mente algunos algoritmos clásicos basados en colonia de hormigas. Iniciando con sistema de hormigas (AS, por sus siglas en inglés[17]) [33], que es la versión más básica de un algoritmo basado en hormigas artificiales, se propuso para resolver el problema del agente viajero (TSP, por sus siglas en inglés[18]) [45]. La matriz de feromonas se inicializa a valores ligeramente mayor a los esperados a los que depositaría la hormiga al principio de las iteraciones. Las hormigas eligen de forma probabilista cada nodo a visitar de acuerdo con la feromona y la información heurística respecto a todos los posibles nodos, asignándole prioridades a estos dos valores. La feromona que las hormigas depositan está en términos de la calidad de la solución que cada una ha construido. La actualización de feromonas se realiza una vez que todas las hormigas de la población han terminado de construir las soluciones y se ha aplicado la evaporación a la matriz de feromonas.

Una mejora para AS es utilizar una estrategia elitista (EAS, por sus siglas en inglés[19]) propuesta por Dorigo et al. [17], que le da mayor importancia a las mejores soluciones en cada iteración. Se basa en tomar la mejor ruta construida en cada iteración y agregar feromona adicional en todos los arcos que pertenecen a esta ruta. Con este cambio, las hormigas pueden converger más rápido a las mejores rutas, sin embargo, con el riesgo de estancarse de forma temprana.

Otra versión de AS es la basada en rangos AS_{rank} propuesta por Bullnheimer et al. [19], donde la feromona que la hormiga deposita depende del rango que tiene, a mayor rango es mayor la feromona que puede depositar. El rango de la hormiga está dado por la calidad de la solución que construyó. Para la actualización de feromonas, primero se ordenan las hormigas por rango y se selecciona un

[17] Ant System
[18] Travelling Salesman Problem
[19] Elitist Ant System

número dado de éstas para actualizar la matriz de feromonas. La cantidad de feromona que se puede depositar está ponderada de acuerdo con el rango.

El Min-Max AS (MMAS) propuesta por Stützle et al. [112], se enfoca en limitar el rango de las feromonas estableciendo límites inferior y superior. La matriz de feromonas se inicializa con un valor alto cercano al límite superior con el objetivo de explorar bastantes soluciones desde las primeras iteraciones. Dentro de este algoritmo se propone reiniciar la matriz de feromonas en un cierto tiempo o cuando no se han mejorado lo suficiente las soluciones.

Una variante que mejora a las anteriores es el sistema de colonia de hormigas (ACS, por sus siglas en inglés[20]) propuesta por Dorigo y Gambardella [32]. En esta variante se utiliza reglas de probabilidad mayormente enfocadas en realizar explotación y de una forma más intensa a la que se usa en AS. Además de que la evaporación y el depósito de feromonas se realizan solo en los arcos que pertenecen a las mejores soluciones. Este algoritmo también trata de hacer exploración removiendo una cantidad de feromona cada vez que una hormiga pasa por algún arco.

2.2.2. Colonia de hormigas multi-objetivo

Los algoritmos de optimización multi-objetivo basados en colonia de hormigas (MOACO, por sus siglas en inglés[21]) ya existentes en la literatura comparten ideas que pueden ser usadas para crear nuevos algoritmos. Como lo propusieron López-Ibáñez y Stützle [77] en un marco general que trata el diseño de algoritmos basados en MOACO. Donde no se especifica el cómo implementar cada uno de estos algoritmos, si no, que componentes se pueden agregar al nuevo algoritmo. Tales componentes se encuentran en tres grupos principales: construcción de soluciones, actualización de feromonas y el uso de múltiples colonias. A continuación se describen estos tres grupos.

Construcción de soluciones

En la modelación del problema puede ser necesario utilizar una matriz de feromona por cada objetivo si es posible descomponer el problema o bien, utilizar solo una matriz de feromona, lo mismo sucede con la matriz de información heurística. Aunque en ocasiones es posible usar una matriz por cada objetivo, durante la construcción de las soluciones puede ser necesario reunir la información de todas las matrices en una sola. Para realizar esto, se pueden utilizar algún método de agregación como pueden ser:

1. *Suma ponderada*

   Dado un escalar $\lambda \in \mathbb{R}$, además de dos matrices de feromonas τ^1_{ij}, τ^2_{ij} y dos matrices de información heurística η^1_{ij}, η^2_{ij}, se pueden definir las nuevas matrices τ_{ij}, η_{ij} de feromona e información heurística respectivamente con la Ecuación (2.23).

$$\tau_{ij} = (1-\lambda)\tau^1_{ij} + \lambda\tau^2_{ij}, \qquad \eta_{ij} = (1-\lambda)\eta^1_{ij} + \lambda\eta^2_{ij}. \tag{2.23}$$

2. *Producto ponderado*

   De manera similar a la suma ponderada pero utilizando el producto como se muestra en la Ecuación (2.24) para actualizar las matrices.

[20] Ant Colony System
[21] Multi-objective Ant Colony Optimization

$$\tau_{ij} = (1-\lambda)\tau_{ij}^1 \cdot \lambda\tau_{ij}^2, \qquad \eta_{ij} = (1-\lambda)\eta_{ij}^1 \cdot \lambda\eta_{ij}^2. \tag{2.24}$$

3. *Aleatoriedad*

   Con una probabilidad uniforme p en cada paso de la construcción de la solución se toma un elemento de la primer matriz y con probabilidad $1-p$ se toma el elemento de la segunda matriz. Esto se realiza también para la matriz de información heurística.

En las funciones de agregación descritas anteriormente se utiliza un peso $\lambda \in \Lambda$ de un conjunto de pesos Λ. Durante la construcción de las soluciones puede ser necesario utilizar pesos para diversificar la búsqueda de las hormigas. Con el conjunto de pesos Λ pueden suceder dos cosas; que todas las hormigas usen el conjunto de pesos en la misma iteración o que solo se use un peso específico por iteración. La elección depende del tomador de decisiones.

Actualización de feromonas

Dado un conjunto de soluciones A^{upd} en la iteración actual, para realizar la actualización de feromonas es necesario tomar un subconjunto de N^{upd} soluciones que puedan ser consideradas para agregar feromona a las matrices, de aquí se pueden considerar algunas estrategias como:

1. *Soluciones no dominadas*

   Del conjunto de soluciones A^{upd} tomar N^{upd} soluciones que sean no dominadas, siguiendo un orden de prioridad en caso de tener más de N^{upd} soluciones y con estas actualizar todas las matrices de feromonas.

2. *Mejor de cada objetivo*

   Tomar las mejores N^{upd} soluciones en cada objetivo y con cada una de estas actualizar la matriz de feromona de cada objetivo. En caso de tener solo una matriz de feromonas, tomar $2 \cdot N^{upd}$ soluciones de cada objetivo.

3. *Mejor de cada objetivo con pesos*

   En el caso de que las hormigas utilicen un vector con un peso λ para cada objetivo, al igual que el anterior, tomar N^{upd} soluciones para cada objetivo y actualizar su correspondiente matriz de feromonas del objetivo indicado. En el caso de solo tener una matriz, tomar $2 \cdot N^{upd}$ soluciones de cada objetivo.

El uso de alguna de estas estrategias depende del tomador de decisiones.

Uso de múltiples colonias

En el diseño de algoritmos basados en ACO, se pueden usar múltiples colonias con una matriz de feromona individual y una matriz de información heurística, lo cual puede resultar en que las colonias intercambien soluciones para actualizar su respectiva matriz de feromona. Si un grupo de hormigas pertenece a una colonia significa que comparten la misma matriz de feromona e información heurística para construir soluciones. Si una colonia tiene sus respectivas matrices, también

puede tener su propio conjunto de pesos. Una de las ideas principales del porqué utilizar múltiples colonias es la capacidad de compartir información para la actualización de feromonas. Dado un número de colonias N^{col}, dos factores a tomar en cuenta son los pesos que pueden tener las colonias y la actualización de feromona que pueden tener, como a continuación se presenta.

1. *Pesos para múltiples colonias*

   Se tienen dos maneras en la que se pueden generar pesos paras las N^{col} colonias, teniendo pesos disjuntos o pesos que se traslapan. Para generar pesos disjuntos, basta con generar pesos de manera uniforme en el intervalo $[0, 1]$ y repartirlos de manera equitativa entre las colonias. En cuanto a los pesos que se traslapan, los pesos se pueden repartir de manera que las colonias compartan el 50 % de sus pesos.

2. *Actualización de feromonas en múltiples colonias*

   Al momento de que las colonias generan sus soluciones, es necesario guardar éstas en un archivo A^{iter} donde quedaran solo aquellas soluciones que sean no dominadas. Después, aquellas soluciones no dominadas son utilizadas para actualizar la matriz de feromona a la que pertenecen. También se puede aumentar la cooperación entre colonias, dividiendo el conjunto de soluciones en A^{iter} entre las colonias de manera que cada colonia se encarga de una región del espacio de soluciones distinta.

En la el Algoritmo 8 se muestra el marco general de MOACO que proponen López-Ibáñez y Stützle [77]. Primero se inicializan las matrices de feromona (línea 2) y el conjunto de pesos (línea 3). Aplica las funciones de agregación para feromona (línea 11) y para información heurística (línea 16). Luego construye soluciones (línea 21) y aplica búsqueda local (línea 22). Guarda soluciones no dominadas (línea 23). Distribuye las soluciones en el archivo externo en las colonias (línea 28). Con las soluciones distribuidas, realiza la actualización de matrices en cada colonia (línea 29). Cuando se haya cumplido el criterio de término devuelve las soluciones no dominadas (línea 33).

Clasificación de ACOs en MOP

En la literatura existen bastantes propuestas para clasificar algoritmos de colonia de hormigas, para este trabajo se considera la propuesta de Coello et al. [38]. Donde se describe una taxonomía de algoritmos multi-objetivo de ACO como se muestra en la Figura 2.7. Aquí se propone un esquema general para el diseño de algoritmos multi-objetivos basados en ACO como se presenta en el Algoritmo 9. Donde propone usar una matriz de feromona por cada uno de los k objetivos (línea 1). Mantener soluciones no dominadas en un archivo externo (línea 2). Luego cada hormiga con la información heurística de todos los objetivos y sus matrices de feromonas construye una solución (líneas 4-9). Se actualiza el archivo externo con las nuevas soluciones (línea 8). Después actualiza las matrices de feromonas de acuerdo con las nuevas soluciones (línea 9) y finalmente regresa el archivo externo (línea 12).

De acuerdo con la taxonomía se tienen algoritmos MOACO clasificados según sus características principales como el uso de múltiples colonias, uso de múltiples matrices de feromonas y su forma de evaluación. Como la propuesta de este trabajo se enfoca en los indicadores de calidad para ACO, es de interés conocer algunos algoritmos representativos según su evaluación. Por lo que a

Algoritmo 8 MOACO for multiple colonies and mutiple matrices

Input: number of colonies N^{col}, number of ants N^a, pheromone matrices τ, information matrices η
Output: non-dominated solutions

for each colony$c \in \{1, .., N^{col}\}$ **do**
  InitializePheromoneInformation()
  $\Lambda_c \leftarrow$ MultiColonyWeights()
end for
$iter \leftarrow 0$
while not stopping criteria met **do**
  $A^{iter} \leftarrow \emptyset$
  for each colony $c \in \{1, .., N^{col}\}$ **do**
    for each ant $k \in \{1, .., N^a\}$ **do**
      $\lambda \leftarrow$ NextWeight$(\Lambda_c, k, iter)$
      if multiple pheromone matrices **then**
        $\tau \leftarrow$ Aggregation$(\lambda, \{\tau_c^1, \tau_c^2\})$
      else
        $\tau \leftarrow \tau_c$
      end if
      if multiple information matrices **then**
        $\eta \leftarrow$ Aggregation$(\lambda, \{\eta_c^1, \eta_c^2\})$
      else
        $\eta \leftarrow \eta$
      end if
      $s \leftarrow$ ConstructSolution(τ, η)
      $s \leftarrow$ WeightedLocalSearch(s, λ)
      $A^{iter} \leftarrow$ RemoveDominated$(A^{iter} \cup \{s\})$
    end for
  end for
  $A^{bf} \leftarrow$ RemoveDominated$(A^{bf} \cup A^{iter})$
  for each colony $c \in \{1, .., N^{col}\}$ **do**
    $A_c^{upd} \leftarrow$ MultiColonyUpdate(A^{upd})
    PheromoneUpdate(A_c^{upd}, N^{upd})
  end for
  $iter \leftarrow iter + 1$
end while
return RemoveDominated(A^{bf})

continuación se presentan algunos algoritmos MOACO representativos basados en la evaluación de Pareto, descomposición e indicadores de calidad.

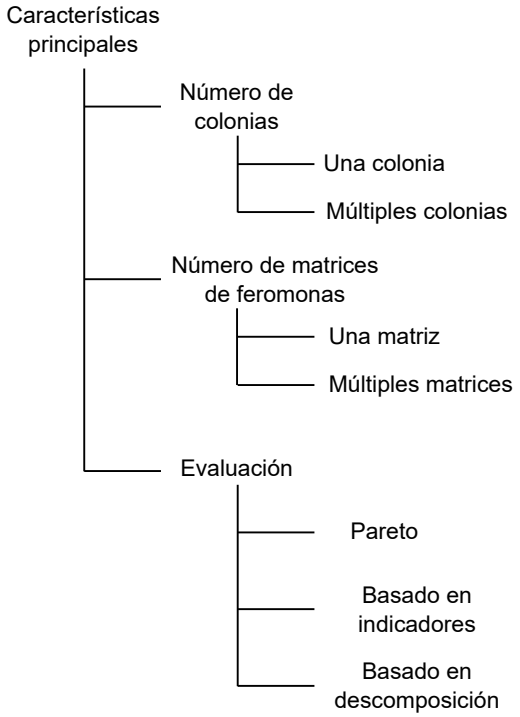


Figura 2.7: Taxonomía de los algoritmos multi-objetivo basados en ACO, propuesta por Coello et al. [38].

Algoritmo 9 MOACO for k objectives

Init_Pheromone_Trails(τ_{rj}^{i}); // for $i = 1, .., k; r, j = 1, .., n$
$\mathcal{A} = \emptyset$; / Archive of ranked non-dominated solutions
$t = 0$
while $t \leq T_{max}$ **do**
    for $h = 1, ..., N_a$ **do**
        BuildSolution$_h(S(t), \tau^1, ..., \tau^k, \eta^1, ..., \eta^k)$;
    end for
    ArchiveUpdate$(S(t), \mathcal{A})$;
    PheromoneUpdate$(\tau^1, ..., \tau^k, \mathcal{A})$;
    $t = t + 1$
end while
return $\mathcal{A}$

Algoritmos basados en evaluación de Pareto

Este tipo de algoritmos evalúan las soluciones de acuerdo con la dominancia de Pareto como el rango de dominancia. Uno de los más representativos es CPACO [6] que es una mejora del algoritmo PACO [52]. En CPACO utiliza una sola matriz de feromonas y una matriz por cada información

heurística a diferencia de PACO donde usa una matriz de feromona y una de información heurística por cada objetivo. También se calcula el rango de dominancia de cada solución como en NSGA-II [29] y se usa un esquema de reemplazo basado en la distancia *crowding* [62]. El algoritmo realiza los siguientes pasos en cada iteración:

Paso 0 Con el esquema de reemplazo y dada una población de soluciones S, crea una nueva población Y donde cada solución de Y reemplaza a alguna solución de S si es mejor.

Paso 1 Calcula el rango mediante el ordenamiento no dominado de NSGA-II.

Paso 2 Inicializa los elementos de la matriz de feromonas a un valor dado.

Paso 3 Todos los elementos en la población actualizan la matriz de la feromona con el inverso de su rango obtenido.

Este mecanismo hace que las hormigas puedan explorar mejor la información heurística usando solo una matriz de feromonas y al tener un mejor rango la cantidad de feromona es mayor. Este algoritmo tiene como objetivo converger hacia el frente de Pareto en lugar de una región específica.

Otras propuestas de ACO basados en dominancia fueron diseñados por Alaya et al. [3] donde definen una familia de algoritmos a partir de m-ACO donde usa m colonias y m matrices de feromona para m objetivos y cada colonia se enfoca en optimizar un objetivo. De aquí deriva el algoritmo m-ACO$_1(m+1, m)$ que usa $m+1$ colonias, con m matrices de feromona, tiene el mismo funcionamiento del algoritmo base pero la colonia extra optimiza todos los objetivos. La variante m-ACO$_2(m+1, m)$ usa misma cantidad de colonias y matrices de feromonas pero la actualización de feromona involucra a todas las colonias. La variante m-ACO$_3(1, 1)$ usa una sola colonia y una sola matriz de feromona y finalmente m-ACO$_4(1, m)$ que usa una colonia y m matrices de feromona.

Algoritmos basados en descomposición

El algoritmo más representativo en esta clase es la optimización multi-objetivo mediante colonia de hormigas basada en descomposición (MOEA/D-ACO, por sus siglas en inglés[22]) que descompone el problema multi-objetivo en subproblemas de un solo objetivo donde cada hormiga trata de resolver un subproblema. Al tratar cada subproblema, las hormigas se dividen en grupos con el objetivo de que cada uno se enfoque en aproximarse a una región del frente de Pareto. Cada grupo tiene su propia matriz de feromonas y cada hormiga su matriz de información heurística. Para la actualización de feromona de la matriz de cada grupo se utilizan las soluciones creadas en su propio grupo. A continuación se describe MOEA/D-ACO.

Inicialmente se descompone el problema en N subproblemas de un solo objetivo asociando a cada uno un vector λ de pesos, de manera que al problema i le corresponde el vector de pesos λ^i, donde la función de un solo objetivo $g(x|\lambda^i)$ se define de acuerdo con la Ecuación (2.25).

$$\min_{x \in \Omega} \; g(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x). \tag{2.25}$$

Una vez descompuesto el problema se realizan los siguientes pasos:

[22] Multiobjective Evolutionary Algorithm Decomposition Ant Colony Optimization

Paso 0 Inicializar una solución para cada uno de los N subproblemas y la matriz de información τ^i. Para cada uno de los K grupos inicializa la matriz de información η^j. Agrega en el archivo externo las soluciones no dominadas de entre las soluciones iniciales.

Paso 1 Para cada hormiga se construye una nueva solución con la regla de probabilidad dada que depende de la solución actual x^i, la matriz de información heurística η^i y la matriz de feromona τ^j del grupo j.

Paso 2 Agrega cada solución que se acaba de construir si no existe alguna solución en el archivo externo que la domine y eliminar del archivo externo aquellas soluciones que son dominadas por la nueva solución.

Paso 3 Si se cumple el criterio de paro, terminar la ejecución de los pasos y regresar el contenido del archivo externo.

Paso 4 Para cada grupo j, actualiza la matriz de feromonas τ^j con las soluciones construidas por las hormigas en ese grupo y que fueron guardadas en el archivo externo.

Paso 5 Para cada hormiga i que construyó su solución x^i, revisa en las hormigas de su vecindario si existe alguna solución y tal que $g(y|\lambda^i) < g(x^i|\lambda^i)$. Esto significa que las hormigas tomaran la mejor solución de sus vecinos solo si es mejor que la suya bajo el vector de pesos que le corresponde. Esto solo si la solución y no había sido usada para reemplazar alguna solución anteriormente. Finalmente repetir el Paso 1 si no se ha cumplido algún criterio de término.

Algoritmos basados en indicadores

El algoritmo de optimización mediante colonia de hormigas basado en indicadores (IBACO, por sus siglas en inglés[23]) [83] utiliza los indicadores binarios de calidad de hipervolumen y ϵ^+ para agregar feromona a las matrices. Pues uno de los retos al implementar ACO en los problemas multi-objetivo es la decisión sobre que cantidad de feromona debe agregar la hormiga en cada arco que elige. En muchas ocasiones definir esta cantidad sujeta a las restricciones del problema y los objetivos es complicado. Por ello, una alternativa más es usar la calidad de la solución dada por los indicadores, una mejor calidad implica mayor feromona.

Dado un número determinado de N hormigas, la construcción de las soluciones y la probabilidad de moverse de entre los nodos de la gráfica pueden realizarse de la misma forma en la que otros algoritmos basados en colonia de hormigas lo hacen. Una vez construidas las soluciones, se guardan en un archivo externo aquellas que no sean dominadas.

En la Ecuación (2.26) se muestra que dada una hormiga k la cantidad de feromona que deposita en el arco (i, j) es la calidad de su solución de acuerdo con la función de aptitud definida en la sección anterior.

$$\Delta\tau_{ij}^k = \begin{cases} Fit(S_{ant}) & \text{si } (i,j) \in S_{ant} \\ 0 & \text{e.o.c} \end{cases}. \tag{2.26}$$

Para cada una de las hormigas se suma el total de veces que pasaron por el mismo arco y se actualiza la entrada en la matriz de feromonas junto al proceso de evaporación dada la constante $\rho > 0$ como se muestra en la Ecuación (2.27).

[23]Indicator-based Ant Colony Optimization

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^{N} \Delta\tau_{ij}^{k}. \tag{2.27}$$

2.2.3. Trabajo relacionado

Se mencionan contribuciones en el estado del arte sobre ACOs multi-objetivo basados en indicadores. Comenzando con el trabajo de Yi et al. [76], donde los autores usan el indicador binario ϵ para evaluar la convergencia de cada solución dentro de un archivo externo. Con esto logran mejorar la diversidad de MOACO tradicional. Este algoritmo usa dos archivos externos, uno para la convergencia y otro para la diversidad. Además de dos matrices de feromonas, una para cada archivo externo. Las soluciones construidas por iteración actualizan su matriz correspondiente y la feromona global es calculada de acuerdo con las dos matrices.

Una versión ponderada del indicador ϵ es usada dentro del algoritmo IWBLS/ACO por Moncef et al. [88]. Este algoritmo tiene el objetivo de mejorar la exploración dentro del problema de la mochila. Dadas las soluciones construidas por las hormigas se evalúan con el indicador ϵ ponderado. Después de evaluar, aplican búsqueda local y eliminan las menos aptas de acuerdo con el indicador.

Falcón-Cardona y Coello [40] proponen el iMOACO$_{\mathbb{R}}$, que es una mejora de ACO$_{\mathbb{R}}$ [110], donde las hormigas construyen soluciones sobre un espacio de búsqueda continuo. En iMOACO$_{\mathbb{R}}$ se utiliza un mecanismo de selección basado en el indicador $R2$ para seleccionar que soluciones se guardan en un archivo de feromonas. La razón de usar rangos basados en $R2$ en lugar de dominancia de Pareto es para tener un mejor desempeño en problemas de muchos objetivos.

Un trabajo usando vectores de dirección de forma gradual fue propuesto por Mansour et al. [82]. Donde las hormigas tienen un vector de dirección que cambia de acuerdo con la frecuencia de movimiento de éstas. En este algoritmo utilizan una matriz de feromonas que es actualizada únicamente con las soluciones no dominadas. Muestran que el método de generación de pesos puede mejorar la diversidad de soluciones.

2.3. Problema de enrutamiento de vehículos multi-objetivo

Desde su formulación básica propuesta por Dantzig et al. [26] como una generalización del problema del agente viajero, el problema de enrutamiento de vehículos (VRP) se ha hecho cada vez más complejo. Debido a los avances en cuanto la forma en que se transportan bienes, la satisfacción del cliente, límites de tiempo, incertidumbre, número de clientes entre otros. Todo esto aumenta la cantidad de objetivos que están en conflicto y que requieren de mejores estrategias.

2.3.1. Planteamiento clásico

El problema clásico de VRP se puede definir formalmente como sigue:

Se tiene un conjunto $N = \{1, 2, .., n\}$ de n puntos los cuales se requieren visitar desde un punto especial denominado *depósito* que se denota como 0. Una matriz de distancias D que especifica la distancia d_{ij}=d_{ji} entre todo par de puntos $i, j \in N$. El vector de entregas Q especifica la demanda q_i para ser entregada en cada punto i. Se tiene un conjunto de vehículos K, donde la capacidad cada vehículo $k \in K$ se denota como C. La variable booleana x_{ijk}=x_{jik}=1 significa que el vehículo k pasa por el arco de los puntos i y j.

La función objetivo (2.28) es minimizar la distancia total:

$$\text{mín} \sum_{k \in K} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ijk}, \tag{2.28}$$

sujeto a:

$$x_{iik} = 0, k \in K, i \in N, \tag{2.29}$$

$$x_{ijk} = x_{jik}, k \in K \ i, j \in N, \tag{2.30}$$

$$x_{0jk} = 1, \ k \in K, \tag{2.31}$$

$$\sum_{k \in K} \sum_{i=0}^{n} \sum_{j=0}^{n} q_j x_{ijk} \leq C, \ k \in K, \tag{2.32}$$

$$C > \max_{i \in N} q_i. \tag{2.33}$$

La restricción (2.29) asegura que no se visite un nodo a si mismo. La restricción (2.30) asegura que el costo de recorrer un arco en cualquier dirección es el mismo. La restricción (2.31) asegura que los vehículos inicien en el depósito. La restricción (2.32) asegura que la demanda total de cada recorrido no excede la capacidad del vehículo. La restricción (2.33) asegura que no existe alguna demanda que supere la capacidad del vehículo.

2.3.2. Variaciones de VRP

En la Tabla 2.1 se muestran algunas variantes básicas de problemas de enrutamiento de vehículos que sirven como base para problemas más específicos. Para encontrar más problemas e instancias de prueba se puede consultar [85].

2.3.3. Metaheurísticas para VRP

Actualmente existen bastantes metaheurísticas que se han aplicado a los problemas de VRP. Las metaheurísticas se pueden clasificar en dos formas, las basadas en una solución y las basadas en población [116]. Las basadas en una solución aplicadas a VRP son principalmente metaheurísticas que usan heurísticas basadas en búsqueda local. Las basadas en población comprenden un mayor alcance pero este trabajo se enfoca en algoritmos bio-inspirados.

Metaheurísticas basadas en una solución

El recocido simulado es una de las más usadas, como en [44] donde tratan el problema básico de CVRP realizando intercambios entre los clientes dentro del vector que representa la solución. En [8] proponen usar múltiples temperaturas para el VRPTW multi-objetivo y modificar el criterio de aceptación de acuerdo con la dominancia de las nuevas soluciones. En [70] proponen un cambio en el esquema de enfriamiento para el TDVRP que mejora las probabilidades para el criterio de aceptación. También se pueden utilizar distintos operadores para modificar las rutas de VRP como se muestra en [75], donde le asignan alguna probabilidad de usar cada operador durante el recocido.

La búsqueda tabú también se ha utilizado desde los planteamientos más básicos como en [74] para el CVRP, donde consideran como movimientos tabú a las inserciones y desplazamientos de clientes

Problema	Descripción	Referencia
Enrutamiento de vehículos con ventanas de tiempo	Para cada cliente se tiene asociado un intervalo de ventana que restringe el horario en que un vehículo puede visitar al cliente.	[63]
Enrutamiento de vehículos con retornos	Se pueden recoger y entregar paquetes para cualquier cliente. Los clientes se dividen en quieres requieren alguna entrega y aquellos que quieren realizar una entrega.	[30]
Enrutamiento con vehículos heterogéneos	Los vehículos tienen un tipo diferente de capacidad y un costo asociado. El costo de cada vehículo está en términos de la distancia que recorre.	[114]
Enrutamiento de vehículos con múltiples depósitos	Los vehículos pueden regresar a distintos depósitos.	[89]
Enrutamiento de vehículos periódico	Existe un horizonte de planeación, donde en un intervalo de días se debe visitar a los clientes necesarios y los vehículos están disponibles en algunos días.	[20]
Enrutamiento de vehículos con entregas divididas	Se pueden dividir las demandas de los clientes y los vehículos pueden variar en su capacidad como disponibilidad.	[109]
Enrutamiento de vehículos estocástico	Distintos componentes del problema son aleatorios o dependen de una distribución de probabilidad. Algunos componentes estocásticos pueden ser las demandas de los clientes o las distancias entre éstos.	[12]
Enrutamiento de vehículos dinámico	Durante la visita de los clientes en el recorrido pueden aparecer nuevos clientes no contemplados.	[68]
Enrutamiento de vehículos dependiente del tiempo	El costo de viaje entre clientes depende del día en que se realice el viaje y se utilizan ventanas de tiempo.	[81]

Tabla 2.1: Variantes clásicas de problemas de enrutamiento de vehículos.

dentro de las rutas. Una modificación interesante de búsqueda tabú se encuentra en [61] donde lo paralelizan para el CVRP, tomando múltiples instancias del algoritmo que independientemente obtienen alguna solución. De entre las soluciones generadas se toma la mejor como inicial para todos las instancias del algoritmo. También es usado en el VRPTW [80]. En [117] se usa la búsqueda tabú para resolver las rutas de cada día en el VRP consistente y luego para mejorar la solución tomando en cuenta todos los días.

La búsqueda local iterativa también se ha usado en [94] para el VRPB. También se puede usar una versión híbrida como en [23], además de definir múltiples operadores para realizar una modificación en las soluciones. En [111] realizan múltiples veces la búsqueda local por iteración para el CVRP. Otra variante de la búsqueda local como la búsqueda de vecindario variable se han propuesto para el VRPTW [10] donde los vecindarios los toma como los clientes más cercanos al actual cumpliendo las ventanas de tiempo. Otra aplicación se encuentra en [71] donde cambian entre vecindarios usando algunos operadores como la inserción de clientes, mezcla de rutas, intercambio de clientes y arcos.

Metaheurísticas basadas en población

Comenzando con los algoritmos genéticos, en [9] para el VRPTW proponen utilizar operadores de cruza y mutación dentro del recocido simulado. Tomando como operadores de mutación a las diferentes formas de insertar un cliente en otra ruta y como operador de cruza el alternar entre las rutas de cada solución padre. En [1] para el DVRP utilizan un algoritmo genético para optimizar cada VRP estático junto a un mecanismo de búsqueda local. En [48] proponen distintas formas de implementar los algoritmos genéticos para la clase de problemas en VRPB. En [127] combinan un algoritmo genético con la búsqueda local para el problema de HFVRPTW.

ACO se adapta bastante bien a los problemas que involucran construir una solución mediante

gráficas. Para problemas de VRP en la mayoría las hormigas recorren los clientes dada alguna probabilidad en términos de los objetivos. En [102] dividen el problema de CVRP en subproblemas, de forma que a cada hormiga le corresponde resolver un problema de TSP además de utilizar búsqueda local. También se han combinado estrategias de ACO con GA, como en [122] para el PVRPTW, donde después de que cada hormiga construya una solución se aplican operadores de cruza en forma secuencial y con las mejores soluciones se actualiza la matriz de feromonas.

Muchos algoritmos de ACO usan búsqueda local dentro de la creación de soluciones como en [35] donde tratan un subproblema de DVRP. Aquí las hormigas construyen de una forma dinámica, cada vez que agregan a un cliente realizan una optimización con búsqueda local. Otro uso de la búsqueda local y ACO se muestra en [115] donde generan soluciones para problemas grandes con ACO y luego mejoran cada subproblema con búsqueda local.

En [69] para el VRP con demanda estocástica agregan operadores de cruza y mutación a PSO, creando soluciones a partir de cruzar cada solución con la mejor hasta el momento. En [1] muestran una aplicación de un PSO híbrido para el VRPMD. En [22] combinan PSO con recocido simulado para una variante del CVRP. En [84] comparan distintas variantes de PSO para el VRPSD, la diferencia entre variantes es sobre el cálculo de velocidad de las partículas. En [87] proponen una forma en codificar una instancia del VRPSDC que mejora a PSO comparado con versiones básicas.

2.3.4. Problema de enrutamiento generalizado consistente multi-objetivo

El problema a tratar en este trabajo es el problema de enrutamiento generalizado consistente multi-objetivo (MOGenConVRP, por sus siglas en inglés[24]) [66], que trata de obtener una planeación de las rutas que los vehículos deben tomar durante un horizonte de planeación. Donde cada cliente tiene demanda por algún servicio y puede ser atendido dentro de algunos días y en horarios específicos dentro del horizonte de planeación.

Para este problema se trata de minimizar tres objetivos: el tiempo total de todas las rutas que toman los vehículos, la máxima diferencia entre los tiempos de llegada de los clientes y el máximo número de vehículos distintos que visitan un cliente.

Sea G una gráfica dirigida $G = (N^0, A)$, donde $N^0 = \{0, 1, .., n\}$ es el conjunto de clientes, excepto el nodo 0 que representa el depósito y N es el conjunto $N^0 - 0$. Los clientes se dividen en dos grupos, clientes en AM (N^{am}) que solo pueden ser visitados en la mañana y los clientes en PM que solo pueden ser visitados en la tarde con (N^{pm}). También se tiene un horizonte de planeación D de $|D|$ días. El conjunto $N^f \subseteq N$ contiene los clientes que requieren al menos dos visitas en el horizonte de planeación.

En cuanto a los arcos de la gráfica, se define $A = \{(i, j) : i, j \in N^0, i \neq j\}$. Como observación, no existen arcos entre clientes en AM a clientes en PM y viceversa. Las aristas tienen pesos, que representan el tiempo de ir del cliente i al j. También existe un conjunto de vehículos $K = \{0, 1, .., v\}$, donde cada vehículo tiene una capacidad q, además de un límite de tiempo y tiene que visitar a los clientes solo una vez por día, además de iniciar y terminar la ruta en el depósito.

Para cada día $d \in D$, cada cliente $i \in N$, tiene una demanda q_{id} y un tiempo de servicio s_{id}. También se tienen conjuntos $N^0_d \subseteq N^0$ sobre los clientes que requieren servicio en el día d. Análogamente, los subconjuntos A^0_d que contiene arcos entre los clientes en N^0_d. Se tienen variables binarias como w_{id}, que es 1 si el cliente i requiere servicio en el día $d(q_{id} > 0)$, de otra forma es 0. Si el vehículo k toma el arco $(i, j) \in A^0_d$ en el día d se denota como x_{ijkd}=1. Para denotar en

[24] Multi-objective Generalized Consistent Vehicle Routing Problem

específico un cliente i visitado por un cliente se usa y_{ikd}=1. Si el cliente i es visitado por el vehículo k al menos una vez en el horizonte de planeación se asigna la variable z_{ik}=1.

El tiempo de llegada del cliente $i \in N_d$ en el día d es a_{id}. El máximo número de distintos vehículos que visitan a algún cliente es z_{max}. Y l_{max} es la diferencia máxima entre el menor tiempo y el mayor tiempo de llegada de cada cliente sobre todos los clientes. En la Tabla 2.2 se resumen todos los términos usados en la descripción del problema.

El objetivo es minimizar un vector de funciones objetivo en conflicto (2.34a), donde las funciones objetivo son; Minimizar el tiempo total de las rutas de los vehículos (2.34b). Minimizar el máximo número de distintos vehículos que visitan a algún cliente (2.34c) y minimizar el máximo de la diferencia máxima entre los tiempos de llegada de cada cliente (2.34d).

$$\text{mín } F = (f_1, f_2, f_3), \tag{2.34a}$$

$$f_1 = \sum_{d \in D} \sum_{k \in K} \sum_{(i,j) \in A_d^0} t_{ij} x_{ijkd}, \tag{2.34b}$$

$$f_2 = z_{max}, \tag{2.34c}$$

$$f_3 = l_{max}, \tag{2.34d}$$

sujeto a:

$$y_{okd} = 1 \ \forall k \in K, d \in D \tag{2.34e}$$

$$\sum_{k \in K} y_{ikd} = 1 \ \forall i \in N_d, d \in D \tag{2.34f}$$

$$\sum_{i \in N_d} q_{id} y_{ikd} \leq Q \ \forall k \in K, d \in D \tag{2.34g}$$

$$\sum_{i \in N_d^0} x_{ijkd} = y_{jkd} \ \forall j \in N_d^0, k \in K, d \in D \tag{2.34h}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) - (1 - x_{ijkd})T \leq a_{jd} \ \forall (i,j) \in A_d, k \in K, d \in D \tag{2.34i}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) + (1 - x_{ijkd})T \geq a_{jd} \ \forall (i,j) \in A_d, k \in K, d \in D \tag{2.34j}$$

$$z_{ik} \geq y_{ikd} \ \forall i \in N_d^f, k \in K, d \in D \tag{2.34k}$$

$$\sum_{k \in K} z_{ik} \leq z_{max} \ \forall i \in N^f \tag{2.34l}$$

$$(\alpha_{i\alpha} - \alpha_{i\beta}) w_{i\alpha} w_{i\beta} \leq l_{max} \forall i \in N^f, \alpha, \beta \in D \tag{2.34m}$$

$$a_{id} \in [t_{0i}, min(\frac{T}{2}, T - t_{i0} - s_{id})] \forall i \in N_d^{qm}, d \in D \tag{2.34n}$$

$$a_{id} \in [max(t_{0i}, \frac{T}{2}), T - t_{i0} - s_{id}] \forall i \in N_d^{pm}, d \in D \tag{2.34ñ}$$

$$x_{ijkd} \in \{0,1\},\ \forall (i,j) \in A_d^0, k \in K, d \in D \tag{2.34o}$$

$$y_{ikd} \in \{0,1\},\ \forall i \in N_d^0, k \in K, d \in D \tag{2.34p}$$

$$z_{ik} \in \{0,1\},\ \forall i \in N^f, k \in K \tag{2.34q}$$

Restricción (2.34e) asegura que cada ruta inicia en el depósito. Restricción (2.34f) garantiza que cada cliente es visitado en cada día que requiera servicio. La Desigualdad (2.34g) limita la cantidad de demanda que puede manejar el vehículo y que no exceda su capacidad. Igualdades (2.34h) asegura que si un vehículo recorre un arco (i, j) entonces se marca como visitado el cliente j. Desigualdades (2.34i) y (2.34j) aseguran un orden entre los tiempos de llegada de visitar el cliente i al j. Restricción (2.34k) asegura que los clientes sean visitados al menos una vez. (2.34l) Indica el máximo número de distintos vehículos por cliente. Restricción (2.34m) define la diferencia máxima de entre los tiempos de llegada de acuerdo con el horizonte de planeación. Restricción (2.34n) define las cotas superior e inferior para los tiempos de llegada en los clientes de horario AM. Restricción (2.34ñ) define las cotas superior e inferior para los tiempos de llegada en los clientes de horario PM. Restricciones (2.34o-2.34q) define como variables booleanas a las variables de decisión relacionadas a visitar un arco, visitar un cliente y visitar múltiples veces.

Término	**Dominio**	**Descripción**
t_{ij}	$\mathbb{R}$	Distancia del cliente i al cliente j
x_{ijkd}	$\{0,1\}$	1 si el vehículo k recorre en el día d el arco (i,j)
q_i	$\mathbb{R}^+$	Demanda del cliente i
s_{id}	$\mathbb{R}^+$	Tiempo en el que requiere servicio el cliente i en el día d
w_{id}	$\{0,1\}$	1 si el cliente i requiere servicio en el día d
y_{ikd}	$\{0,1\}$	1 si el cliente i es visitado por el vehículo k en el día d
z_{ik}	$\{0,1\}$	1 si el vehículo k ha visitado más de una vez el cliente i
a_{id}	$\mathbb{R}^+$	Tiempo de llegada al cliente i en el día d
z_{max}	$\mathbb{R}^+$	Máximo número de distintos vehículos que visitan un cliente
l_{max}	$\mathbb{R}^+$	Diferencia máxima entre los tiempos de llega de entre los clientes
T	$\mathbb{R}^+$	Máximo tiempo que puede circular un vehículo en cada día

Tabla 2.2: Términos del MOGenConVRP.

2.3.5. Trabajo relacionado

Se presentan los trabajos más recientes relacionados a VRP multi-objetivo bajo incertidumbre. Comenzando con Mahdieh et al. [4], donde proponen tres problemas de VRP multi-objetivo bajo incertidumbre que resuelve con un algoritmo genético en paralelo. En el primer problema lo definen

como el VRP difuso selectivo con tiempos de ventana (FSVRPTW, por sus siglas en inglés[25]) que tiene dos funciones objetivo; el primero es el tiempo de los arcos y el segundo la importancia de visitar todos los clientes y esta importancia está dada por números difusos. El segundo problema es el VRP fiable selectivo (RSVRP) que también minimiza el tiempo de los arcos y el segundo objetivo es maximizar la utilidad de visitar a cada nodo y esta utilidad está dada por una distribución de probabilidad normal. El tercer problema es el VRP robusto selectivo (MVSVRP) donde define escenarios por cada cliente y con ellos una probabilidad de que el escenario dado afecte la utilidad de visitar al cliente actual.

Otro trabajo que trata la incertidumbre de las demandas es propuesto por Carmona et al. [91], donde la demanda está dada por números difusos. Proponen usar algoritmo genético elitista donde las soluciones con alguna probabilidad son mejoradas con búsqueda local. También se puede manejar la incertidumbre dentro de las restricciones como lo hacen Rahbari et al. [100]. Donde toman un valor nominal que define un intervalo en el que se encuentran los tiempos de cada viaje y también definen una función objetiva robusta que toma en cuenta los posibles escenarios.

En el trabajo de Bernardo [96], se usa robustez Min-Max basada en conjuntos para el VRP con clientes y demandas estocásticos (SSCVRPSD). Generan distintos escenarios con distintas distribuciones de probabilidad para los clientes y demandas. Una vez que obtienen el escenario más pesimista realizan una optimización con recocido simulado. En el trabajo de Shen et al. [108] se optimiza el VRP eléctrico con ventana de tiempo bajo incertidumbre en las demandas. Para aplicar optimización robusta, definen un valor de demanda nominal sobre el cual calculan un costo robusto basado en penalizaciones para la solución. Con este nuevo costo realizan una optimización de búsqueda local y prueban esto para múltiples valores nominales de demandas.

Otro trabajo donde tratan robustez para el peor escenario es propuesto por Santos et al. [104], donde optimizan el VRPB teniendo como incertidumbre la ganancia de recoger artículos de los clientes. Los escenarios se generan a partir de una distribución de probabilidad y el algoritmo principal se basa en búsqueda local. Lu y Gzara [78] comparan las soluciones robustas en VRPTW contra el mismo problema sin incertidumbre, ambos con incertidumbre en las demandas. Utilizan un algoritmo de ramificación y poda, simulando múltiples escenarios generados. En el trabajo de Hu et al. [60] se busca optimizar el tiempo y el número de vehículos donde la incertidumbre está en los tiempos y demandas. En ambos se tienen distintos escenarios que afectan las restricciones del problema. Se optimiza con búsqueda local variable adaptativa en orden lexicográfico.

[25] Fuzzy Selective Vehicle Routing Problem with Time Windows

Capítulo 3

Colonia de hormigas basada en múltiples indicadores

En este capítulo se presenta la propuesta principal de este proyecto, que es la cooperación de múltiples indicadores mediante colonia de hormigas para el MOGenConVRP bajo incertidumbre. En la Sección 3.1, se muestra la definición de los componentes necesarios para realizar optimización mediante colonia de hormigas para MOGenConVRP tales como la construcción de soluciones, reglas de probabilidad, información heurística, actualización de feromona y búsqueda local. En la Sección 3.2, se presenta el algoritmo principal aplicado a soluciones aproximadas en MOGenConVRP que trata la cooperación de múltiples colonias basadas en indicadores de calidad y un algoritmo base que trata la suma ponderada de indicadores. En la Sección 3.3, se describe el uso de este algoritmo para la optimización ligeramente robusta dentro del problema de interés.

3.1. Componentes básicos de colonia de hormigas

Se presentan los componentes básicos para resolver el MOGenConVRP usando colonia de hormigas multi-objetivo. De acuerdo con el MOGenConVRP, éste se compone principalmente de: las rutas para cada horario en cada día y la asignación correcta de los vehículos a clientes sujeto a las restricciones del problema. Las hormigas artificiales construyen estas soluciones mediante los recorridos sobre la gráfica del problema. Visitando los clientes de acuerdo con las reglas de probabilidad y agregándolos a la ruta actual en un vehículo dado. De acuerdo con la definición del MOGenConVRP, en cada día y horario se tiene un problema clásico de enrutamiento de vehículos, por lo que es necesario implementar una matriz de feromonas para cada día y horario posible. A continuación se describe la información heurística de todos los objetivos, la regla de probabilidad que siguen las hormigas para construir las soluciones, la forma de actualizar la matriz de feromonas y los operadores genéticos de cruza y mutación.

3.1.1. Información heurística

Primero se debe calcular la información heurística para cada uno de los tres objetivos en MOGenConVRP debido a que es necesaria al momento de calcular las probabilidades que debe seguir

la hormiga para tomar algún camino.

El valor η_{ij} es el inverso de la distancia euclidiana entre dos clientes, si la distancia d_{ij} entre i y j es corta, se vuelve un arco con mayor potencial a ser elegido, se calcula como sigue:

$$\eta_{ij} = \frac{1}{d_{ij}}. \tag{3.1}$$

El valor ψ_{ij} es la información heurística sobre el objetivo de la máxima diferencia entre los tiempos de llegada y se calcula como sigue:

$$\psi_{ij} = \frac{1}{\text{máx}(1, wait_{ij})}, \tag{3.2}$$

donde el valor $wait_{ij}$ se calcula como:

$$wait_{ij} = \text{máx}(\{\alpha_{j_d} - estimated_{ij} : \ d \in D, \alpha_{j_d} > 0\}). \tag{3.3}$$

El valor de $wait_{ij}$ es la diferencia más grande entre los tiempos de llegada a j en caso de tomar la decisión de ir de i a j en el día actual $d^* \in D$. La variable $estimated_{ij}$ en la Ecuación (3.4) es el tiempo de llegada a j en caso de tomar el arco (i, j). Cuanto más grande sea el valor de $wait_{ij}$, significa que tomar el arco (i, j) incrementaría el tiempo de llegada a j, como consecuencia ψ_{ij} es pequeño y no sería conveniente agregar al cliente j al recorrido.

$$estimated_{ij} = \alpha_{id^*} + s_i + t_{ij}. \tag{3.4}$$

El valor ϕ_j es la información heurística sobre el objetivo de la máxima cantidad de distintos vehículos que visitan a un cliente y se calcula de acuerdo con la Ecuación (3.5) con $z_{jk} = 1$ si el vehículo k visita a j. Por lo que este valor es más bajo cuando muchos vehículos distintos han visitado el cliente j.

$$\phi_j = \frac{1}{\sum_{k \in K} z_{jk}}. \tag{3.5}$$

3.1.2. Probabilidad de tomar un arco

La probabilidad de la hormiga para avanzar desde el cliente i al cliente j depende de si se está realizando explotación o exploración. Para esto se tiene un parámetro $0 < q_0 < 1$ que indica la probabilidad de explotación, por lo que dado un cliente i, el siguiente cliente j es escogido de acuerdo con la Ecuación (3.7). Donde τ_{ijdh} es la feromona del arco (i, j) en la matriz M_{dh} del día d y horario $h \in \{\text{AM,PM}\}$. En cuanto al conjunto $N(i)$, son los clientes l que son vecinos de i que no han sido visitados y que aún pueden ser visitados por el vehículo actual. Las constantes $\alpha, \beta, \gamma, \delta$ indican la relevancia de cada información heurística descritas anteriormente.

$$p_{ij} = \frac{\tau_{ijdh}^{\alpha} \eta_{ij}^{\beta} \psi_{ij}^{\gamma} \phi_j^{\delta}}{\sum_{l \in N(i)} \tau_{ildh}^{\alpha} \eta_{il}^{\beta} \psi_{il}^{\gamma} \phi_l^{\delta}}, \tag{3.6}$$

$$j = \begin{cases} argmax\{\tau_{ijdh}^{\alpha} \eta_{ij}^{\beta} \psi_{ij}^{\gamma} \phi_j^{\delta}\} & q \leq q_0 \\ \text{con probabilidad } p_{ij} & q > q_0 \end{cases}. \tag{3.7}$$

3.1.3. Construcción de soluciones

Ya obtenida la información heurística de cada objetivo y la probabilidad de tomar un arco para cada hormiga, se define la forma en que se construyen soluciones. Dados los horarios AM, PM en que los clientes pueden requerir visitas un número dado de días, una hormiga se encarga de construir las rutas de cada día en cada horario para agregarlo a la planeación. En el Algoritmo 10 se presenta la forma para crear específicamente cada ruta en un determinado día y horario. Se inicializa la hormiga en el depósito con un nuevo vehículo (líneas 2-5). De acuerdo con las probabilidades y la información heurística, toma el siguiente cliente para insertarlo a la ruta siempre que no se haya excedido el tiempo T ni la capacidad actual del vehículo (línea 10). Actualiza las capacidades del vehículo (línea 11), el tiempo de llegada al nuevo cliente (línea 12) y el tiempo actual (línea 13). En caso de no poder agregar un nuevo cliente, el vehículo regresa al depósito y guarda la ruta (líneas 16-18). Todo esto se realiza mientras existan clientes que en el día y horario actual requieran visitas.

Algoritmo 10 Build subsolution time h day d

Input: timetables {AM, PM}, days $d \in \mathbb{Z}$, pheromone constant $\alpha > 0 \in \mathbb{R}$, time travel constant $\beta > 0 \in \mathbb{R}$, driver consistency constant $\gamma > 0 \in \mathbb{R}$, arrival constant $\delta > 0 \in \mathbb{R}$, explotation probability $q_0 > 0 \in \mathbb{R}$, total time $T \in \mathbb{R}$, Pheromone matrix for day d and time h M_{hd}
Output: Ant solution k for day d, time h

1: Initialize ant k
2: Initialize tour in depot 0
3: $i \leftarrow depot$
4: Take vehicle $v \in K$
5: $T_c \leftarrow 0$
6: **while** There are clients without visit in day d time t **do**
7: **if** There is a client to visit from i without exceeding vehicle capacity **and**
8: not reached limit time T for v **and** $random(0,1) > \frac{T_c}{T}$ **then**
9: *Get* next client j with probability p_{ij} and explotation probability q_0
10: Update vehicle capacity
11: $a_{jd} \leftarrow a_{id} + s_{id} + t_{ij}$ //update arrival time j
12: Update current time T_c
13: $i \leftarrow j$
14: **else** //cannot visit another client
15: Return vehicle v to depot
16: Save created tour in v
17: Take another vehicle
18: **end if**
19: **end while**
20: Save all vehicles and created tours in day d, time h
21: **return** k

3.1.4. Actualización de feromonas

La actualización de feromonas consiste en utilizar a los indicadores de calidad para asignar las cantidades de feromonas en los arcos que recorren las hormigas. Lo primero es obtener la calidad de

la solución mediante la función de aptitud para cada indicador $I \in \mathcal{I}=\{HV, R2, \epsilon^+\}$ que se calcula con la función Fit_I presentada en la Ecuación (3.8) con una constante $k > 0$.

$$Fit_I(x_1) = \sum_{x_2 \in P \setminus \{x_1\}} -e^{-I(x_2,x_1)/k}. \tag{3.8}$$

Una vez calculada la aptitud para cada indicador, se calcula la suma ponderada de éstos con un conjunto de pesos $W=\{w_{HV}, w_{R2}, w_{\epsilon^+}\}$ como se muestra en la Ecuación (3.9).

$$Fit(s) = \sum_{I \in \mathcal{I}} w_I \cdot Fit_I(s). \tag{3.9}$$

Las hormigas guardan la calidad de la solución que previamente calcularon para cada arco (i, j) en cada dia d y horario h como se muestra en la Ecuación (3.10).

$$\Delta_{dhij}(p) = \begin{cases} Fit(p) & (i,j) \in p \\ 0 & (i,j) \notin p \end{cases}. \tag{3.10}$$

Y dada una constante de evaporación $\rho > 0$, se actualiza cada entrada (i, j) para cada matriz M_{dh} con la feromona de todas las hormigas como se muestra en la Ecuación (3.11).

$$M_{dh}(ij) \leftarrow M_{dh}(ij) \cdot (1 - \rho) + \sum_{p \in P} \Delta_{dhij}(p). \tag{3.11}$$

En el Algoritmo 11 se muestra la actualización de feromonas dada una población de soluciones P, un conjunto de matrices de feromona M y un horizonte de planeación D.

Algoritmo 11 Update pheromone

Input: $1 > \rho > 0$, Set of pheromone matrices for each day and timetable M, Solutions set P, Days of planing horizon D
Output: Pheromone matrices updated for each day and timetable

```
1: H = {AM,PM}
2: for h in H do
3:     for d in D do
4:         M_dh(ij) ← M_dh(ij) · (1 − ρ) + Σ_{p∈P} Δ_dhij(p)
5:     end for
6: end for
7: return M
```

3.1.5. Operador de cruza

Se presenta el operador de cruza usado en la propuesta de este trabajo, el objetivo de agregar este operador es para diversificar las soluciones. Una vez que las hormigas han construido las soluciones en la iteración actual, este conjunto de soluciones se toma como conjunto de soluciones padres. Para seleccionar cada par de soluciones a cruzar se utiliza un mecanismo de torneo de acuerdo con su valor $Fit(s)$ de su respectivo indicador. Dadas dos soluciones s y s', para generar una solución

hija se toman todas las rutas del primer día en ambos horarios de la solución s y se agregan como rutas de la nueva solución c_1 en el mismo día y horario. Luego se toman las rutas del segundo día en ambos horarios de la solución s' y se agregan a las rutas en c_1 respetando el horario y día. En la siguiente iteración se toman las rutas del tercer día de la solución s y se repite el proceso, alternando entre las rutas de s y s'. Para generar la otra solución hija c_2, se debe realizar el mismo proceso pero iniciando ahora con las rutas de la solución s'. En las Figuras 3.1 y 3.2 se muestran dos soluciones padres y sus dos soluciones hijas en las Figuras 3.3 y 3.4 para un problema con dos días de planeación en los horarios AM y PM. Donde la solución hija en la Figura 3.3 tiene en el primer día las dos planeaciones en AM y PM de la primer solución padre (Figura 3.1) y en el segundo día a la planeación de la segunda solución padre (Figura 3.2). Se puede notar que para la segunda solución hija (Figura 3.4) su planeación en el primer día es la del segundo padre y en el segundo día su planeación es la del primer padre.

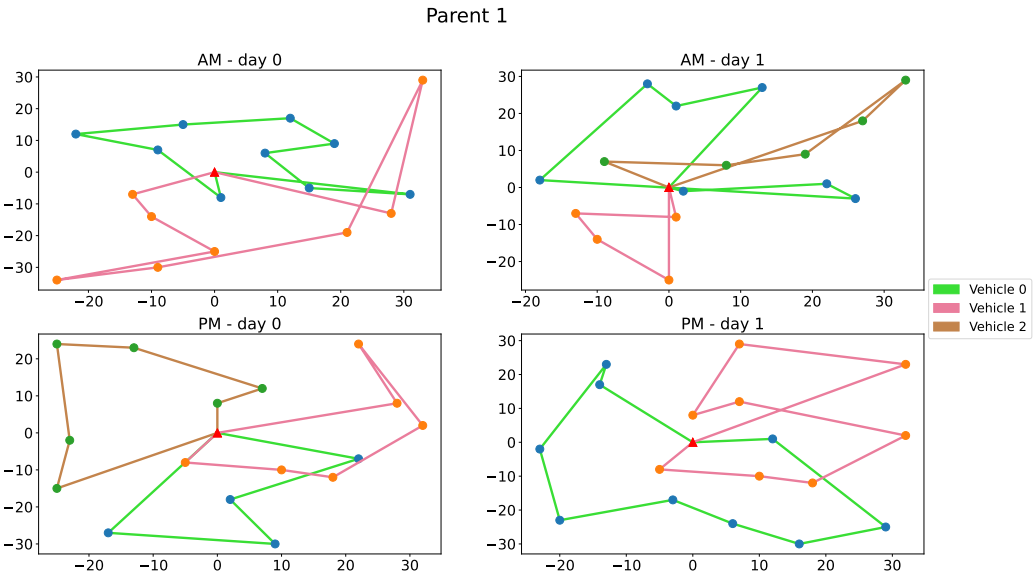


Figura 3.1: Ejemplo del operador de cruza (primer solución padre).

3.1.6. Operador de mutación

Una mutación para una solución del MOGenConVRP es realizando intercambios en las posiciones de los clientes sobre las rutas de un determinado día y horario dada una probabilidad p_m y para esto se tienen tres operadores de mutación:

- Intercambio de posiciones: Toma un determinado número de clientes en la ruta e intercambia aleatoriamente sus posiciones en el recorrido actual.
- Desplazamiento cíclico: Toma aleatoriamente dos posiciones en el recorrido que no sean el depósito, todos los clientes entre ese par de clientes son desplazados un lugar a la derecha, el

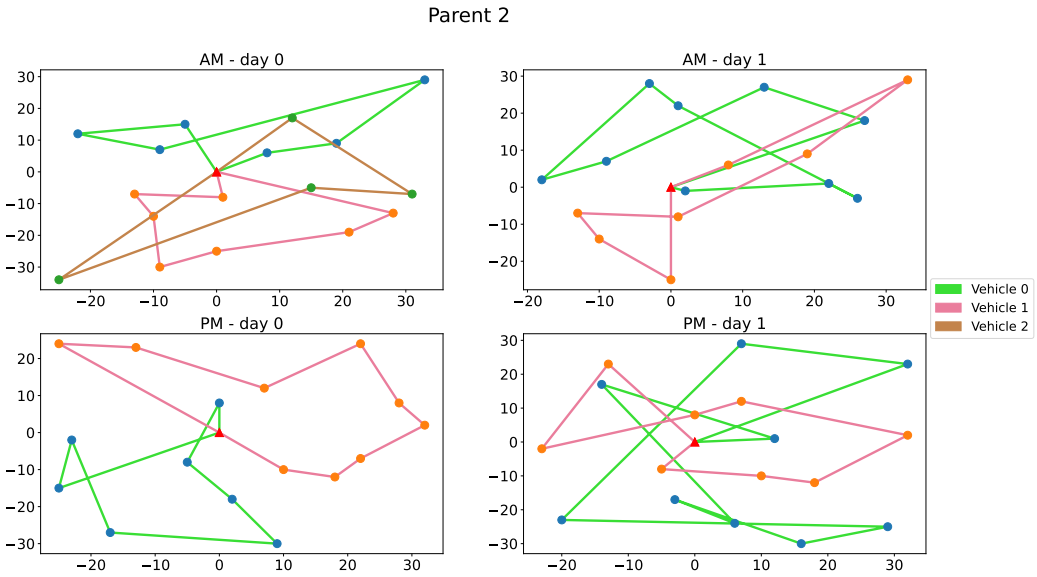


Figura 3.2: Ejemplo del operador de cruza (segunda solución padre).

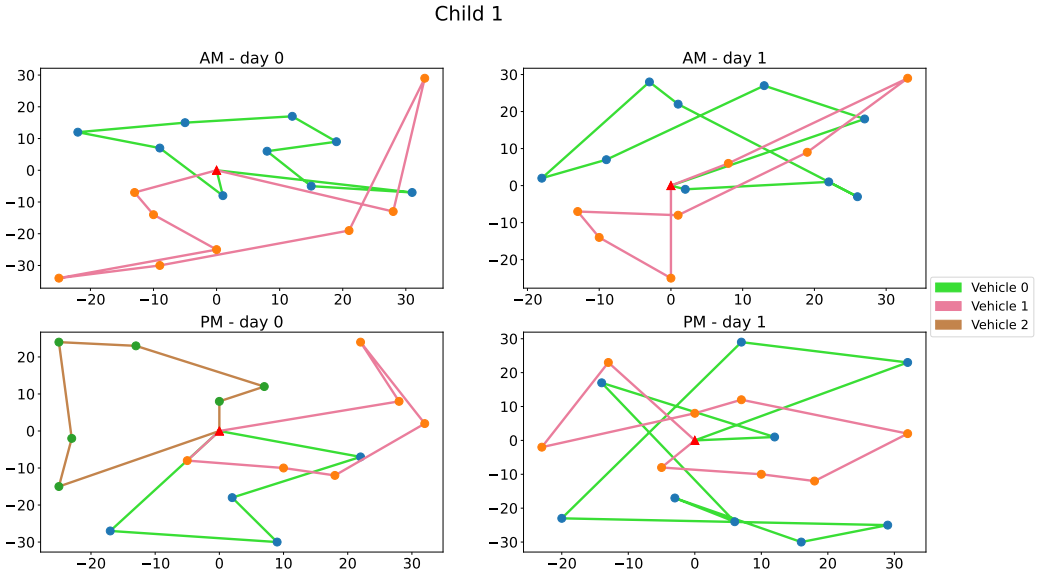


Figura 3.3: Ejemplo del operador de cruza (primer solución hija).

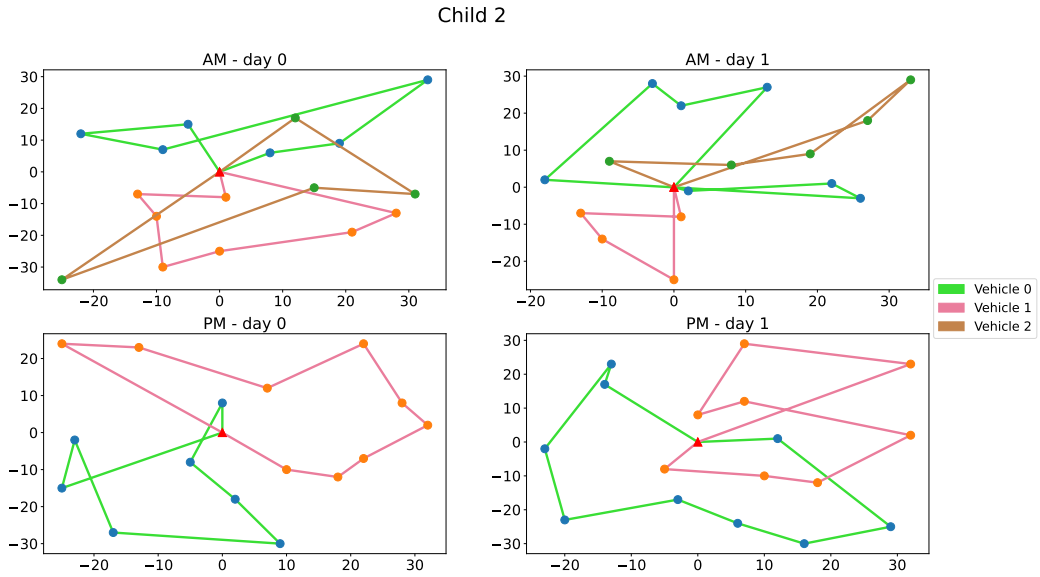


Figura 3.4: Ejemplo del operador de cruza (segunda solución hija).

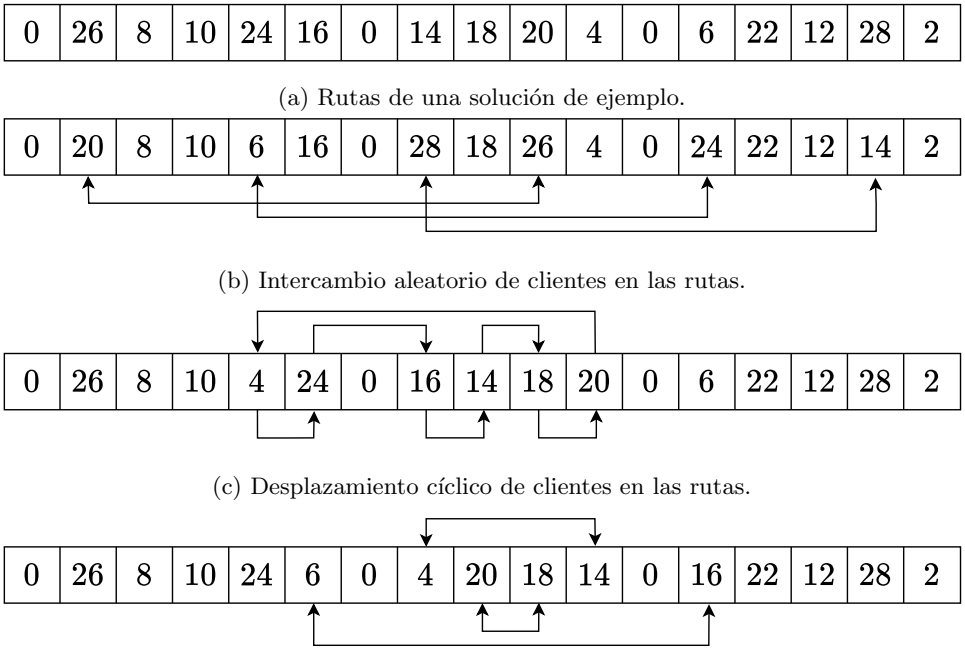
último de ellos se mueve al principio de la subsecuencia seleccionada.

- Inversa de la secuencia: Selecciona dos puntos del recorrido y a los clientes de esta subsecuencia se cambian a una posición invertida. Es decir; al último cliente en la subsecuencia se intercambia con el cliente en la primera posición de la subsecuencia, el penúltimo cliente se intercambia con el segundo elemento de la subsecuencia y así en adelante.

En la Figura 3.5 se muestran ejemplos de los distintos tipos de mutación para un vector de rutas en la Figura 3.5a. Por ejemplo, en la Figura 3.5b se intercambian de posición los clientes 20 y 26, los clientes 6 y 24 y los clientes 14 y 28. En la Figura 3.5c se mueven a la derecha los clientes que van del 24 al 4, dejando al 4 en la primera posición a la izquierda. En la Figura 3.5d se toma la inversa de la secuencia, cambiando al 16 con 6, el 4 con 14 y el 20 con 18.

3.1.7. Búsqueda local

Como parte de la aplicación de colonia de hormigas para el MOGenConVRP, es necesario implementar un mecanismo de búsqueda local a la población construida en cada iteración dentro de cada ejecución de cada IBACO$_I$. El objetivo de usar la búsqueda local es mejorar las soluciones construidas por las hormigas mediante perturbaciones a la solución actual, tales como aplicar desplazamientos de tiempo a las rutas construidas, eliminación de cruces entre caminos de rutas y el uso de operadores de destrucción y reparación. El tipo de búsqueda local que se implementó es una versión modificada de LNS que proponen en [66] y [65]. Primero se definen todos los componentes básicos que se usan en el algoritmo principal.



(a) Rutas de una solución de ejemplo.

(b) Intercambio aleatorio de clientes en las rutas.

(c) Desplazamiento cíclico de clientes en las rutas.

(d) Inversa de la secuencia.

Figura 3.5: Ejemplos de operadores de mutación para el vector en la Figura 3.5a.

Operador de destrucción

El objetivo del operador de destrucción es quitar de las rutas aquellos clientes que tienen una mayor cantidad de distintos vehículos y una mayor diferencia entre los tiempos de llegada. Para cada cliente i de la solución actual s, define una variable $\mathscr{D}(i)$ que es la suma del número de distintos vehículos z_i del cliente i más la relación entre el tiempo de llegada a i respecto a la diferencia máxima en los tiempos de llegada con una constante $\epsilon > 0$. La variable l_i es el máximo tiempo de llegada para el cliente i y la diferencia máxima entre los tiempos de llegada se denota como l_{max}, por lo que la variable $\mathscr{D}(i)$ se calcula de acuerdo con la Ecuación (3.12). Una vez calculada las variables $\mathscr{D}(i)$, los clientes se ordenan de forma ascendente de acuerdo con esta variable y se eliminan los primeros n clientes de la solución actual.

$$\mathscr{D}(i) = z_i + \frac{l_i}{l_{max} + \epsilon}. \tag{3.12}$$

Operador de reparación

Para agregar los clientes eliminados de nuevo a la solución se debe calcular un costo c_{ik} que es el tiempo incrementado por agregar a un cliente i en el recorrido del vehículo k en la posición que toma un menor incremento en el tiempo. En caso de que no sea factible agregar a un cliente en alguna ruta porque excedería el tiempo límite de las rutas o bien, la capacidad de los vehículos se

excede, entonces su costo se define como infinito. Cuando el costo es infinito, la única manera de agregarlo en la solución es agregándolo en una nueva ruta con un nuevo vehículo. Una vez obtenidos los costos c_{ik} de los clientes por agregar, se actualizan a $c_{ik} + y$ donde $y \in [-\eta, \eta]$ con η un factor para agregar aleatoriedad. Finalmente, en cada iteración se va agregando el i-ésimo cliente con el menor costo de acuerdo con la Ecuación (3.13) hasta que se hayan agregado todos los clientes.

$$i^* = \arg \min_{i \in N \setminus \{0\}, k \in K} \{c_{ik}\}. \tag{3.13}$$

Mejora de la diferencia entre los tiempos de llegada

Una heurística para poder mejorar la diferencia entre los tiempos de llegada para un cliente es que dada una ruta ya construida, se calcule un tiempo de espera para el vehículo antes de iniciar el recorrido. Con esto se puede desplazar los tiempos de llegada de todos los clientes y en consecuencia, se minimiza la diferencia entre los tiempos de llegada. El hecho de agregar un tiempo de espera antes de iniciar el recorrido, significa que se debe encontrar la forma de empujar hacia adelante o hacia atrás todos los tiempos de llegada.

Comenzando sobre cómo empujar hacia adelante, a partir de un cliente j con algún otro cliente k, se revisan los días en que su tiempo de llegada es el menor, sean $d_j^{earliest}$ y $d_k^{earliest}$ respectivamente, se tienen dos casos:

- $d_j^{earliest}$ y $d_k^{earliest}$ son iguales: Para esto se toma el tiempo de llegada $a_k^{current}$ para el cliente k en el día $d_j^{earliest}$ y se toma el tiempo más tardío a_k^{latest} entre los tiempos de llegada para el cliente k y se calcula lo máximo que se puede empujar hacia adelante de acuerdo con la Ecuación (3.14).

$$pf(j,k) = \frac{l_j - l_k + (a_k^{latest} - a_k^{current})}{2}. \tag{3.14}$$

- $d_j^{earliest}$ y $d_k^{earliest}$ son distintos: Se toma el tiempo de llegada $a_k^{current}$ para el cliente k en el día $d_j^{earliest}$ y se toma el segundo menor tiempo de llegada $a_k^{2nd\ earliest}$ para el cliente k y se calcula lo máximo que se puede empujar hacia adelante de acuerdo con la Ecuación (3.15).

$$pf(j,k) = \frac{l_j + a_k^{2nd\ earliest} - a_k^{current}}{2}. \tag{3.15}$$

De lo anterior se obtiene el tiempo máximo $pf(j,k)$ que se puede desplazar hacia adelante el tiempo de llegada de un cliente j respecto a otro cliente k en el día $d_j^{earliest}$, todo se encuentra en el Algoritmo 12. Primero, calcula los tiempos de llegada más temprano para ambos clientes (líneas 1-2) y el tiempo de llegada en el día actual para el cliente k (línea 3). Si el día mas temprano en sus tiempo de llegada es el mismo, entonces calcula el empuje usando la diferencia entre los tiempos de llegada del día actual con el tiempo más largo para k (líneas 4-6). En otro caso, usa la diferencia entre el segundo tiempo de llegada más temprano para k y el tiempo de llegada en el día actual (líneas 8-9).

Ahora, para empujar hacia atrás los tiempos de llegada a partir de un cliente j con algún otro cliente k, se revisan los días en que su tiempo de llegada es el último, sean d_j^{latest} y d_k^{latest} respectivamente, se tienen dos casos:

Algoritmo 12 Maximum push foward

Input: client j to push arrival time, client k to compare
Output: Maximum push foward (pf_{jk}) for j subject to k
1: $d_j^{earliest} \leftarrow dayEarliestArrivalTime(j)$ //Earliest arrival time day of j
2: $d_k^{earliest} \leftarrow dayEarliestArrivalTime(k)$ //Earliest arrival time day of k
3: $a_k^{current} \leftarrow arrivalTime(d_j^{earliest})$ //Arrival time in current day for k
4: **if** $d_j^{earliest} == d_k^{earliest}$ **then**
5: $a_k^{latest} \leftarrow latestArrivalTime(k)$ //Latest arrival time for k
6: $pf_{jk} \leftarrow \frac{l_j - l_k + (a_k^{latest} - a_k^{current})}{2}$
7: **else**
8: $a_k^{2nd\ earliest} \leftarrow secondEarliestArrivalTime(k)$ //Second earliest arrival time for k
9: $pf_{jk} \leftarrow \frac{l_j + a_k^{2nd\ earliest} - a_k^{current}}{2}$
10: **end if**
11: **return** pf_{jk}

- d_j^{latest} y d_k^{latest} son iguales: Se toma el tiempo de llegada $a_k^{current}$ en el día d_j^{latest} para el cliente k, además del segundo tiempo de llegada más temprano $a_k^{2nd\ earliest}$ para k y se calcula lo máximo que se puede empujar hacia atrás de acuerdo con la Ecuación (3.16).

$$pb(j,k) = \frac{l_j - l_k + (a_k^{current} - a_k^{2nd\ earliest})}{2}. \tag{3.16}$$

- d_j^{latest} y d_k^{latest} son distintos: Se toma el tiempo de llegada $a_k^{current}$ en el día d_j^{latest} para el cliente k, además del segundo tiempo de llegada más temprano $a_k^{2nd\ earliest}$ para k y se calcula lo máximo que se puede empujar hacia atrás de acuerdo con la Ecuación (3.17).

$$pb(j,k) = \frac{l_j + (a_k^{current} - a_k^{2nd\ earliest})}{2}. \tag{3.17}$$

De lo anterior se obtiene el tiempo máximo $pb(j,k)$ que se puede desplazar hacia atrás el tiempo de llegada de un cliente j respecto a otro cliente k en el día d_j^{latest}, todo esto se encuentra en el Algoritmo 13. Se obtienen los días con los tiempos de llegada más tarde para los clientes j y k (líneas 1-2), luego el tiempo de llegada en el día actual (línea 3). Si el último día de los tiempos de llegada de los clientes es distinto, entonces calcula el empuje usando la diferencia entre el tiempo de llegada actual para k y su segundo tiempo de llegada más temprano (líneas 5-6). En otro caso, calcula el tiempo de llegada sin considerar el tiempo máximo de k (líneas 8-9).

Una vez definida la forma de obtener los tiempos en que se puede empujar hacia adelante o atrás los tiempos de llegada entre pares de clientes, se puede aplicar esto considerando todos los clientes en la planeación actual. Como se muestra en el Algoritmo 14 para mejorar los tiempos de llegada para una solución s. Lo primero que realiza este algoritmo es inicializar un conjunto BC definido como aquellos clientes tal que su tiempo de llegada debe ser mejorado para reducir l_{max}. Este conjunto toma inicialmente el cliente con la diferencia máxima entre los tiempos de llegada (líneas 6-7). Luego se toma el menor de entre lo máximo que se puede desplazar hacia adelante los

Algoritmo 13 Maximum push back

Input: client j to push arrival time, client k to compare
Output: Maximum push back (pb_{jk}) for j subject to k

1: $d_j^{latest} \leftarrow dayLatestArrivalTime(j)$ //latest arrival time day of j
2: $d_k^{latest} \leftarrow dayLatestArrivalTime(k)$ //latest arrival time day of k
3: $a_k^{current} \leftarrow arrivalTime(d_j^{latest})$ //arrival time in current day for k
4: **if** $d_j^{latest} != d_k^{latest}$ **then**
5: $\quad a_k^{2nd\ earliest} \leftarrow secondEarliestArrivalTime(k)$ //second earliest arrival time for k
6: $\quad pb_{jk} \leftarrow \frac{l_j - l_k + (a_k^{current} - a_k^{2nd\ earliest})}{2}$
7: **else**
8: $\quad a_k^{2nd\ earliest} \leftarrow secondEarliestArrivalTime(k)$ //second earliest arrival time for k
9: $\quad pb_{jk} \leftarrow \frac{l_j + (a_k^{current} - a_k^{2nd\ earliest})}{2}$
10: **end if**
11: **return** pb_{jk}

tiempos de llegada de cada cliente en BC (líneas 9-11) usando el Algoritmo 15. Donde dado un cliente j, toma el día en que es visitado más temprano y para los clientes que pertenecen a la misma ruta en ese día revisa el valor $pf(j,k)$. Tomando el máximo empuje de estos vecinos respecto a j. En caso de no poder desplazar el tiempo por arriba de una constante ϵ pequeña, entonces se agrega ese cliente al conjunto BC. Regresando al algoritmo principal, aplica el desplazamiento máximo hacia adelante que se encontró sobre la ruta donde se encontró este valor (línea 13). Después, se revisa si ahora es necesario desplazar hacía atrás (líneas 17-19), con el objetivo de disminuir el desplazamiento realizado hacia adelante usando el Algoritmo 16 y lo aplica (línea 21). Finalmente, el algoritmo se detiene cuando ya no es posible aplicar un desplazamiento hacia adelante o atrás que sea mayor a la constante ϵ o se haya cumplido un límite de iteraciones.

Para ejemplificar el efecto de la mejora en las diferencias entre los tiempos de llegada. En la Tabla 3.1 se muestran estos valores para una instancia de 18 clientes y 2 días de planeación. Se muestran las diferencias antes de búsqueda local y después de la búsqueda junto a la mejora obtenida que es el tiempo que se logró reducir. Aunque en algunos casos es negativa porque empeora este objetivo después de la búsqueda local. Se puede notar que 14 clientes logran mejorar la diferencia máxima entre sus tiempos de llegada y la mitad de ellos con una mejora significativa.

Mejora del tiempo de los ciclos

Ahora es necesario mejorar los tiempos de las rutas que recorren lo vehículos, para esto se usa el Algoritmo 17. Donde se toma el cliente con la diferencia más grande entre los tiempos de llegada (línea 4) y se toma su menor tiempo de llegada (línea 5), el mayor tiempo de llegada (línea 6) y el promedio de entre sus tiempos de llegada (línea 7). Para luego comparar respecto al promedio si el menor tiempo se encuentra más alejado al promedio o es el mayor tiempo el más alejado. Dependiendo de esto se selecciona el día en que se aplica la heurística de 2-OPT [73] que elimina cruces en los arcos sobre la ruta de este cliente (líneas 8-12). Todo esto se repite hasta que la función objetivo ya no mejore o se cumpla un límite de iteraciones.

En las Figuras B.1a y B.1b se muestra el antes y después respectivamente de aplicar búsqueda

Algoritmo 14 Improve arrival time differences

Input: Solution to improve s
Output: Improvement of arrival time difference for s

$maxPF \leftarrow 0$
$maxPF \leftarrow 0$
$\epsilon \leftarrow 10^{-4}$
while $maxPF > \epsilon \vee maxPB > \epsilon \vee$ **not** reached limit iterations **do**
    $i^{max} \leftarrow taskWithMaxATD(s)$
    $BC \leftarrow \{i^{max}\}$
    $maxPF \leftarrow maxATD(s)$
    for all $j \in BC$ **do**
        $maxPF \leftarrow min\{maxPF, getMaxPF(j, BC, s)\}$
    end for
    if $(maxPF > \epsilon)$ **then**
        $applyPF(maxPF, BC, s)$
    else
        $BC \leftarrow \{i^{max}\}$
        $maxPB \leftarrow maxATD(s)$
        for $j \in BC$ **do**
            $maxPB \leftarrow min\{maxPB, getMaxPB(j, BC, s)\}$
        end for
        if $(maxPB > \epsilon)$ **then**
            $applyPB(maxPB, BC, s)$
        end if
    end if
end while
return s

Algoritmo 15 Get max push front

Input: Current solution s, blocking costumer set BC, current costumer j
Output: Maximum push foward ($maxpf$)

$d_j^{earliest} \leftarrow dayEarliestArrivalTime(j)$
$maxpf \leftarrow 0$
for k **in** $tour(j, d_j^{earliest})$ **do** //check tour on day $d_j^{earliest}$ where j appears
    if $pf(j, k) < \epsilon$ **then**
        $BC.add(k)$
    end if
    $maxpf \leftarrow max(maxpf, pf(j, k))$
end for
return $maxpf$

local a una instancia de MOGenConVRP. Se puede notar que se han eliminado casi todos los cruces y aquellos nodos cercanos entre sí, se encuentran en recorridos similares. La forma en la que se obtuvo esta solución se detalla en el Apéndice B.

Algoritmo 16 Get max push back

Input: Current solution s, blocking costumer set BC, current costumer j
Output: Maximum push back ($maxpb$)
1: $d_j^{latest} \leftarrow dayLatestArrivalTime(j)$
2: $maxpb \leftarrow 0$
3: **for** k **in** $tour(j, d_j^{latest})$ **do** //Check tour on day d_j^{latest} where j appears
4: **if** $pb(j,k) < \epsilon$ **then**
5: $BC.add(k)$
6: **end if**
7: $maxpb \leftarrow max(maxpb, pb(j,k))$
8: **end for**
9: **return** $maxpb$

Cliente	Tiempo de llegada antes de LNS	Tiempo de llegada después de LNS	Mejora obtenida
1	9.19	0.0	9.19
2	17.55	1.64	15.91
3	0.527	0.5	0.027
4	15.59	4.09	11.50
5	0.14	0.0	0.14
6	2.72	4.09	-1.36
7	0.0	0.0	0.0
8	1.006	4.94	-3.93
9	12.19	4.70	7.49
10	4.92	4.52	0.39
11	9.19	0.0	9.19
12	0.0	3.97	-3.97
13	4.34	1.18	3.16
14	12.49	2.64	9.85
15	1.52	0.5	1.02
16	19.33	1.64	17.68
17	0.42	1.03	-0.61
18	5.92	4.52	1.39

Tabla 3.1: Efecto de la búsqueda local en los tiempos de llegada para una instancia de 18 clientes y 2 días de planeación.

Función de costo para MOGenConVRP

Una vez definidos todos los componentes de LNS, esta heurística se presenta en el Algoritmo 18 que dada una solución inicial $s^{initial}$, le aplica operadores de destrucción y reparación (líneas 5-6). De aquí se obtiene una solución distinta s' para la cual aplica una mejora en los tiempos de llegada aplicando desplazamientos en los tiempos de las rutas (línea 7). Después le aplica una mejora a las rutas, eliminando cruces entre los caminos (línea 9). La solución nueva es aceptada como la actual siempre que su costo sea mejor a la actual (líneas 11-13) y se compara con la mejor

Algoritmo 17 Improve time

Input: Current solution s
Output: Improvement of time for s

1: **procedure** IMPROVETIMECONSISTENCY(s)
2: **repeat**
3: $f' \leftarrow f(s)$
4: $i^{max} \leftarrow taskWithMaxArrivalTimeDifference(s)$
5: $eat \leftarrow earliestArrivalTime(i^{max}, s)$
6: $lat \leftarrow latestArrivalTime(i^{max}, s)$
7: $aat \leftarrow averagaArrivalTime(i^{max}, s)$
8: **if** $(aat - eat > lat - aat)$ **then**
9: $d \leftarrow dayEarliestArrival(i^{max}, s)$
10: **else**
11: $d \leftarrow dayLatestArrival(i^{max}, s)$
12: **end if**
13: $apply\text{-}2OPT(i^{max}, d, s)$
14: **until** $f(s) > f' \vee$ reached limit iterations
15: **end procedure**

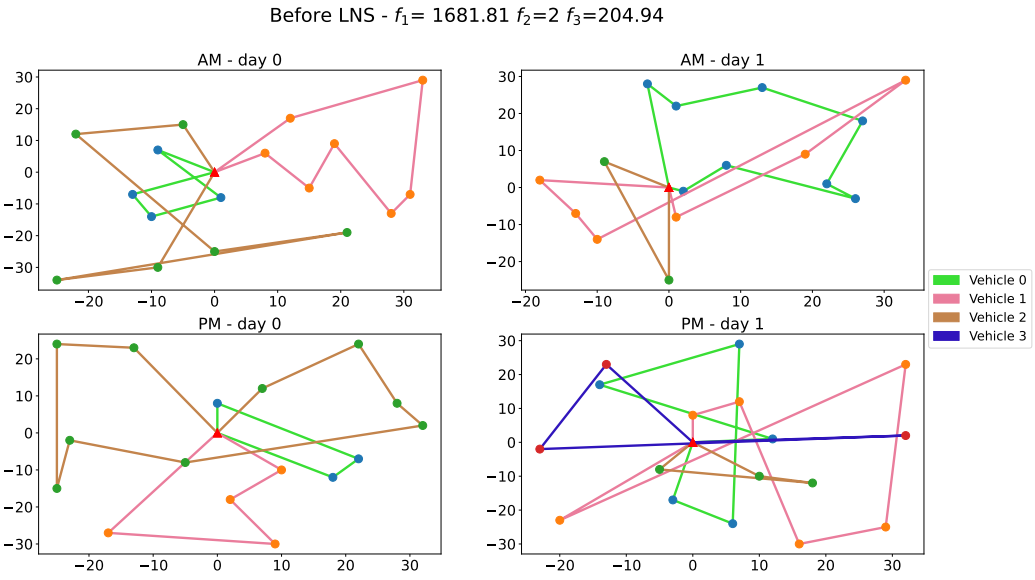


Figura 3.6: Solución construida por una hormiga antes de búsqueda local.

solución encontrada al momento (líneas 14-16). Finalmente regresa la mejor solución encontrada al momento. A continuación se definen el operador de destrucción, el operador de reparación, los métodos que mejoran la diferencia máxima entre los tiempos de llegada y el tiempo de las rutas

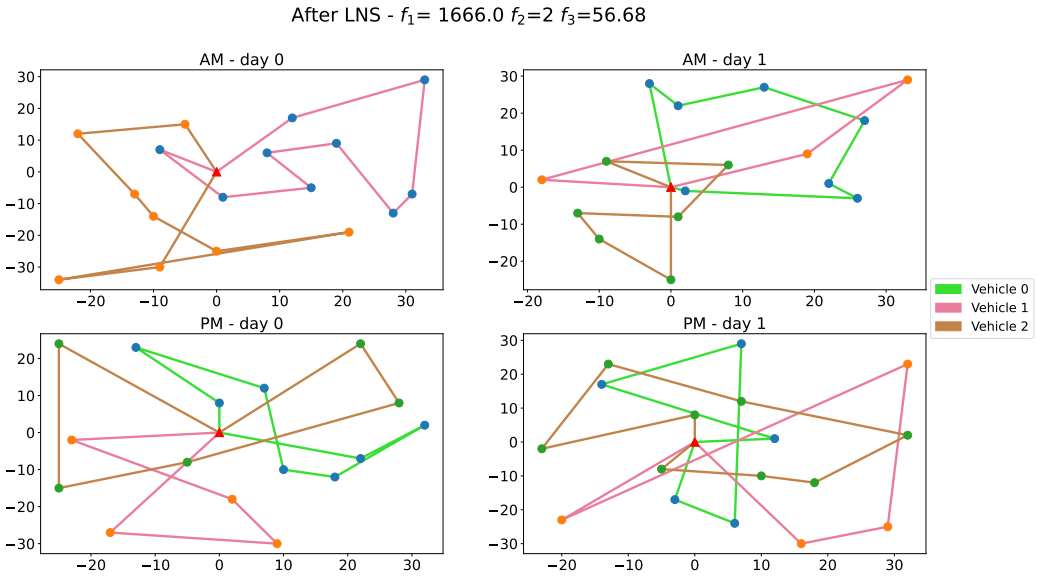


Figura 3.7: Solución después de aplicarle búsqueda local

obtenidas. Como esto minimiza un solo objetivo, más adelante se presenta la forma de usarlo para el caso multi-objetivo.

Anteriormente se detalló el algoritmo de LNS para optimizar un sólo objetivo, para usarlo en el MOGenConVRP es necesario mejorar las soluciones por cada función objetivo en orden lexicográfico. La función a optimizar en cada ejecución de LNS por objetivo va a ser una suma ponderada entre los objetivos como sigue:

$$f'_j = \alpha_j f_1 + (1 - \alpha_j) f_3 \; j = 1, 2, 3. \tag{3.18}$$

Donde los pesos $\alpha_1, \alpha_2, \alpha_3$ para cada objetivo se definen en las Ecuaciones (3.19), (3.20), (3.21) respectivamente.

$$\alpha_1 = \frac{1}{1 + \frac{\delta}{UB_3}}, \tag{3.19}$$

$$\alpha_2 = \frac{\alpha_1 + \alpha_3}{2}, \tag{3.20}$$

$$\alpha_3 = \frac{1}{1 + \frac{UB_1}{\delta}}. \tag{3.21}$$

Tanto UB_1 como UB_3 son cotas superiores para f_1 y f_3 respectivamente. También se tiene la constante $\delta = 10^{-3}$. Ya definidas las funciones objetivo, el Algoritmo 19 mejora un conjunto de soluciones en orden lexicográfico para cada solución de la población.

Algoritmo 18 Large Neighborhood Search (LNS)

Input: Initial solution $s^{initial}$, number of clients to remove $n_{removes}$, maximum number of iterations i_{LNSmax}
Output: Best solution found s^{best}

$s \leftarrow s^{initial}$
$s^{best} \leftarrow s^{initial}$
while not reach termination criteria **do**
    Apply destruction operator d and repair operator r
    $s' \leftarrow$ generateNewSolution$(s, d, r, n_{removes})$
    $adjustDepartureTimes(s')$ // ver Algoritmo 14
    if isDriverFeasible(s') **then**
        $improveTimeConsistency(s')$ // ver Algoritmo 17
    end if
    if $f(s') < f(s)$ **then**
        $s \leftarrow s'$
    end if
    if $f(s') < f(s^{best})$ **then**
        $s^{best} \leftarrow s'$
    end if
end while
return s^{best}

Algoritmo 19 Multi Directional Large Neighborhood Search (MDLNS)

Input: Population of solutions to improve P
Output: Improved population P

for s **in** P **do**
    for all $j \in \{1, 2, 3\}$ **do**
        $f'_j \leftarrow \alpha_j f_1 + (1 - \alpha_j) f_3$
        $LNS(s, f'_j)$
    end for
end for
return P

Tomando como pesos $\alpha_1 = 0.999983, \alpha_2 = 0.499992, \alpha_3 = 0.000020$ obtenidos con $UB_1 = 500$ y $UB_3 = 800$ para la suma ponderada de MDLNS y 20 soluciones usando 200 iteraciones de LNS. En la Figuras 3.8, 3.9, 3.10 se muestran el valor de la suma ponderada f'_1 y los objetivos f_1 y f_2 respectivamente durante la primera iteración de LNS dentro de MDLNS. Se puede notar que mientras se minimiza la función objetivo f'_1, la función objetivo f_1 sobre el tiempo de las rutas se llega a mejorar, esto gracias al valor de α_1 que es cercano a 1 y por tanto, tanto el valor de la suma ponderada f'_1 como f_1 son similares. A comparación de la diferencia entre los tiempos de llegada f_3, que en muchos casos no mejora o termina empeorando. De los gráficos mostrados se puede notar que al mejorar la función objetivo ponderada por iteración se le puede dar mayor prioridad para mejorar algún objetivo específico.

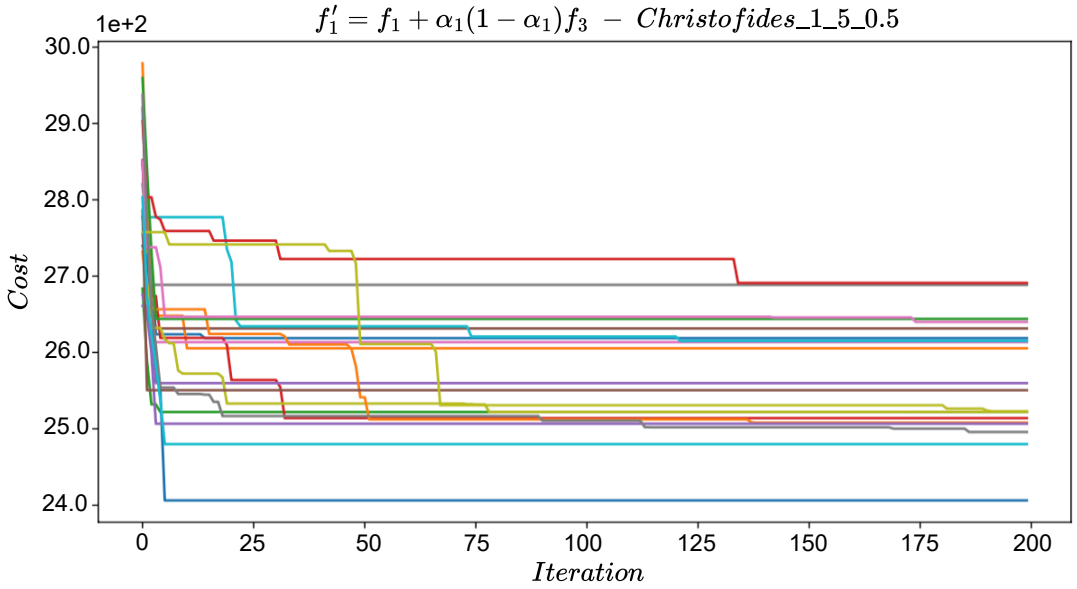


Figura 3.8: Función $f_1' = f_1\alpha_1 + (1-\alpha_1)f_3$ durante la primer iteración de LNS dentro de MDLNS.

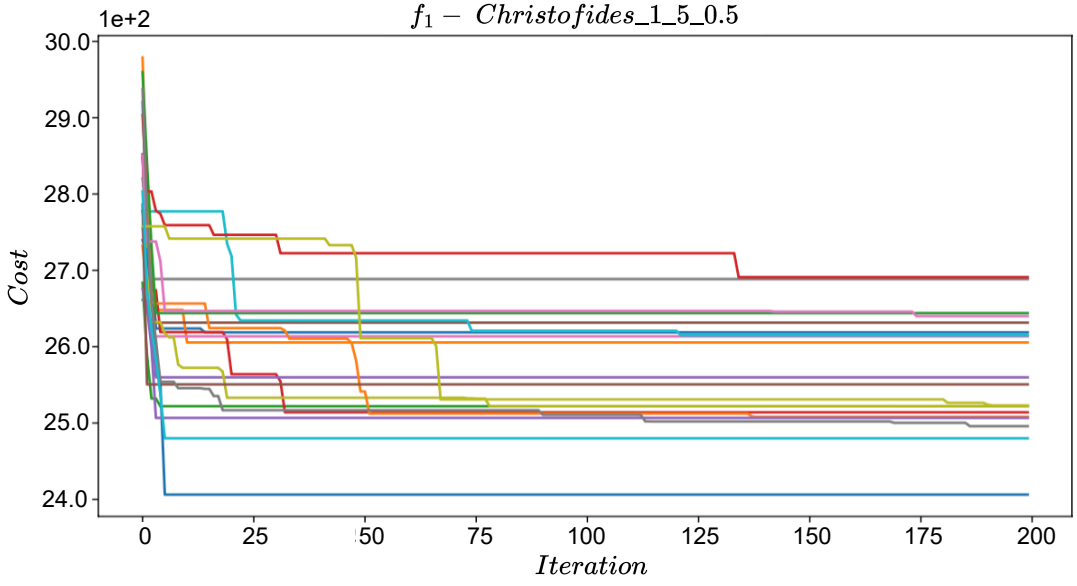


Figura 3.9: Función f_1 durante la primer iteración de LNS dentro de MDLNS.

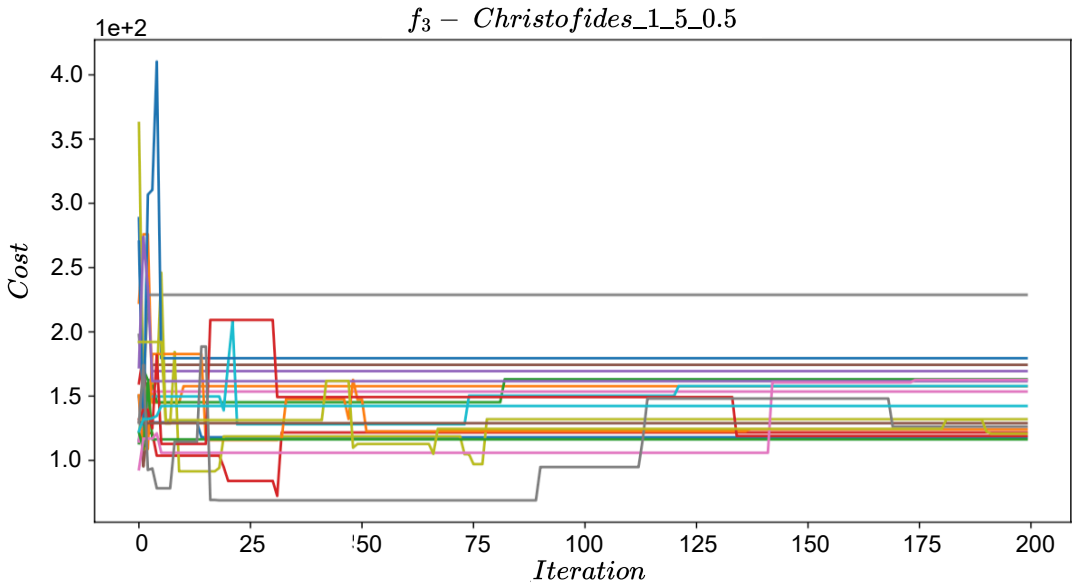


Figura 3.10: Función f_3 durante la primer iteración de LNS dentro de MDLNS.

3.2. Cooperación de múltiples indicadores en colonia de hormigas

En la literatura referente a los algoritmos de colonia de hormigas basados en indicadores desde el primero de estos [83], se han desarrollado pocos de este tipo de algoritmos, en su mayoría emplean de forma individual algunos indicadores como hipervolumen, R2 y ϵ^+. Como parte introductoria de la sección, se presenta una propuesta base que usa la suma ponderada de los indicadores de calidad para evaluar la aptitud de las soluciones. De esta propuesta se derivan los algoritmos basados en un solo indicador que se utilizan más adelante para el algoritmo cooperativo.

3.2.1. IBACO$_{ws}$

Usar algoritmos basados en un solo indicador como IBACO [83] pueden dar buenos resultados para el problema a tratar. Sin embargo, ciertos indicadores pueden dirigir la búsqueda hacía zonas del espacio de los objetivos que eviten realizar una mejor exploración. De acuerdo con [42], combinar la ejecución de distintos indicadores y migrar sus soluciones puede beneficiar la diversidad de soluciones y mejorar la dirección que toman los algoritmos basados en indicadores.

Se presenta el IBACO$_{ws}$ en el Algoritmo 20 que es usado como base para la propuesta principal, este algoritmo es una versión híbrida modificada del presentado por primera vez en [83]. Inicialmente las hormigas construyen p soluciones (línea 3). Con la población actual crea una nueva usando operadores de cruza y mutación (líneas 4 y 5). Aplica búsqueda local a las soluciones (línea 7). Guarda soluciones aproximadas en el archivo externo (línea 8). De entre todas las soluciones generadas, calcula la aptitud de cada individuo de acuerdo con la suma ponderada de los indicadores

y elimina el menos apto de forma iterativa (líneas 9-15). Con las soluciones restantes, actualiza la matriz de feromonas (línea 16).

La relevancia de tener este algoritmo es que una colonia de hormigas puede hacer uso de un indicador o la suma de éstos. Se puede notar que IBACO$_{HV}$, IBACO$_{R2}$ e IBACO$_{\epsilon^+}$ son un caso particular de IBACO$_{ws}$ donde cada indicador tiene peso uno y los demás peso cero. Como se presentó en el algoritmo, múltiples indicadores pueden ayudar a evaluar una solución de acuerdo con las preferencias de cada uno. Por lo que es interesante comparar si es mejor evaluar una solución con varios indicadores o evaluar distintas soluciones de distintos indicadores como se presenta en la siguiente sección.

Algoritmo 20 IBACO$_{ws}$

Input: number of ants $p > 0 \in \mathbb{Z}$, pheromone constant $\alpha > 0 \in \mathbb{R}$, time travel constant $\beta > 0 \in \mathbb{R}$, driver consistency constant $\gamma > 0 \in \mathbb{R}$, arrival constant $\delta > 0 \in \mathbb{R}$, evaporation rate $\rho > 0 \in \mathbb{R}$, crossover probability $p_c > 0 \in \mathbb{R}$, crossover mutation $p_m > 0 \in \mathbb{R}$, explotation probability $q_0 > 0 \in \mathbb{R}$, iterations $max\_iter > 0 \in \mathbb{R}$, quality indicators $\mathcal{I} = \{I_1, .., I_k\}$

Output: $P_{Q,\epsilon}$

1: $A \leftarrow \emptyset$
2: **for** $i = 1$ **to** max_iter **do**
3: $P \leftarrow$ BUILD_SOLUTIONS$(p, d, \alpha, \beta, \gamma, \delta, q_0, M, T)$ // ver Sección 3.1
4: $Q \leftarrow$ CROSSOVER(P, p_c) // ver Sección 3.1
5: $Q \leftarrow$ MUTATION(Q, p_m) // ver Sección 3.1
6: $P \leftarrow P \cup Q$
7: Local_Search(P) // ver Sección 3.1
8: $A \leftarrow ArchiveUpdateP_{Q,\epsilon}D_y(A, P)$
9: **while** $|P| > n$ **do**
10: **for** s **in** P **do**
11: $Fit(s) \leftarrow \sum_{\mathcal{I}_k \in \mathcal{I}} w_{\mathcal{I}_k} \cdot Fit_{\mathcal{I}_k}(s)$
12: **end for**
13: $worst \leftarrow \text{mín}_{s \in P}\, Fit(s)$
14: $P \leftarrow P \backslash \{worst\}$
15: **end while**
16: UPDATE_PHEROMONE(M, P, D)
17: **end for**
18: **return** A

3.2.2. cMIBACO

La propuesta de esta tesis es el algoritmo cooperativo basado en múltiples indicadores para la optimización mediante colonia de hormigas (cMIBACO, por sus siglas en inglés[1]). En el cual múltiples algoritmos IBACO$_I$ basados en solo indicador $I = \{HV, R2, \epsilon^+\}$ representan una colonia de hormigas, donde cada una de éstas obtiene soluciones aproximadas. Al tener distintas colonias aproximando a distintas regiones del frente de Pareto se puede tener una mejor diversidad de

[1]Cooperative Multi-Indicator Based Ant Colony Optimization Algorithm

soluciones. Al momento de que las colonias de hormigas ya han actualizado bastantes veces su matriz de feromonas, puede ser necesario realizar algún intercambio de información, con el fin de evitar una convergencia temprana. Así que una alternativa es actualizar la matriz de feromonas de cada colonia con soluciones externas de otra colonia. Para obtener soluciones externas se realiza un proceso de migración de soluciones, el cual consiste en que para cada colonia se toma aleatoriamente soluciones creadas por las otras colonias. Luego se evalúan las soluciones seleccionadas con el nuevo indicador y este conjunto se utiliza para actualizar la matriz de feromonas de la colonia seleccionada.

En el Algoritmo 21 se muestra el cMIBACO, primero se inicializan las matrices de feromona para cada IBACO$_I$ en un valor aleatorio entre 0 y 1 (líneas 2-4). Para un número de iteraciones, se ejecutan los múltiples IBACO$_I$ para cada indicador (líneas 6-8). Luego, se guardan las soluciones aproximadas de entre todas las soluciones obtenidas (línea 9). Como siguiente paso, realiza el proceso de migración de soluciones como se presenta en el Algoritmo 22 (línea 10) y se actualizan las matrices de feromona con las soluciones externas para cada indicador (líneas 11-13). Para el proceso de migración, para cada indicador se seleccionan aleatoriamente $nmig \in \mathbb{Z}^+$ soluciones del archivo externo que no fueron producidas por el indicador actual y evaluando cada solución de acuerdo con el indicador de su nueva colonia. Estas soluciones externas se utilizan para actualizar la matriz de feromonas de su nueva colonia de hormigas.

Algoritmo 21 cMIBACO general framework

Input: number of external solutions $nmig \in \mathbb{Z}$, quality indicators $\mathcal{I} = \{I_1, .., I_k\}$, iterations $max\_iter\_c \in \mathbb{Z}$, number of ants $p > 0 \in \mathbb{Z}$, days $d > 0 \in \mathbb{Z}$, pheromone constant $\alpha > 0 \in \mathbb{R}$, time travel constant $\beta > 0 \in \mathbb{R}$, driver consistency constant $\gamma > 0 \in \mathbb{R}$, arrival constant $\delta > 0 \in \mathbb{R}$, evaporation rate $\rho > 0 \in \mathbb{R}$, crossover probability $p_c > 0 \in \mathbb{R}$, crossover mutation $p_m > 0 \in \mathbb{R}$, iterations $max\_iter > 0 \in \mathbb{R}$

Output: $P_{Q,\epsilon}$

1: $A \leftarrow \emptyset$
2: **for** $j = 1$ **to** k **do**
3: Initialize pheromone matrix M^j of IBACO$_j$.
4: **end for**
5: **for** $t = 1$ to max_iter_c **do**
6: **for** $j = 1$ **to** k **do**
7: $P_j \leftarrow$IBACO$_j(p, d, \alpha, \beta, \gamma, \delta, \rho, p_c, p_m, q_0, max_iter, \mathcal{I})$
8: **end for**
9: $A \leftarrow ArchiveUpdate P_{Q,\epsilon} D_y(A, \{P_1, .., P_k\})$
10: $E_1, .., E_k \leftarrow$ MIGRATION$(A, nmig, \{I_1, .., I_k\})$
11: **for** $j = 1$ **to** k **do**
12: UPDATE_PHEROMONE(M^j, E_j, D)
13: **end for**
14: **end for**
15: **return** A

Algoritmo 22 Migration in cMIBACO

Input: archive A, number of external solutions $nmig$, quality indicators $\mathcal{I} = \{I_1, .., I_k\}$
Output: $E_1, .., E_k$

```
1: if |A| > nmig then
2:     for j = 1 to k do
3:         E_j ← ∅
4:         if there are nmig solutions in A that were not produced by IBACO_j then
5:             Randomly select nmig solutions from A that were not generated by IBACO_j
6:             Store the solutions in E_j
7:             for s in E_j do
8:                 Fit(s) ← Σ_{s'∈E_j\{s}} −e^{−I_j({s'},{s})/k}
9:             end for
10:        end if
11:    end for
12: end if
13: return {E_1, .., E_k}
```

3.2.3. Complejidad en tiempo

Para el análisis de complejidad primero se debe analizar la complejidad de cada iteración en IBACO$_{ws}$ y los indicadores individuales en términos de las comparaciones realizadas dada una población de soluciones P y n la longitud del vector que contiene todos los recorridos en todos los días de una instancia de MOGenConVRP con m clientes.

Primero, realizar la operación de cruza descrita en la Sección 3.1 toma tiempo $O(n)$ debido a que se unen vectores de tamaño lineal de entre las soluciones padres. Para el operador de mutación, la complejidad también es $O(n)$ con cualquier de los tres operadores de mutación definidos ya que solo se cambia una vez la posición de cualquier elemento.

A continuación se analiza la complejidad de IBACO$_{ws}$ por partes.

- Crear $|P|$ soluciones a partir de P con el operador de cruza en tiempo $O(n)$ y aplicar cualquier operador de mutación de tiempo $O(n)$ toma en total tiempo $O(2|P|n)$.
- Guardar soluciones aproximadas en el archivo externo toma tiempo $O(|P||A|)$, ya que cada vez que se agrega una solución al archivo se tiene que revisar si alguna otra la domina.
- Sobre la complejidad de evaluar $Fit(x_1)$, para cualquier solución $x_1 \in P$, este operador compara cada solución contra el resto y aplicando cada operador binario, por lo que se debe evaluar cada indicador y sumar las complejidades:
  - Para $I_{HV}(x_2, x_1)$, obtener el hipervolumen de un conjunto de soluciones toma tiempo $O(d\ log\ d)$ para tres objetivos y d soluciones no dominadas de acuerdo con la implementación utilizada por Beume et al. [13]. Debido a que en esta implementación se usa un árbol binario balanceado como estructura de datos, donde mantiene todas las soluciones de la población. Al árbol balanceado le aplica operaciones de búsqueda, inserción y eliminación, estas operaciones toman tiempo logarítmico. Por lo que se requiere tiempo $O((d\ log\ d)|P|)$ para calcular el hipervolumen en $Fit(x_1)$ de una solución contra el resto y en total $O((d\ log\ d)|P|^2)$ para toda la población.

- Para $I_{HV}(x_2, x_1)$, calcular el indicador $R2$ para conjuntos de tamaño uno, requiere realizar un promedio de los pesos W en tres dimensiones. Por lo que requiere tiempo $O(3|W||P|)$ para $Fit(x_1)$ de cada solución y en total $O(3|W||P|^2)$ para toda la población.
- Para $I_{\epsilon^+}(x_2, x_1)$, se calcula la diferencia ϵ más grande entre dos soluciones de tres objetivos, tomando $O(3|P|)$ para calcular $Fit(x_1)$ de cada solución y en total $O(3|P|^2)$ para toda la población.

Tomando las complejidades de los tres indicadores se tiene que evaluar todas las soluciones en P con $Fit(x_1)$ toma tiempo $O((d\ log\ d)|P|^2 + 3|W||P|^2 + 3|P|^2)$ y esto se realiza hasta reducir el tamaño de población, lo que lleva a una complejidad $O((d\ log\ d)|P|^3 + 3|W||P|^3 + 3|P|^3)$. Tomando el tiempo máximo se tiene que la complejidad de utilizar la suma ponderada de los tres indicadores es $O((d\ log\ d)|P|^3)$.

- En la actualización de feromonas, cada solución debe guardar su aptitud en el conjunto de matrices de feromonas M de acuerdo con los días y horarios. Cada una de estas matrices de dimension $m \times m$ con m el número de clientes, en el peor caso teniendo complejidad $O(m^2|M||P|)$.

Tomando el máximo de las complejidades anteriores, la complejidad en tiempo por iteración de IBACO$_{ws}$ es $O((d\ log\ d)|P|^3)$. En cuanto a usar los indicadores de forma individual, para IBACO$_{HV}$ la complejidad es $O((d\ log\ d)|P|^3)$, para IBACO$_{R2}$ se tiene complejidad $O(3|W||P|^3)$ y el IBACO$_{\epsilon^+}$ complejidad $O(3|P|^3)$.

Ahora la complejidad en cada iteración del algoritmo cMIBACO dada una población P y m clientes de MOGenConVRP.

- Inicializar las matrices M de cada día y horario de dimensión $m \times m$ a un valor mínimo o aleatorio toma tiempo $O(m^2|M|)$.
- Del análisis anterior de IBACO$_{ws}$ y las variantes de un solo indicador se tienen las complejidades:
  - Complejidad $O((d\ log\ d)|P|^3)$ para IBACO$_{HV}$.
  - Complejidad $O(3|W||P|^3)$ para IBACO$_{R2}$.
  - Complejidad $O(3|P|^3)$ para IBACO$_{\epsilon^+}$.

  Sumando las complejidades de los algoritmos, se tiene en total tiempo $O((d\ log\ d)|P|^3 + 3|W||P|^3 + 3|P|^3)$ y tomando el máximo $O((d\ log\ d)|P|^3)$.
- Guardar soluciones aproximadas en el archivo externo toma tiempo $O(|P||A|)$.
- En la migración de soluciones, para cada conjunto de soluciones externas E_k se evalúa la aptitud de sus soluciones en tiempo $O(|E_k|^2)$ ya que utiliza la función $Fit(x_1)$ comparando uno contra el resto. Considerando todos los conjuntos de soluciones externas para los indicadores y los tiempos de los indicadores binarios, se tiene que el proceso de migración toma tiempo $O((d\ log\ d)|E_k|^2 + 3|W||E_k|^2 + 3|E_k|^2)$.

- En la actualización de feromonas, cada solución del conjunto de soluciones externas E_k debe guardar su aptitud en las M matrices de los días y horarios. Cada una de estas matrices de dimension $m \times m$, en el peor caso teniendo complejidad $O(m^2|M||E_k|)$.

De todos los pasos en cMIBACO, donde se tiene mayor complejidad en tiempo es en la ejecución de los algoritmos individuales, teniendo $O((d\ log\ d)|P|^3)$ como la máxima.

3.3. Soluciones ligeramente robustas con cMIBACO

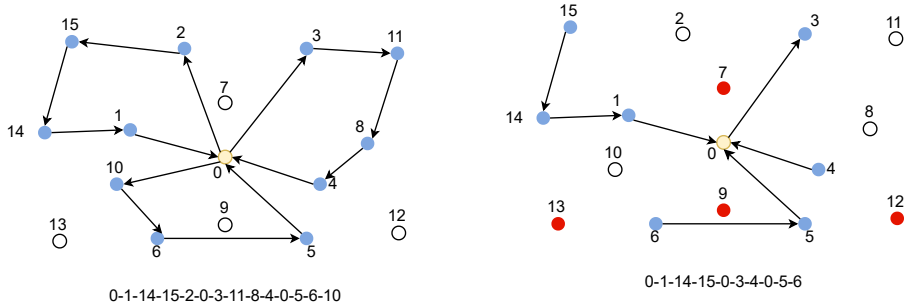
Ahora se define la forma de aplicar el cMIBACO para la optimización robusta en el MOGenConVRP bajo incertidumbre. Iniciando con la manera en que se define la incertidumbre para el MOGenConVRP y la evaluación de los escenarios. Después se muestra la aplicación de cMIBACO para obtener las soluciones robustamente ligeras.

3.3.1. Incertidumbre en MOGenConVRP

Para el MOGenConVRP la incertidumbre se encuentra en los clientes a visitar, se tienen distintos escenarios que contienen estos clientes a visitar y que no necesariamente se encuentran todos en un escenario nominal. Por ejemplo, para una instancia con dos días de planeación que tiene como clientes a $2, 4, 6, 8, 10, 12, 14$ en AM y $1, 3, 5, 7, 9, 11, 13$ en PM, un escenario válido es visitar en el primer día solo a los clientes en $\{2, 3, 7, 9, 12, 14\}$ y en el segundo día solo visitar los clientes $\{1, 2, 5, 6, 7, 11, 14\}$. Otro escenario válido es visitar a todos los clientes en todos los días.

Por lo que dada una solución obtenida del cMIBACO en un escenario nominal, para convertirla en una solución para otro escenario se realiza lo descrito en el Algoritmo 23. Donde toma las rutas de la solución nominal s y las copia en la nueva solución s', copiando solo aquellos arcos válidos en el nuevo escenario (líneas 2-3). Estos arcos son aquellos que solo contienen los nodos del escenario dado. De forma que se tienen rutas incompletas, sin embargo se deben agregar nodos a éstas para hacerlas válidas. Por lo que, para cada nodo que aún no está incluido en alguna ruta, se busca el cliente más cercano en alguna ruta incompleta donde aún se pueda añadir (línea 8). Toma el cliente con la distancia mínima (línea 11) y agrégalo a un lado de su nodo más cercano (línea 12). Finalmente se tendrán los nuevos clientes dentro de las rutas más cercanas a ellos. Estas rutas en su mayoría provienen del escenario nominal donde se realizó la optimización, por lo que probablemente contengan los mejores tiempos.

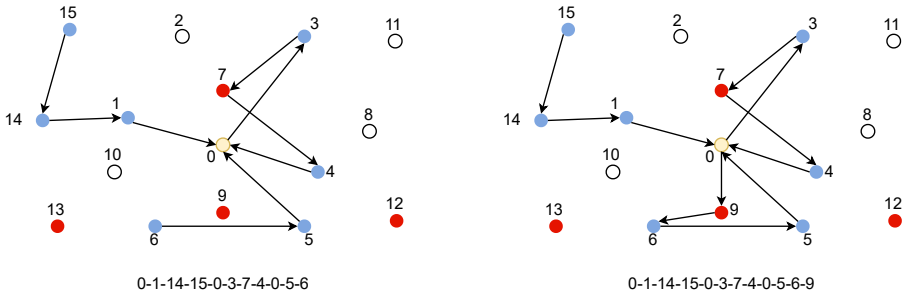
En las Figuras 3.11, 3.12, 3.13 se muestra un ejemplo paso a paso sobre la evaluación de una solución en un día y horario específico para otro escenario. Comenzando con las dos imágenes de la Figura 3.11, donde la Figura 3.11a muestra una solución en un escenario nominal y la Figura 3.11b muestra como se eliminan los caminos que contienen clientes que ya no aparecen en el escenario y añade clientes que no estaban presentes. En la Figura 3.12 se muestran los siguientes pasos, donde ambas figuras muestran como se agregan los nuevos clientes a las rutas que aún están incompletas. En la Figura 3.13 se muestra como se añaden los últimos clientes a la ruta más cercana y aún con capacidad según el algoritmo.



(a) Paso 1. Rutas obtenidas para un escenario nominal en un día y horario.

(b) Paso 2. Se agregan clientes que no estaban en el escenario nominal (rojo) y se eliminan aquellos que ya no pertenecen (blanco).

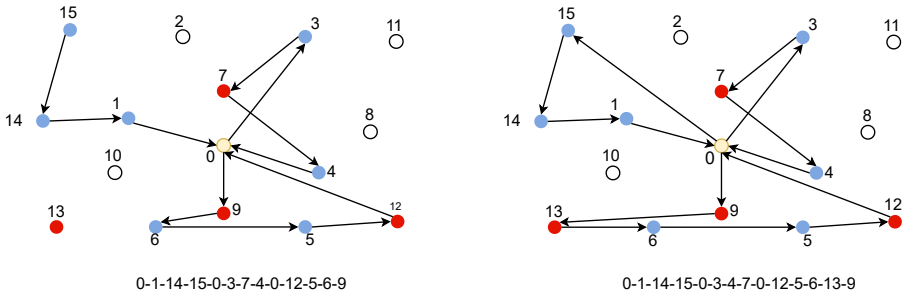
Figura 3.11: Ejemplo gráfico para transformar una solución dado un escenario (Pasos 1-2). Clientes que no pertenecen al escenario (blanco). Clientes que pertenecen al escenario (azul). Clientes que pertenecen a otro escenario (rojo).



(a) Paso 3. Se agrega el cliente 7 a la ruta de su nodo más cercano que es el 3.

(b) Paso 4. Se agrega el cliente 9 a la ruta de su nodo más cercano que es el 6.

Figura 3.12: Ejemplo gráfico para transformar una solución dado un escenario (Pasos 3-4). Clientes que no pertenecen al escenario (blanco). Clientes que pertenecen al escenario (azul). Clientes que pertenecen a otro escenario (rojo).



(a) Paso 5. Se agrega el cliente 12 a la ruta de su nodo más cercano que es el 5.

(b) Paso 6. Se agrega el cliente 13 a la ruta de su nodo más cercano que es el 6.

Figura 3.13: Ejemplo gráfico para transformar una solución dado un escenario (Pasos 5-6). Clientes que no pertenecen al escenario (blanco). Clientes que pertenecen al escenario (azul). Clientes que pertenecen a otro escenario (rojo).

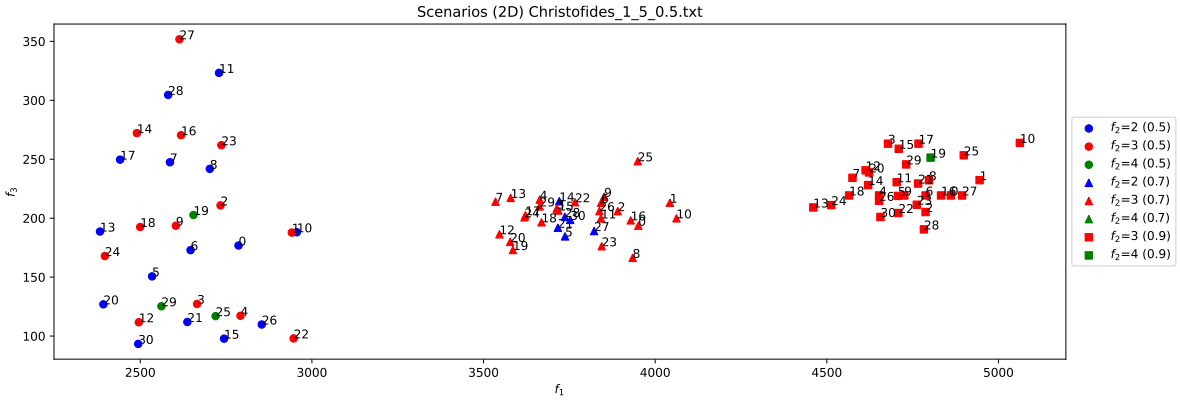
Algoritmo 23 Create solution from scenario

Input: Nominal scenario ξ', current solution s, timetable h, day for planning d, nodes from nominal scenario $V_{\xi'}^{hd}$, nodes from a given scenario V_{ξ}^{hd}

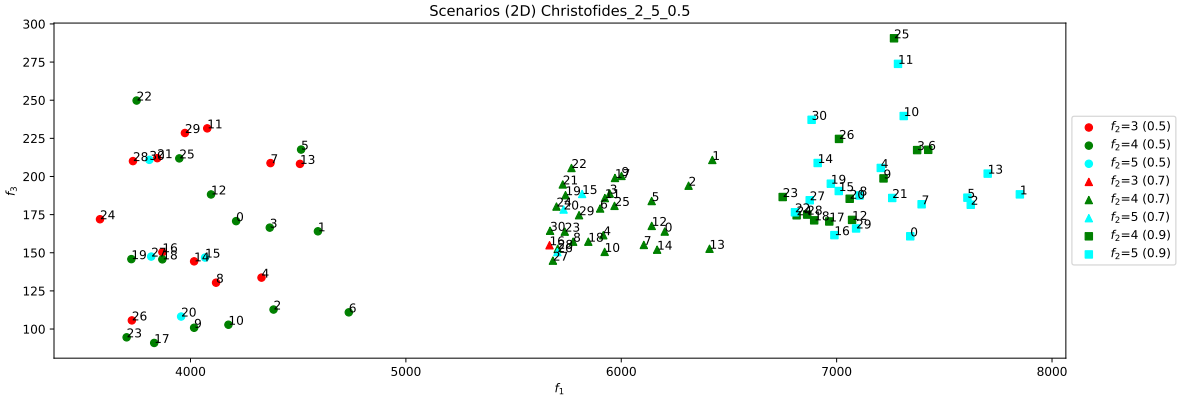
Output: New solution s' for new scenario

1: Copy the tours from s that contains at least one client from V_{ξ}^{hd}
2: Remove from tours every client not contained in V_{ξ}^{hd}
3: $K \leftarrow V_{\xi'}^{hd} \cap V_{\xi}^{hd}$
4: **while** $|K| < |V_{\xi}^{hd}|$ **do**
5: $\quad R \leftarrow \emptyset$
6: $\quad$ **for** v **in** $V_{\xi}^{hd} \setminus K$ **do**
7: $\quad\quad$ Let $v_{nearest}$ the nearest neighbor in K for v with distance d and feasible tour
8: $\quad\quad R \leftarrow R \cup \{(v, v_{nearest}, d)\}$
9: $\quad$ **end for**
10: $\quad$ Get $(u, u_{nearest}, d)$ from R with the minimal distance d
11: $\quad$ Append u next to $u_{nearest}$ in its current tour.
12: $\quad K \leftarrow K \cup \{u\}$
13: **end while**
14: **return** s'

En la Figura 3.14 se muestra un ejemplo de soluciones evaluadas en tres escenarios optimizando el escenario nominal con cMIBACO en dos instancias de MOGenConVRP. Debido a la cantidad de soluciones que se encontraron solo se presentan 30 soluciones por instancia.



(a) Primeras 30 soluciones para `Christofides_1_5_0.5` evaluadas en tres escenarios (0.5,0.7,0.9).



(b) Primeras 30 soluciones para `Christofides_2_5_0.5` evaluadas en tres escenarios (0.5,0.7,0.9).

Figura 3.14: Se muestran las distintas evaluaciones en espacio de los objetivos para soluciones en varios escenarios para dos instancias de MOGenConVRP. Usando tres escenarios de frecuencias; 50 %(puntos circulares), 70 % (puntos triangulares), 90 % (puntos cuadrados). Denotando las soluciones del segundo objetivo como f_2=2 (azul), f_2=3 (rojo), f_2=4 (verde) y f_2=5 (cian).

3.3.2. Filtrado de soluciones ligeramente robustas

La forma en que se obtienen soluciones ligeramente robustas con cMIBACO se presenta en el Algoritmo 24. Comenzando con la inicialización de los clientes para el escenario nominal (línea 1). Luego se realiza la optimización sobre el escenario nominal, obteniendo el conjunto de soluciones

aproximadas $P_{Q,\epsilon}$ (línea 2). Para cada solución aproximada se evalúa en todos los escenarios excepto el nominal (línea 5) de acuerdo con el algoritmo anterior y se guardan sus peores escenarios (línea 6). Finalmente se obtienen aquellas que sean robustas Min-Max basadas en conjuntos de acuerdo con el archivo externo $ArchiveUpdate\mathcal{R}$ (línea 8).

Algoritmo 24 cMIBACO for Light Robust Solutions

Input: $nmig \in \mathbb{Z}, \mathcal{I} = \{I_1, .., I_k\}, max\_iter\_c \in \mathbb{Z}, p > 0 \in \mathbb{Z}, \alpha > 0 \in \mathbb{R}, \beta > 0 \in \mathbb{R}, \gamma > 0 \in \mathbb{R}, \delta > 0 \in \mathbb{R}, \rho > 0 \in \mathbb{R}, p_c > 0 \in \mathbb{R}, p_m > 0 \in \mathbb{R}, q_0 > 0 \in \mathbb{R}, max\_iter > 0 \in \mathbb{R}, \xi' \in \mathcal{U}$
Output: Set LR of lightly robust solutions
1: Initialize costumers of nominal scenario ξ'
2: $A \leftarrow$ cMIBACO$(nmig, \mathcal{I}, max_iter_c, p, d, \alpha, \beta, \gamma, \delta, \rho, p_c, p_m, q_0, max_iter)$
3: WC $\leftarrow \emptyset$
4: **for** a **in** A **do**
5: Evaluate a in each scenario $\xi \in \mathcal{U} \setminus \{\xi'\}$
6: Add worst solutions to WC
7: **end for**
8: $LR \leftarrow ArchiveUpdate\mathcal{R}$(WC, [])
9: **return** LR

En la Figura 3.15 se muestran los peores escenarios y su evaluación para cada solución a partir del escenario nominal en una instancia de MOGenConVRP. En cuanto a las soluciones ligeramente robustas, en la Figura 3.16 se muestran estas soluciones para la misma instancia. Para cada conjunto de soluciones se denota gráficamente el área debajo de éstas.

En la Figura 3.17 se diferencian estas soluciones de otras que no lo son, como en el caso de las soluciones identificadas con 0,1,2 que no son robustas porque el conjunto de la solución 32 está contenido dentro de el área de estas soluciones. Las soluciones 3 y 4 tampoco son robustas porque la solución 24 está contenida dentro del área de estas soluciones.

En la Figura 3.18 se tienen más soluciones a comparar, por ejemplo la solución 8 tiene una buena solución para el escenario nominal pero una pésima solución en el escenario de frecuencias 90 % y dentro del área de esta solución se encuentra contenida la solución robusta 32. Este mismo caso sucede con la solución 7 que contiene a la solución eficiente 24 por tener un mal desempeño en un escenario. Las soluciones 5,6 y 9 contienen a la solución 30 por lo que tampoco son robustas.

En la Figura 3.19 se muestran las soluciones 10,11,12,14 y 15 que no son robustas porque las soluciones robustas 13 y 24 están contenidas en su área, esto debido a que en un escenario tienen un mal desempeño.

Otras soluciones que no son robustas se muestran en la Figura 3.20, tales soluciones son la 16,17, 19 y 20 debido a que la solución 13 está contenida en el área de estas soluciones.

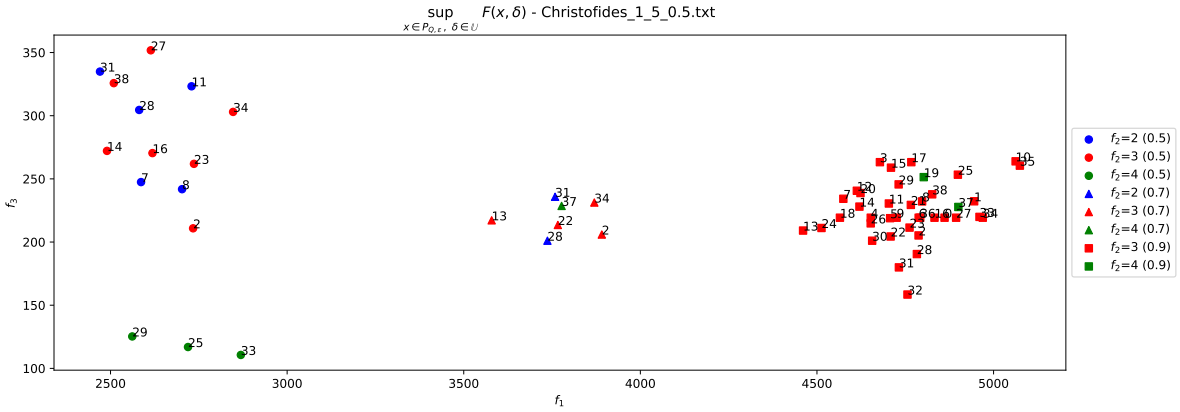


Figura 3.15: Peores escenarios de cada solución para `Christofides_1_5_0.5`. Usando tres escenarios de frecuencias; 50 %(puntos circulares), 70 % (puntos triangulares), 90 % (puntos cuadrados). Denotando las soluciones del segundo objetivo como f_2=2 (azul), f_2=3 (rojo) y f_2=4 (verde).

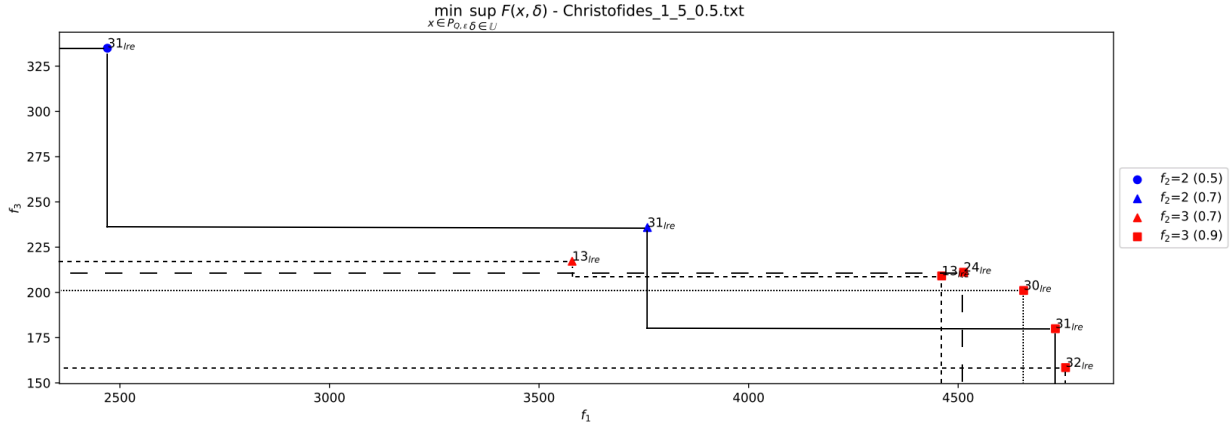


Figura 3.16: Soluciones ligeramente robustas obtenidas para `Christofides_1_5_0.5` y el área debajo de cada conjunto de soluciones. Usando tres escenarios de frecuencias; 50 %(puntos circulares), 70 % (puntos triangulares), 90 % (puntos cuadrados). Denotando las soluciones del segundo objetivo como f_2=2 (azul), f_2=3 (rojo).

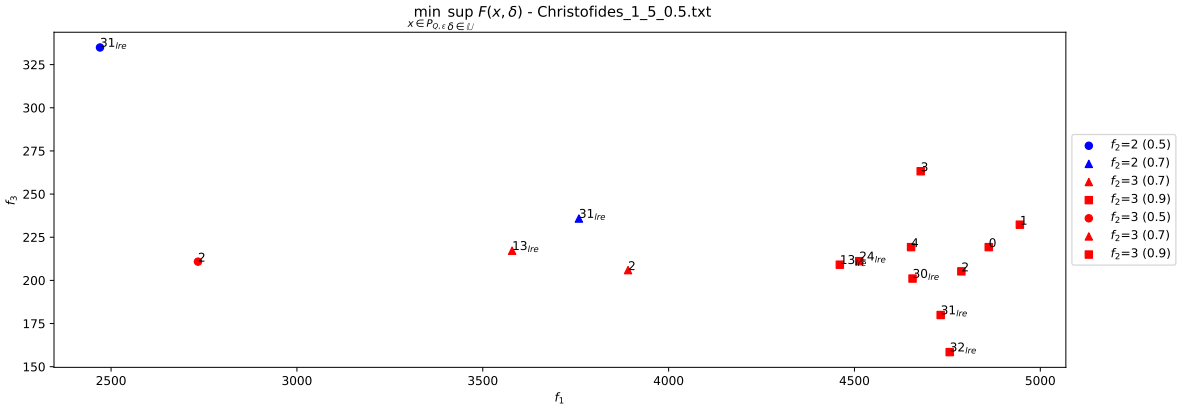


Figura 3.17: Comparación de soluciones ligeramente robustas de `Christofides_1_5_0.5`. Donde la solución 32 está contenida en el área del conjunto de soluciones 0,1,2. De la misma forma, la solución 24 está contenida en el área de 3 y 4.

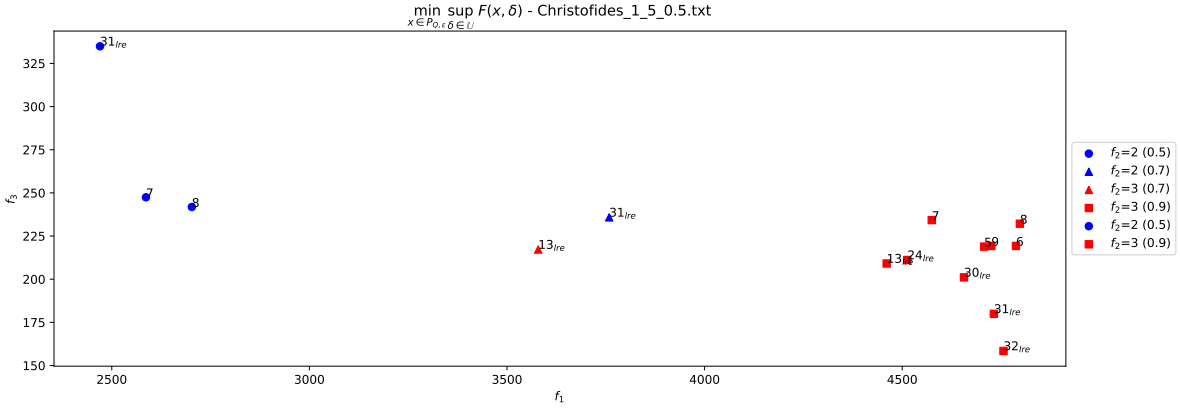


Figura 3.18: Comparación de soluciones ligeramente robustas de `Christofides_1_5_0.5`. Donde la solución 32 está contenida en el área del conjunto de soluciones de 8. Similar mente, la solución 24 está contenida en el área de la solución 7. Las soluciones 5,6 y 9 contienen a la solución 30.

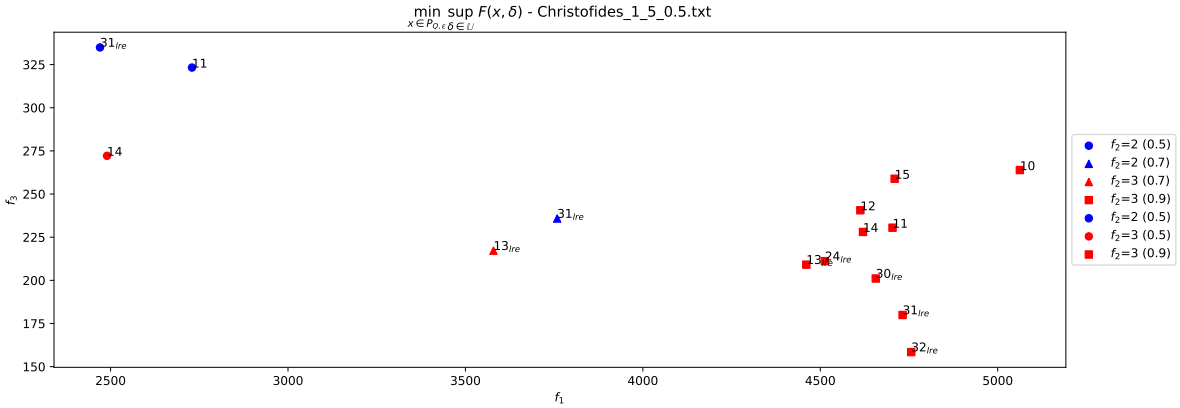


Figura 3.19: Comparación de soluciones ligeramente robustas de `Christofides_1_5_0.5`. Las soluciones 10,11,12,14 y 15 no son robustas porque las soluciones 13 y 24 están contenidas en su área.

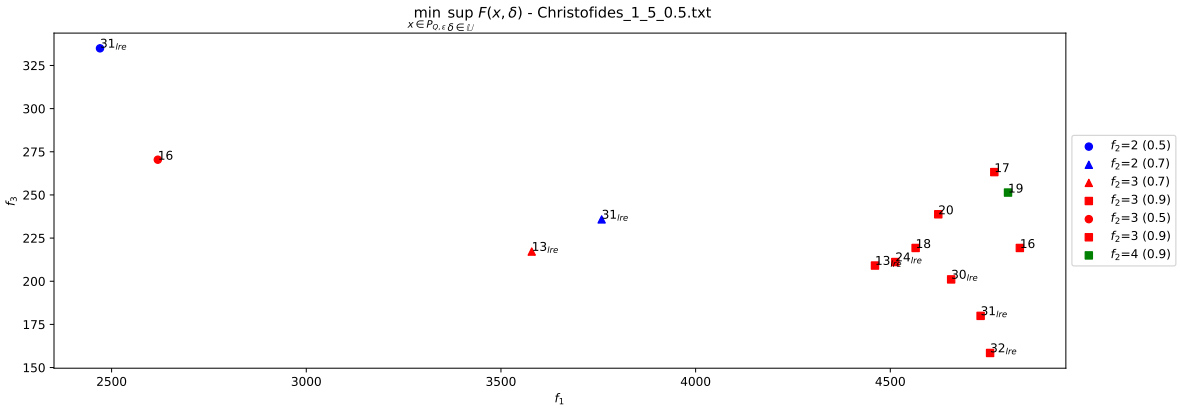


Figura 3.20: Comparación de soluciones ligeramente robustas de `Christofides_1_5_0.5`. Las soluciones 16,17,19 y 20 no son robustas porque la solución 13 está contenida en el área de estas soluciones.

Capítulo 4

Estudio experimental

En este capítulo se presentan los resultados de todos los experimentos que involucran los algoritmos basados en indicadores del capítulo anterior y la optimización ligeramente robusta para el MOGenConVRP bajo incertidumbre. En la Sección 4.1, se describe el conjunto de problemas utilizados, los parámetros utilizados y las métricas usadas para la comparación de algoritmos. En la Sección 4.2, se presenta el efecto individual de cada componente de cMIBACO y su comparación con el resto. Dentro de esta sección es de interés conocer qué componentes mejoran el desempeño de cMIBACO y que justifican su diseño. Junto a las desventajas que puedan tener cada componente de cMIBACO.

En la Sección 4.3, se compara los algoritmos basados en un indicador, la suma ponderada y el cooperativo para obtener soluciones aproximadas en MOGenConVRP. En esta sección se busca conocer si existe diferencia entre usar indicadores de forma individual contra la forma cooperativa. Además de conocer cuáles son los algoritmos que tienen un desempeño cercano a cMIBACO y cuáles son las debilidades del cMIBACO frente a los demás algoritmos. En la Sección 4.4, se presentan los resultados de aplicar el cMIBACO para el MOGenConVRP bajo incertidumbre y su comparación con el resto. En esta sección se quiere saber si optimizar con cMIBACO para soluciones aproximadas puede funcionar para filtrar soluciones robustamente ligeras. También conocer el desempeño de obtener soluciones robustas mediante cMIBACO contra soluciones robustas usando indicadores individuales.

En el Apéndice A se encuentran los gráficos de todos los resultados de todas las secciones en este capítulo. El código de este proyecto se encuentra en un repositorio de Github[1].

4.1. Metodología

En cada uno de los experimentos, para cada algoritmo se realizaron 20 ejecuciones independientes y su desempeño se midió con los indicadores de $HV, R2, E_s$. Para el análisis estadístico se utilizó la prueba de rango de Wilcoxon [120] con un intervalo de confianza del 95 % y diagramas de diferencias críticas [56]. También se comparan las medianas de cMIBACO respecto a los demás algoritmos, junto a sus frentes obtenidos. Todos los experimentos se realizaron en una computadora Apple Mac con chip M2, frecuencia de procesador de 3.49 GHz y 16 GB RAM.

[1] `https://github.com/rodrigofvc/cmibaco-lre-mogenconvrp`

4.1.1. Problemas utilizados

El conjunto de problemas se obtuvo de [66], donde los autores proponen instancias de MOGenConVRP con distintos números de clientes y distintas frecuencias de visita a cada cliente. Todas las instancias tienen el mismo horizonte de planeación de 5 días y un tiempo límite para los vehículos de $T = 1000$. Cada instancia con una frecuencia de visita a los clientes de 50 % tiene una versión de frecuencias de visita de 70 % y 90 %. Estas versiones por instancia conservan la misma cantidad, ubicación de los clientes y mismas propiedades de vehículos. En la Tabla 4.1 se muestran el número de clientes y la capacidad de los vehículos de cada instancia.

Problema de Frecuencia 50 %	**Problema de Frecuencia 70 %**	**Problema de Frecuencia 90 %**	**Cantidad de clientes**	**Capacidad del vehículo**
Christofides_1_5_0.5	Christofides_1_5_0.7	Christofides_1_5_0.9	50	160
Christofides_2_5_0.5	Christofides_2_5_0.7	Christofides_2_5_0.9	75	140
Christofides_3_5_0.5	Christofides_3_5_0.7	Christofides_3_5_0.9	100	200
Christofides_4_5_0.5	Christofides_4_5_0.7	Christofides_4_5_0.9	150	200
Christofides_5_5_0.5	Christofides_5_5_0.7	Christofides_5_5_0.9	200	200
Christofides_6_5_0.5	Christofides_6_5_0.7	Christofides_6_5_0.9	50	160
Christofides_7_5_0.5	Christofides_7_5_0.7	Christofides_7_5_0.9	75	140
Christofides_8_5_0.5	Christofides_8_5_0.7	Christofides_8_5_0.9	100	200
Christofides_9_5_0.5	Christofides_9_5_0.7	Christofides_9_5_0.9	150	200
Christofides_10_5_0.5	Christofides_10_5_0.7	Christofides_10_5_0.9	200	200
Christofides_11_5_0.5	Christofides_11_5_0.7	Christofides_11_5_0.9	120	200
Christofides_12_5_0.5	Christofides_12_5_0.7	Christofides_12_5_0.9	100	200

Tabla 4.1: Número de clientes y capacidades de los vehículos para las instancias de todas las frecuencias en MOGenConVRP.

4.1.2. Parámetros

Los parámetros utilizados se encontraron realizando una optimización de hiperparámetros para cada algoritmo con iRace [79]. Todas las ejecuciones de los algoritmos usan las mismas semillas aleatorias, en este caso son 20 semillas de números aleatorios que son números primos de ocho dígitos. En la Tabla 4.2 se muestran los parámetros que involucran el comportamiento de las hormigas y búsqueda local. Para obtener soluciones aproximadas con $ArchiveUpdateP_{Q,\epsilon}D_y$, se usaron como parámetros $\epsilon = [9000, 5, 550]$ y $\Delta_y = [100, 0, 50]$ para los objetivos f_1, f_2, f_3 respectivamente.

4.1.3. Evaluación del desempeño

Para evaluar el desempeño de cada algoritmo en cada ejecución, se utilizaron los indicadores de calidad de hipervolumen, $R2$ y Riesz s-energy. Para el hipervolumen, cada cierto número de evaluaciones en cada ejecución de cada algoritmo se evaluó el contenido del archivo externo de soluciones aproximadas usando los puntos de referencia en la Tabla 4.3. Estos puntos de referencia fueron obtenidos como el máximo para cada objetivo de entre múltiples ejecuciones empíricas más una constante.

En cuanto al indicador de $R2$, el punto ideal que se consideró fue el origen y se utilizaron 30 vectores de pesos generados mediante Riesz s-energy con la implementación de PYMOO [16], estos

Parámetros	cMIBACO	IBACO$_{ws}$	IBACO$_{\epsilon+}$	IBACO$_{R2}$	IBACO$_{HV}$	cMIBACO$_{base}$	cMIBACO$_{hybrid}$	cMIBACO$_{crossover}$	cMIBACO$_{mutation}$
ρ	0.4439	0.4859	0.3584	0.3755	0.8996	0.7443	0.1794	0.4624	0.6378
α	1.3123	2.1998	2.5445	0.9869	2.7025	0.7133	0.8227	0.9547	0.4094
β	4.6125	4.672	4.1522	4.6904	4.4091	0.5443	0.9473	0.582	0.9165
γ	0.0495	0.5205	0.3538	0.4866	0.5077	0.1433	0.22	0.2158	0.2239
δ	3.2124	1.4294	3.1899	1.5255	2.9445	0.502	0.0701	0.5402	0.5341
Q	0.4765	0.6373	0.4162	0.7186	0.0779	0.9837	0.2194	0.2767	0.9437
q_0	0.5695	0.1364	0.4167	0.2921	0.2926	0.7179	0.6138	0.6282	0.7549
p_m	0.0338	0.0473	0.0372	0.1048	0.1066	-	0.1079	-	0.2978
p_c	0.8733	0.945	0.5982	0.5838	0.5264	-	0.7856	0.8845	-
$nmig$	17	-	-	-	-	15	11	13	10
r_c (LNS)	18	22	23	18	16	-	-	-	-
η (LNS)	104873	102319	103250	106670	102276	-	-	-	-
δ_{lns} (LNS)	0.0471	0.05	0.0485	0.0312	0.0468	-	-	-	-
UB_1 (LNS)	1397	931	946	1001	939	-	-	-	-
UB_2 (LNS)	843	564	912	573	549	-	-	-	-

Tabla 4.2: Mejores parámetros encontrados para cada algoritmo de acuerdo con iRace.

Instancia de Frecuencia 50 %	**Instancia de Frecuencia 70 %**	**Instancia de Frecuencia 90 %**	**Punto de referencia** $[f_1, f_2, f_3]$
Christofides_1_5_0.5	Christofides_1_5_0.7	Christofides_1_5_0.9	[10000, 7, 700]
Christofides_2_5_0.5	Christofides_2_5_0.7	Christofides_2_5_0.9	[10000, 6, 600]
Christofides_3_5_0.5	Christofides_3_5_0.7	Christofides_3_5_0.9	[10000, 6, 600]
Christofides_4_5_0.5	Christofides_4_5_0.7	Christofides_4_5_0.9	[12000, 7, 900]
Christofides_5_5_0.5	Christofides_5_5_0.7	Christofides_5_5_0.9	[16000, 9, 900]
Christofides_6_5_0.5	Christofides_6_5_0.7	Christofides_6_5_0.9	[16000, 9, 900]
Christofides_7_5_0.5	Christofides_7_5_0.7	Christofides_7_5_0.9	[12000, 8, 900]
Christofides_8_5_0.5	Christofides_8_5_0.7	Christofides_8_5_0.9	[15000, 8, 900]
Christofides_9_5_0.5	Christofides_9_5_0.7	Christofides_9_5_0.9	[15000, 9, 1000]
Christofides_10_5_0.5	Christofides_10_5_0.7	Christofides_10_5_0.9	[17000, 9, 1300]
Christofides_11_5_0.5	Christofides_11_5_0.7	Christofides_11_5_0.9	[17000, 9, 1300]
Christofides_12_5_0.5	Christofides_12_5_0.7	Christofides_12_5_0.9	[17000, 9, 1300]

Tabla 4.3: Vectores de referencia usados para calcular el hipervolumen de cada instancia.

pesos se muestran en la Tabla 4.4. Para evaluar con el indicador de Riesz s-energy se calcularon distancias con la norma euclidiana.

[0,0,1]	[0,0.824,0.175]	[0.162, 0, 0.837]	[0.321, 0, 0.678]	[0.510, 0, 0.489]	[0.674, 0.325, 0]
[0,0.167,0.832]	[0,1,0]	[0.171, 0.417, 0.411]	[0.342, 0.261, 0.395]	[0.517, 0.159, 0.323]	[0.677, 0.161, 0.161]
[0,0.331,0.668]	[0.157, 0.842, 0]	[0.174, 0.277, 0.548]	[0.343, 0.397, 0.259]	[0.517, 0.482, 0]	[0.831, 0, 0.168]
[0,0.493,0.507]	[0.158, 0.146, 0.694]	[0.179, 0.551, 0.269]	[0.345, 0.123, 0.531]	[0.518, 0.320, 0.161]	[0.837, 0.162, 0]
[0,0.656,0.343]	[0.160, 0.697, 0.142]	[0.321, 0.678, 0]	[0.352, 0.529, 0.118]	[0.672, 0, 0.327]	[1 , 0, 0]

Tabla 4.4: Vectores de referencia para calcular el indicador de $R2$.

4.2. Efecto de los componentes en cMIBACO

De acuerdo con el diseño de cMIBACO, se tienen componentes híbridos y de búsqueda local que los algoritmos basados en ACO en su mayoría no poseen. Por lo que es necesario evaluar qué tanto aporta cada uno de estos componentes y evaluarlo contra la versión básica. Así que se muestran los resultados de comparar cinco versiones de cMIBACO:

- cMIBACO$_{base}$: cMIBACO sin cruza, mutación ni búsqueda local.

- cMIBACO$_{lns}$: cMIBACO con cruza, mutación y búsqueda local.
- cMIBACO$_{hybrid}$: cMIBACO con solo cruza y mutación.
- cMIBACO$_{crossover}$: cMIBACO con solo cruza.
- cMIBACO$_{mutation}$: cMIBACO con solo mutación.

A continuación se presentan los resultados de acuerdo con los indicadores de calidad y el análisis estadístico de comparar cMIBACO$_{base}$ contra las demás variantes.

4.2.1. Análisis de indicadores de calidad

En la Tabla 4.5 se muestran el promedio y desviación estándar de cada indicador de calidad de las 20 ejecuciones por cada algoritmo para cada problema, comparando la versión cMIBACO$_{base}$ contra los demás. Junto al valor de cada indicador se indica con una flecha ↑ cuando existe una diferencia significativa de acuerdo con la prueba de Wilcoxon entre los resultados de cMIBACO$_{base}$ con cada uno y si éste fue mejor. Pero en el caso de existir una diferencia significativa y los resultados de cMIBACO no son mejores, se utiliza la flecha ↓. Cuando no existe diferencia significativa se utiliza la flecha ↔.

Comenzando con el indicador de hipervolumen, en todos los problemas la variante cMIBACO$_{lns}$ tiene mejor desempeño que los demás y con diferencia significativa respecto a cMIBACO$_{base}$. En segundo lugar se tiene frecuentemente a cMIBACO$_{hybrid}$ y en tercer lugar el cMIBACO$_{crossover}$. Esto muestra que usar búsqueda local con componentes híbridos obtiene una mejor aproximación del frente de Pareto que usar solo cruza o mutación. Además de que en todos los problemas se supera a cMIBACO$_{base}$, por lo que los nuevos componentes mejoran a la estrategia clásica de solo construir soluciones. Aunque la versión básica fue separada por tres algoritmos, esta versión fue mejor que la variante cMIBACO$_{mutation}$.

Para el indicador de R2 se muestran mejores resultados de cMIBACO$_{lns}$ respecto a los demás y al igual que en hipervolumen, la variante cMIBACO$_{hybrid}$ termina en segunda para todos los problemas. Por otro lado, el cMIBACO$_{mutation}$ tiene los peores resultados para este indicador a pesar de que solo supera en dos problemas a la versión base.

Para Riesz s-energy, el cMIBACO$_{lns}$ no tuvo diferencia significativa respecto a cMIBACO$_{base}$ en 11 problemas, sin embargo, las otras versiones de cMIBACO$_{hybrid}$ y cMIBACO$_{crossover}$ perdieron con diferencia significativa en al menos cuatro instancias. En cuanto al cMIBACO$_{mutation}$, este algoritmo superó en casi la mitad de problemas a la versión base y en ningún problema perdió. Esto significa que usar solo mutación puede incrementar la diversidad de soluciones, aunque el desempeño de éstas de acuerdo con los indicadores de hipervolumen y R2 fue bastante bajo.

En la Figura 4.1 se muestra el diagrama de diferencias críticas maximizando los indicadores y considerando el tiempo de ejecución de los algoritmos. Donde la versión cMIBACO$_{lns}$ obtuvo un mejor rango, seguido de la versión cMIBACO$_{hybrid}$ y para todos se muestra una diferencia significativa. Tomando en cuenta todos los resultados de los indicadores de calidad, la búsqueda local con operadores genéticos influyen bastante para mejorar el comportamiento de las colonias de hormigas. El desempeño de usar solo operadores genéticos va a depender de la cantidad de veces que se realicen cruza o mutación. Pues cMIBACO$_{crossover}$ aunque mostró mejor desempeño en hipervolumen que cMIBACO$_{mutation}$ no tiene un mejor desempeño a cMIBACO$_{hybrid}$.

En el Apéndice A se muestran todos los resultados numéricos de estos experimentos, como las soluciones aproximadas obtenidas en espacio de los objetivos para todos los problemas. Donde

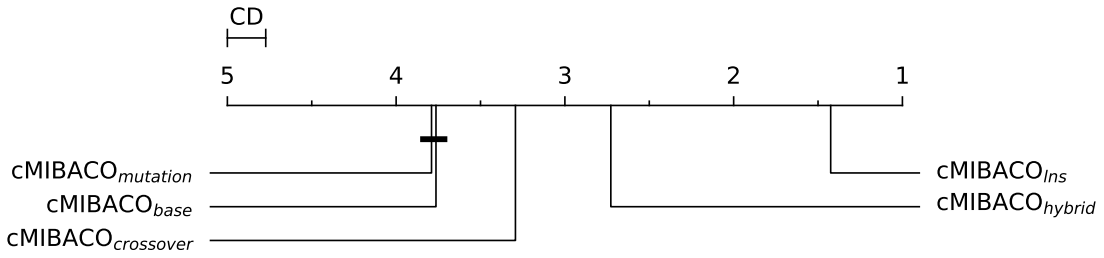


Figura 4.1: Diagrama de diferencias críticas para componentes de cMIBACO.

la versión cMIBACO$_{hybrid}$ muestra soluciones que son más cercanas entre sí. Por otro lado, la versión cMIBACO$_{mutation}$ tiene soluciones más dispersas y con mayor distribución, sin embargo, la versión cMIBACO$_{lns}$ tiene soluciones con mejores costos. En el mismo apéndice también se muestra el hipervolumen durante las evaluaciones de la mediana de cMIBACO contra los demás algoritmos. En todas las gráficas se muestra que la versión cMIBACO$_{lns}$ en la mayor parte del tiempo lleva un mejor desempeño sin quedarse estancado tanto tiempo. Seguido de la versión cMIBACO$_{hybrid}$ que se mantiene por arriba de las demás variantes y la variante cMIBACO$_{mutation}$ como la de menor desempeño. En el mismo apéndice se muestran los diagramas de caja, donde la variante cMIBACO$_{lns}$ se mantiene en todos los problemas como el algoritmo con mejor distribución y varianza de acuerdo con hipervolumen. Por el contrario, el cMIBACO$_{mutation}$ obtiene intervalos menores en todos los problemas y algunas ocasiones con datos atípicos y lo mismo sucede con cMIBACO$_{base}$.

De acuerdo con los resultados se listan las siguientes conclusiones:

- El cMIBACO$_{lns}$ mantiene mejores resultados en cuanto a los indicadores de hipervolumen y $R2$, mostrando que los componentes genéticos y la búsqueda local mejoran considerablemente las soluciones obtenidas.
- Usar solo mutación como en cMIBACO$_{mutation}$ puede apoyar bastante a la diversidad de soluciones como se muestra Riesz s-energy, sin embargo, la calidad de las soluciones se reduce.
- Es importante tomar en cuenta el balance entre la cruza y cuanta mutación que se realiza dentro de cMIBACO$_{hybrid}$, pues fue el segundo mejor algoritmo y como una alternativa más.
- De acuerdo con la prueba de Wilcoxon, existe una diferencia significativa entre todos los algoritmos, mostrando que cMIBACO$_{lns}$ es el mejor clasificado.

4.3. Desempeño individual y cooperativo de indicadores

En esta sección se presentan los resultados en la comparación de cMIBACO (de ahora en adelante se refiere así a la versión cMIBACO$_{lns}$ de la sección anterior) contra los algoritmos basados en indicadores para ACO que se describieron anteriormente como lo son IBACO$_{ws}$, IBACO$_{HV}$, IBACO$_{R2}$, IBACO$_{\epsilon+}$ para soluciones aproximadas.

Problema	Indicador	cMIBACO$_{base}$	cMIBACO$_{lns}$	cMIBACO$_{hybrid}$	cMIBACO$_{crossover}$	cMIBACO$_{mutation}$
Christofides_1_5_0.5	HV	2.145e+07 (2.965e+05)	**2.356e+07 (4.600e+05)** ↑	2.257e+07 (2.675e+05) ↑	2.248e+07 (3.463e+05) ↑	2.086e+07 (4.365e+05) ↓
Christofides_2_5_0.5	HV	8.967e+06 (1.831e+05)	**1.253e+07 (1.472e+06)** ↑	9.980e+06 (5.399e+05) ↑	9.759e+06 (5.894e+05) ↑	8.898e+06 (1.980e+05) ↔
Christofides_3_5_0.5	HV	7.241e+06 (2.260e+05)	**9.193e+06 (7.931e+05)** ↑	8.145e+06 (1.289e+05) ↑	7.781e+06 (1.197e+05) ↑	6.941e+06 (1.487e+05) ↓
Christofides_4_5_0.5	HV	1.788e+07 (3.906e+05)	**2.244e+07 (2.588e+05)** ↑	2.032e+07 (2.471e+05) ↑	1.917e+07 (3.408e+05) ↑	1.770e+07 (3.013e+05) ↔
Christofides_5_5_0.5	HV	3.244e+07 (1.062e+06)	**4.660e+07 (3.363e+05)** ↑	3.660e+07 (1.534e+06) ↑	3.511e+07 (1.900e+06) ↑	3.155e+07 (6.067e+05) ↓
Christofides_6_5_0.5	HV	7.021e+07 (1.087e+06)	**7.633e+07 (9.282e+05)** ↑	7.271e+07 (9.651e+05) ↑	7.186e+07 (9.061e+05) ↑	6.789e+07 (1.345e+06) ↓
Christofides_7_5_0.5	HV	3.040e+07 (5.036e+05)	**3.700e+07 (3.022e+06)** ↑	3.215e+07 (2.245e+05) ↑	3.174e+07 (3.733e+05) ↑	2.999e+07 (5.450e+05) ↓
Christofides_8_5_0.5	HV	3.584e+07 (4.312e+05)	**4.160e+07 (1.573e+06)** ↑	3.818e+07 (5.191e+05) ↑	3.745e+07 (3.047e+05) ↑	3.473e+07 (7.177e+05) ↓
Christofides_9_5_0.5	HV	4.013e+07 (1.718e+06)	**4.953e+07 (5.146e+05)** ↑	4.469e+07 (6.460e+05) ↑	4.292e+07 (6.006e+05) ↑	3.864e+07 (2.403e+06) ↓
Christofides_10_5_0.5	HV	4.823e+07 (5.440e+05)	**6.925e+07 (8.445e+05)** ↑	5.354e+07 (5.181e+05) ↑	5.119e+07 (5.105e+05) ↑	4.764e+07 (9.343e+05) ↔
Christofides_11_5_0.5	HV	6.395e+07 (1.123e+06)	**7.933e+07 (4.862e+06)** ↑	7.138e+07 (8.794e+05) ↑	6.827e+07 (6.556e+05) ↑	6.399e+07 (9.123e+05) ↔
Christofides_12_5_0.5	HV	8.277e+07 (7.009e+05)	**9.563e+07 (6.783e+06)** ↑	8.702e+07 (9.704e+05) ↑	8.562e+07 (7.613e+05) ↑	8.201e+07 (1.017e+06) ↓
+\≈\−			12 \0 \0	12 \0 \0	12 \0 \0	0 \4 \8
Christofides_1_5_0.5	R2	8.897e+02 (2.084e+01)	**7.926e+02 (1.636e+01)** ↑	8.233e+02 (1.246e+01) ↑	8.347e+02 (1.653e+01) ↑	9.097e+02 (1.843e+01) ↓
Christofides_2_5_0.5	R2	1.306e+03 (2.511e+01)	**1.161e+03 (1.809e+01)** ↑	1.187e+03 (1.308e+01) ↑	1.219e+03 (2.011e+01) ↑	1.297e+03 (2.161e+01) ↔
Christofides_3_5_0.5	R2	1.604e+03 (3.508e+01)	**1.362e+03 (2.338e+01)** ↑	1.457e+03 (1.802e+01) ↑	1.517e+03 (1.986e+01) ↑	1.598e+03 (2.844e+01) ↔
Christofides_4_5_0.5	R2	2.034e+03 (3.838e+01)	**1.679e+03 (2.756e+01)** ↑	1.818e+03 (2.490e+01) ↑	1.925e+03 (3.121e+01) ↑	2.021e+03 (3.074e+01) ↔
Christofides_5_5_0.5	R2	2.549e+03 (5.934e+01)	**2.101e+03 (1.732e+01)** ↑	2.285e+03 (2.686e+01) ↑	2.396e+03 (3.436e+01) ↑	2.557e+03 (3.613e+01) ↔
Christofides_6_5_0.5	R2	9.183e+02 (2.015e+01)	**8.038e+02 (1.429e+01)** ↑	8.472e+02 (1.212e+01) ↑	8.601e+02 (1.143e+01) ↑	9.251e+02 (1.674e+01) ↔
Christofides_7_5_0.5	R2	1.347e+03 (2.200e+01)	**1.190e+03 (1.476e+01)** ↑	1.231e+03 (1.551e+01) ↑	1.258e+03 (1.847e+01) ↑	1.331e+03 (2.196e+01) ↑
Christofides_8_5_0.5	R2	1.720e+03 (3.068e+01)	**1.483e+03 (1.812e+01)** ↑	1.571e+03 (1.908e+01) ↑	1.623e+03 (2.421e+01) ↑	1.717e+03 (2.220e+01) ↔
Christofides_9_5_0.5	R2	2.230e+03 (3.232e+01)	**1.869e+03 (3.464e+01)** ↑	2.009e+03 (2.148e+01) ↑	2.106e+03 (3.251e+01) ↑	2.212e+03 (3.042e+01) ↔
Christofides_10_5_0.5	R2	2.811e+03 (3.186e+01)	**2.345e+03 (4.193e+01)** ↑	2.541e+03 (2.441e+01) ↑	2.665e+03 (2.525e+01) ↑	2.794e+03 (4.834e+01) ↔
Christofides_11_5_0.5	R2	2.636e+03 (5.239e+01)	**2.215e+03 (5.230e+01)** ↑	2.344e+03 (4.428e+01) ↑	2.458e+03 (2.730e+01) ↑	2.556e+03 (3.886e+01) ↑
Christofides_12_5_0.5	R2	1.737e+03 (3.703e+01)	**1.520e+03 (2.505e+01)** ↑	1.599e+03 (2.365e+01) ↑	1.643e+03 (1.851e+01) ↑	1.728e+03 (3.515e+01) ↔
+\≈\−			12 \0 \0	12 \0 \0	12 \0 \0	2 \9 \1
Christofides_1_5_0.5	E_s	1.740e-02 (2.917e-02)	6.900e-02 (2.458e-01) ↔	**1.588e-02 (1.140e-02)** ↔	4.709e-02 (5.444e-02) ↓	6.783e-03 (8.411e-03) ↔
Christofides_2_5_0.5	E_s	3.208e-02 (5.171e-02)	6.551e-02 (1.278e-01) ↔	**2.033e-02 (3.099e-02)** ↔	2.776e-02 (4.086e-02) ↔	6.429e-03 (8.863e-03) ↑
Christofides_3_5_0.5	E_s	3.590e-02 (9.443e-02)	**1.746e-02 (2.597e-02)** ↔	6.740e-02 (1.908e-01) ↔	2.548e-02 (3.404e-02) ↔	7.173e-03 (3.698e-03) ↑
Christofides_4_5_0.5	E_s	**1.092e-02 (9.776e-03)**	3.010e-02 (4.459e-02) ↔	1.858e-02 (1.757e-02) ↓	1.727e-02 (1.544e-02) ↔	4.766e-03 (3.121e-03) ↑
Christofides_5_5_0.5	E_s	9.708e-03 (1.652e-02)	1.088e-02 (8.863e-03) ↔	**1.578e-02 (1.308e-02)** ↓	2.792e-02 (3.537e-02) ↓	7.653e-03 (6.868e-03) ↔
Christofides_6_5_0.5	E_s	9.403e-03 (6.424e-03)	1.585e-02 (1.516e-02) ↔	4.063e-02 (4.801e-02) ↓	**7.534e-02 (2.475e-01)** ↓	1.123e-02 (1.142e-02) ↔
Christofides_7_5_0.5	E_s	5.024e-02 (1.786e-01)	**1.137e-02 (1.150e-02)** ↔	2.160e-02 (2.288e-02) ↔	2.642e-02 (2.183e-02) ↑	2.257e-02 (3.559e-02) ↔
Christofides_8_5_0.5	E_s	**2.076e-02 (2.086e-02)**	3.154e-02 (4.284e-02) ↔	2.995e-02 (4.260e-02) ↔	2.867e-02 (2.710e-02) ↔	9.756e-03 (7.278e-03) ↑
Christofides_9_5_0.5	E_s	**1.691e-02 (2.457e-02)**	2.122e-02 (9.598e-03) ↔	4.884e-02 (1.119e-01) ↔	3.745e-02 (6.514e-02) ↓	8.085e-03 (1.548e-02) ↑
Christofides_10_5_0.5	E_s	1.899e-02 (1.772e-02)	3.632e-02 (8.284e-02) ↔	2.131e-02 (1.989e-02) ↔	2.816e-02 (4.400e-02) ↔	**1.738e-02 (3.193e-02)** ↔
Christofides_11_5_0.5	E_s	**1.331e-02 (2.003e-02)**	1.393e-02 (1.722e-02) ↔	3.366e-02 (4.952e-02) ↓	2.411e-02 (3.747e-02) ↓	6.926e-03 (3.587e-03) ↔
Christofides_12_5_0.5	E_s	6.954e-02 (1.651e-01)	**1.312e-02 (1.105e-02)** ↑	2.244e-02 (3.760e-02) ↔	8.850e-02 (2.524e-01) ↔	2.523e-02 (5.054e-02) ↔
+\≈\−			0 \11 \1	0 \8 \4	1 \6 \5	5 \7 \0
Christofides_1_5_0.5	tiempo (secs)	3.187e+02 (6.764e+00)	5.063e+02 (8.489e+00) ↓	5.785e+02 (7.062e+00) ↓	1.437e+03 (2.844e+01) ↓	**1.999e+02 (2.955e+00)** ↑
Christofides_2_5_0.5	tiempo (secs)	5.101e+02 (4.762e+00)	7.158e+02 (3.721e+00) ↓	6.556e+02 (5.365e+00) ↓	1.574e+03 (2.853e+01) ↓	**3.019e+02 (3.775e+00)** ↑
Christofides_3_5_0.5	tiempo (secs)	7.598e+02 (7.212e+00)	1.205e+03 (1.410e+01) ↓	7.780e+02 (7.015e+00) ↓	1.750e+03 (4.589e+01) ↓	**4.356e+02 (4.307e+00)** ↑
Christofides_4_5_0.5	tiempo (secs)	1.435e+03 (2.662e+01)	1.788e+03 (2.926e+01) ↓	1.073e+03 (1.888e+01) ↑	2.101e+03 (4.394e+01) ↓	**7.713e+02 (6.593e+00)** ↑
Christofides_5_5_0.5	tiempo (secs)	2.451e+03 (5.097e+01)	2.614e+03 (3.515e+01) ↓	1.485e+03 (4.634e+01) ↑	2.773e+03 (3.177e+01) ↓	**1.321e+03 (2.427e+01)** ↑
Christofides_6_5_0.5	tiempo(secs)	3.079e+02 (4.142e+00)	4.989e+02 (9.453e+00) ↓	5.837e+02 (5.751e+01) ↓	1.432e+03 (1.534e+01) ↓	**1.947e+02 (2.048e+00)** ↑
Christofides_7_5_0.5	tiempo (secs)	5.152e+02 (9.341e+00)	7.409e+02 (7.008e+00) ↓	6.696e+02 (9.277e+00) ↓	1.622e+03 (1.849e+02) ↓	**3.033e+02 (5.586e+00)** ↑
Christofides_8_5_0.5	tiempo (secs)	7.840e+02 (2.035e+01)	1.140e+03 (1.574e+01) ↓	8.022e+02 (2.087e+01) ↓	1.740e+03 (2.107e+01) ↓	**4.508e+02 (1.616e+01)** ↑
Christofides_9_5_0.5	tiempo (secs)	1.465e+03 (2.695e+01)	1.739e+03 (1.143e+01) ↓	1.091e+03 (2.388e+01) ↑	2.206e+03 (7.875e+01) ↓	**7.776e+02 (5.122e+00)** ↑
Christofides_10_5_0.5	tiempo (secs)	2.468e+03 (2.390e+01)	2.610e+03 (5.411e+01) ↓	1.510e+03 (3.300e+01) ↑	2.796e+03 (1.156e+02) ↓	**1.288e+03 (3.719e+01)** ↑
Christofides_11_5_0.5	tiempo (secs)	1.013e+03 (5.086e+01)	1.387e+03 (1.794e+01) ↓	9.138e+02 (1.309e+01) ↑	1.757e+03 (1.685e+01) ↓	**5.549e+02 (1.685e+01)** ↑
Christofides_12_5_0.5	tiempo (secs)	7.906e+02 (2.220e+01)	1.102e+03 (9.269e+00) ↓	8.223e+02 (4.998e+01) ↓	1.945e+03 (6.719e+02) ↓	**4.518e+02 (4.652e+01)** ↑
+\≈\−			0 \0 \12	5 \0 \7	0 \0 \12	12 \0 \0

Tabla 4.5: Calidad de las soluciones aproximadas para cada variante de cMIBACO.

4.3.1. Análisis de indicadores de calidad

En la Tabla 4.6 se muestran el promedio y desviación estándar para cada indicador de calidad de las 20 ejecuciones por cada algoritmo para cada problema. Junto al valor de cada indicador se indica con una flecha ↑ cuando existe una diferencia significativa de acuerdo con la prueba de Wilcoxon entre los resultados de cMIBACO con cada uno y si éste fue mejor. Pero en el caso de existir una diferencia significativa y los resultados de cMIBACO no son mejores, se utiliza la flecha ↓. Cuando no existe diferencia significativa se utiliza la flecha ↔.

De acuerdo con el hipervolumen, en siete problemas el cMIBACO logra tener mejor promedio de este indicador y aquellos algoritmos que en alguna instancia son mejores que cMIBACO no logran ser mejores en más instancias con diferencia significativa. Pues los otros algoritmos logran ser mejor que cMIBACO en a lo más 2 instancias con diferencia significativa,por ejemplo, IBACO_ϵ^+ es mejor en tres problemas pero solo en dos existe diferencia significa (Christofides_8_5_0.5 y Christofides_11_5_0.5). También se tiene que cMIBACO fue mejor en la mayoría de problemas grandes de entre 150 y 200 clientes, por lo que es mejor cMIBACO ante problemas de MOGen-

ConVRP más difíciles. Por otro lado, el IBACO$_{HV}$ tuvo el desempeño más bajo, siendo el algoritmo más superado.

Para el indicador de $R2$, el cMIBACO es mejor en cuatro problemas y aunque es superado en a lo más cinco problemas, los problemas para los que cMIBACO se desempeña mejor son los más grandes de 150 y 200 clientes. El algoritmo con mejor desempeño de acuerdo con este indicador es el IBACO$_{ws}$, superando a cMIBACO en cinco instancias y perdiendo en solo una. Aunque las instancias en las que es mejor tienen de 75 y 100 clientes, por lo que IBACO$_{ws}$ funciona bien para problemas medianos y lo mismo sucede con IBACO$_{\epsilon+}$ donde es el mejor para problemas de entre 50 y 120 clientes. Al igual que el indicador anterior, el IBACO$_{HV}$ fue el algoritmo con peor desempeño.

En la Figura 4.2 se muestra el diagrama de diferencias críticas maximizando los indicadores de calidad y considerando el tiempo de ejecución de los algoritmos. Se puede notar que cMIBACO obtiene un mejor rango y con diferencia significativa, seguido por IBACO$_{\epsilon+}$ y luego IBACO$_{R2}$.

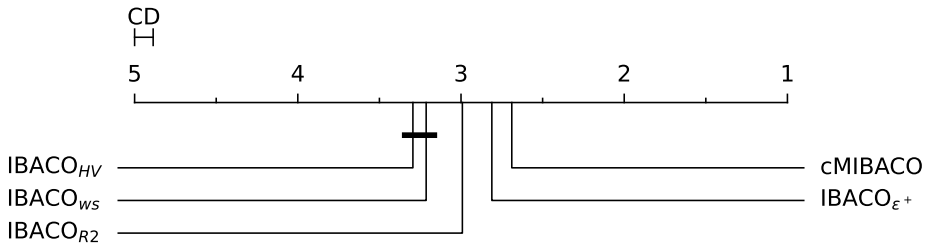


Figura 4.2: Diagrama de diferencias críticas para la comparación de indicadores cooperativos, individual y suma ponderada.

De acuerdo con los resultados se listan las siguientes conclusiones:

- El cMIBACO logra obtener mejores soluciones aproximadas de acuerdo con los tres indicadores.
- Existe diferencia significativa entre usar un indicador individual y usar múltiples indicadores.
- El IBACO$_{\epsilon+}$ fue el más cercano a cMIBACO.
- La desventaja de usar IBACO$_{ws}$ está en los tiempos que tarda en calcular cada indicador para cada solución.
- Una desventaja en la ejecución de cMIBACO es controlar que tanta población le asigna a cada subalgoritmo.

En el Apéndice A se muestran frentes en espacio de los objetivos obtenidos con los algoritmos para algunos problemas. De acuerdo con los resultados, el cMIBACO logra tener en la mayoría, una mejor distribución de soluciones. Se puede notar en todos los algoritmos que las instancias no tienen un espacio objetivo tan uniforme o estructurado como suele suceder en problemas de optimización continuos bien conocidos. De acuerdo con las gráficas de convergencia en el mismo apéndice, en algunas instancias la mediana de cMIBACO puede superar a los algoritmos y en otras compite con los demás.

Problema	Indicador	cMIBACO	IBACO$_{\epsilon^+}$	IBACO$_{HV}$	IBACO$_{R2}$	IBACO$_{ws}$
Christofides_1_5_0.5	HV	2.356e+07 (4.600e+05)	2.364e+07 (3.629e+05) ↔	**2.408e+07 (3.762e+05)** ↑	2.381e+07 (3.926e+05) ↑	2.363e+07 (3.911e+05) ↔
Christofides_2_5_0.5	HV	**1.253e+07 (1.472e+060)**	1.114e+07 (1.323e+06) ↓	1.189e+07 (1.631e+06) ↔	1.139e+07 (1.451e+06) ↔	1.107e+07 (1.177e+06) ↓
Christofides_3_5_0.5	HV	9.193e+06 (7.931e+05)	**9.299e+06 (6.488e+05)** ↔	9.126e+06 (8.565e+05) ↔	9.172e+06 (7.113e+05) ↔	9.188e+06 (7.123e+05) ↔
Christofides_4_5_0.5	HV	**2.244e+07 (2.588e+05)**	2.214e+07 (2.765e+05) ↓	2.199e+07 (3.181e+05) ↓	2.239e+07 (1.585e+05) ↔	2.240e+07 (2.495e+05) ↔
Christofides_5_5_0.5	HV	**4.660e+07 (3.363e+05)**	4.591e+07 (4.525e+05) ↓	4.559e+07 (5.198e+05) ↓	4.622e+07 (3.686e+05) ↓	4.634e+07 (3.906e+05) ↓
Christofides_6_5_0.5	HV	7.633e+07 (9.282e+05)	7.577e+07 (1.215e+06) ↔	**7.719e+07 (7.305e+05)** ↑	7.686e+07 (1.058e+06) ↔	7.574e+07 (1.009e+06) ↔
Christofides_7_5_0.5	HV	**3.700e+07 (3.022e+06)**	3.505e+07 (2.264e+06) ↔	3.631e+07 (2.849e+06) ↔	3.497e+07 (1.946e+06) ↓	3.565e+07 (2.484e+06) ↔
Christofides_8_5_0.5	HV	4.160e+07 (1.573e+06)	**4.176e+07 (5.183e+05)** ↑	4.165e+07 (1.678e+06) ↔	4.142e+07 (4.032e+05) ↔	4.197e+07 (1.364e+06) ↑
Christofides_9_5_0.5	HV	**4.953e+07 (5.146e+05)**	4.913e+07 (4.763e+05) ↓	4.887e+07 (5.185e+05) ↓	4.928e+07 (4.409e+05) ↔	4.942e+07 (3.839e+05) ↔
Christofides_10_5_0.5	HV	**6.925e+07 (8.445e+05)**	6.789e+07 (2.448e+06) ↓	6.818e+07 (5.736e+05) ↓	6.882e+07 (6.953e+05) ↔	6.817e+07 (2.457e+06) ↓
Christofides_11_5_0.5	HV	7.933e+07 (4.862e+06)	**8.165e+07 (5.450e+06)** ↑	8.015e+07 (5.812e+06) ↔	7.729e+07 (3.095e+06) ↔	7.759e+07 (1.738e+06) ↔
Christofides_12_5_0.5	HV	**9.563e+07 (6.783e+06)**	9.304e+07 (5.449e+06) ↔	9.384e+07 (6.536e+06) ↔	9.448e+07 (6.613e+06) ↔	9.110e+07 (2.940e+06) ↓
+ \≈ \−			2 \5 \5	2 \6 \4	1 \9 \2	1 \7 \4
Christofides_1_5_0.5	R2	7.926e+02 (1.636e+01)	**7.838e+02 (1.075e+01)** ↔	7.892e+02 (8.816e+00) ↔	7.857e+02 (9.598e+00) ↔	7.853e+02 (6.107e+00) ↔
Christofides_2_5_0.5	R2	1.161e+03 (1.809e+01)	1.146e+03 (1.655e+01) ↑	1.171e+03 (1.943e+01) ↔	1.149e+03 (1.354e+01) ↔	**1.142e+03 (1.729e+01)** ↑
Christofides_3_5_0.5	R2	1.362e+03 (2.338e+01)	1.351e+03 (1.663e+01) ↔	1.386e+03 (1.800e+01) ↓	1.375e+03 (1.310e+01) ↓	**1.348e+03 (1.632e+01)** ↑
Christofides_4_5_0.5	R2	**1.679e+03 (2.756e+01)**	1.707e+03 (2.152e+01) ↓	1.728e+03 (2.001e+01) ↓	1.685e+03 (1.532e+01) ↔	1.682e+03 (2.088e+01) ↔
Christofides_5_5_0.5	R2	**2.101e+03 (1.732e+01)**	2.144e+03 (2.757e+01) ↓	2.160e+03 (2.723e+01) ↓	2.114e+03 (1.852e+01) ↓	2.117e+03 (2.183e+01) ↓
Christofides_6_5_0.5	R2	8.038e+02 (1.429e+01)	7.960e+02 (9.215e+00) ↑	7.986e+02 (8.883e+00) ↔	**7.904e+02 (1.080e+01)** ↑	7.988e+02 (1.063e+01) ↔
Christofides_7_5_0.5	R2	1.190e+03 (1.476e+01)	1.171e+03 (1.779e+01) ↑	1.199e+03 (1.072e+01) ↓	1.184e+03 (6.881e+00) ↔	**1.171e+03 (1.375e+01)** ↑
Christofides_8_5_0.5	R2	1.483e+03 (1.812e+01)	**1.454e+03 (2.228e+01)** ↑	1.496e+03 (2.000e+01) ↔	1.476e+03 (2.459e+01) ↔	1.468e+03 (1.818e+01) ↑
Christofides_9_5_0.5	R2	**1.869e+03 (3.464e+01)**	1.886e+03 (2.225e+01) ↓	1.917e+03 (2.649e+01) ↓	1.878e+03 (2.557e+01) ↔	1.873e+03 (1.451e+01) ↔
Christofides_10_5_0.5	R2	**2.345e+03 (4.193e+01)**	2.365e+03 (2.123e+01) ↔	2.384e+03 (3.216e+01) ↓	2.357e+03 (2.205e+01) ↔	2.352e+03 (2.551e+01) ↔
Christofides_11_5_0.5	R2	2.215e+03 (5.230e+01)	**2.104e+03 (7.186e+01)** ↑	2.198e+03 (4.696e+01) ↔	2.181e+03 (4.970e+01) ↔	2.140e+03 (5.552e+01) ↑
Christofides_12_5_0.5	R2	1.520e+03 (2.505e+01)	1.517e+03 (2.063e+01) ↔	1.538e+03 (2.903e+01) ↓	1.515e+03 (2.187e+01) ↔	**1.511e+03 (1.587e+01)** ↔
+ \≈ \−			5 \4 \3	0 \5 \7	1 \9 \2	5 \6 \1
Christofides_1_5_0.5	E_a	6.900e-02 (2.458e-01)	9.202e-03 (1.012e-02) ↔	**1.289e-02 (9.903e-03)** ↔	3.424e-02 (1.054e-01) ↔	1.918e-02 (2.971e-02) ↔
Christofides_2_5_0.5	E_a	6.551e-02 (1.278e-01)	**1.319e-02 (1.219e-02)** ↔	2.655e-02 (4.178e-02) ↔	3.892e-02 (9.674e-02) ↔	2.323e-02 (3.446e-02) ↔
Christofides_3_5_0.5	E_a	1.746e-02 (2.597e-02)	2.727e-02 (5.449e-02) ↔	1.046e-02 (1.133e-02) ↔	1.126e-02 (8.593e-03) ↔	**1.039e-02 (1.112e-02)** ↔
Christofides_4_5_0.5	E_a	3.010e-02 (4.459e-02)	2.520e-02 (3.343e-02) ↔	4.887e-02 (1.214e-01) ↔	2.459e-02 (2.256e-02) ↔	**1.958e-02 (1.988e-02)** ↔
Christofides_5_5_0.5	E_a	**1.088e-02 (8.863e-03)**	6.029e-02 (2.248e-01) ↔	1.510e-02 (2.048e-02) ↔	1.228e-02 (7.524e-03) ↔	2.152e-02 (1.694e-02) ↓
Christofides_6_5_0.5	E_a	1.585e-02 (1.516e-02)	2.061e-02 (3.160e-02) ↔	2.504e-02 (2.153e-02) ↔	**1.445e-02 (1.247e-02)** ↔	3.445e-02 (6.497e-02) ↔
Christofides_7_5_0.5	E_a	1.137e-02 (1.150e-02)	1.497e-02 (1.710e-02) ↔	2.099e-02 (2.786e-02) ↔	1.430e-02 (1.124e-02) ↔	**1.030e-02 (7.273e-03)** ↔
Christofides_8_5_0.5	E_a	3.154e-02 (4.284e-02)	**1.837e-02 (2.086e-02)** ↔	2.461e-02 (3.371e-02) ↔	4.099e-02 (6.784e-02) ↔	2.077e-02 (3.484e-02) ↔
Christofides_9_5_0.5	E_a	2.122e-02 (9.598e-03)	7.590e-02 (1.892e-01) ↔	3.849e-02 (4.231e-02) ↔	3.001e-02 (4.016e-02) ↔	**1.589e-02 (8.206e-03)** ↔
Christofides_10_5_0.5	E_a	3.632e-02 (8.284e-02)	2.565e-02 (3.172e-02) ↔	2.573e-02 (3.497e-02) ↔	**1.284e-02 (8.846e-03)** ↔	1.986e-02 (1.081e-02) ↔
Christofides_11_5_0.5	E_a	**1.393e-02 (1.722e-02)**	5.468e-02 (1.408e-01) ↔	2.049e-02 (3.403e-02) ↔	1.862e-02 (3.679e-02) ↔	2.349e-02 (5.161e-02) ↔
Christofides_12_5_0.5	E_a	1.312e-02 (1.105e-02)	2.606e-02 (5.258e-02) ↔	**1.225e-02 (7.042e-03)** ↔	3.312e-02 (9.931e-02) ↔	5.006e-02 (1.148e-01) ↔
+ \≈ \−			0 \12 \0	0 \12 \0	0 \12 \0	0 \11 \1
Christofides_1_5_0.5	tiempo (secs)	5.063e+02 (8.489e+00)	**4.767e+02 (9.984e+00)** ↑	5.843e+02 (4.492e+00) ↓	5.571e+02 (4.127e+00) ↓	8.280e+02 (8.321e+00) ↓
Christofides_2_5_0.5	tiempo (secs)	**7.158e+02 (3.721e+00)**	7.542e+02 (1.471e+01) ↓	7.670e+02 (6.427e+00) ↓	7.685e+02 (3.952e+00) ↓	1.083e+03 (4.258e+00) ↓
Christofides_3_5_0.5	tiempo (secs)	1.205e+03 (1.410e+01)	1.400e+03 (2.114e+01) ↓	**1.167e+03 (1.365e+01)** ↑	1.236e+03 (8.110e+00) ↓	1.718e+03 (2.806e+01) ↓
Christofides_4_5_0.5	tiempo (secs)	1.788e+03 (2.926e+01)	2.239e+03 (5.221e+01) ↓	**1.657e+03 (1.406e+01)** ↑	1.805e+03 (9.246e+00) ↓	2.478e+03 (3.002e+01) ↓
Christofides_5_5_0.5	tiempo (secs)	2.614e+03 (3.515e+01)	3.621e+03 (5.399e+01) ↓	**2.393e+03 (2.910e+01)** ↑	2.829e+03 (6.540e+01) ↓	3.557e+03 (4.742e+01) ↓
Christofides_6_5_0.5	tiempo (secs)	**4.989e+02 (9.453e+00)**	4.999e+02 (6.346e+00) ↔	5.674e+02 (2.527e+00) ↓	5.788e+02 (5.381e+00) ↓	8.162e+02 (7.446e+00) ↓
Christofides_7_5_0.5	tiempo (secs)	**7.409e+02 (7.008e+00)**	7.674e+02 (1.124e+01) ↓	7.845e+02 (8.373e+00) ↓	8.244e+02 (8.334e+00) ↓	1.119e+03 (1.059e+01) ↓
Christofides_8_5_0.5	tiempo (secs)	1.140e+03 (1.574e+01)	1.323e+03 (1.130e+01) ↓	**1.127e+03 (9.262e+00)** ↑	1.252e+03 (1.957e+01) ↓	1.584e+03 (2.136e+01) ↓
Christofides_9_5_0.5	tiempo (secs)	1.739e+03 (1.143e+01)	2.144e+03 (1.887e+01) ↓	**1.647e+03 (1.318e+01)** ↑	1.918e+03 (7.318e+00) ↓	2.392e+03 (1.168e+01) ↓
Christofides_10_5_0.5	tiempo (secs)	2.610e+03 (5.411e+01)	3.566e+03 (7.957e+01) ↓	**2.430e+03 (2.790e+01)** ↑	2.886e+03 (3.081e+01) ↓	3.574e+03 (3.270e+01) ↓
Christofides_11_5_0.5	tiempo (secs)	1.387e+03 (1.794e+01)	1.796e+03 (7.061e+01) ↓	**1.366e+03 (2.329e+01)** ↑	1.595e+03 (1.634e+01) ↓	1.922e+03 (3.604e+01) ↓
Christofides_12_5_0.5	tiempo (secs)	1.102e+03 (9.269e+00)	1.338e+03 (1.295e+01) ↓	**1.095e+03 (1.286e+01)** ↔	1.249e+03 (1.111e+01) ↓	1.560e+03 (2.030e+01) ↓
+ \≈ \−			1 \1 \10	7 \1 \4	0 \0 \12	0 \0 \12

Tabla 4.6: Calidad de las soluciones aproximadas para cada algoritmo basado en indicadores.

4.4. Desempeño en optimización robusta

En esta sección se analizan los resultados de realizar optimización robusta con cMIBACO para el MOGenConVRP bajo incertidumbre. Usando como escenario nominal aquellas instancias donde el cliente requiere ser visitado el 50 % de los días en el horizonte de planeación.

Primero se analiza la calidad de las soluciones nominales frente a los escenarios con frecuencias de 70 % y 90 %. Luego la comparación de las soluciones ligeramente robustas obtenidas con cMIBACO frente a las soluciones ligeramente robustas obtenidas con los indicadores individuales y la suma ponderada. Después se analiza la pérdida de calidad de las soluciones robustas respecto a las soluciones aproximadas con cMIBACO.

4.4.1. Calidad de soluciones nominales frente a otros escenarios

Se presenta la comparación entre las soluciones aproximadas obtenidas con cMIBACO en el escenario nominal frente a estas mismas soluciones evaluadas en los escenarios de 70 % y 90 %. En la Tabla 4.7 se muestran el promedio y desviación estándar por indicador de las soluciones

aproximadas obtenidas en 20 ejecuciones para el escenario nominal y las mismas soluciones en los demás escenarios.

Comenzando con el indicador de hipervolumen, es claro que para las soluciones nominales su valor es mayor a comparación de los otros escenarios. Seguido por el escenario de frecuencias 70 % y en cuanto al escenario de 90 %, en éste se presenta una mayor dificultad.

Debido a los valores de los objetivos, el valor de hipervolumen entre algoritmos en algunos casos es muy notable. Por lo que también conviene tomar en cuenta los demás indicadores, continuando con R2 donde las soluciones del escenario nominal son las más cercanas al punto ideal.

Aunque los otros dos escenarios son peores bajo este indicador, se puede notar que las diferencias no son tan amplias. En cuanto a la diversidad de soluciones con Riesz s-energy, el escenario nominal sigue teniendo mejores resultados. Sin embargo, el escenario 90 % tiene en bastantes problemas mejores valores que el escenario 70 %.

4.4.2. Desempeño de soluciones robustamente ligeras con cMIBACO

Se compara el desempeño entre los algoritmos IBACO$_{HV}$, IBACO$_{R2}$, IBACO$_{\epsilon^+}$, IBACO$_{ws}$ contra el cMIBACO para obtener soluciones ligeramente robustas. Se considera como escenario nominal el conjunto de soluciones aproximadas de cada uno de los algoritmos. En la Tabla 4.9 se muestran el promedio y desviación estándar de cada indicador de las soluciones robustas obtenidas en las 20 ejecuciones de cada algoritmo. Realizando análisis estadístico con la prueba de Wilcoxon con intervalo de confianza del 95 %.

De acuerdo con el hipervolumen, solo IBACOϵ^+ supera en un problema a cMIBACO con diferencia significativa, por otro lado, los otros algoritmos son superados en a lo más tres problemas. Se tienen más problemas donde no existe diferencia significativa, aunque cMIBACO logra ser mejor en cinco problemas que tienen de 75 a 200 clientes y es el algoritmo que más veces termina en primer lugar. Para R2 sucede un caso similar, donde solo IBACOϵ^+ e IBACO$_{ws}$ superan con diferencia significativa en a lo más dos problemas a cMIBACO, estos problemas son de 50 y 120 clientes. Mientras que IBACO$_{HV}$ es el que tiene un menor desempeño respecto a hipervolumen y R2.

En cuanto a Riesz s-energy, solo IBACO$_{ws}$ logra superar con diferencia significativa en un problema a cMIBACO. El IBACO$_{R2}$ logra ser el algoritmo con mejor promedio en la mayoría de los problemas pero sin diferencia significativa.

En las Figuras 4.3, 4.4 y 4.5 se muestran los frentes en espacio de los objetivos de las soluciones ligeramente robustas. En la mayoría de los problemas se puede notar que el tercer objetivo referente a la diferencia entre los tiempos de llegada tiene una mayor diversidad de soluciones. Para el segundo objetivo, varios problemas tienen soluciones bastante diversas con valores de 3 a 5. Por otro lado, las soluciones en todos los problemas tienen menor variación en el primer objetivo.

También se analizó la pérdida de calidad de las soluciones ligeramente robustas respecto a las soluciones aproximadas con cMIBACO. En la Tabla 4.8 se muestran el promedio, desviación estándar y porcentajes de las diferencias entre soluciones robustas y aproximadas de acuerdo con el hipervolumen y los tres objetivos.

Los resultados para el hipervolumen muestran que la mitad de los problemas tienen una pérdida de calidad debajo de 40 %, cuatro tienen una pérdida debajo de 50 % y la pérdida máxima es de 57.56 % para Christofides_4_5_0.5. En muchas instancias la pérdida de calidad es debido al primer objetivo, para éste siete instancias tienen una pérdida de entre 30 % − 40 %, con 36.75 % como el máximo. Por otro lado, la máxima pérdida de calidad para el segundo objetivo es de 17.29 % en Christofides_9_5_0.5. Y el tercer objetivo no tiene pérdidas tan grandes como el primero, pues siete

Problema	Indicador	Escenario nominal %50	Escenario %70	Escenario %90
Christofides_1_5_0.5	HV	**2.362e+07 (3.630e+05)**	1.705e+07 (3.022e+05)	1.142e+07 (2.830e+05)
Christofides_2_5_0.5	HV	**1.089e+07 (1.195e+06)**	4.811e+06 (8.293e+05)	2.948e+06 (6.775e+04)
Christofides_3_5_0.5	HV	**8.694e+06 (5.425e+05)**	5.113e+06 (1.170e+05)	2.280e+06 (7.752e+04)
Christofides_4_5_0.5	HV	**2.006e+07 (2.609e+05)**	8.511e+06 (2.024e+05)	4.480e+06 (5.252e+05)
Christofides_5_5_0.5	HV	**4.261e+07 (1.430e+06)**	2.055e+07 (1.638e+06)	1.190e+07 (3.261e+05)
Christofides_6_5_0.5	HV	**7.557e+07 (9.252e+05)**	5.713e+07 (8.513e+05)	4.514e+07 (5.744e+05)
Christofides_7_5_0.5	HV	**3.412e+07 (1.715e+06)**	1.791e+07 (7.978e+05)	1.479e+07 (1.783e+05)
Christofides_8_5_0.5	HV	**4.020e+07 (3.961e+05)**	2.681e+07 (1.824e+06)	1.907e+07 (3.525e+05)
Christofides_9_5_0.5	HV	**4.589e+07 (3.642e+05)**	2.533e+07 (5.171e+05)	1.722e+07 (1.406e+06)
Christofides_10_5_0.5	HV	**6.104e+07 (3.414e+06)**	3.567e+07 (2.967e+06)	2.254e+07 (5.310e+05)
Christofides_11_5_0.5	HV	**7.452e+07 (2.238e+06)**	5.511e+07 (1.272e+06)	4.084e+07 (2.475e+06)
Christofides_12_5_0.5	HV	**8.810e+07 (7.316e+05)**	7.084e+07 (8.223e+05)	4.847e+07 (8.021e+05)
Christofides_1_5_0.5	R2	**8.079e+02 (6.003e+00)**	1.203e+03 (1.530e+01)	1.521e+03 (1.654e+01)
Christofides_2_5_0.5	R2	**1.203e+03 (1.474e+01)**	1.841e+03 (2.619e+01)	2.238e+03 (2.986e+01)
Christofides_3_5_0.5	R2	**1.464e+03 (2.265e+01)**	1.964e+03 (2.591e+01)	2.414e+03 (3.164e+01)
Christofides_4_5_0.5	R2	**1.914e+03 (2.985e+01)**	2.719e+03 (2.505e+01)	3.178e+03 (3.411e+01)
Christofides_5_5_0.5	R2	**2.337e+03 (2.988e+01)**	3.326e+03 (3.260e+01)	3.981e+03 (3.535e+01)
Christofides_6_5_0.5	R2	**8.157e+02 (6.814e+00)**	1.238e+03 (1.636e+01)	1.486e+03 (1.548e+01)
Christofides_7_5_0.5	R2	**1.231e+03 (1.959e+01)**	1.912e+03 (2.766e+01)	2.233e+03 (1.796e+01)
Christofides_8_5_0.5	R2	**1.565e+03 (2.210e+01)**	2.056e+03 (2.564e+01)	2.423e+03 (3.311e+01)
Christofides_9_5_0.5	R2	**2.092e+03 (2.089e+01)**	2.727e+03 (2.793e+01)	3.139e+03 (3.759e+01)
Christofides_10_5_0.5	R2	**2.610e+03 (3.335e+01)**	3.243e+03 (2.814e+01)	3.915e+03 (3.913e+01)
Christofides_11_5_0.5	R2	**2.240e+03 (3.439e+01)**	2.931e+03 (6.057e+01)	3.298e+03 (6.601e+01)
Christofides_12_5_0.5	R2	**1.604e+03 (2.335e+01)**	2.120e+03 (3.021e+01)	2.738e+03 (4.400e+01)
Christofides_1_5_0.5	E_s	**1.716e-01 (9.222e-02)**	1.965e+00 (2.274e+00)	3.644e+00 (8.806e+00)
Christofides_2_5_0.5	E_s	**3.310e-01 (2.474e-01)**	2.267e+00 (1.475e+00)	1.436e+00 (1.833e+00)
Christofides_3_5_0.5	E_s	**6.038e-01 (3.908e-01)**	2.415e+00 (8.357e-01)	2.293e+00 (1.521e+00)
Christofides_4_5_0.5	E_s	**5.585e-01 (1.932e-01)**	2.448e+00 (7.647e-01)	3.745e+00 (5.226e+00)
Christofides_5_5_0.5	E_s	**4.370e-01 (1.242e-01)**	1.910e+00 (3.921e-01)	2.277e+00 (1.588e+00)
Christofides_6_5_0.5	E_s	**2.412e-01 (7.816e-02)**	1.427e+00 (8.028e-01)	2.458e+00 (1.965e+00)
Christofides_7_5_0.5	E_s	**4.831e-01 (3.560e-01)**	4.247e+00 (8.826e+00)	1.779e+00 (6.647e-01)
Christofides_8_5_0.5	E_s	**5.068e-01 (1.216e-01)**	2.871e+00 (2.225e+00)	1.575e+00 (5.012e-01)
Christofides_9_5_0.5	E_s	**5.813e-01 (2.103e-01)**	3.399e+00 (2.230e+00)	2.034e+00 (9.530e-01)
Christofides_10_5_0.5	E_s	**4.740e-01 (2.132e-01)**	1.813e+00 (8.624e-01)	2.684e+00 (2.146e+00)
Christofides_11_5_0.5	E_s	**6.965e-01 (1.844e-01)**	3.972e+00 (4.875e+00)	2.506e+00 (8.016e-01)
Christofides_12_5_0.5	E_s	**6.201e-01 (2.986e-01)**	3.647e+00 (2.062e+00)	1.848e+00 (4.741e-01)

Tabla 4.7: Promedio y desviación estándar de las soluciones obtenidas por escenario en cada problema de acuerdo con los indicadores $HV, R2, E_s$.

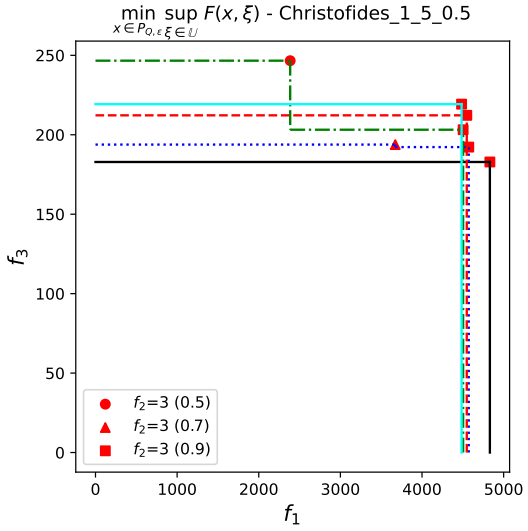
instancias tienen una pérdida debajo de 20 % y el máximo de 26.52 %.

Problema	HV	f_1	f_2	f_3
Christofides_1_5_0.5	7.563e+06 (2.790e+06) [32.10 %]	1.512e+03 (1.604e+02) [36.75 %]	3.902e-01 (1.756e-01) [12.73 %]	2.515e+01 (1.487e+01) [12.41 %]
Christofides_2_5_0.5	5.517e+06 (2.171e+06) [44.03 %]	2.185e+03 (1.225e+02) [36.19 %]	6.820e-01 (1.195e-01) [16.20 %]	3.476e+01 (1.011e+01) [19.48 %]
Christofides_3_5_0.5	3.786e+06 (1.276e+06) [41.19 %]	1.937e+03 (1.602e+02) [29.36 %]	4.607e-01 (8.637e-02) [11.58 %]	1.999e+01 (9.056e+00) [8.55 %]
Christofides_4_5_0.5	1.291e+07 (3.013e+06) [57.56 %]	3.152e+03 (1.920e+02) [35.41 %]	7.839e-01 (1.594e-01) [16.77 %]	4.127e+01 (1.040e+01) [18.10 %]
Christofides_5_5_0.5	2.529e+07 (6.533e+06) [54.27 %]	3.776e+03 (3.085e+02) [34.45 %]	7.047e-01 (7.580e-02) [14.22 %]	4.438e+01 (1.117e+01) [20.36 %]
Christofides_6_5_0.5	2.015e+07 (5.669e+06) [26.39 %]	1.324e+03 (1.724e+02) [33.14 %]	3.298e-01 (1.456e-01) [10.47 %]	5.564e+01 (1.612e+01) [20.93 %]
Christofides_7_5_0.5	1.262e+07 (6.003e+06) [34.11 %]	2.113e+03 (1.514e+02) [34.69 %]	6.114e-01 (1.639e-01) [14.43 %]	4.953e+01 (9.931e+00) [20.43 %]
Christofides_8_5_0.5	1.598e+07 (4.353e+06) [38.40 %]	1.735e+03 (1.569e+02) [25.58 %]	6.223e-01 (1.399e-01) [14.64 %]	8.685e+01 (1.039e+01) [26.52 %]
Christofides_9_5_0.5	2.393e+07 (3.348e+06) [48.32 %]	2.354e+03 (1.041e+02) [26.90 %]	8.401e-01 (1.266e-01) [17.29 %]	6.774e+01 (1.062e+01) [22.46 %]
Christofides_10_5_0.5	3.421e+07 (7.437e+06) [49.40 %]	2.814e+03 (3.485e+02) [25.96 %]	6.117e-01 (7.080e-02) [12.32 %]	4.918e+01 (1.025e+01) [17.26 %]
Christofides_11_5_0.5	2.234e+07 (7.242e+06) [28.15 %]	1.547e+03 (1.659e+02) [16.04 %]	5.038e-01 (1.721e-01) [12.42 %]	4.119e+01 (1.580e+01) [16.56 %]
Christofides_12_5_0.5	2.554e+07 (1.303e+07) [26.71 %]	2.314e+03 (1.824e+02) [30.71 %]	4.555e-01 (1.081e-01) [10.88 %]	4.713e+01 (1.090e+01) [19.18 %]

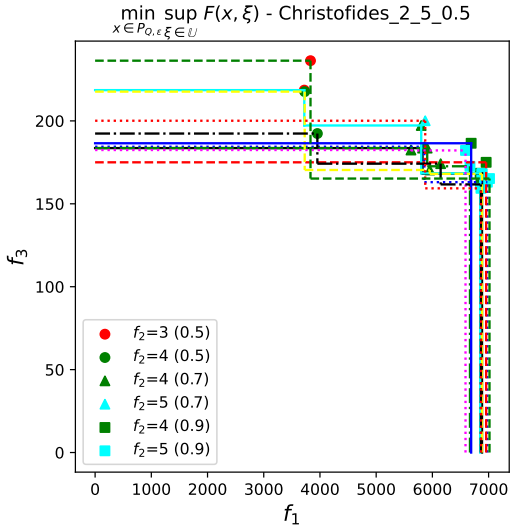
Tabla 4.8: Pérdida de calidad entre soluciones aproximadas y robustas.

Problema	Indicador	cMIBACO	IBACO$_{\epsilon+}$	IBACO$_{HV}$	IBACO$_{R2}$	IBACO$_{ws}$
Christofides_1_5_0.5	HV	1.599e+07 (2.841e+06)	1.636e+07 (3.053e+06) ↔	1.695e+07 (2.696e+06) ↔	1.690e+07 (2.795e+06) ↔	**1.732e+07 (2.003e+06)** ↔
Christofides_2_5_0.5	HV	**7.012e+06 (1.588e+06)**	6.952e+06 (1.437e+06) ↔	6.637e+06 (1.526e+06) ↔	6.361e+06 (2.154e+06) ↔	6.641e+06 (1.867e+06) ↔
Christofides_3_5_0.5	HV	**5.406e+06 (1.169e+06)**	5.150e+06 (1.245e+06) ↔	4.627e+06 (1.027e+06) ↓	5.077e+06 (1.224e+06) ↔	3.959e+06 (1.356e+06) ↓
Christofides_4_5_0.5	HV	9.523e+06 (2.983e+06)	9.795e+06 (3.312e+06) ↔	**1.093e+07 (3.464e+06)** ↔	1.060e+07 (2.657e+06) ↔	8.993e+06 (2.900e+06) ↔
Christofides_5_5_0.5	HV	2.131e+07 (6.721e+06)	2.021e+07 (5.167e+06) ↔	2.113e+07 (5.296e+06) ↔	**2.138e+07 (6.943e+06)** ↔	2.096e+07 (6.113e+06) ↔
Christofides_6_5_0.5	HV	5.618e+07 (5.967e+06)	**5.717e+07 (6.887e+06)** ↔	5.469e+07 (7.426e+06) ↔	5.393e+07 (5.925e+06) ↔	5.469e+07 (5.187e+06) ↔
Christofides_7_5_0.5	HV	2.438e+07 (3.916e+06)	2.449e+07 (3.629e+06) ↔	2.413e+07 (4.372e+06) ↔	2.326e+07 (4.176e+06) ↔	**2.539e+07 (3.806e+06)** ↔
Christofides_8_5_0.5	HV	**2.562e+07 (4.047e+06)**	2.413e+07 (3.730e+06) ↔	2.273e+07 (2.519e+06) ↓	2.207e+07 (1.042e+06) ↓	2.436e+07 (3.807e+06) ↔
Christofides_9_5_0.5	HV	**2.559e+07 (3.313e+06)**	2.469e+07 (4.538e+06) ↔	2.529e+07 (4.680e+06) ↔	2.395e+07 (3.015e+06) ↔	2.347e+07 (2.448e+06) ↓
Christofides_10_5_0.5	HV	**3.504e+07 (7.531e+06)**	3.121e+07 (2.391e+06) ↔	3.125e+07 (3.084e+06) ↔	3.096e+07 (4.724e+06) ↓	3.166e+07 (3.910e+06) ↓
Christofides_11_5_0.5	HV	5.700e+07 (6.772e+06)	**6.125e+07 (4.892e+06)** ↑	4.815e+07 (5.250e+06) ↓	5.388e+07 (6.921e+06) ↔	5.832e+07 (7.050e+06) ↔
Christofides_12_5_0.5	HV	7.009e+07 (9.121e+06)	6.875e+07 (9.181e+06) ↔	**7.079e+07 (8.949e+06)** ↔	6.845e+07 (8.676e+06) ↔	6.575e+07 (8.999e+06) ↔
+\≈\−			1 \11 \0	0 \9 \3	0 \10 \2	0 \9 \3
Christofides_1_5_0.5	R2	1.010e+03 (1.965e+02)	1.024e+03 (2.196e+02) ↔	9.345e+02 (1.728e+02) ↔	9.532e+02 (2.262e+02) ↔	**8.667e+02 (1.081e+02)** ↑
Christofides_2_5_0.5	R2	1.312e+03 (2.060e+02)	**1.279e+03 (1.710e+02)** ↔	1.331e+03 (2.111e+02) ↔	1.438e+03 (3.173e+02) ↔	1.412e+03 (2.888e+02) ↔
Christofides_3_5_0.5	R2	1.582e+03 (2.207e+02)	**1.573e+03 (1.935e+02)** ↔	1.641e+03 (2.376e+02) ↔	1.669e+03 (2.659e+02) ↔	1.882e+03 (2.915e+02) ↓
Christofides_4_5_0.5	R2	2.466e+03 (3.983e+02)	2.448e+03 (3.753e+02) ↔	2.317e+03 (4.012e+02) ↔	**2.264e+03 (4.152e+02)** ↔	2.516e+03 (3.484e+02) ↔
Christofides_5_5_0.5	R2	**3.051e+03 (5.059e+02)**	3.101e+03 (4.124e+02) ↔	3.115e+03 (3.413e+02) ↔	3.098e+03 (4.540e+02) ↔	3.071e+03 (4.587e+02) ↔
Christofides_6_5_0.5	R2	**9.559e+02 (1.791e+02)**	9.616e+02 (1.864e+02) ↔	1.040e+03 (2.278e+02) ↔	1.022e+03 (2.216e+02) ↔	1.009e+03 (2.114e+02) ↔
Christofides_7_5_0.5	R2	1.382e+03 (2.452e+02)	1.356e+03 (2.122e+02) ↔	1.418e+03 (2.727e+02) ↔	1.462e+03 (2.889e+02) ↔	**1.340e+03 (1.985e+02)** ↔
Christofides_8_5_0.5	R2	**1.833e+03 (2.115e+02)**	1.949e+03 (1.807e+02) ↔	2.001e+03 (1.376e+02) ↓	2.037e+03 (1.213e+02) ↓	1.916e+03 (2.317e+02) ↔
Christofides_9_5_0.5	R2	2.619e+03 (1.210e+02)	**2.604e+03 (2.437e+02)** ↔	2.614e+03 (2.204e+02) ↔	2.608e+03 (1.743e+02) ↔	2.638e+03 (1.243e+02) ↔
Christofides_10_5_0.5	R2	**3.100e+03 (2.919e+02)**	3.209e+03 (1.938e+02) ↔	3.262e+03 (1.463e+02) ↓	3.299e+03 (2.392e+02) ↓	3.223e+03 (1.735e+02) ↔
Christofides_11_5_0.5	R2	2.686e+03 (2.140e+02)	**2.431e+03 (2.327e+02)** ↑	2.945e+03 (1.767e+02) ↓	2.754e+03 (2.307e+02) ↔	2.537e+03 (2.456e+02) ↑
Christofides_12_5_0.5	R2	**1.853e+03 (2.801e+02)**	1.862e+03 (2.400e+02) ↔	1.854e+03 (2.104e+02) ↔	1.914e+03 (2.441e+02) ↔	1.931e+03 (2.534e+02) ↔
+\≈\−			1 \11 \0	0 \9 \3	0 \10 \2	2 \10 \1
Christofides_1_5_0.5	E_s	4.617e-02 (5.596e-02)	2.951e-02 (3.445e-02) ↔	**1.211e-01 (4.372e-01)** ↔	7.419e-02 (1.463e-01) ↔	6.231e-02 (1.307e-01) ↔
Christofides_2_5_0.5	E_s	6.185e-01 (2.366e+00)	4.130e-02 (5.600e-02) ↔	7.924e-02 (1.335e-01) ↔	4.966e-02 (8.783e-02) ↔	**3.951e-02 (5.137e-02)** ↔
Christofides_3_5_0.5	E_s	6.864e-02 (7.353e-02)	9.323e-02 (2.375e-01) ↔	8.895e-02 (1.957e-01) ↔	5.518e-02 (8.374e-02) ↔	**3.159e-02 (7.301e-02)** ↑
Christofides_4_5_0.5	E_s	4.229e-02 (6.308e-02)	3.557e-02 (3.972e-02) ↔	3.932e-02 (3.956e-02) ↔	**2.472e-02 (3.065e-02)** ↔	2.836e-02 (3.723e-02) ↔
Christofides_5_5_0.5	E_s	5.827e-02 (1.096e-01)	**1.716e-02 (2.862e-02)** ↔	5.503e-02 (9.383e-02) ↔	4.480e-02 (1.003e-01) ↔	2.888e-02 (4.217e-02) ↔
Christofides_6_5_0.5	E_s	2.883e-02 (4.177e-02)	9.429e-02 (2.154e-01) ↔	2.607e-02 (4.871e-02) ↔	**2.229e-02 (3.348e-02)** ↔	2.444e-02 (4.415e-02) ↔
Christofides_7_5_0.5	E_s	7.199e-02 (6.495e-02)	2.344e-01 (4.716e-01) ↔	1.545e-01 (2.283e-01) ↔	**1.375e-01 (1.928e-01)** ↔	1.678e-01 (3.251e-01) ↔
Christofides_8_5_0.5	E_s	**1.302e-01 (4.380e-01)**	2.496e-02 (3.561e-02) ↔	6.875e-01 (2.983e+00) ↔	5.343e-02 (1.347e-01) ↔	3.216e-02 (3.994e-02) ↔
Christofides_9_5_0.5	E_s	5.476e-02 (7.414e-02)	4.742e-02 (1.107e-01) ↔	7.901e-02 (2.451e-01) ↔	3.603e-02 (3.774e-02) ↔	**2.565e-02 (3.250e-02)** ↔
Christofides_10_5_0.5	E_s	9.204e-03 (1.130e-02)	8.037e-03 (9.730e-03) ↔	6.173e-02 (1.787e-01) ↔	**4.565e-01 (2.011e+00)** ↔	7.700e-03 (1.627e-02) ↔
Christofides_11_5_0.5	E_s	4.129e-02 (6.382e-02)	3.543e-02 (3.581e-02) ↔	**1.305e-02 (1.570e-02)** ↔	1.545e-02 (1.943e-02) ↔	4.835e-02 (4.801e-02) ↔
Christofides_12_5_0.5	E_s	7.055e-02 (1.221e-01)	5.093e-02 (1.097e-01) ↔	3.590e-02 (4.263e-02) ↔	**3.292e-02 (5.017e-02)** ↔	4.320e-02 (4.416e-02) ↔
+\≈\−			0 \12 \0	0 \12 \0	0 \12 \0	1 \11 \0

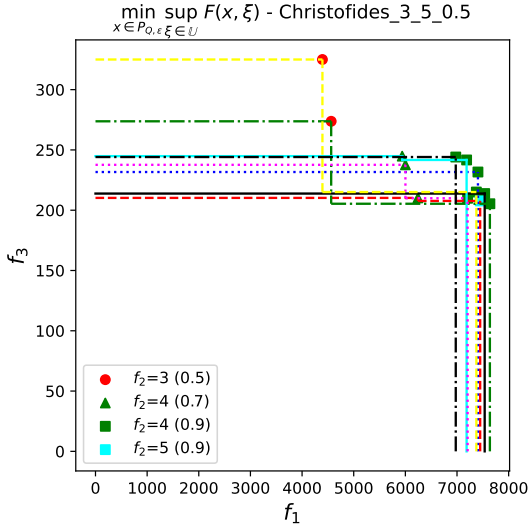
Tabla 4.9: Calidad de las soluciones ligeramente robustas para cada algoritmo basado en indicadores.



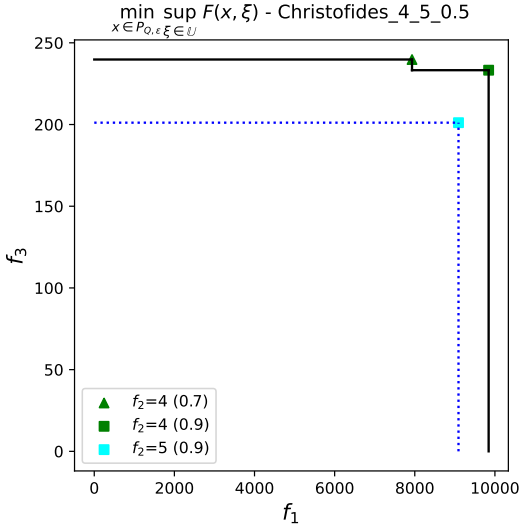
(a) Soluciones ligeramente robustas para Christofides_1_5_0.5.



(b) Soluciones ligeramente robustas para Christofides_2_5_0.5.

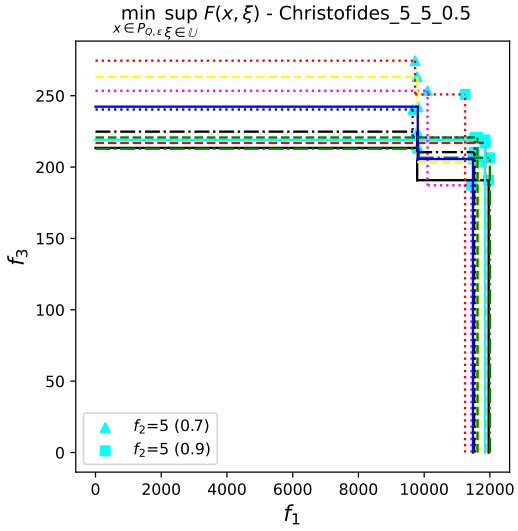


(c) Soluciones ligeramente robustas para Christofides_3_5_0.5.

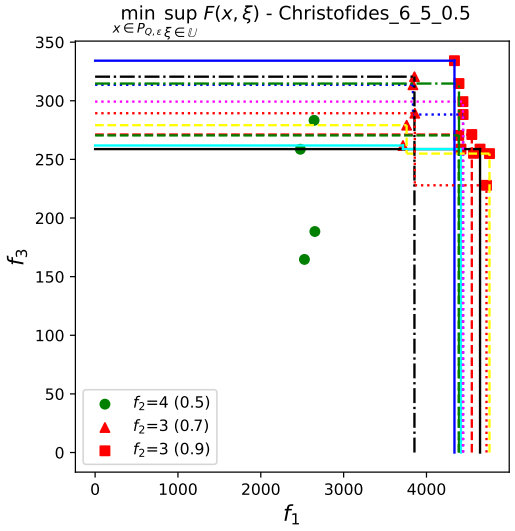


(d) Soluciones ligeramente robustas para Christofides_4_5_0.5.

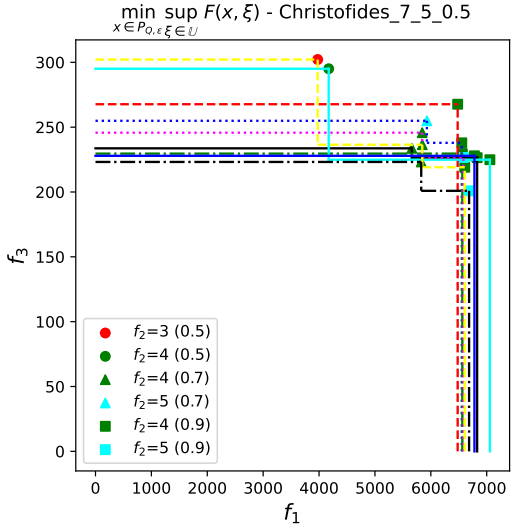
Figura 4.3: Soluciones ligeramente robustas para problemas 1-4.



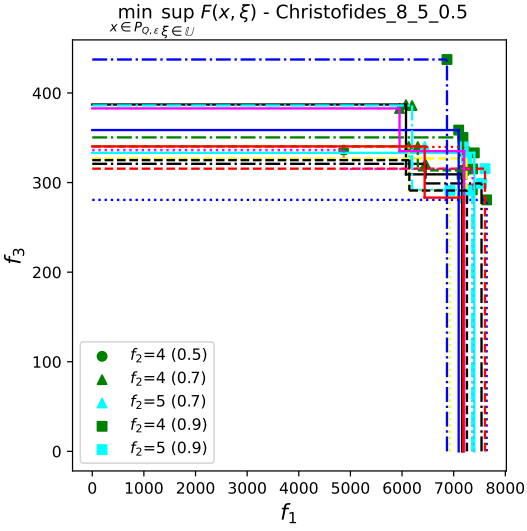
(a) Soluciones ligeramente robustas para Christofides_5_5_0.5.



(b) Soluciones ligeramente robustas para Christofides_6_5_0.5.

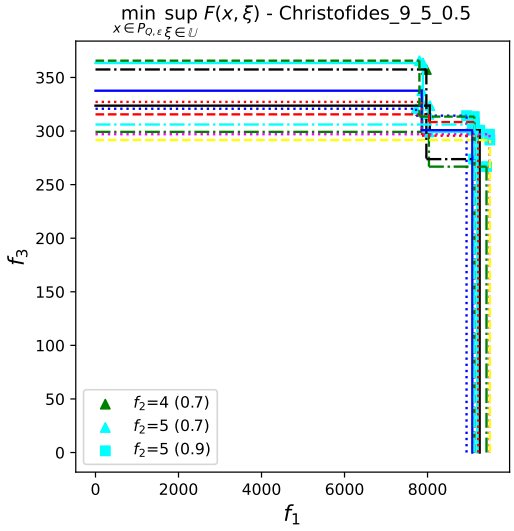


(c) Soluciones ligeramente robustas para Christofides_7_5_0.5.

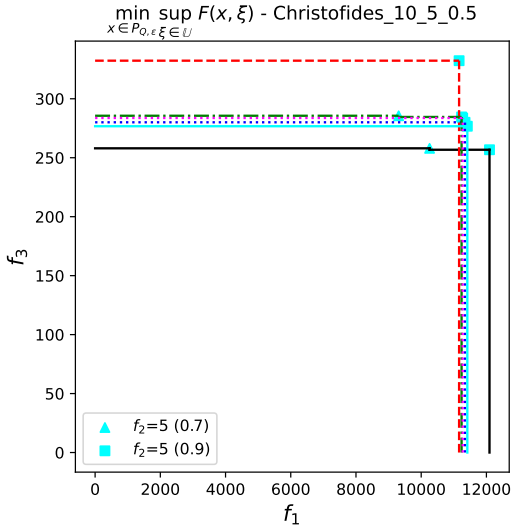


(d) Soluciones ligeramente robustas para Christofides_8_5_0.5.

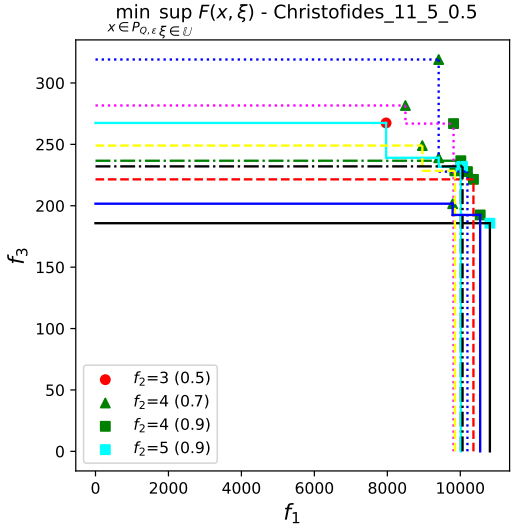
Figura 4.4: Soluciones ligeramente robustas para problemas 5-8.



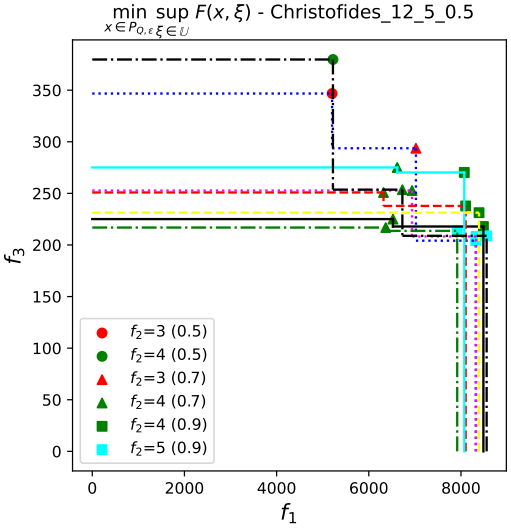
(a) Soluciones ligeramente robustas para Christofides_9_5_0.5.



(b) Soluciones ligeramente robustas para Christofides_10_5_0.5.



(c) Soluciones ligeramente robustas para Christofides_11_5_0.5.



(d) Soluciones ligeramente robustas para Christofides_12_5_0.5.

Figura 4.5: Soluciones ligeramente robustas para problemas 9-12.

4.5. Resumen

En este capítulo se presentaron los resultados de comparar la nueva propuesta cMIBACO contra los algoritmos basados en un solo indicador y suma ponderada para obtener soluciones aproximadas. También se analizó el impacto de los componentes híbridos y de búsqueda local en cMIBACO. Además de que se presentó la aplicación de cMIBACO para el MOGenConVRP bajo incertidumbre, obteniendo soluciones ligeramente robustas. En todos los experimentos se midió la calidad de soluciones con los indicadores de hipervolumen, R2 y Riesz s-energy. Se realizó análisis estadístico con la prueba de Wilcoxon.

En la comparación de los componentes de cMIBACO, se compararon cinco versiones, mostrando que la versión con cruza, mutación y búsqueda local tiene un mejor rango de acuerdo con el diagrama de diferencias críticas. Además de que esta variante supera en el 100 % de los problemas a las demás variantes de acuerdo con hipervolumen y $R2$.

La variante con cruza y mutación se mantiene como segundo lugar la mayor parte del tiempo en los indicadores de hipervolumen y $R2$. Por el contrario, la variante con cruza se mantiene en tercera posición frecuentemente. Y usar solo mutación no tiene un impacto significativo, esto se confirma en la optimización de parámetros, pues los mejores valores encontrados para la probabilidad de mutación son cercanos a cero para todos los algoritmos.

En los resultados de la comparación entre los algoritmos basados en un indicador y la suma ponderada contra cMIBACO para soluciones aproximadas, el diagrama de diferencias críticas muestra a cMIBACO con mejor rango y con diferencia significativa frente a los demás algoritmos. Mientras que IBACO$_{\epsilon+}$ es una buena alternativa debido a que es el segundo mejor algoritmo y con tiempos de ejecución más cortos.

De acuerdo con el indicador de hipervolumen, cuando cMIBACO no es mejor que otro algoritmo no se tiene diferencia estadística significativa en la mayoría de problemas. Además de que cMIBACO logra estar clasificado como primero en el 60 % de las instancias para este indicador y muy pocas veces en los demás lugares. Además de que es el mejor en los problemas con mayor cantidad de clientes.

En el indicador de $R2$, el cMIBACO se clasifica primero en el 30 % de las instancias y es seguido por el IBACO$_{ws}$, aunque cMIBACO se desempeña mejor en los problemas grandes, mientras que IBACO$_{ws}$ es mejor en problemas medianos. Tanto las versiones de IBACO$_{R2}$ e IBACO$_{HV}$ obtienen un desempeño bajo respecto a cMIBACO, donde el segundo pierde más veces con diferencia significativa.

Para la optimización robusta, se obtuvieron soluciones robustamente ligeras usando como escenario nominal a las soluciones aproximadas obtenidas por cada algoritmo en los experimentos anteriores. Primero se mostró la calidad de las soluciones del escenario nominal respecto a la calidad de las soluciones en los otros escenarios, resultando en que los otros escenarios resultan más difíciles de optimizar.

Se compararon las soluciones robustas obtenidas con cMIBACO frente a los algoritmos individuales y suma ponderada, resultando en que cMIBACO obtiene soluciones ligeramente robustas de mejor calidad y con diferencia significativa en bastantes instancias de MOGenConVRP. Respecto a los indicadores de hipervolumen y $R2$, en ambos indicadores el cMIBACO se clasificó la mayor parte del tiempo en primer lugar y en segundo el IBACO$_{\epsilon+}$.

Se realizó un análisis sobre la pérdida de calidad de las soluciones ligeramente robustas obtenidas con cMIBACO respecto a las soluciones aproximadas. De acuerdo con hipervolumen, la mayoria tiene un pérdida de calidad debajo del 40 %, esto debido al primer objetivo. Para el primer objetivo

la mayoría de los problemas tuvieron una pérdida alrededor del 40 %. En contraste con el segundo objetivo donnde el máximo fue de 17.29 % y para el tercer objetivo la mayoría de problemas tuvo alrededor del 20 %. Por lo que las soluciones obtenidas presentan un mayor deterioro en el primer objetivo, aunque tanto el segundo como el tercer objetivo son considerablemente más difíciles de resolver y éstos tuvieron un deterioro bajo.

Capítulo 5

Conclusiones y trabajo futuro

En esta tesis se presentó un nuevo algoritmo multi-objetivo de colonia de hormigas basada en múltiples indicadores que tiene como objetivo principal mejorar la búsqueda de soluciones con distintas colonias de hormigas. Donde cada colonia aproxima distintas regiones del frente de soluciones óptimas de acuerdo con su indicador de calidad para luego realizar un intercambio de soluciones que permitan mejorar la diversidad de soluciones. También se propuso un algoritmo que utiliza la suma ponderada de indicadores como función de aptitud en las hormigas, de este algoritmo se derivan los algoritmos basados en un solo indicador para los experimentos. Todos los algoritmos se probaron en el MOGenConVRP bajo incertidumbre donde la incertidumbre se encuentra en los clientes a visitar y se optimizan tres objetivos. Para este problema se compararon soluciones aproximadas y ligeramente robustas en todos los algoritmos. A continuación se detallan las conclusiones y futuro trabajo.

5.1. Conclusiones

La aportación principal de este proyecto es un nuevo algoritmo definido como cMIBACO, que se basa en la cooperación de múltiples indicadores de calidad para colonia de hormigas. Este algoritmo se utilizó para la optimización robusta para el MOGenConVRP bajo incertidumbre, encontrando soluciones aproximadas y ligeramente robustas. El diseño de cMIBACO se basa en los trabajos sobre EMOA's del estado del arte que tratan la cooperación de indicadores junto a los elementos básicos de la propuesta original de IBACO y ACO's híbridos. Dentro de cMIBACO se incorporan componentes del estado del arte como son los archivos externos, teniendo un archivo externo para soluciones aproximadas y otro archivo externo para soluciones robustas basadas en conjuntos. Además de que en este trabajo se propone usar la suma ponderada de indicadores como función de aptitud en IBACO$_{ws}$, de éste se deriva el algoritmo IBACO$_{R2}$ que también es una contribución al estado del arte.

Se analizó el efecto de los componentes en cMIBACO debido a que tiene componentes que no están presentes en el ACO tradicional, encontrando que usar operadores de cruza, mutación y búsqueda local logran tener un mejor desempeño que de forma individual y que no incorporarlos. Siendo esta variante la mejor de acuerdo con los tres indicadores de calidad y con diferencia significativa según la prueba estadística de Wilcoxon. En segundo lugar se encuentra usar la variante con solo cruza y mutación en cMIBACO, mientras que usar solo mutación tiene un bajo desempeño a

pesar de tener soluciones más diversas.

También se analizó el desempeño de cMIBACO contra solo usar un indicador o la suma ponderada en IBACO para soluciones aproximadas y ligeramente robustas. Para el conjunto de soluciones aproximadas, el cMIBACO es el de mejor desempeño de acuerdo con los indicadores de calidad y pruebas estadísticas. Este algoritmo se desempeña mejor para los problemas con mayor cantidad de clientes. Mientras que IBACO$_{\epsilon+}$ a pesar de ser el segundo mejor algoritmo, resulta ser una buena alternativa para problemas de tamaño medio y debido a sus cortos tiempos de ejecución. Por otro lado, el IBACO$_{ws}$ tiene bastante desventaja en el tiempo que tarda cualquiera de sus ejecuciones debido a la evaluación de todos sus indicadores.

Para la optimización robusta, se utilizó como escenario nominal a las soluciones aproximadas de los experimentos anteriores. La comparación se realizó de la misma forma que en soluciones aproximadas, teniendo a cMIBACO dominando en los indicadores de hipervolumen y R2 en varios problemas con diferencia significativa respecto a los demás algoritmos. Se analizó la pérdida de calidad de las soluciones ligeramente robustas obtenidas con cMIBACO respecto a sus soluciones aproximadas. Donde el primer objetivo es el más afectado con un porcentaje de pérdida entre 30 % y 40 % en la mayoría de los problemas. Sin embargo, los otros dos objetivos tuvieron un mejor desempeño, donde el segundo objetivo tuvo una pérdida máxima de 17.29 % y el tercer objetivo una pérdida máxima de 26.52 %. Estos resultados muestran que cMIBACO puede obtener soluciones ligeramente robustas con una pérdida aceptable en los tres objetivos para MOGenConVRP bajo incertidumbre.

5.2. Trabajo futuro

Como parte del trabajo futuro, el cMIBACO aún tiene elementos por mejorar, como es la forma en que se seleccionan las soluciones externas para cada indicador. Por lo que puede ser conveniente implementar un mecanismo que a partir de la diversidad que aporta cada indicador, determine el tamaño de soluciones externas para cada indicador. También sería interesante usar otros indicadores dentro de cMIBACO que mejoren las debilidades que presentan los indicadores actuales, así como usarlos en la suma ponderada. Otra idea es hacer que los distintos IBACO$_I$ compitan como ya lo proponen en algunos EMOA's del estado del arte, creando poblaciones de cada indicador bajo ciertos criterios. En cuanto a la aplicación de MOGenConVRP, se tienen matrices de feromona por cada día y horario a planear, la desventaja de esto es que con instancias grandes se puede tardar la actualización de feromonas. Por lo que sería interesante manejar una sola matriz de feromona que pueda construir las rutas de todos los días en el horizonte de planeación.

De acuerdo con el diseño de cMIBACO, este puede encontrar soluciones aproximadas, sin embargo, una de las carencias de este conjunto es la cantidad de soluciones Pareto débiles que acepta. Ante esto, sería interesante cambiar el archivo externo de $P_{Q,\epsilon}$ por un archivo externo de soluciones cercanas al óptimo $N_{Q,\epsilon}$. Otra aplicación interesante de cMIBACO está en tratar otros tipos de robustez en problemas de optimización.

Apéndice A

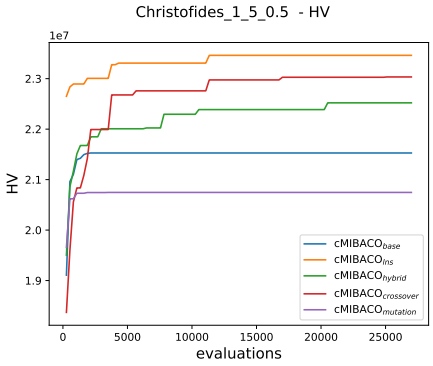
Resultados numéricos

En este apéndice se presentan los resultados numéricos de los experimentos realizados en el Capítulo 4. Que son las comparaciones de los componentes en cMIBACO, la comparación de soluciones aproximadas y la comparación de soluciones ligeramente robustas. Para cada experimento se muestra:

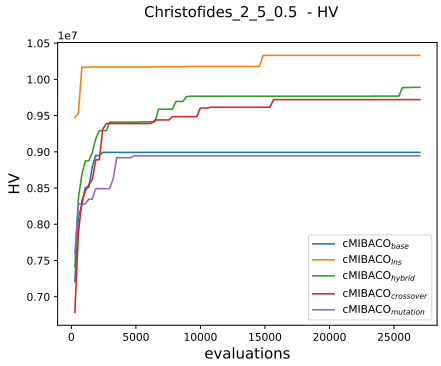
- Gráficos de hipervolumen del archivo externo cada 270 evaluaciones de la mediana entre las ejecuciones de cada algoritmo excepto para la optimización robusta.
- Soluciones en espacio de los objetivos obtenidas de la mediana entre las ejecuciones de cada algoritmo de acuerdo con el indicador de hipervolumen.
- Gráficos de caja del hipervolumen de los archivos externos al final de las 20 ejecuciones para soluciones aproximadas y las soluciones ligeramente robustas obtenidas.

Comenzando con los experimentos sobre los componentes de cMIBACO, en la Figura A.1 se presenta el hipervolumen del archivo externo durante evaluaciones de la mediana entre las ejecuciones para cada variante de cMIBACO. En la Figura A.2 se presentan los gráficos de caja del hipervolumen del archivo externo final de todas las ejecuciones para las variantes de cMIBACO. En la Figura A.3 se presentan los frentes de soluciones aproximadas obtenidas para la mediana de las ejecuciones de las variantes de cMIBACO.

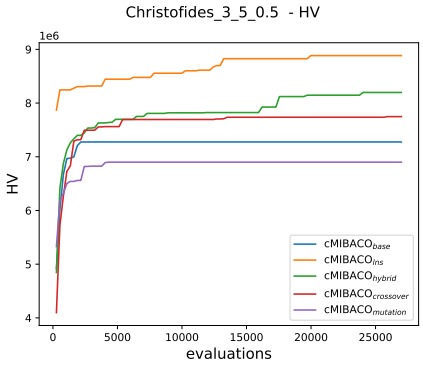
Sobre la comparación de indicadores para soluciones aproximadas, en la Figura A.4 se muestran el hipervolumen del archivo externo durante evaluaciones de la mediana entre las ejecuciones de cada algoritmo. En la Figura A.5 se presentan los gráficos de caja del hipervolumen del archivo externo final de todas las ejecuciones de los algoritmos basados en indicadores. En la Figura A.6 se presentan los frentes de soluciones aproximadas obtenidas para la mediana de las ejecuciones de los algoritmos basados en indicadores. Finalmente, en la Figura A.7 se presentan los gráficos de caja de las soluciones ligeramente robustas obtenidas de todas las ejecuciones con todos los algoritmos.



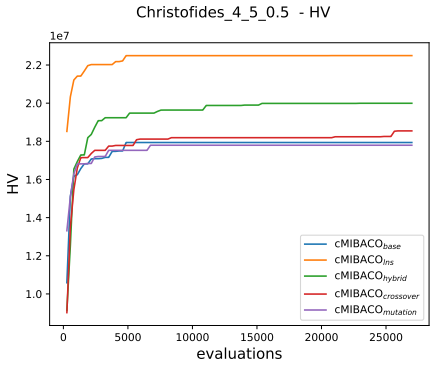
(a) Mediana de hipervolumen para Christofides_1_5_0.5.



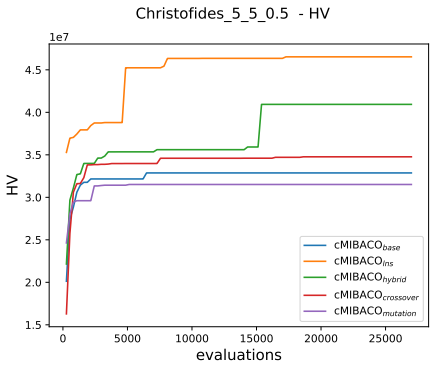
(b) Mediana de hipervolumen para Christofides_2_5_0.5.



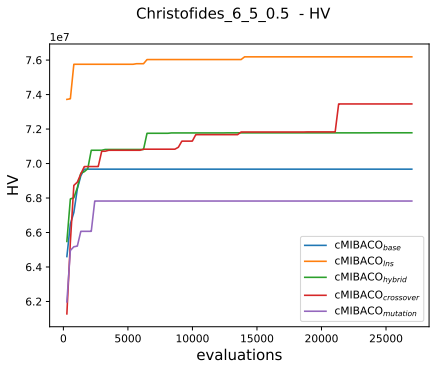
(c) Mediana de hipervolumen para Christofides_3_5_0.5.



(d) Mediana de hipervolumen para Christofides_4_5_0.5.

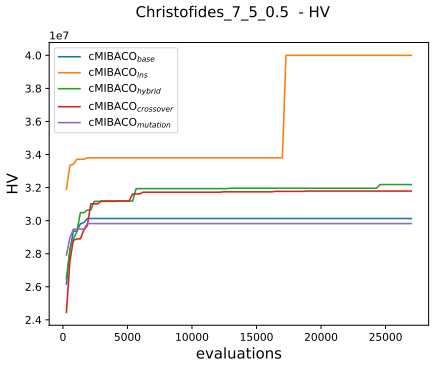


(e) Mediana de hipervolumen para Christofides_5_5_0.5.

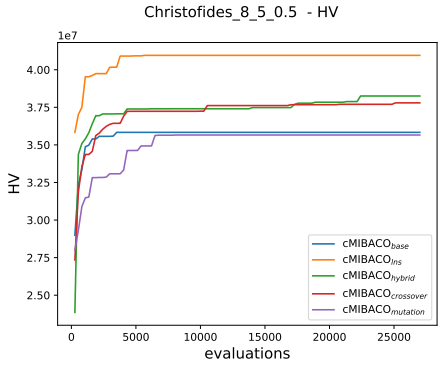


(f) Mediana de hipervolumen para Christofides_6_5_0.5.

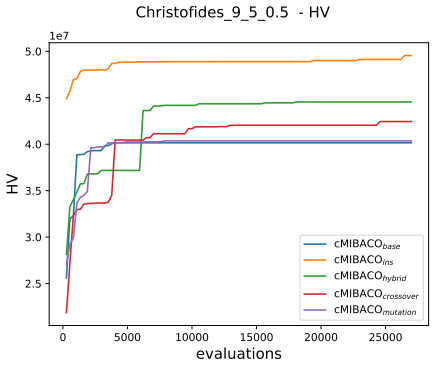
Figura A.1: Comparación de la mediana de hipervolumen entre variantes de cMIBACO para soluciones aproximadas.



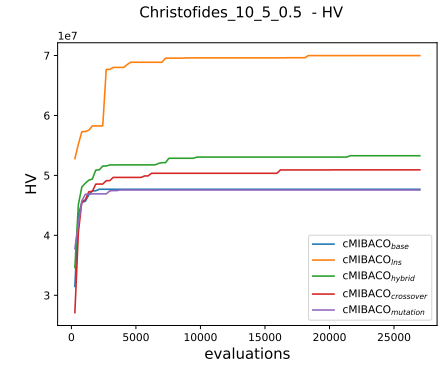
(g) Mediana de hipervolumen para Christofides_7_5_0.5.



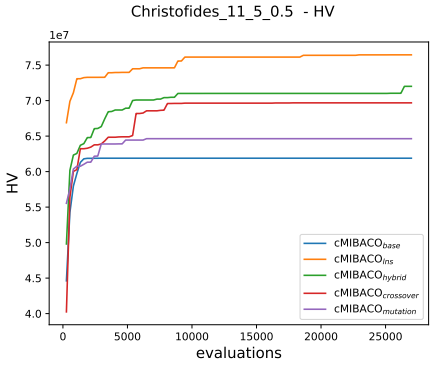
(h) Mediana de hipervolumen para Christofides_8_5_0.5.



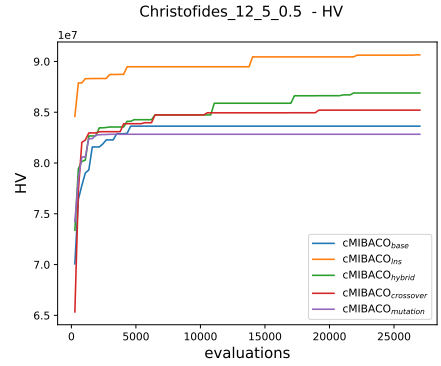
(i) Mediana de hipervolumen para Christofides_9_5_0.5.



(j) Mediana de hipervolumen para Christofides_10_5_0.5.

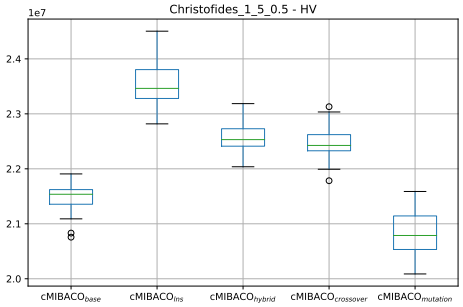


(k) Mediana de hipervolumen para Christofides_11_5_0.5.

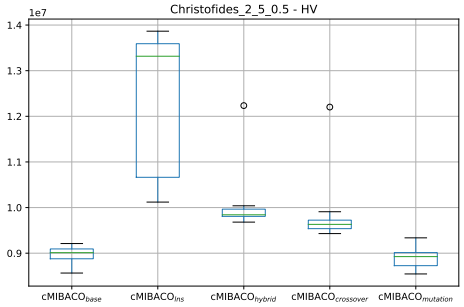


(l) Mediana de hipervolumen para Christofides_12_5_0.5.

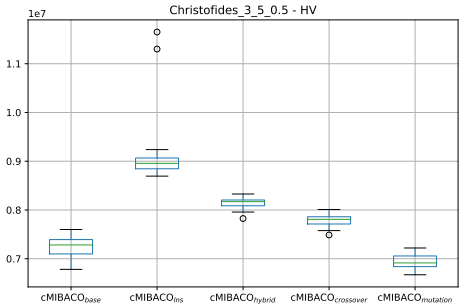
Figura A.1: Comparación de la mediana de hipervolumen entre variantes de cMIBACO para soluciones aproximadas.



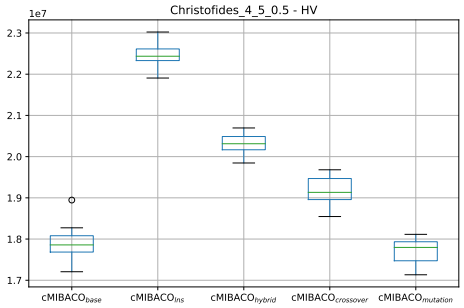
(a) Distribución del hipervolumen de los datos para Christofides_1_5_0.5.



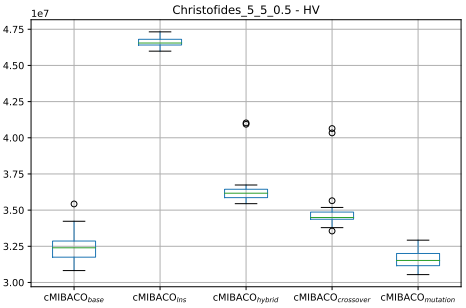
(b) Distribución del hipervolumen de los datos para Christofides_2_5_0.5.



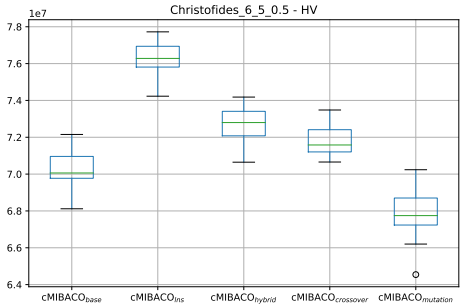
(c) Distribución del hipervolumen de los datos para Christofides_3_5_0.5.



(d) Distribución del hipervolumen de los datos para Christofides_4_5_0.5.

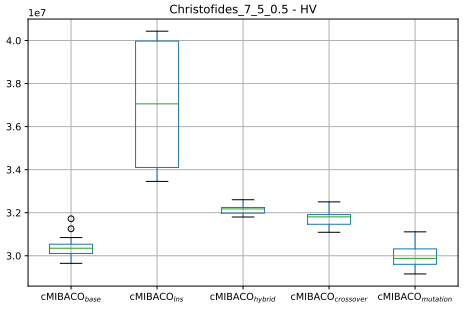


(e) Distribución del hipervolumen de los datos para Christofides_5_5_0.5.

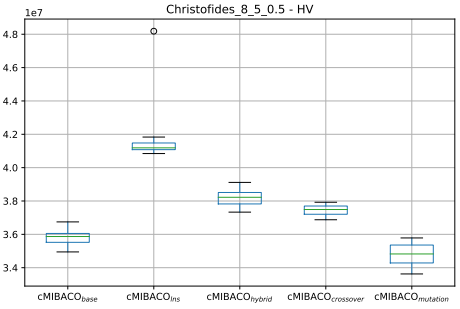


(f) Distribución del hipervolumen de los datos para Christofides_6_5_0.5.

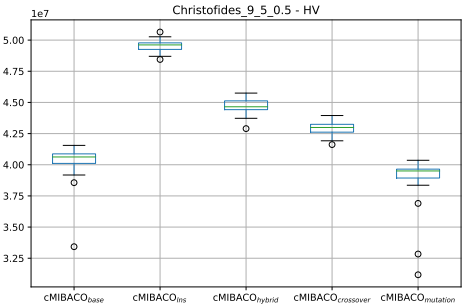
Figura A.2: Distribución de los datos de hipervolumen de las variantes de cMIBACO para soluciones aproximadas.



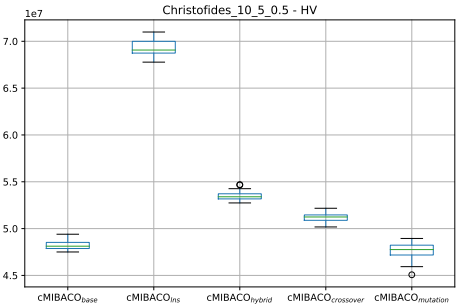
(g) Distribución del hipervolumen de los datos para Christofides_7_5_0.5.



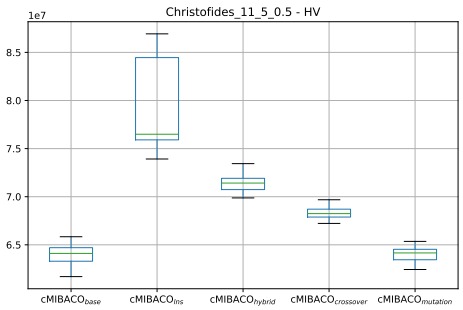
(h) Distribución del hipervolumen de los datos para Christofides_8_5_0.5.



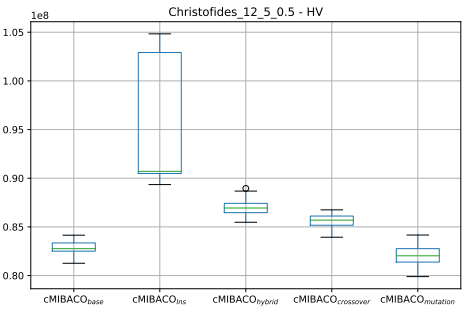
(i) Distribución del hipervolumen de los datos para Christofides_9_5_0.5.



(j) Distribución del hipervolumen de los datos para Christofides_10_5_0.5.

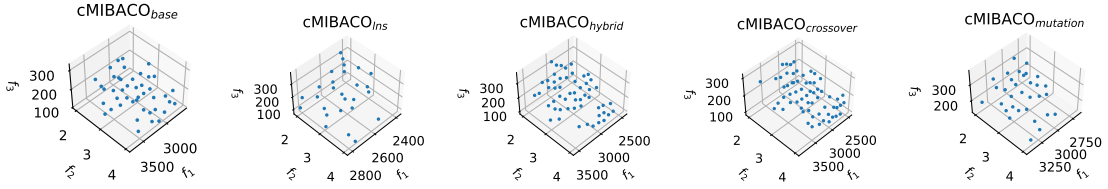


(k) Distribución del hipervolumen de los datos para Christofides_11_5_0.5.

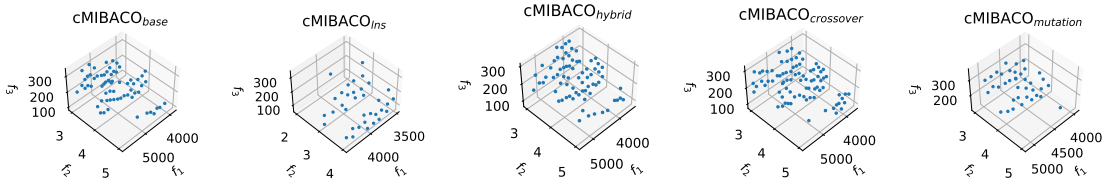


(l) Distribución del hipervolumen de los datos para Christofides_12_5_0.5.

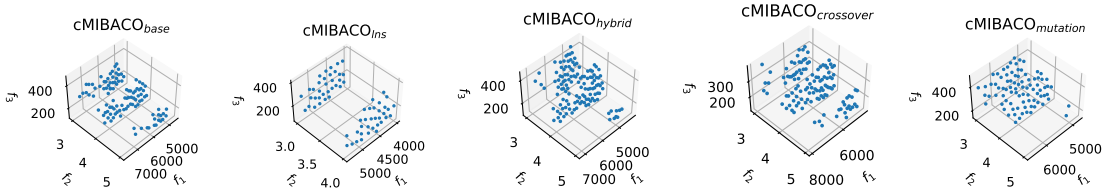
Figura A.2: Distribución de los datos de hipervolumen de las variantes de cMIBACO para soluciones aproximadas.



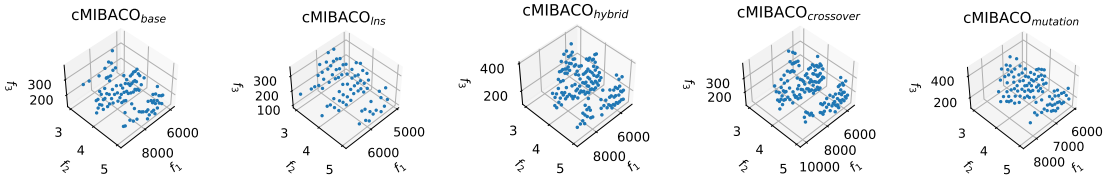
(a) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_1_5_0.5.



(b) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_2_5_0.5.

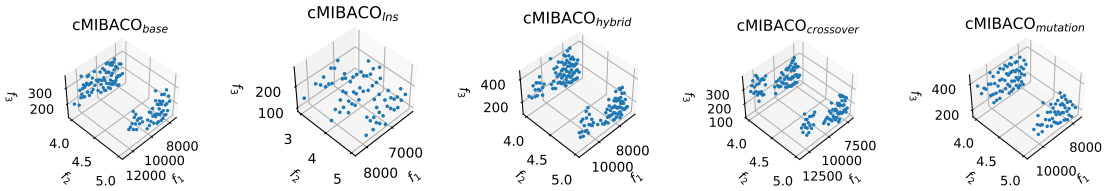


(c) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_3_5_0.5.

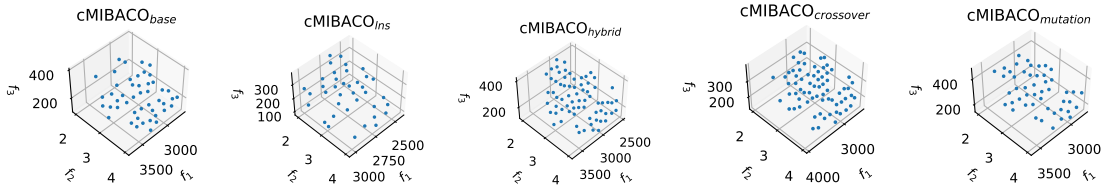


(d) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_4_5_0.5.

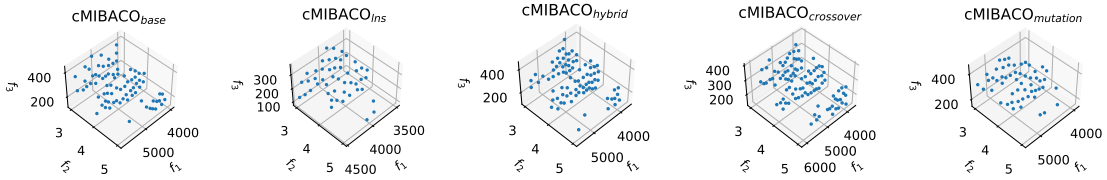
Figura A.3: Frentes obtenidos con cada versión de cMIBACO.



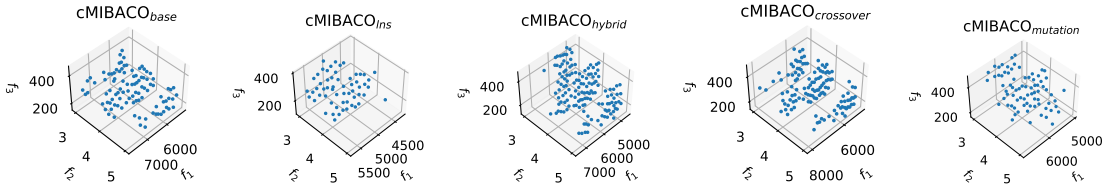
(e) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_5_5_0.5.



(f) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_6_5_0.5.

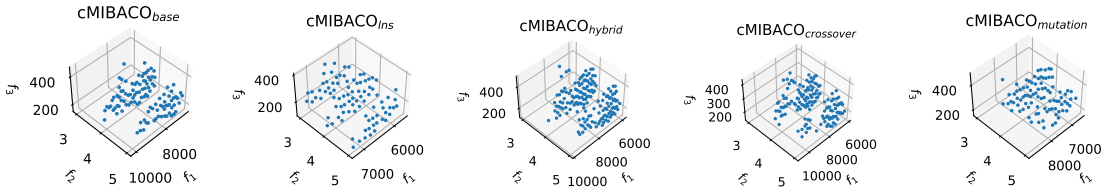


(g) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_7_5_0.5.

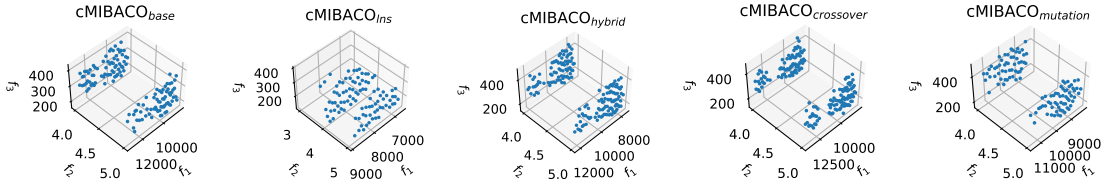


(h) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_8_5_0.5.

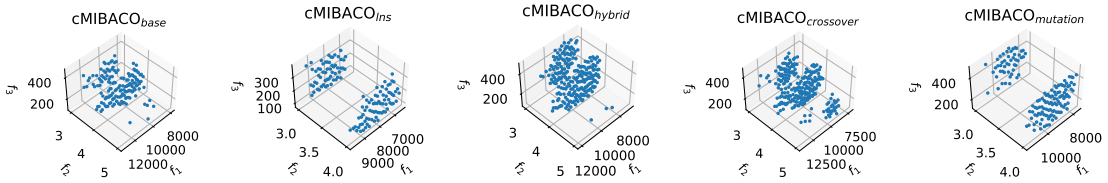
Figura A.3: Frentes obtenidos con cada versión de cMIBACO.



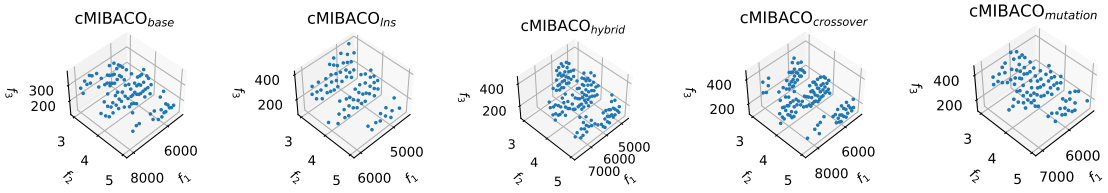
(i) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_9_5_0.5.



(j) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_10_5_0.5.

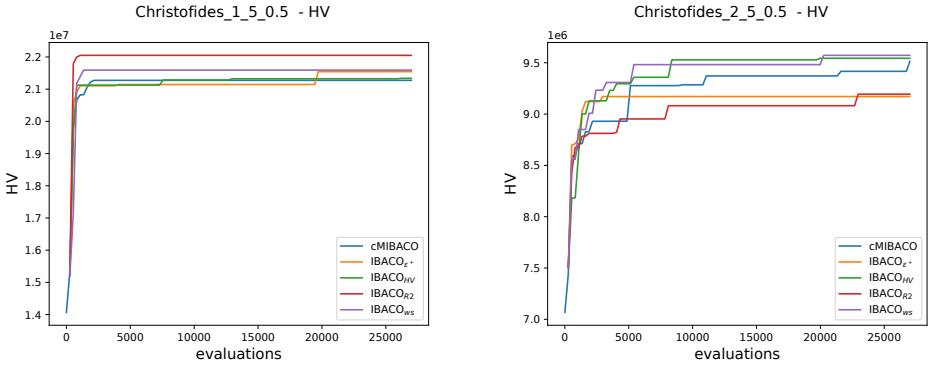


(k) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_11_5_0.5.



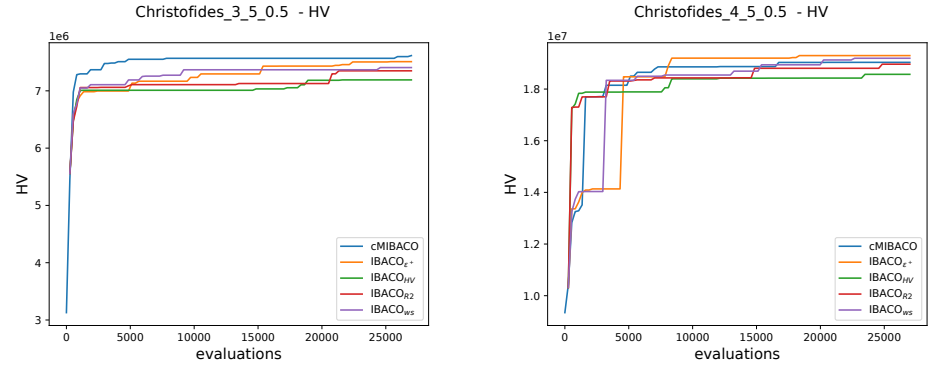
(l) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada versión de cMIBACO para Christofides_12_5_0.5.

Figura A.3: Frentes obtenidos con cada versión de cMIBACO.



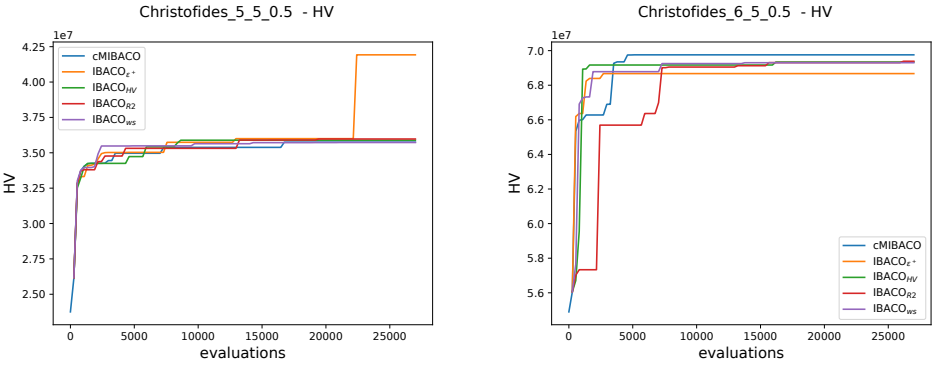
(a) Mediana de hipervolumen para Christofides_1_5_0.5.

(b) Mediana de hipervolumen para Christofides_2_5_0.5.



(c) Mediana de hipervolumen para Christofides_3_5_0.5.

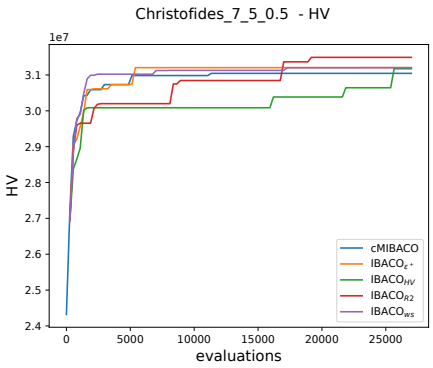
(d) Mediana de hipervolumen para Christofides_4_5_0.5.



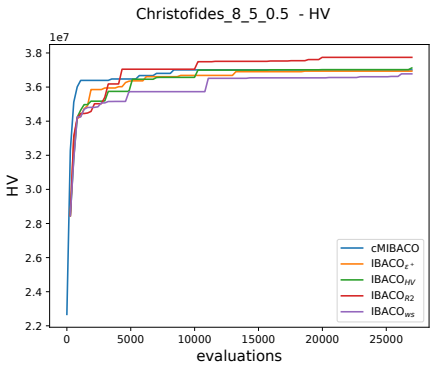
(e) Mediana de hipervolumen para Christofides_5_5_0.5.

(f) Mediana de hipervolumen para Christofides_6_5_0.5.

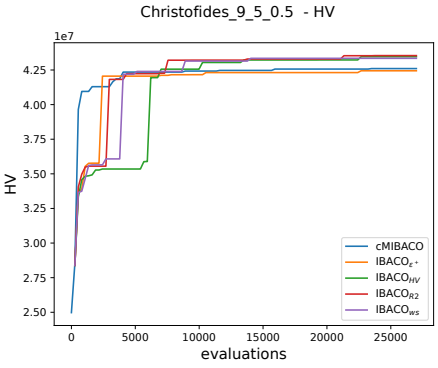
Figura A.4: Comparación de la mediana de hipervolumen entre algoritmos para soluciones aproximadas.



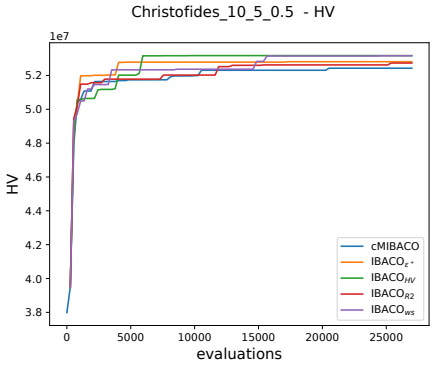
(g) Mediana de hipervolumen para Christofides_7_5_0.5.



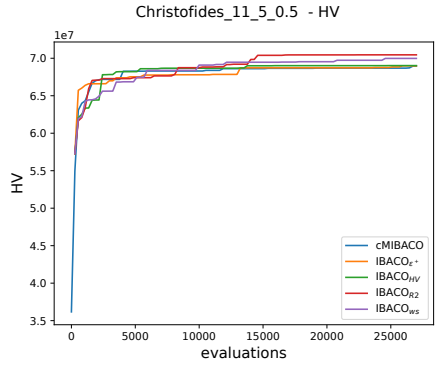
(h) Mediana de hipervolumen para Christofides_8_5_0.5.



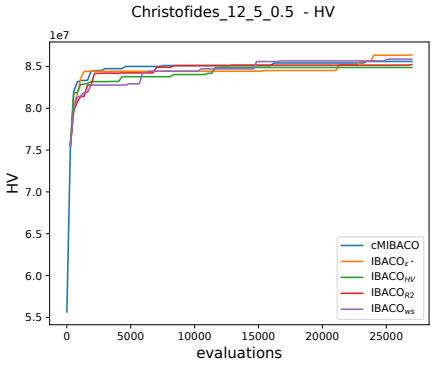
(i) Mediana de hipervolumen para Christofides_9_5_0.5.



(j) Mediana de hipervolumen para Christofides_10_5_0.5.

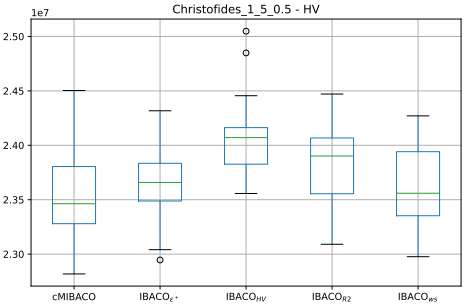


(k) Mediana de hipervolumen para Christofides_11_5_0.5.

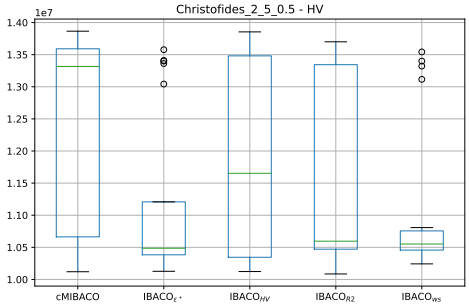


(l) Mediana de hipervolumen para Christofides_12_5_0.5.

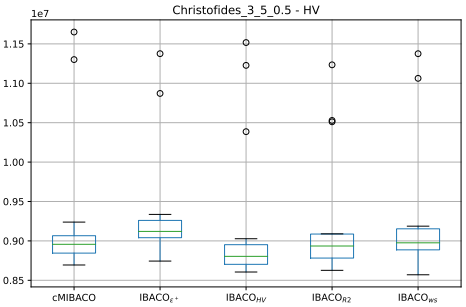
Figura A.4: Comparación de la mediana de hipervolumen entre algoritmos para soluciones aproximadas.



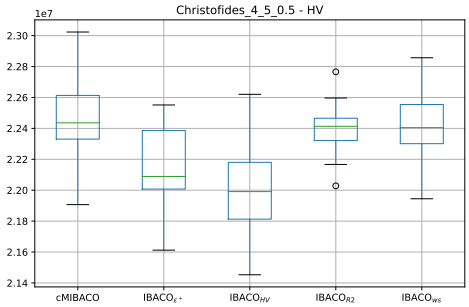
(a) Distribución del hipervolumen de los datos para Christofides_1_5_0.5.



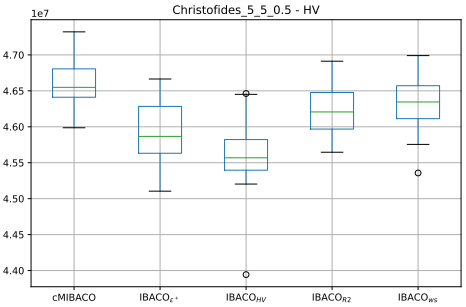
(b) Distribución del hipervolumen de los datos para Christofides_2_5_0.5.



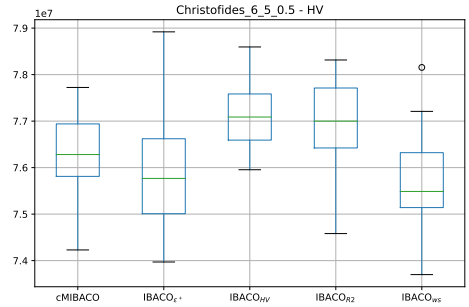
(c) Distribución del hipervolumen de los datos para Christofides_3_5_0.5.



(d) Distribución del hipervolumen de los datos para Christofides_4_5_0.5.

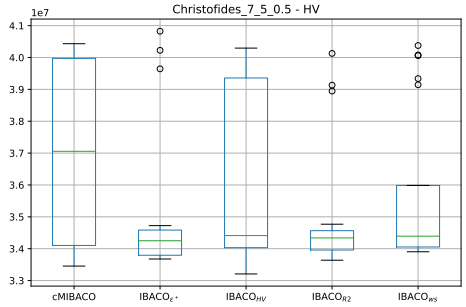


(e) Distribución del hipervolumen de los datos para Christofides_5_5_0.5.

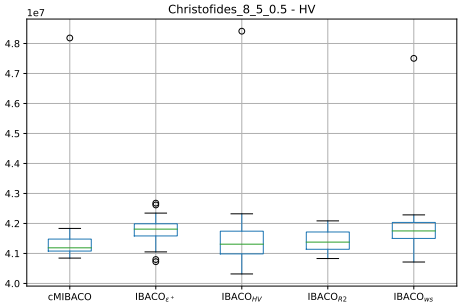


(f) Distribución del hipervolumen de los datos para Christofides_6_5_0.5.

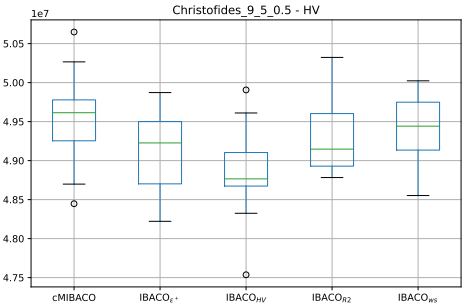
Figura A.5: Distribución de los datos de hipervolumen para soluciones aproximadas.



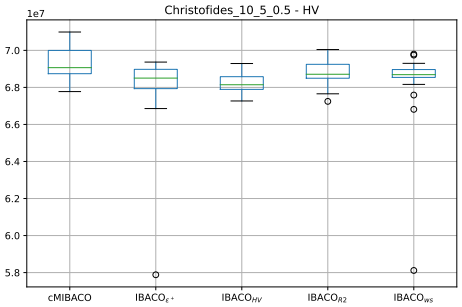
(g) Distribución del hipervolumen de los datos para Christofides_7_5_0.5.



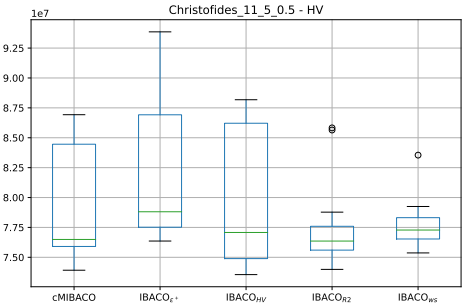
(h) Distribución del hipervolumen de los datos para Christofides_8_5_0.5.



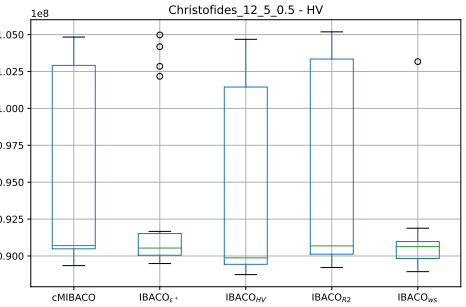
(i) Distribución del hipervolumen de los datos para Christofides_9_5_0.5.



(j) Distribución del hipervolumen de los datos para Christofides_10_5_0.5.

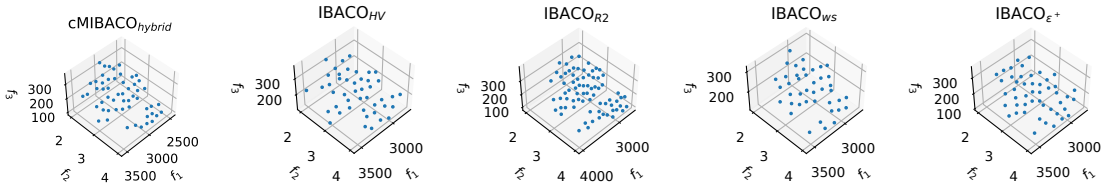


(k) Distribución del hipervolumen de los datos para Christofides_11_5_0.5.

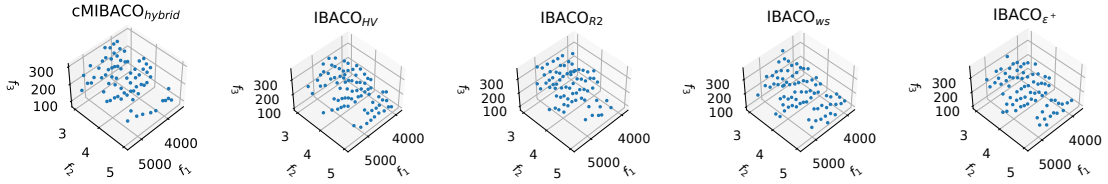


(l) Distribución del hipervolumen de los datos para Christofides_12_5_0.5.

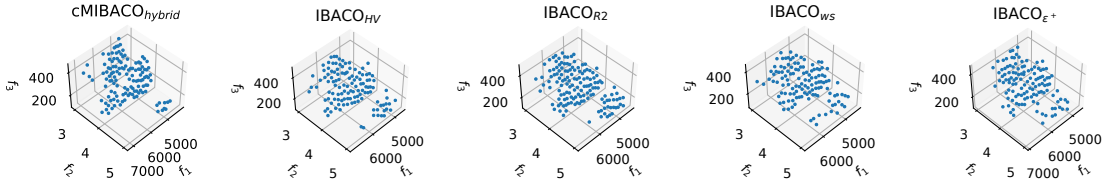
Figura A.5: Distribución de los datos de hipervolumen para soluciones aproximadas.



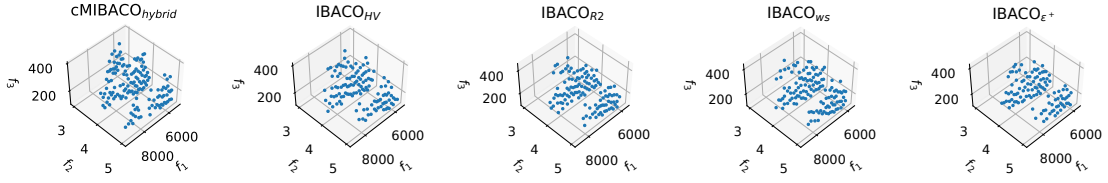
(a) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_1_5_0.5.



(b) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_2_5_0.5.

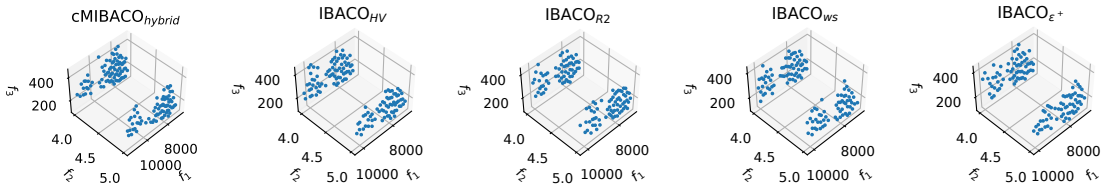


(c) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_3_5_0.5.

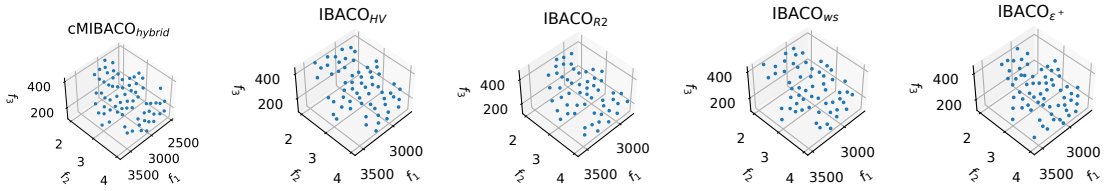


(d) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_4_5_0.5.

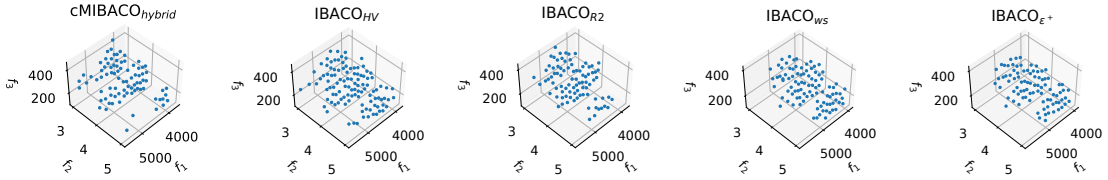
Figura A.6: Frentes de soluciones aproximadas obtenidas con cada algoritmo.



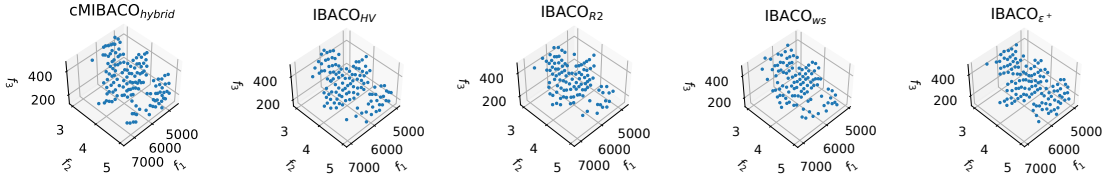
(e) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_5_5_0.5.



(f) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_6_5_0.5.

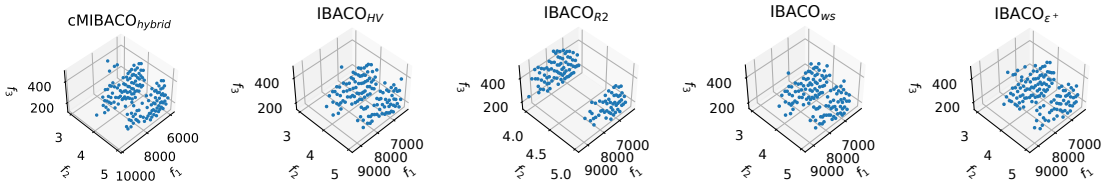


(g) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_7_5_0.5.

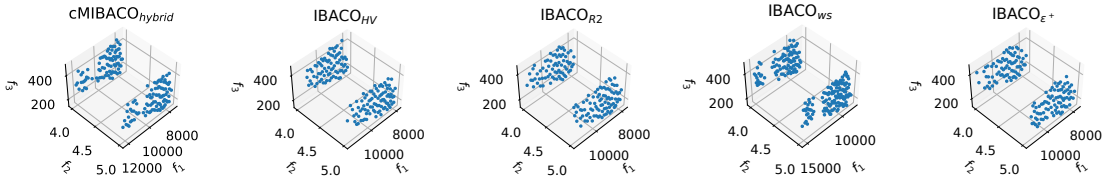


(h) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_8_5_0.5.

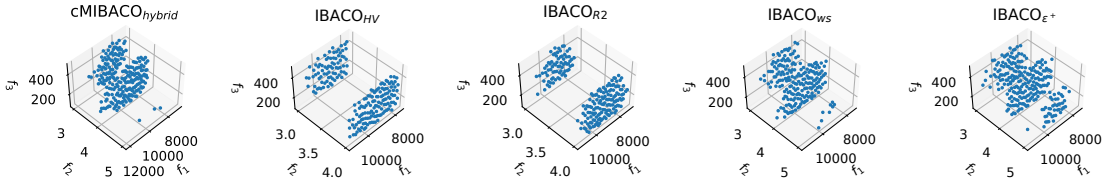
Figura A.6: Frentes de soluciones aproximadas obtenidas con cada algoritmo.



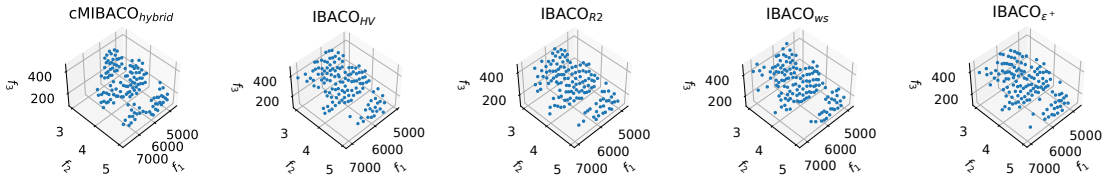
(i) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_9_5_0.5.



(j) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_10_5_0.5.

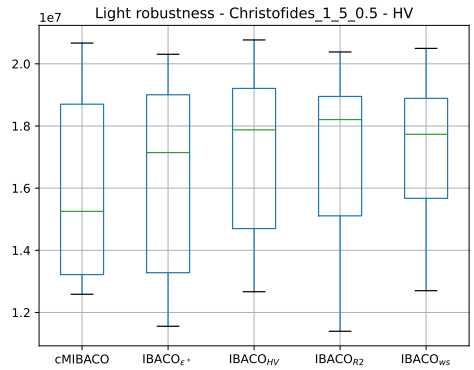


(k) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_11_5_0.5.

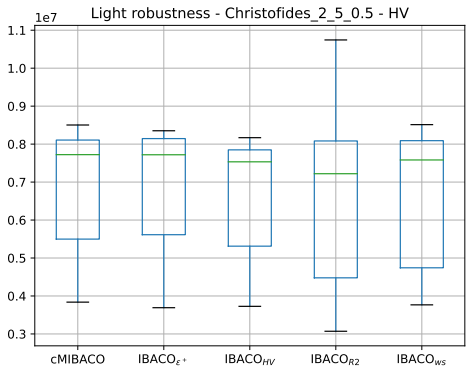


(l) Soluciones en espacio de los objetivos para la mediana de hipervolumen de cada algoritmo para Christofides_12_5_0.5.

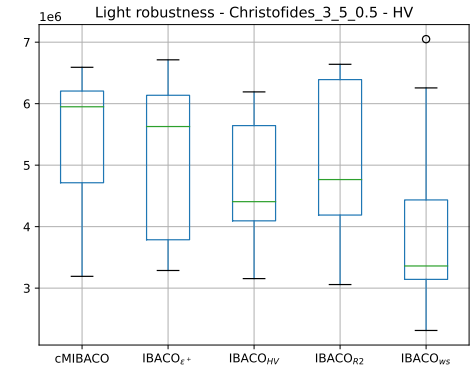
Figura A.6: Frentes de soluciones aproximadas obtenidas con cada algoritmo.



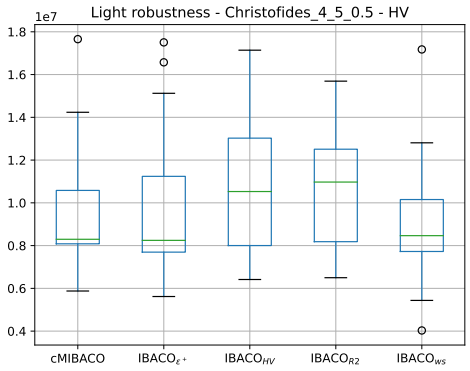
(a) Distribución del hipervolumen de los datos para Christofides_1_5_0.5.



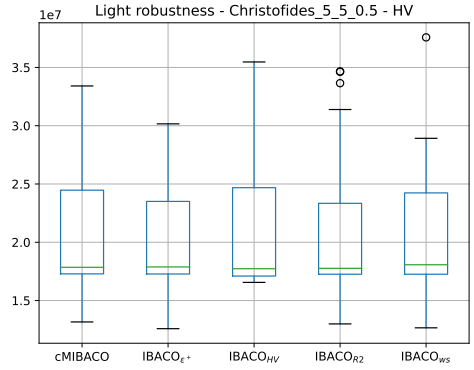
(b) Distribución del hipervolumen de los datos para Christofides_2_5_0.5.



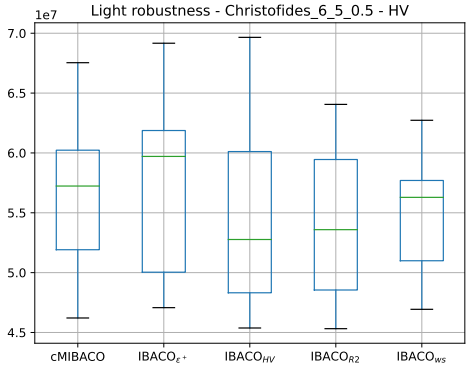
(c) Distribución del hipervolumen de los datos para Christofides_3_5_0.5.



(d) Distribución del hipervolumen de los datos para Christofides_4_5_0.5.

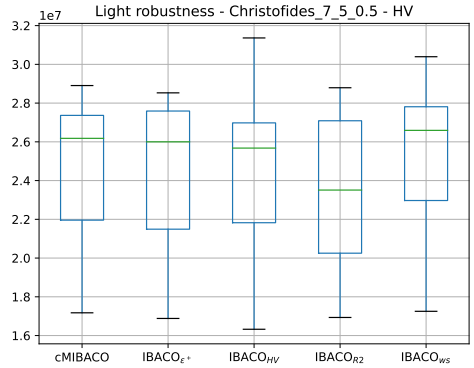


(e) Distribución del hipervolumen de los datos para Christofides_5_5_0.5.

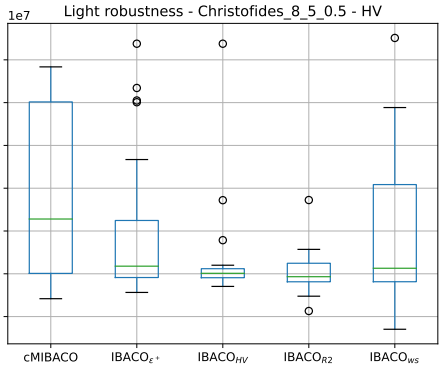


(f) Distribución del hipervolumen de los datos para Christofides_6_5_0.5.

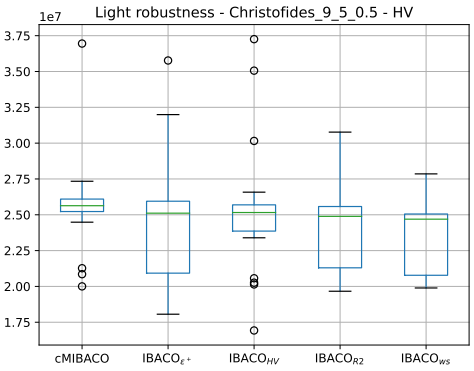
Figura A.7: Distribución de los datos de hipervolumen para soluciones ligeramente robustas.



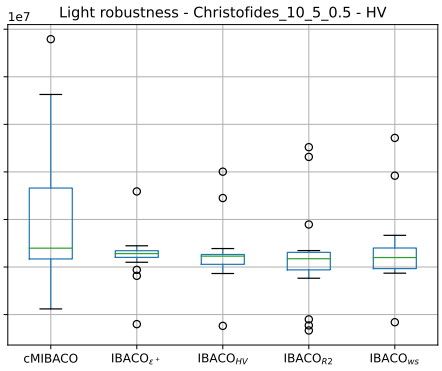
(g) Distribución del hipervolumen de los datos para Christofides_7_5_0.5.



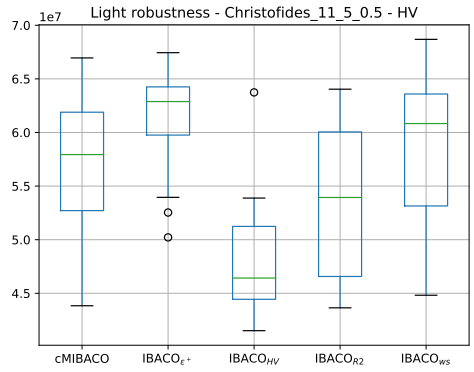
(h) Distribución del hipervolumen de los datos para Christofides_8_5_0.5.



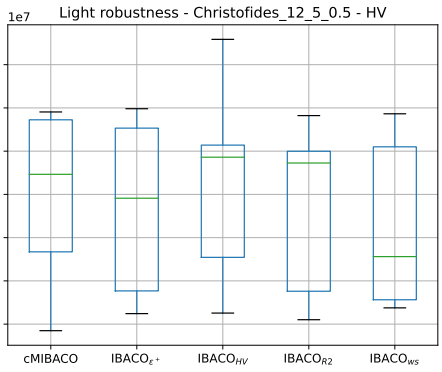
(i) Distribución del hipervolumen de los datos para Christofides_9_5_0.5.



(j) Distribución del hipervolumen de los datos para Christofides_10_5_0.5.



(k) Distribución del hipervolumen de los datos para Christofides_11_5_0.5.



(l) Distribución del hipervolumen de los datos para Christofides_12_5_0.5.

Figura A.7: Distribución de los datos de hipervolumen para soluciones ligeramente robustas.

Apéndice B

Comportamiento de las hormigas

El comportamiento de las hormigas depende principalmente de los valores que los parámetros tengan en la ejecución. Tales parámetros como las prioridades de los objetivos, taza de evaporación de feromonas y su valor inicial afectan considerablemente la decisión de las hormigas al momento de revisar los clientes. En este apéndice de presenta un ejemplo detallado de la construcción paso a paso de una solución con dos días de planeación en horarios AM y PM. Inicializando la matriz de feromonas de cada día y horario en un valor mínimo de 1×10^{-2}, taza de evaporación de 0.4, importancia de feromona $\alpha = 0.6$ y para los objetivos $\beta =, \gamma = 0.5, \delta = 0.6$. Con una probabilidad de explotación de 0.1, probabilidaes de cruza y mutación de 0.8 y 0.3 respectivamente. En la Tabla B.1 se muestra la información de los clientes usados en la instancia de ejemplo, teniendo 18 clientes más el depósito y vehículos de capacidad 7 con límite de tiempo $T = 100$.

Id	**Coordenadas** (x, y)	**Demanda** $[d_1, d_2]$	**Tiempos de servicio** $[s_1, s_2]$
1	(-3.180, -3.781)	[3, 3]	[1,1]
2	(0.864, 3.187)	[2, 1]	[0,1]
3	(-2.960, 2.820)	[1, 2]	[1,0]
4	(-0.866, -2.470)	[1, 3]	[1,1]
5	(2.045, 3.571)	[3, 3]	[1,1]
6	(-4.371, -2.022)	[2, 1]	[0,1]
7	(2.040, 2.697)	[2, 3]	[0,1]
8	(-1.409, -4.733)	[3, 3]	[1,1]
9	(6.236, 1.998)	[2, 3]	[0,1]
10	(3.755, 4.412)	[3, 2]	[1,0]
11	(-2.755, -4.412)	[1, 1]	[1,0]
12	(5.325, 3.212)	[2, 2]	[1,0]
13	(4.755, -1.412)	[3, 3]	[1,0]
14	(-2.755, 5.412)	[1, 2]	[1,0]
15	(6.755, 3.412)	[3, 1]	[1,0]
16	(5.725, 2.432)	[2, 1]	[1,0]
17	(5.315, 0.412)	[2, 1]	[1,0]
18	(4.375, 0.432)	[1, 1]	[1,0]

Tabla B.1: Clientes en la instancia de ejemplo de dos días de planeación con demandas d_1, d_2 para cada día y sus respectivos tiempos de servicio s_1, s_2.

Para el ejemplo mostramos las probabilidades y la información heurística de cada cliente mientras se van agregando a la ruta actual. Además de las rutas actuales y capacidad del vehículo.

Para el primer día en horario AM:
Cliente actual: 0:

- Para ir al cliente 8 desde 0: $\tau_{0,8} = 0.010,\ \eta_{0,8} = 0.202,\ \psi_{0,8} = 1.000,\ \phi_{0,8} = 1.000 \Rightarrow\ p_{0,8} = 0.107$
- Para ir al cliente 10 desde 0: $\tau_{0,10} = 0.010,\ \eta_{0,10} = 0.173,\ \psi_{0,10} = 1.000,\ \phi_{0,10} = 1.000 \Rightarrow\ p_{0,10} = 0.097$
- Para ir al cliente 12 desde 0: $\tau_{0,12} = 0.010,\ \eta_{0,12} = 0.161,\ \psi_{0,12} = 1.000,\ \phi_{0,12} = 1.000 \Rightarrow\ p_{0,12} = 0.093$
- Para ir al cliente 2 desde 0: $\tau_{0,2} = 0.010,\ \eta_{0,2} = 0.303,\ \psi_{0,2} = 1.000,\ \phi_{0,2} = 1.000 \Rightarrow\ p_{0,2} = 0.136$
- Para ir al cliente 16 desde 0: $\tau_{0,16} = 0.010,\ \eta_{0,16} = 0.161,\ \psi_{0,16} = 1.000,\ \phi_{0,16} = 1.000 \Rightarrow\ p_{0,16} = 0.093$
- Para ir al cliente 14 desde 0: $\tau_{0,14} = 0.010,\ \eta_{0,14} = 0.165,\ \psi_{0,14} = 1.000,\ \phi_{0,14} = 1.000 \Rightarrow\ p_{0,14} = 0.094$
- Para ir al cliente 4 desde 0: $\tau_{0,4} = 0.010,\ \eta_{0,4} = 0.382,\ \psi_{0,4} = 1.000,\ \phi_{0,4} = 1.000 \Rightarrow\ p_{0,4} = 0.156$
- Para ir al cliente 18 desde 0: $\tau_{0,18} = 0.010,\ \eta_{0,18} = 0.227,\ \psi_{0,18} = 1.000,\ \phi_{0,18} = 1.000 \Rightarrow\ p_{0,18} = 0.115$
- Para ir al cliente 6 desde 0: $\tau_{0,6} = 0.010,\ \eta_{0,6} = 0.208,\ \psi_{0,6} = 1.000,\ \phi_{0,6} = 1.000 \Rightarrow\ p_{0,6} = 0.108$

Cliente elegido: 12, Rutas: [0, 12], Capacidad restante del vehículo: 5
Cliente actual: 12

- Para ir al cliente 8 desde 12: $\tau_{12,8} = 0.010,\ \eta_{12,8} = 0.096,\ \psi_{12,8} = 1.000,\ \phi_{12,8} = 1.000 \Rightarrow\ p_{12,8} = 0.066$
- Para ir al cliente 10 desde 12: $\tau_{12,10} = 0.010,\ \eta_{12,10} = 0.506,\ \psi_{12,10} = 1.000,\ \phi_{12,10} = 1.000 \Rightarrow\ p_{12,10} = 0.179$
- Para ir al cliente 2 desde 12: $\tau_{12,2} = 0.010,\ \eta_{12,2} = 0.224,\ \psi_{12,2} = 1.000,\ \phi_{12,2} = 1.000 \Rightarrow\ p_{12,2} = 0.110$
- Para ir al cliente 16 desde 12: $\tau_{12,16} = 0.010,\ \eta_{12,16} = 1.141,\ \psi_{12,16} = 1.000,\ \phi_{12,16} = 1.000 \Rightarrow\ p_{12,16} = 0.291$
- Para ir al cliente 14 desde 12: $\tau_{12,14} = 0.010,\ \eta_{12,14} = 0.119,\ \psi_{12,14} = 1.000,\ \phi_{12,14} = 1.000 \Rightarrow\ p_{12,14} = 0.075$

- Para ir al cliente 4 desde 12: $\tau_{12,4} = 0.010,\ \eta_{12,4} = 0.119,\ \psi_{12,4} = 1.000,\ \phi_{12,4} = 1.000 \Rightarrow p_{12,4} = 0.075$
- Para ir al cliente 18 desde 12: $\tau_{12,18} = 0.010,\ \eta_{12,18} = 0.340,\ \psi_{12,18} = 1.000,\ \phi_{12,18} = 1.000 \Rightarrow p_{12,18} = 0.141$
- Para ir al cliente 6 desde 12: $\tau_{12,6} = 0.010,\ \eta_{12,6} = 0.091,\ \psi_{12,6} = 1.000,\ \phi_{12,6} = 1.000 \Rightarrow p_{12,6} = 0.064$

Cliente elegido: 18, Rutas: [0,12,18], Capacidad restante del vehículo: 4
Cliente actual: 18

- Para ir al cliente 8 desde 18: $\tau_{18,8} = 0.010,\ \eta_{18,8} = 0.129,\ \psi_{18,8} = 1.000,\ \phi_{18,8} = 1.000 \Rightarrow p_{18,8} = 0.112$
- Para ir al cliente 10 desde 18: $\tau_{18,10} = 0.010,\ \eta_{18,10} = 0.248,\ \psi_{18,10} = 1.000,\ \phi_{18,10} = 1.000 \Rightarrow p_{18,10} = 0.166$
- Para ir al cliente 2 desde 18: $\tau_{18,2} = 0.010,\ \eta_{18,2} = 0.224,\ \psi_{18,2} = 1.000,\ \phi_{18,2} = 1.000 \Rightarrow p_{18,2} = 0.157$
- Para ir al cliente 16 desde 18: $\tau_{18,16} = 0.010,\ \eta_{18,16} = 0.414,\ \psi_{18,16} = 1.000,\ \phi_{18,16} = 1.000 \Rightarrow p_{18,16} = 0.226$
- Para ir al cliente 14 desde 18: $\tau_{18,14} = 0.010,\ \eta_{18,14} = 0.115,\ \psi_{18,14} = 1.000,\ \phi_{18,14} = 1.000 \Rightarrow p_{18,14} = 0.105$
- Para ir al cliente 4 desde 18: $\tau_{18,4} = 0.010,\ \eta_{18,4} = 0.167,\ \psi_{18,4} = 1.000,\ \phi_{18,4} = 1.000 \Rightarrow p_{18,4} = 0.131$
- Para ir al cliente 6 desde 18: $\tau_{18,6} = 0.010,\ \eta_{18,6} = 0.110,\ \psi_{18,6} = 1.000,\ \phi_{18,6} = 1.000 \Rightarrow p_{18,6} = 0.102$

Cliente elegido: 14, Rutas: [0,12,18,14], Capacidad restante del vehículo: 3
Cliente actual: 14

- Para ir al cliente 8 desde 14: $\tau_{14,8} = 0.010,\ \eta_{14,8} = 0.098,\ \psi_{14,8} = 1.000,\ \phi_{14,8} = 1.000 \Rightarrow p_{14,8} = 0.135$
- Para ir al cliente 10 desde 14: $\tau_{14,10} = 0.010,\ \eta_{14,10} = 0.152,\ \psi_{14,10} = 1.000,\ \phi_{14,10} = 1.000 \Rightarrow p_{14,10} = 0.175$
- Para ir al cliente 2 desde 14: $\tau_{14,2} = 0.010,\ \eta_{14,2} = 0.235,\ \psi_{14,2} = 1.000,\ \phi_{14,2} = 1.000 \Rightarrow p_{14,2} = 0.228$
- Para ir al cliente 16 desde 14: $\tau_{14,16} = 0.010,\ \eta_{14,16} = 0.111,\ \psi_{14,16} = 1.000,\ \phi_{14,16} = 1.000 \Rightarrow p_{14,16} = 0.146$
- Para ir al cliente 4 desde 14: $\tau_{14,4} = 0.010,\ \eta_{14,4} = 0.123,\ \psi_{14,4} = 1.000,\ \phi_{14,4} = 1.000 \Rightarrow p_{14,4} = 0.155$
- Para ir al cliente 6 desde 14: $\tau_{14,6} = 0.010,\ \eta_{14,6} = 0.131,\ \psi_{14,6} = 1.000,\ \phi_{14,6} = 1.000 \Rightarrow p_{14,6} = 0.161$

Cliente elegido: 2, Rutas: [0,12,18,14,2], Capacidad restante del vehículo: 1
Cliente actual: 0

- Para ir al cliente 8 desde 0: $\tau_{0,8} = 0.010,\ \eta_{0,8} = 0.202,\ \psi_{0,8} = 1.000,\ \phi_{0,8} = 1.000 \Rightarrow\ p_{0,8} = 0.190$
- Para ir al cliente 10 desde 0: $\tau_{0,10} = 0.010,\ \eta_{0,10} = 0.173,\ \psi_{0,10} = 1.000,\ \phi_{0,10} = 1.000 \Rightarrow\ p_{0,10} = 0.173$
- Para ir al cliente 16 desde 0: $\tau_{0,16} = 0.010,\ \eta_{0,16} = 0.161,\ \psi_{0,16} = 1.000,\ \phi_{0,16} = 1.000 \Rightarrow\ p_{0,16} = 0.166$
- Para ir al cliente 4 desde 0: $\tau_{0,4} = 0.010,\ \eta_{0,4} = 0.382,\ \psi_{0,4} = 1.000,\ \phi_{0,4} = 1.000 \Rightarrow\ p_{0,4} = 0.278$
- Para ir al cliente 6 desde 0: $\tau_{0,6} = 0.010,\ \eta_{0,6} = 0.208,\ \psi_{0,6} = 1.000,\ \phi_{0,6} = 1.000 \Rightarrow\ p_{0,6} = 0.193$

Cliente elegido: 8, Rutas: [0, 12, 18, 14, 2, 0,8], Capacidad restante del vehículo: 4
Cliente actual: 8

- Para ir al cliente 10 desde 8: $\tau_{8,10} = 0.010,\ \eta_{8,10} = 0.095,\ \psi_{8,10} = 1.000,\ \phi_{8,10} = 1.000 \Rightarrow\ p_{8,10} = 0.159$
- Para ir al cliente 16 desde 8: $\tau_{8,16} = 0.010,\ \eta_{8,16} = 0.099,\ \psi_{8,16} = 1.000,\ \phi_{8,16} = 1.000 \Rightarrow\ p_{8,16} = 0.163$
- Para ir al cliente 4 desde 8: $\tau_{8,4} = 0.010,\ \eta_{8,4} = 0.430,\ \psi_{8,4} = 1.000,\ \phi_{8,4} = 1.000 \Rightarrow\ p_{8,4} = 0.394$
- Para ir al cliente 6 desde 8: $\tau_{8,6} = 0.010,\ \eta_{8,6} = 0.249,\ \psi_{8,6} = 1.000,\ \phi_{8,6} = 1.000 \Rightarrow\ p_{8,6} = 0.284$

Cliente elegido: 6, Rutas: [0, 12, 18, 14, 2, 0,8,6], Capacidad restante del vehículo: 2
Cliente actual: 6

- Para ir al cliente 10 desde 6: $\tau_{6,10} = 0.010,\ \eta_{6,10} = 0.096,\ \psi_{6,10} = 1.000,\ \phi_{6,10} = 1.000 \Rightarrow\ p_{6,10} = 0.258$
- Para ir al cliente 16 desde 6: $\tau_{6,16} = 0.010,\ \eta_{6,16} = 0.091,\ \psi_{6,16} = 1.000,\ \phi_{6,16} = 1.000 \Rightarrow\ p_{6,16} = 0.249$
- Para ir al cliente 4 desde 6: $\tau_{6,4} = 0.010,\ \eta_{6,4} = 0.283,\ \psi_{6,4} = 1.000,\ \phi_{6,4} = 1.000 \Rightarrow\ p_{6,4} = 0.493$

Cliente elegido: 4, Rutas: [0, 12, 18, 14, 2, 0,8,6,4,0], Capacidad restante del vehículo: 1
Cliente actual: 0

- Para ir al cliente 10 desde 0: $\tau_{0,10} = 0.010,\ \eta_{0,10} = 0.173,\ \psi_{0,10} = 1.000,\ \phi_{0,10} = 1.000 \Rightarrow\ p_{0,10} = 0.511$
- Para ir al cliente 16 desde 0: $\tau_{0,16} = 0.010,\ \eta_{0,16} = 0.161,\ \psi_{0,16} = 1.000,\ \phi_{0,16} = 1.000 \Rightarrow\ p_{0,16} = 0.489$

Cliente elegido: 10, Rutas: [0, 12, 18, 14, 2, 0,8,6,4,0,10], Capacidad restante del vehículo: 4
Cliente actual: 10
Cliente elegido: 10, Rutas: [0, 12, 18, 14, 2, 0,8,6,4,0,10,16], Capacidad restante del vehículo: 16

Para el segundo día en horario AM:
Cliente actual: 0

- Para ir al cliente 16 desde 0: $\tau_{0,16} = 0.010,\ \eta_{0,16} = 0.161,\ \psi_{0,16} = 0.297,\ \phi_{0,16} = 1.000 \Rightarrow p_{0,16} = 0.093$
- Para ir al cliente 4 desde 0: $\tau_{0,4} = 0.010,\ \eta_{0,4} = 0.382,\ \psi_{0,4} = 0.092,\ \phi_{0,4} = 1.000 \Rightarrow p_{0,4} = 0.087$
- Para ir al cliente 14 desde 0: $\tau_{0,14} = 0.010,\ \eta_{0,14} = 0.165,\ \psi_{0,14} = 0.073,\ \phi_{0,14} = 1.000 \Rightarrow p_{0,14} = 0.047$
- Para ir al cliente 18 desde 0: $\tau_{0,18} = 0.010,\ \eta_{0,18} = 0.227,\ \psi_{0,18} = 0.174,\ \phi_{0,18} = 1.000 \Rightarrow p_{0,18} = 0.088$
- Para ir al cliente 6 desde 0: $\tau_{0,6} = 0.010,\ \eta_{0,6} = 0.208,\ \psi_{0,6} = 0.195,\ \phi_{0,6} = 1.000 \Rightarrow p_{0,6} = 0.088$
- Para ir al cliente 2 desde 0: $\tau_{0,2} = 0.010,\ \eta_{0,2} = 0.303,\ \psi_{0,2} = 0.046,\ \phi_{0,2} = 1.000 \Rightarrow p_{0,2} = 0.053$
- Para ir al cliente 10 desde 0: $\tau_{0,10} = 0.010,\ \eta_{0,10} = 0.173,\ \psi_{0,10} = 1.000,\ \phi_{0,10} = 1.000 \Rightarrow p_{0,10} = 0.178$
- Para ir al cliente 8 desde 0: $\tau_{0,8} = 0.010,\ \eta_{0,8} = 0.202,\ \psi_{0,8} = 1.000,\ \phi_{0,8} = 1.000 \Rightarrow p_{0,8} = 0.196$
- Para ir al cliente 12 desde 0: $\tau_{0,12} = 0.010,\ \eta_{0,12} = 0.161,\ \psi_{0,12} = 1.000,\ \phi_{0,12} = 1.000 \Rightarrow p_{0,12} = 0.171$

Cliente elegido: 2, Rutas: [0, 2] Capacidad restante del vehículo: 6
Cliente actual: 2

- Para ir al cliente 16 desde 2: $\tau_{2,16} = 0.010,\ \eta_{2,16} = 0.203,\ \psi_{2,16} = 1.000,\ \phi_{2,16} = 1.000 \Rightarrow p_{2,16} = 0.001$
- Para ir al cliente 4 desde 2: $\tau_{2,4} = 0.010,\ \eta_{2,4} = 0.169,\ \psi_{2,4} = 0.306,\ \phi_{2,4} = 1.000 \Rightarrow p_{2,4} = 0.001$
- Para ir al cliente 14 desde 2: $\tau_{2,14} = 0.010,\ \eta_{2,14} = 0.235,\ \psi_{2,14} = 0.088,\ \phi_{2,14} = 1.000 \Rightarrow p_{2,14} = 0.001$
- Para ir al cliente 18 desde 2: $\tau_{2,18} = 0.010,\ \eta_{2,18} = 0.224,\ \psi_{2,18} = 0.719,\ \phi_{2,18} = 1.000 \Rightarrow p_{2,18} = 0.001$
- Para ir al cliente 6 desde 2: $\tau_{2,6} = 0.010,\ \eta_{2,6} = 0.135,\ \psi_{2,6} = 0.577,\ \phi_{2,6} = 1.000 \Rightarrow p_{2,6} = 0.001$

- Para ir al cliente 10 desde 2: $\tau_{2,10} = 0.010,\ \eta_{2,10} = 0.318,\ \psi_{2,10} = 0.607,\ \phi_{2,10} = 1.000 \Rightarrow p_{2,10} = 0.991$
- Para ir al cliente 8 desde 2: $\tau_{2,8} = 0.010,\ \eta_{2,8} = 0.121,\ \psi_{2,8} = 0.132,\ \phi_{2,8} = 1.000 \Rightarrow p_{2,8} = 0.001$
- Para ir al cliente 12 desde 2: $\tau_{2,12} = 0.010,\ \eta_{2,12} = 0.224,\ \psi_{2,12} = 0.393,\ \phi_{2,12} = 1.000 \Rightarrow p_{2,12} = 0.001$

Cliente elegido: 10, Rutas: [0, 2,10] Capacidad restante del vehículo: 4
Cliente actual: 10

- Para ir al cliente 16 desde 10: $\tau_{10,16} = 0.010,\ \eta_{10,16} = 0.358,\ \psi_{10,16} = 1.000,\ \phi_{10,16} = 1.000 \Rightarrow p_{10,16} = 0.306$
- Para ir al cliente 4 desde 10: $\tau_{10,4} = 0.010,\ \eta_{10,4} = 0.121,\ \psi_{10,4} = 0.446,\ \phi_{10,4} = 1.000 \Rightarrow p_{10,4} = 0.106$
- Para ir al cliente 14 desde 10: $\tau_{10,14} = 0.010,\ \eta_{10,14} = 0.152,\ \psi_{10,14} = 0.172,\ \phi_{10,14} = 1.000 \Rightarrow p_{10,14} = 0.076$
- Para ir al cliente 18 desde 10: $\tau_{10,18} = 0.010,\ \eta_{10,18} = 0.248,\ \psi_{10,18} = 0.761,\ \phi_{10,18} = 1.000 \Rightarrow p_{10,18} = 0.214$
- Para ir al cliente 6 desde 10: $\tau_{10,6} = 0.010,\ \eta_{10,6} = 0.096,\ \psi_{10,6} = 0.127,\ \phi_{10,6} = 1.000 \Rightarrow p_{10,6} = 0.050$
- Para ir al cliente 8 desde 10: $\tau_{10,8} = 0.010,\ \eta_{10,8} = 0.095,\ \psi_{10,8} = 0.077,\ \phi_{10,8} = 1.000 \Rightarrow p_{10,8} = 0.038$
- Para ir al cliente 12 desde 10: $\tau_{10,12} = 0.010,\ \eta_{10,12} = 0.506,\ \psi_{10,12} = 0.313,\ \phi_{10,12} = 1.000 \Rightarrow p_{10,12} = 0.210$

Cliente elegido: 16, Rutas: [0, 2,10,16] Capacidad restante del vehículo: 3
Cliente actual: 16

- Para ir al cliente 4 desde 16: $\tau_{16,4} = 0.010,\ \eta_{16,4} = 0.122,\ \psi_{16,4} = 0.202,\ \phi_{16,4} = 1.000 \Rightarrow p_{16,4} = 0.091$
- Para ir al cliente 14 desde 16: $\tau_{16,14} = 0.010,\ \eta_{16,14} = 0.111,\ \psi_{16,14} = 1.000,\ \phi_{16,14} = 1.000 \Rightarrow p_{16,14} = 0.192$
- Para ir al cliente 18 desde 16: $\tau_{16,18} = 0.010,\ \eta_{16,18} = 0.414,\ \psi_{16,18} = 0.401,\ \phi_{16,18} = 1.000 \Rightarrow p_{16,18} = 0.268$
- Para ir al cliente 6 desde 16: $\tau_{16,6} = 0.010,\ \eta_{16,6} = 0.091,\ \psi_{16,6} = 0.088,\ \phi_{16,6} = 1.000 \Rightarrow p_{16,6} = 0.051$
- Para ir al cliente 8 desde 16: $\tau_{16,8} = 0.010,\ \eta_{16,8} = 0.099,\ \psi_{16,8} = 0.065,\ \phi_{16,8} = 1.000 \Rightarrow p_{16,8} = 0.046$
- Para ir al cliente 12 desde 16: $\tau_{16,12} = 0.010,\ \eta_{16,12} = 1.141,\ \psi_{16,12} = 0.204,\ \phi_{16,12} = 1.000 \Rightarrow p_{16,12} = 0.352$

Cliente elegido: 18, Rutas: [0, 2,10,16,18] Capacidad restante del vehículo: 2
Cliente actual: 18

- Para ir al cliente 4 desde 18: $\tau_{18,4} = 0.010,\ \eta_{18,4} = 0.167,\ \psi_{18,4} = 0.194,\ \phi_{18,4} = 1.000 \Rightarrow p_{18,4} = 0.216$
- Para ir al cliente 14 desde 18: $\tau_{18,14} = 0.010,\ \eta_{18,14} = 0.115,\ \psi_{18,14} = 0.671,\ \phi_{18,14} = 1.000 \Rightarrow p_{18,14} = 0.321$
- Para ir al cliente 6 desde 18: $\tau_{18,6} = 0.010,\ \eta_{18,6} = 0.110,\ \psi_{18,6} = 0.085,\ \phi_{18,6} = 1.000 \Rightarrow p_{18,6} = 0.111$
- Para ir al cliente 8 desde 18: $\tau_{18,8} = 0.010,\ \eta_{18,8} = 0.129,\ \psi_{18,8} = 0.065,\ \phi_{18,8} = 1.000 \Rightarrow p_{18,8} = 0.107$
- Para ir al cliente 12 desde 18: $\tau_{18,12} = 0.010,\ \eta_{18,12} = 0.340,\ \psi_{18,12} = 0.107,\ \phi_{18,12} = 1.000 \Rightarrow p_{18,12} = 0.245$

Cliente elegido: 14, Rutas: [0, 2,10,16,18,14], Capacidad restante del vehículo: 0
Cliente actual: 0

- Para ir al cliente 4 desde 0: $\tau_{0,4} = 0.010,\ \eta_{0,4} = 0.382,\ \psi_{0,4} = 0.092,\ \phi_{0,4} = 1.000 \Rightarrow p_{0,4} = 0.161$
- Para ir al cliente 6 desde 0: $\tau_{0,6} = 0.010,\ \eta_{0,6} = 0.208,\ \psi_{0,6} = 0.195,\ \phi_{0,6} = 1.000 \Rightarrow p_{0,6} = 0.162$
- Para ir al cliente 8 desde 0: $\tau_{0,8} = 0.010,\ \eta_{0,8} = 0.202,\ \psi_{0,8} = 1.000,\ \phi_{0,8} = 1.000 \Rightarrow p_{0,8} = 0.362$
- Para ir al cliente 12 desde 0: $\tau_{0,12} = 0.010,\ \eta_{0,12} = 0.161,\ \psi_{0,12} = 1.000,\ \phi_{0,12} = 1.000 \Rightarrow p_{0,12} = 0.315$

Cliente elegido: 12, Rutas: [0, 2,10,16,18,14,0,12] Capacidad restante del vehículo: 5
Cliente actual: 12

- Para ir al cliente 4 desde 12: $\tau_{12,4} = 0.010,\ \eta_{12,4} = 0.119,\ \psi_{12,4} = 0.881,\ \phi_{12,4} = 1.000 \Rightarrow p_{12,4} = 0.621$
- Para ir al cliente 6 desde 12: $\tau_{12,6} = 0.010,\ \eta_{12,6} = 0.091,\ \psi_{12,6} = 0.137,\ \phi_{12,6} = 1.000 \Rightarrow p_{12,6} = 0.208$
- Para ir al cliente 8 desde 12: $\tau_{12,8} = 0.010,\ \eta_{12,8} = 0.096,\ \psi_{12,8} = 0.086,\ \phi_{12,8} = 1.000 \Rightarrow p_{12,8} = 0.170$

Cliente elegido: 12, Rutas: [0, 2,10,16,18,14,0,12,4] Capacidad restante del vehículo: 2
Cliente actual: 0

- Para ir al cliente 6 desde 0: $\tau_{0,6} = 0.010,\ \eta_{0,6} = 0.208,\ \psi_{0,6} = 0.195,\ \phi_{0,6} = 1.000 \Rightarrow p_{0,6} = 0.309$
- Para ir al cliente 8 desde 0: $\tau_{0,8} = 0.010,\ \eta_{0,8} = 0.202,\ \psi_{0,8} = 1.000,\ \phi_{0,8} = 1.000 \Rightarrow p_{0,8} = 0.691$

Cliente elegido: 6, Rutas:[0, 2,10,16,18,14,0,12,4,0,6], Capacidad restante del vehículo: 6
Cliente elegido: 8, Rutas:[0, 2,10,16,18,14,0,12,4,0,6,8], Capacidad restante del vehículo: 3

Para el primer día en horario PM:
Cliente actual: 0

- Para ir al cliente 5 desde 0: $\tau_{0,5} = 0.010,\ \eta_{0,5} = 0.243,\ \psi_{0,5} = 1.000,\ \phi_{0,5} = 1.000 \Rightarrow\ p_{0,5} = 0.124$
- Para ir al cliente 17 desde 0: $\tau_{0,17} = 0.010,\ \eta_{0,17} = 0.188,\ \psi_{0,17} = 1.000,\ \phi_{0,17} = 1.000 \Rightarrow\ p_{0,17} = 0.106$
- Para ir al cliente 11 desde 0: $\tau_{0,11} = 0.010,\ \eta_{0,11} = 0.192,\ \psi_{0,11} = 1.000,\ \phi_{0,11} = 1.000 \Rightarrow\ p_{0,11} = 0.106$
- Para ir al cliente 7 desde 0: $\tau_{0,7} = 0.010,\ \eta_{0,7} = 0.296,\ \psi_{0,7} = 1.000,\ \phi_{0,7} = 1.000 \Rightarrow\ p_{0,7} = 0.139$
- Para ir al cliente 1 desde 0: $\tau_{0,1} = 0.010,\ \eta_{0,1} = 0.202,\ \psi_{0,1} = 1.000,\ \phi_{0,1} = 1.000 \Rightarrow\ p_{0,1} = 0.139$
- Para ir al cliente 15 desde 0: $\tau_{0,15} = 0.010,\ \eta_{0,15} = 0.132,\ \psi_{0,15} = 1.000,\ \phi_{0,15} = 1.000 \Rightarrow\ p_{0,15} = 0.086$
- Para ir al cliente 13 desde 0: $\tau_{0,13} = 0.010,\ \eta_{0,13} = 0.202,\ \psi_{0,13} = 1.000,\ \phi_{0,13} = 1.000 \Rightarrow\ p_{0,13} = 0.110$
- Para ir al cliente 3 desde 0: $\tau_{0,3} = 0.010,\ \eta_{0,3} = 0.245,\ \psi_{0,3} = 1.000,\ \phi_{0,3} = 1.000 \Rightarrow\ p_{0,3} = 0.124$
- Para ir al cliente 9 desde 0: $\tau_{0,9} = 0.010,\ \eta_{0,9} = 0.153,\ \psi_{0,9} = 1.000,\ \phi_{0,9} = 1.000 \Rightarrow\ p_{0,9} = 0.093$

Cliente elegido: 11, Rutas: [0,11], Capacidad restante del vehículo: 6
Cliente actual: 11

- Para ir al cliente 5 desde 11: $\tau_{11,5} = 0.010,\ \eta_{11,5} = 0.107,\ \psi_{11,5} = 1.000,\ \phi_{11,5} = 1.000 \Rightarrow\ p_{11,5} = 0.087$
- Para ir al cliente 17 desde 11: $\tau_{11,17} = 0.010,\ \eta_{11,17} = 0.106,\ \psi_{11,17} = 1.000,\ \phi_{11,17} = 1.000 \Rightarrow\ p_{11,17} = 0.086$
- Para ir al cliente 7 desde 11: $\tau_{11,7} = 0.010,\ \eta_{11,7} = 0.117,\ \psi_{11,7} = 1.000,\ \phi_{11,7} = 1.000 \Rightarrow\ p_{11,7} = 0.091$
- Para ir al cliente 1 desde 11: $\tau_{11,1} = 0.010,\ \eta_{11,1} = 1.314,\ \psi_{11,1} = 1.000,\ \phi_{11,1} = 1.000 \Rightarrow\ p_{11,1} = 0.389$
- Para ir al cliente 15 desde 11: $\tau_{11,15} = 0.010,\ \eta_{11,15} = 0.081,\ \psi_{11,15} = 1.000,\ \phi_{11,15} = 1.000 \Rightarrow\ p_{11,15} = 0.073$

- Para ir al cliente 13 desde 11: $\tau_{11,13} = 0.010,\ \eta_{11,13} = 0.124,\ \psi_{11,13} = 1.000,\ \phi_{11,13} = 1.000 \Rightarrow\ p_{11,13} = 0.094$
- Para ir al cliente 3 desde 11: $\tau_{11,3} = 0.010,\ \eta_{11,3} = 0.138,\ \psi_{11,3} = 1.000,\ \phi_{11,3} = 1.000 \Rightarrow p_{11,3} = 0.101$
- Para ir al cliente 9 desde 11: $\tau_{11,9} = 0.010,\ \eta_{11,9} = 0.091,\ \psi_{11,9} = 1.000,\ \phi_{11,9} = 1.000 \Rightarrow p_{11,9} = 0.078$

Cliente elegido: 1, Rutas: [0,11,1], Capacidad restante del vehículo: 3
Cliente actual: 1

- Para ir al cliente 5 desde 1: $\tau_{1,5} = 0.010,\ \eta_{1,5} = 0.111,\ \psi_{1,5} = 1.000,\ \phi_{1,5} = 1.000 \Rightarrow\ p_{1,5} = 0.143$
- Para ir al cliente 17 desde 1: $\tau_{1,17} = 0.010,\ \eta_{1,17} = 0.106,\ \psi_{1,17} = 1.000,\ \phi_{1,17} = 1.000 \Rightarrow p_{1,17} = 0.139$
- Para ir al cliente 7 desde 1: $\tau_{1,7} = 0.010,\ \eta_{1,7} = 0.120,\ \psi_{1,7} = 1.000,\ \phi_{1,7} = 1.000 \Rightarrow\ p_{1,7} = 0.150$
- Para ir al cliente 15 desde 1: $\tau_{1,15} = 0.010,\ \eta_{1,15} = 0.082,\ \psi_{1,15} = 1.000,\ \phi_{1,15} = 1.000 \Rightarrow p_{1,15} = 0.119$
- Para ir al cliente 13 desde 1: $\tau_{1,13} = 0.010,\ \eta_{1,13} = 0.121,\ \psi_{1,13} = 1.000,\ \phi_{1,13} = 1.000 \Rightarrow p_{1,13} = 0.150$
- Para ir al cliente 3 desde 1: $\tau_{1,3} = 0.010,\ \eta_{1,3} = 0.151,\ \psi_{1,3} = 1.000,\ \phi_{1,3} = 1.000 \Rightarrow\ p_{1,3} = 0.172$
- Para ir al cliente 9 desde 1: $\tau_{1,9} = 0.010,\ \eta_{1,9} = 0.091,\ \psi_{1,9} = 1.000,\ \phi_{1,9} = 1.000 \Rightarrow\ p_{1,9} = 0.127$

Cliente elegido: 3, Rutas:[0,11,1,3], Capacidad restante del vehículo: 2
Cliente actual: 3

- Para ir al cliente 5 desde 3: $\tau_{3,5} = 0.010,\ \eta_{3,5} = 0.198,\ \psi_{3,5} = 1.000,\ \phi_{3,5} = 1.000 \Rightarrow\ p_{3,5} = 0.207$
- Para ir al cliente 17 desde 3: $\tau_{3,17} = 0.010,\ \eta_{3,17} = 0.116,\ \psi_{3,17} = 1.000,\ \phi_{3,17} = 1.000 \Rightarrow p_{3,17} = 0.151$
- Para ir al cliente 7 desde 3: $\tau_{3,7} = 0.010,\ \eta_{3,7} = 0.200,\ \psi_{3,7} = 1.000,\ \phi_{3,7} = 1.000 \Rightarrow\ p_{3,7} = 0.209$
- Para ir al cliente 15 desde 3: $\tau_{3,15} = 0.010,\ \eta_{3,15} = 0.103,\ \psi_{3,15} = 1.000,\ \phi_{3,15} = 1.000 \Rightarrow p_{3,15} = 0.140$
- Para ir al cliente 13 desde 3: $\tau_{3,13} = 0.010,\ \eta_{3,13} = 0.114,\ \psi_{3,13} = 1.000,\ \phi_{3,13} = 1.000 \Rightarrow p_{3,13} = 0.149$
- Para ir al cliente 9 desde 3: $\tau_{3,9} = 0.010,\ \eta_{3,9} = 0.108,\ \psi_{3,9} = 1.000,\ \phi_{3,9} = 1.000 \Rightarrow\ p_{3,9} = 0.145$

Cliente elegido: 7, Rutas:[0,11,1,3,7], Capacidad restante del vehículo: 0
Cliente actual: 0

- Para ir al cliente 5 desde 0: $\tau_{0,5} = 0.010,\ \eta_{0,5} = 0.243,\ \psi_{0,5} = 1.000,\ \phi_{0,5} = 1.000 \Rightarrow\ p_{0,5} = 0.238$
- Para ir al cliente 17 desde 0: $\tau_{0,17} = 0.010,\ \eta_{0,17} = 0.188,\ \psi_{0,17} = 1.000,\ \phi_{0,17} = 1.000 \Rightarrow\ p_{0,17} = 0.204$
- Para ir al cliente 15 desde 0: $\tau_{0,15} = 0.010,\ \eta_{0,15} = 0.132,\ \psi_{0,15} = 1.000,\ \phi_{0,15} = 1.000 \Rightarrow\ p_{0,15} = 0.165$
- Para ir al cliente 13 desde 0: $\tau_{0,13} = 0.010,\ \eta_{0,13} = 0.202,\ \psi_{0,13} = 1.000,\ \phi_{0,13} = 1.000 \Rightarrow\ p_{0,13} = 0.213$
- Para ir al cliente 9 desde 0: $\tau_{0,9} = 0.010,\ \eta_{0,9} = 0.153,\ \psi_{0,9} = 1.000,\ \phi_{0,9} = 1.000 \Rightarrow\ p_{0,9} = 0.180$

Cliente elegido: 9, Rutas:[0,11,1,3,7,9], Capacidad restante del vehículo: 5
Cliente actual: 9

- Para ir al cliente 5 desde 9: $\tau_{9,5} = 0.010,\ \eta_{9,5} = 0.223,\ \psi_{9,5} = 1.000,\ \phi_{9,5} = 1.000 \Rightarrow\ p_{9,5} = 0.174$
- Para ir al cliente 17 desde 9: $\tau_{9,17} = 0.010,\ \eta_{9,17} = 0.545,\ \psi_{9,17} = 1.000,\ \phi_{9,17} = 1.000 \Rightarrow\ p_{9,17} = 0.297$
- Para ir al cliente 15 desde 9: $\tau_{9,15} = 0.010,\ \eta_{9,15} = 0.664,\ \psi_{9,15} = 1.000,\ \phi_{9,15} = 1.000 \Rightarrow\ p_{9,15} = 0.334$
- Para ir al cliente 13 desde 9: $\tau_{9,13} = 0.010,\ \eta_{9,13} = 0.269,\ \psi_{9,13} = 1.000,\ \phi_{9,13} = 1.000 \Rightarrow\ p_{9,13} = 0.194$

Cliente elegido: 17, Rutas: [0,11,1,3,7,9,17], Capacidad restante del vehículo: 3
Cliente actual: 17

- Para ir al cliente 5 desde 17: $\tau_{17,5} = 0.010,\ \eta_{17,5} = 0.220,\ \psi_{17,5} = 1.000,\ \phi_{17,5} = 1.000 \Rightarrow\ p_{17,5} = 0.257$
- Para ir al cliente 15 desde 17: $\tau_{17,15} = 0.010,\ \eta_{17,15} = 0.301,\ \psi_{17,15} = 1.000,\ \phi_{17,15} = 1.000 \Rightarrow\ p_{17,15} = 0.310$
- Para ir al cliente 13 desde 17: $\tau_{17,13} = 0.010,\ \eta_{17,13} = 0.524,\ \psi_{17,13} = 1.000,\ \phi_{17,13} = 1.000 \Rightarrow\ p_{17,13} = 0.433$

Cliente elegido: 15, Rutas: [0,11,1,3,7,9,17,15], Capacidad restante del vehículo: 0
Cliente actual: 0

- Para ir al cliente 5 desde 0: $\tau_{0,5} = 0.010,\ \eta_{0,5} = 0.243,\ \psi_{0,5} = 1.000,\ \phi_{0,5} = 1.000 \Rightarrow\ p_{0,5} = 0.528$
- Para ir al cliente 13 desde 0: $\tau_{0,13} = 0.010,\ \eta_{0,13} = 0.202,\ \psi_{0,13} = 1.000,\ \phi_{0,13} = 1.000 \Rightarrow\ p_{0,13} = 0.472$

Cliente elegido: 5, Rutas:[0,11,1,3,7,9,17,15,0,5], Capacidad restante del vehículo: 4
Cliente elegido: 13, Rutas:[0,11,1,3,7,9,17,15,0,5,13], Capacidad restante del vehículo: 1

Para el segundo día en horario PM:
Cliente actual: 0

- Para ir al cliente 11 desde 0: $\tau_{0,11} = 0.010,\ \eta_{0,11} = 0.192,\ \psi_{0,11} = 0.002,\ \phi_{0,11} = 1.000 \Rightarrow p_{0,11} = 0.108$
- Para ir al cliente 5 desde 0: $\tau_{0,5} = 0.010,\ \eta_{0,5} = 0.243,\ \psi_{0,5} = 0.002,\ \phi_{0,5} = 1.000 \Rightarrow p_{0,5} = 0.124$
- Para ir al cliente 7 desde 0: $\tau_{0,7} = 0.010,\ \eta_{0,7} = 0.296,\ \psi_{0,7} = 0.002,\ \phi_{0,7} = 1.000 \Rightarrow p_{0,7} = 0.137$
- Para ir al cliente 1 desde 0: $\tau_{0,1} = 0.010,\ \eta_{0,1} = 0.202,\ \psi_{0,1} = 0.002,\ \phi_{0,1} = 1.000 \Rightarrow p_{0,1} = 0.111$
- Para ir al cliente 17 desde 0: $\tau_{0,17} = 0.010,\ \eta_{0,17} = 0.188,\ \psi_{0,17} = 0.002,\ \phi_{0,17} = 1.000 \Rightarrow p_{0,17} = 0.106$
- Para ir al cliente 15 desde 0: $\tau_{0,15} = 0.010,\ \eta_{0,15} = 0.132,\ \psi_{0,15} = 0.002,\ \phi_{0,15} = 1.000 \Rightarrow p_{0,15} = 0.086$
- Para ir al cliente 9 desde 0: $\tau_{0,9} = 0.010,\ \eta_{0,9} = 0.153,\ \psi_{0,9} = 0.002,\ \phi_{0,9} = 1.000 \Rightarrow p_{0,9} = 0.094$
- Para ir al cliente 3 desde 0: $\tau_{0,3} = 0.010,\ \eta_{0,3} = 0.245,\ \psi_{0,3} = 0.002,\ \phi_{0,3} = 1.000 \Rightarrow p_{0,3} = 0.123$
- Para ir al cliente 13 desde 0: $\tau_{0,13} = 0.010,\ \eta_{0,13} = 0.202,\ \psi_{0,13} = 0.002,\ \phi_{0,13} = 1.000 \Rightarrow p_{0,13} = 0.110$

Cliente elegido: 7, Rutas:[0,7], Capacidad restante del vehículo: 4
Cliente actual: 7

- Para ir al cliente 11 desde 7: $\tau_{7,11} = 0.010,\ \eta_{7,11} = 0.117,\ \psi_{7,11} = 0.129,\ \phi_{7,11} = 1.000 \Rightarrow p_{7,11} = 0.037$
- Para ir al cliente 5 desde 7: $\tau_{7,5} = 0.010,\ \eta_{7,5} = 1.144,\ \psi_{7,5} = 0.877,\ \phi_{7,5} = 1.000 \Rightarrow p_{7,5} = 0.383$
- Para ir al cliente 1 desde 7: $\tau_{7,1} = 0.010,\ \eta_{7,1} = 0.120,\ \psi_{7,1} = 0.174,\ \phi_{7,1} = 1.000 \Rightarrow p_{7,1} = 0.044$
- Para ir al cliente 17 desde 7: $\tau_{7,17} = 0.010,\ \eta_{7,17} = 0.250,\ \psi_{7,17} = 1.000,\ \phi_{7,17} = 1.000 \Rightarrow p_{7,17} = 0.165$
- Para ir al cliente 15 desde 7: $\tau_{7,15} = 0.010,\ \eta_{7,15} = 0.210,\ \psi_{7,15} = 0.281,\ \phi_{7,15} = 1.000 \Rightarrow p_{7,15} = 0.078$

- Para ir al cliente 9 desde 7: $\tau_{7,9} = 0.010,\ \eta_{7,9} = 0.235,\ \psi_{7,9} = 0.479,\ \phi_{7,9} = 1.000 \Rightarrow\ p_{7,9} = 0.110$
- Para ir al cliente 3 desde 7: $\tau_{7,3} = 0.010,\ \eta_{7,3} = 0.200,\ \psi_{7,3} = 0.193,\ \phi_{7,3} = 1.000 \Rightarrow\ p_{7,3} = 0.063$
- Para ir al cliente 13 desde 7: $\tau_{7,13} = 0.010,\ \eta_{7,13} = 0.203,\ \psi_{7,13} = 0.675,\ \phi_{7,13} = 1.000 \Rightarrow\ p_{7,13} = 0.119$

Cliente elegido: 5, Rutas:[0,7,5], Capacidad restante del vehículo: 1
Cliente actual: 5

- Para ir al cliente 11 desde 5: $\tau_{5,11} = 0.010,\ \eta_{5,11} = 0.107,\ \psi_{5,11} = 0.096,\ \phi_{5,11} = 1.000 \Rightarrow\ p_{5,11} = 0.055$
- Para ir al cliente 1 desde 5: $\tau_{5,1} = 0.010,\ \eta_{5,1} = 0.111,\ \psi_{5,1} = 0.120,\ \phi_{5,1} = 1.000 \Rightarrow\ p_{5,1} = 0.063$
- Para ir al cliente 17 desde 5: $\tau_{5,17} = 0.010,\ \eta_{5,17} = 0.220,\ \psi_{5,17} = 0.413,\ \phi_{5,17} = 1.000 \Rightarrow\ p_{5,17} = 0.176$
- Para ir al cliente 15 desde 5: $\tau_{5,15} = 0.010,\ \eta_{5,15} = 0.212,\ \psi_{5,15} = 0.574,\ \phi_{5,15} = 1.000 \Rightarrow\ p_{5,15} = 0.203$
- Para ir al cliente 9 desde 5: $\tau_{5,9} = 0.010,\ \eta_{5,9} = 0.223,\ \psi_{5,9} = 0.239,\ \phi_{5,9} = 1.000 \Rightarrow\ p_{5,9} = 0.135$
- Para ir al cliente 3 desde 5: $\tau_{5,3} = 0.010,\ \eta_{5,3} = 0.198,\ \psi_{5,3} = 0.308,\ \phi_{5,3} = 1.000 \Rightarrow\ p_{5,3} = 0.143$
- Para ir al cliente 13 desde 5: $\tau_{5,13} = 0.010,\ \eta_{5,13} = 0.176,\ \psi_{5,13} = 0.877,\ \phi_{5,13} = 1.000 \Rightarrow\ p_{5,13} = 0.225$

Cliente elegido: 11, Rutas:Rutas:[0,7,5,11], Capacidad restante del vehículo: 0
Cliente actual: 0

- Para ir al cliente 1 desde 0: $\tau_{0,1} = 0.010,\ \eta_{0,1} = 0.202,\ \psi_{0,1} = 0.002,\ \phi_{0,1} = 1.000 \Rightarrow\ p_{0,1} = 0.176$
- Para ir al cliente 17 desde 0: $\tau_{0,17} = 0.010,\ \eta_{0,17} = 0.188,\ \psi_{0,17} = 0.002,\ \phi_{0,17} = 1.000 \Rightarrow\ p_{0,17} = 0.168$
- Para ir al cliente 15 desde 0: $\tau_{0,15} = 0.010,\ \eta_{0,15} = 0.132,\ \psi_{0,15} = 0.002,\ \phi_{0,15} = 1.000 \Rightarrow\ p_{0,15} = 0.136$
- Para ir al cliente 9 desde 0: $\tau_{0,9} = 0.010,\ \eta_{0,9} = 0.153,\ \psi_{0,9} = 0.002,\ \phi_{0,9} = 1.000 \Rightarrow\ p_{0,9} = 0.149$
- Para ir al cliente 3 desde 0: $\tau_{0,3} = 0.010,\ \eta_{0,3} = 0.245,\ \psi_{0,3} = 0.002,\ \phi_{0,3} = 1.000 \Rightarrow\ p_{0,3} = 0.196$
- Para ir al cliente 13 desde 0: $\tau_{0,13} = 0.010,\ \eta_{0,13} = 0.202,\ \psi_{0,13} = 0.002,\ \phi_{0,13} = 1.000 \Rightarrow\ p_{0,13} = 0.175$

Cliente elegido: 3, Rutas: [0,7,5,11,0,3], Capacidad restante del vehículo: 5
Cliente actual: 3

- Para ir al cliente 1 desde 3: $\tau_{3,1} = 0.010,\ \eta_{3,1} = 0.151,\ \psi_{3,1} = 0.268,\ \phi_{3,1} = 1.000 \Rightarrow p_{3,1} = 0.001$
- Para ir al cliente 17 desde 3: $\tau_{3,17} = 0.010,\ \eta_{3,17} = 0.116,\ \psi_{3,17} = 0.231,\ \phi_{3,17} = 1.000 \Rightarrow p_{3,17} = 0.001$
- Para ir al cliente 15 desde 3: $\tau_{3,15} = 0.010,\ \eta_{3,15} = 0.103,\ \psi_{3,15} = 0.900,\ \phi_{3,15} = 1.000 \Rightarrow p_{3,15} = 0.994$
- Para ir al cliente 9 desde 3: $\tau_{3,9} = 0.010,\ \eta_{3,9} = 0.108,\ \psi_{3,9} = 0.148,\ \phi_{3,9} = 1.000 \Rightarrow p_{3,9} = 0.001$
- Para ir al cliente 13 desde 3: $\tau_{3,13} = 0.010,\ \eta_{3,13} = 0.114,\ \psi_{3,13} = 0.476,\ \phi_{3,13} = 1.000 \Rightarrow p_{3,13} = 0.001$

Cliente elegido: 15, Rutas: [0,7,5,11,0,3,15], Capacidad restante del vehículo: 4
Cliente actual: 15

- Para ir al cliente 1 desde 15: $\tau_{15,1} = 0.010,\ \eta_{15,1} = 0.082,\ \psi_{15,1} = 0.052,\ \phi_{15,1} = 1.000 \Rightarrow p_{15,1} = 0.084$
- Para ir al cliente 17 desde 15: $\tau_{15,17} = 0.010,\ \eta_{15,17} = 0.301,\ \psi_{15,17} = 0.114,\ \phi_{15,17} = 1.000 \Rightarrow p_{15,17} = 0.270$
- Para ir al cliente 9 desde 15: $\tau_{15,9} = 0.010,\ \eta_{15,9} = 0.664,\ \psi_{15,9} = 0.114,\ \phi_{15,9} = 1.000 \Rightarrow p_{15,9} = 0.434$
- Para ir al cliente 13 desde 15: $\tau_{15,13} = 0.010,\ \eta_{15,13} = 0.191,\ \psi_{15,13} = 0.121,\ \phi_{15,13} = 1.000 \Rightarrow p_{15,13} = 0.212$

Cliente elegido: 9, Rutas: [0,7,5,11,0,3,15,9], Capacidad restante del vehículo: 1
Cliente actual: 0

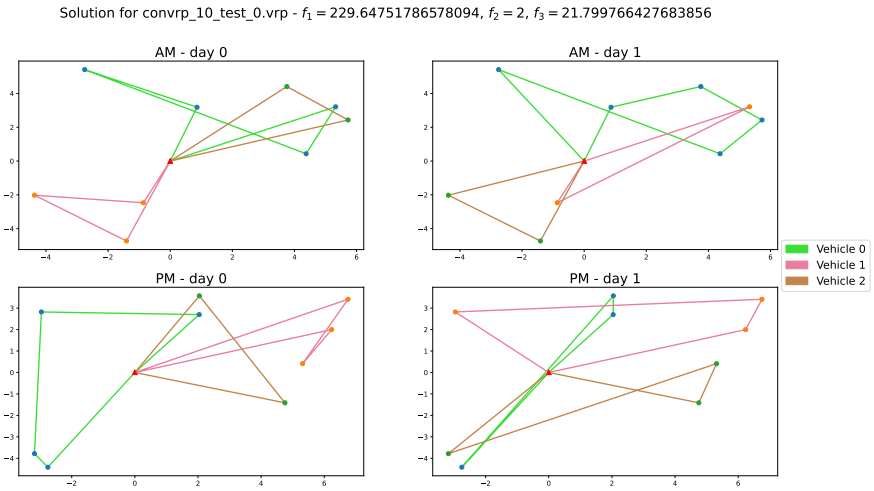
- Para ir al cliente 1 desde 0: $\tau_{0,1} = 0.010,\ \eta_{0,1} = 0.202,\ \psi_{0,1} = 0.002,\ \phi_{0,1} = 1.000 \Rightarrow p_{0,1} = 0.339$
- Para ir al cliente 17 desde 0: $\tau_{0,17} = 0.010,\ \eta_{0,17} = 0.188,\ \psi_{0,17} = 0.002,\ \phi_{0,17} = 1.000 \Rightarrow p_{0,17} = 0.324$
- Para ir al cliente 13 desde 0: $\tau_{0,13} = 0.010,\ \eta_{0,13} = 0.202,\ \psi_{0,13} = 0.002,\ \phi_{0,13} = 1.000 \Rightarrow p_{0,13} = 0.337$

Cliente elegido: 1, Rutas: [0,7,5,11,0,3,15,9,0,1], Capacidad restante del vehículo: 4
Cliente actual: 1

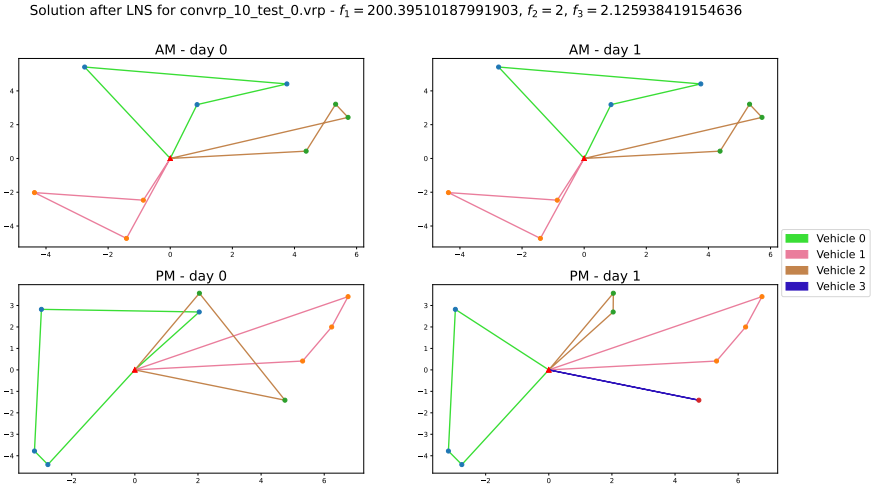
- Para ir al cliente 17 desde 1: $\tau_{1,17} = 0.010,\ \eta_{1,17} = 0.106,\ \psi_{1,17} = 0.142,\ \phi_{1,17} = 1.000 \Rightarrow p_{1,17} = 0.392$
- Para ir al cliente 13 desde 1: $\tau_{1,13} = 0.010,\ \eta_{1,13} = 0.121,\ \psi_{1,13} = 0.291,\ \phi_{1,13} = 1.000 \Rightarrow p_{1,13} = 0.608$

Cliente elegido: 17, Rutas:[0,7,5,11,0,3,15,9,0,1,17], Capacidad restante del vehículo: 3
Cliente elegido: 13, Rutas:[0,7,5,11,0,3,15,9,0,1,17,13], Capacidad restante del vehículo: 0

En la Figura B.1 se muestran tanto la solución construida por la hormiga y su mejora con búsqueda local. Para este ejemplo mostramos también el efecto de usar 10 hormigas con los mismos parámetros para cada uno de los tres indicadores $HV, R2, \epsilon^+$. Una vez que las hormigas de cada colonia construyen las soluciones y las mejoran con búsqueda local. Calculan los valores de aptitud y descartan a las soluciones que menor calidad aportan. En la Figura B.2 se muestran las matrices de feromonas de cada colonia de indicadores después de 10 iteraciones del algoritmo. Las colonias tienen similitudes en varias entradas, aunque estos valores aún pueden dar distintas probabilidades. En la Figura B.3 se muestran las matrices de cada indicador después de 30 iteraciones en cMIBACO, donde se puede notar que en todas las matrices los valores de feromona aún no convergen en su mayoría hacia un máximo. Lo cual significa que aún hay posibilidades de tener diversidad de soluciones por colonia. Finalmente en la Figura B.4 se muestran las feromonas después de 50 iteraciones en cMIBACO donde se mantienen en el mismo rango los valores de feromona respecto a la figura anterior, por lo que las soluciones por indicador pueden estar variando poco debido a que es una instancia de MOGenConVRP pequeña.

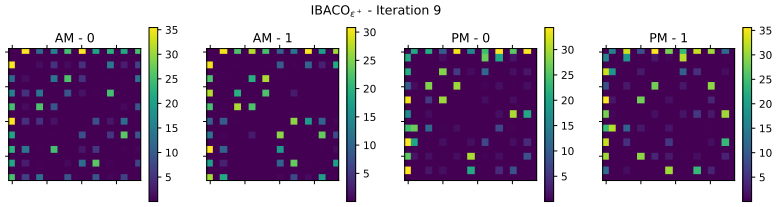


(a) Solución construida por una hormiga antes de búsqueda local.

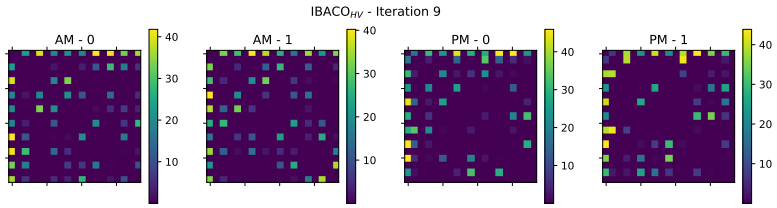


(b) Solución después de aplicarle búsqueda local

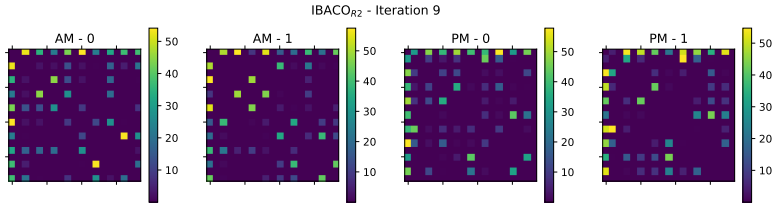
Figura B.1: La solución paso a paso construida en el ejemplo y su mejora con búsqueda local.



(a) Matrices de feromonas de IBACO$_{\epsilon+}$.

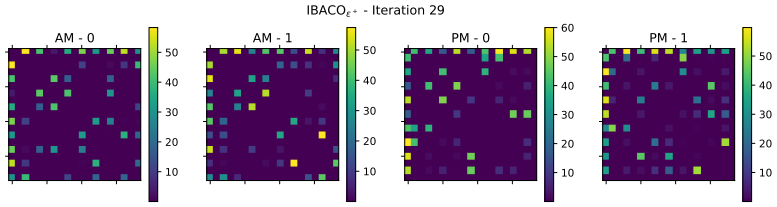


(b) Matrices de feromonas de IBACO$_{HV}$.

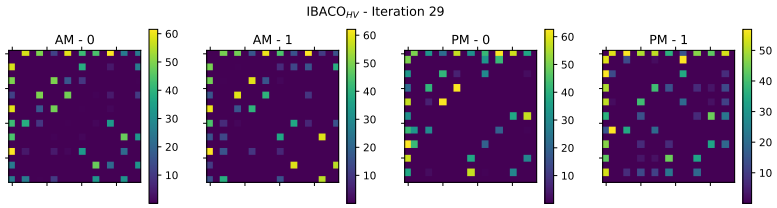


(c) Matrices de feromonas de IBACO$_{R2}$.

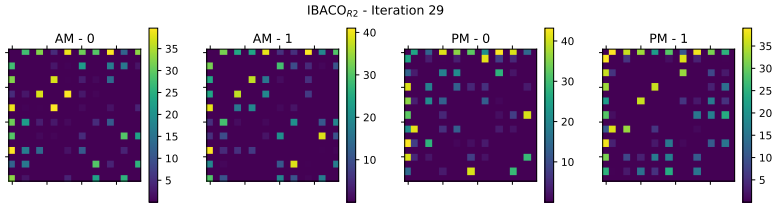
Figura B.2: Matrices de feromonas de cada indicador después de 10 iteraciones en cMIBACO.



(a) Matrices de feromonas de IBACO$_{\epsilon+}$.

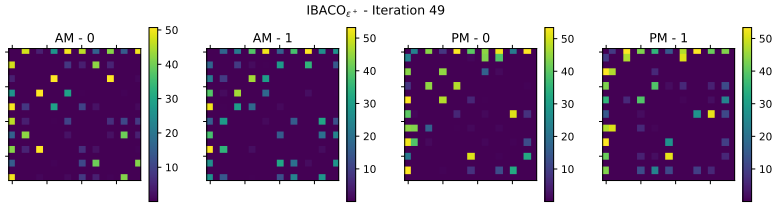


(b) Matrices de feromonas de IBACO$_{HV}$.

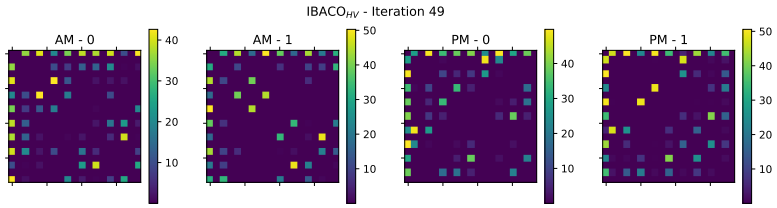


(c) Matrices de feromonas de IBACO$_{R2}$.

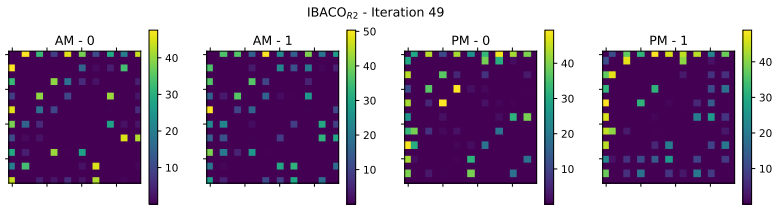
Figura B.3: Matrices de feromonas de cada indicador después de 30 iteraciones en cMIBACO.



(a) Matrices de feromonas de IBACO$_{\epsilon+}$.



(b) Matrices de feromonas de IBACO$_{HV}$.



(c) Matrices de feromonas de IBACO$_{R2}$.

Figura B.4: Matrices de feromonas de cada indicador después de 50 iteraciones en cMIBACO.

Bibliografía

[1] AbdAllah, A. M. F., Essam, D. L., and Sarker, R. A. On solving periodic re-optimization dynamic vehicle routing problems. *Applied Soft Computing 55* (2017), 1–12.

[2] Ahuja, R., Möhring, R., and Zaroliagis, C. *Robust and Online Large-Scale Optimization*, vol. 5868. 01 2009.

[3] Alaya, I., and Ghedira, K. Ant colony optimization for multi-objective optimization problems. vol. 1, pp. 450 – 457.

[4] Allahviranloo, M., Chow, J. Y., and Recker, W. W. Selective vehicle routing problems under uncertainty without recourse. *Transportation Research Part E: Logistics and Transportation Review 62* (2014), 68–88.

[5] Amuso, V. J., Schneible, R. S., Antonik, P., and Zhang, Y. A strength pareto evolutionary algorithm (spea) for multi-mission radar waveform optimization. In *2004 International Waveform Diversity & Design Conference* (2004), pp. 1–7.

[6] Angus, D. Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem. pp. 333 – 340.

[7] Asghari, M., and Mirzapour Al-e-hashem, S. M. J. Green vehicle routing problem: A state-of-the-art review. *International Journal of Production Economics 231* (2021), 107899.

[8] Baños, R., Ortega, J., Gil, C., Fernández, A., and de Toro, F. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications 40*, 5 (2013), 1696–1707.

[9] Baños, R., Ortega, J., Gil, C., Márquez, A. L., and de Toro, F. A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. *Computers & Industrial Engineering 65*, 2 (2013), 286–296.

[10] Benjamin, A., and Beasley, J. Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research 37*, 12 (2010), 2270–2280.

[11] Bertsimas, D. A vehicle routing problem with stochastic demand. *Operations Research 40* (06 1992), 574–585.

[12] Bertsimas, D. A vehicle routing problem with stochastic demand. *Operations Research 40* (06 1992), 574–585.

[13] Beume, N., Fonseca, C. M., Lopez-Ibanez, M., Paquete, L., and Vahrenhold, J. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation 13*, 5 (2009), 1075–1082.

[14] Beume, N., Naujoks, B., and Emmerich, M. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research 181*, 3 (2007), 1653–1669.

[15] Birattari, M., Paquete, L., and Stützle, T. Classification of metaheuristics and design of experiments for the analysis of components.

[16] Blank, J., and Deb, K. pymoo: Multi-objective optimization in python. *IEEE Access 8* (2020), 89497–89509.

[17] Bonabeau, E., Dorigo, M., and Theraulaz, G. Swarm intelligence : From natural to artificial systems / e. bonabeau, m. dorigo, g. theraulaz.

[18] Brassard, G., and Bratley, P. *Algorithmics: Theory and Practice*. Prentice Hall, 1988.

[19] Bullnheimer, B., Hartl, R., and Strauss, C. A new rank based version of the ant system - a computational study. *Central European Journal of Operations Research 7* (01 1999), 25–38.

[20] Cacchiani, V., Hemmelmayr, V., and Tricoire, F. A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics 163* (2014), 53–64. Matheuristics 2010.

[21] Cerný, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications 45* (1985), 41–51.

[22] Chen, A.-l., Yang, G.-k., and Wu, Z.-m. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University - Science A 7*, 4 (2006), 607 – 614.

[23] Chen, P., kuan Huang, H., and Dong, X.-Y. Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. *Expert Systems with Applications 37*, 2 (2010), 1620–1627.

[24] Cheng, J., Zhang, G., Li, Z., and Li, Y. Multi-objective ant colony optimization based on decomposition for bi-objective traveling salesman problems. *Soft Computing 16* (04 2012).

[25] Coello, C. C., Lamont, G. B., and van Veldhuizen, D. A. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer US, 2007.

[26] Dantzig, G. B., and Ramser, J. H. The truck dispatching problem. *Management Science 6* (1959), 80–91.

[27] Deb, K. *Multiobjective Optimization Using Evolutionary Algorithms. Wiley, New York*. 01 2001.

[28] Deb, K., and Jain, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation 18*, 4 (2014), 577–601.

[29] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation 6*, 2 (2002), 182–197.

[30] Deif, I., and Bodin, L. Extension of the clarke and wright algorithm for solving the vehicle routing problem with backhauling. In *Proceedings of the Babson conference on software uses in transportation and logistics management* (1984), Babson Park, MA, pp. 75–96.

[31] Dorigo, M., Birattari, M., and Stutzle, T. Ant colony optimization. *IEEE Computational Intelligence Magazine 1*, 4 (2006), 28–39.

[32] Dorigo, M., and Gambardella, L. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation 1*, 1 (1997), 53–66.

[33] Dorigo, M., Maniezzo, V., and Colorni, A. Ant system: Optimization by a colony of cooperating agents. ieee trans syst man cybernetics - part b. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society 26* (02 1996), 29–41.

[34] Ehrgott, M., Ide, J., and Schöbel, A. Minmax robustness for multi-objective optimization problems. *European Journal of Operational Research 239*, 1 (2014), 17–31.

[35] Euchi, J., Yassine, A., and Chabchoub, H. The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach. *Swarm and Evolutionary Computation 21* (2015), 41–53.

[36] Falcón-Cardona, J. G., and Coello, C. A. C. A multi-objective evolutionary hyperheuristic based on multiple indicator-based density estimators. In *Proceedings of the Genetic and Evolutionary Computation Conference* (New York, NY, USA, 2018), GECCO '18, Association for Computing Machinery, p. 633–640.

[37] Falcón-Cardona, J. G., and Coello, C. A. C. Indicator-based multi-objective evolutionary algorithms: A comprehensive survey. *ACM Comput. Surv. 53*, 2 (mar 2020).

[38] Falcón-Cardona, J. G., Leguizamón, G., Coello Coello, C. A., and Castillo Tapia, M. G. *Multi-objective Ant Colony Optimization: An Updated Review of Approaches and Applications*. Springer Nature Singapore, Singapore, 2022, pp. 1–32.

[39] Falcón-Cardona, J. New findings on indicator-based multi-objective evolutionary algorithms: A brief summary. *CLEI Electronic Journal 25* (04 2022).

[40] Falcón-Cardona, J., and Coello, C. $imoaco_{\mathbb{R}}$: A new indicator-based multi-objective ant colony optimization algorithm for continuous search spaces. vol. 9921, pp. 389–398.

[41] Falcón-Cardona, J. G., Ishibuchi, H., and Coello, C. A. C. Exploiting the trade-off between convergence and diversity indicators. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)* (2020), pp. 141–148.

[42] Falcón-Cardona, J. G., Ishibuchi, H., Coello Coello, C. A., and Emmerich, M. On the effect of the cooperation of indicator-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation 25*, 4 (2021), 681–695.

[43] Falcón-Cardona, J. G., Uribe, L., and Rosas, P. Riesz s-energy as a diversity indicator in evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation* (2024), 1–1.

[44] Fleming, C. L., Griffis, S. E., and Bell, J. E. The effects of triangle inequality on the vehicle routing problem. *European Journal of Operational Research 224*, 1 (2013), 1–7.

[45] Flood, M. M. The traveling-salesman problem. *Operations Research 4* (1956), 61–75.

[46] Fogel, D. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press Series on Computational Intelligence. Wiley, 2006.

[47] Fogel, D. B. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, 2nd ed. Wiley-IEEE Press, 1999.

[48] García-Nájera, A., Bullinaria, J. A., and Gutiérrez-Andrade, M. A. An evolutionary approach for multi-objective vehicle routing problems with backhauls. *Computers & Industrial Engineering 81* (2015), 90–108.

[49] Geoffrion, A. M. Solving bicriterion mathematical programs. *Operations Research 15*, 1 (1967), 39–54.

[50] Glover, F., and Kochenberger, G. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Springer US, 2006.

[51] Glover, F., and Laguna, M. *Tabu search I*, vol. 1. 01 1999.

[52] Guntsch, M., and Middendorf, M. A population based approach for aco. In *Applications of Evolutionary Computing* (Berlin, Heidelberg, 2002), S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. R. Raidl, Eds., Springer Berlin Heidelberg, pp. 72–81.

[53] Hansen, M. P., and Jaszkiewicz, A. Evaluating the quality of approximation to the non-dominated set.

[54] Hansen, M. P., and Jaszkiewicz, A. Evaluating the quality of approximation to the non-dominated set.

[55] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4*, 2 (1968), 100–107.

[56] Herbold, S. Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software 5* (04 2020), 2173.

[57] Hernández Castellanos, C. I., Schütze, O., Sun, J.-Q., and Ober-Blöbaum, S. Non-epsilon dominated evolutionary algorithm for the set of approximate solutions. *Mathematical and Computational Applications 25*, 1 (2020), 3.

[58] Hernández Castellanos, C., Ober-Blöbaum, S., and Peitz, S. Explicit multiobjective model predictive control for nonlinear systems under uncertainty. *International Journal of Robust and Nonlinear Control 30* (09 2020).

[59] Holland, J. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.

[60] Hu, C., Lu, J., Liu, X., and Zhang, G. Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. *Computers & Operations Research 94* (2018), 139–153.

[61] JIANYONG, J., GABRIEL CRAINIC, T., and LØKKETANGEN, A. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European journal of operational research 222*, 3 (2012), 441 – 451.

[62] Jong, K. A. D. An analysis of the behavior of a class of genetic adaptive systems.

[63] Kallehauge, B., Larsen, J., Madsen, O. B., and Solomon, M. M. *Vehicle Routing Problem with Time Windows*. Springer US, Boston, MA, 2005, pp. 67–98.

[64] Knowles, J., and Corne, D. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)* (1999), vol. 1, pp. 98–105 Vol. 1.

[65] Kovacs, A., Golden, B., Hartl, R., and Parragh, S. The generalized consistent vehicle routing problem. *Transportation Science 24* (06 2014).

[66] Kovacs, A., Parragh, S., and Hartl, R. The multi-objective generalized consistent vehicle routing problem. *European Journal of Operational Research 247* (01 2015), 441–458.

[67] Koza, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. A Bradford book. Bradford, 1992.

[68] Kucharska, E., Grobler-Debska, K., and Klimek, R. Collective decision making in dynamic vehicle routing problem. In *MATEC Web of Conferences* (2019), vol. 252, EDP Sciences, p. 03003.

[69] Kuo, R., Zulvia, F. E., and Suryadi, K. Hybrid particle swarm optimization with genetic algorithm for solving capacitated vehicle routing problem with fuzzy demand – a case study on garbage collection system. *Applied Mathematics and Computation 219*, 5 (2012), 2574–2588.

[70] Kuo, Y. Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering 59*, 1 (2010), 157–165.

[71] Kuo, Y., and Wang, C.-C. A variable neighborhood search for the multi-depot vehicle routing problem with loading cost. *Expert Systems with Applications 39*, 8 (2012), 6949–6954.

[72] Land, A. H., and Doig, A. G. *An automatic method for solving discrete programming problems*. Springer, 2010.

[73] Lin, S. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal 44*, 10 (1965), 2245–2269.

[74] Lin, S.-W., Lee, Z.-J., Ying, K.-C., and Lee, C.-Y. Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications 36*, 2, Part 1 (2009), 1505–1512.

[75] Lin, S.-W., Yu, V. F., and Chou, S.-Y. Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research 36*, 5 (2009), 1683–1692. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

[76] Liu, Y., Zhou, H., Wang, Y., Ren, X., and Diao, X. A new ant colony optimization algorithm for multiobjective subset selection problems. *Electronics Letters 55* (11 2019).

[77] Lopez-Ibanez, M., and Stutzle, T. The automatic design of multiobjective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation 16*, 6 (2012), 861–875.

[78] Lu, D., and Gzara, F. The robust vehicle routing problem with time windows: Solution by branch and price and cut. *European Journal of Operational Research 275*, 3 (2019), 925–938.

[79] López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., and Stützle, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives 3* (2016), 43–58.

[80] Maden, W., Eglese, R., and Black, D. Vehicle routing and scheduling with time-varying data: A case study. *Journal of the Operational Research Society 61* (03 2010), 515–522.

[81] Malandraki, C., and Daskin, M. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science 26* (08 1992), 185–200.

[82] Mansour, I., Alaya, I., and Tagina, M. A gradual weight-based ant colony approach for solving the multiobjective multidimensional knapsack problem. *Evolutionary Intelligence 12* (06 2019).

[83] Mansour, I. B., and Alaya, I. Indicator based ant colony optimization for multi-objective knapsack problem. *Procedia Computer Science 60* (2015), 448–457. Knowledge-Based and Intelligent Information and Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings.

[84] Marinakis, Y., Iordanidou, G.-R., and Marinaki, M. Particle swarm optimization for the vehicle routing problem with stochastic demands. *Applied Soft Computing 13*, 4 (2013), 1693–1704.

[85] Mendoza, J., Guéret, C., Hoskins, M., Lobit, H., Pillac, V., Vidal, T., and Vigo, D. Vrp-rep: a vehicle routing community repository.

[86] Mladenović, N., and Hansen, P. Variable neighborhood search. *Computers & Operations Research 24*, 11 (1997), 1097–1100.

[87] Moghaddam, B. F., Ruiz, R., and Sadjadi, S. J. Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm. *Computers & Industrial Engineering 62*, 1 (2012), 306–317.

[88] Moncef, T., Ines, A., and Imen, B. Indicator weighted based multi-objective approach using self-adaptive neighborhood operator. *Procedia Computer Science 192* (2021), 338 – 347.

[89] Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., and Herazo-Padilla, N. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering 79* (2015), 115–129.

[90] Murata, T., and Ishibuchi, H. Moga: multi-objective genetic algorithms. *Proceedings of 1995 IEEE International Conference on Evolutionary Computation 1* (1995), 289–.

[91] Muñoz, C. C., Palacios-Alonso, J. J., Vela, C. R., and Afsar, S. Solving a vehicle routing problem with uncertain demands and adaptive credibility thresholds. In *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (2022), pp. 1–8.

[92] Nocedal, J., and Wright, S. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[93] Pajares, A., Blasco, X., Herrero, J. M., and Reynoso-Meza, G. A multiobjective genetic algorithm for the localization of optimal and nearly optimal solutions which are potentially useful: nevmoga. *Complexity* (2018).

[94] Palhazi Cuervo, D., Goos, P., Sörensen, K., and Arráiz, E. An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research 237*, 2 (2014), 454–464.

[95] Pan, J., Huang, M., Zhang, Q., and Yu, Y. Dynamic vehicle routing problem considering customer satisfaction. In *2020 39th Chinese Control Conference (CCC)* (2020), pp. 5602–5606.

[96] Papini, M. B. *Robust Capacitated Vehicle Routing Problem with Uncertain Demands*. PhD thesis, 2019.

[97] Pareto, V., Montesano, A., Zanni, A., Bruni, L., Chipman, J., and McLure, M. *Manual of Political Economy: A Variorum Translation and Critical Edition*. OUP Oxford, 2014.

[98] Phan, D. H., and Suzuki, J. R2-ibea: R2 indicator based evolutionary algorithm for multiobjective optimization. In *2013 IEEE Congress on Evolutionary Computation* (2013), pp. 1836–1845.

[99] Pisinger, D., and Ropke, S. *Large Neighborhood Search*. 09 2010, pp. 399–419.

[100] Rahbari, A., Nasiri, M. M., Werner, F., Musavi, M., and Jolai, F. The vehicle routing and scheduling problem with cross-docking for perishable products under uncertainty: Two robust bi-objective models. *Applied Mathematical Modelling 70* (2019), 605–625.

[101] Rechenberg, I. *Evolutionsstrategie '94*. Werkstatt Bionik und Evolutionstechnik. Frommann-Holzboog, 1994.

[102] Reimann, M., Doerner, K., and Hartl, R. F. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research 31*, 4 (2004), 563–591.

[103] Russell, S., Norvig, P., and Davis, E. *Artificial Intelligence: A Modern Approach.* Prentice Hall series in artificial intelligence. Prentice Hall, 2010.

[104] Santos, M. J., Curcio, E., Mulati, M. H., Amorim, P., and Miyazawa, F. K. A robust optimization approach for the vehicle routing problem with selective backhauls. *Transportation Research Part E: Logistics and Transportation Review 136* (2020), 101888.

[105] Schütze, O., and Hernández, C. *Archiving Strategies for Evolutionary Multi-objective Optimization Algorithms.* Studies in Computational Intelligence. Springer International Publishing, 2021.

[106] Schütze, O., Vasile, M., and Coello, C. Approximate solutions in space mission design. pp. 805–814.

[107] Schütze, O., Vasile, M., and Coello, C. Computing the set of epsilon-efficient solutions in multiobjective space mission design. *Journal of Aerospace Computing, Information, and Communication 8* (03 2011), 53–70.

[108] Shen, Y., Yu, L., and Li, J. Robust electric vehicle routing problem with time windows under demand uncertainty and weight-related energy consumption. *Complex System Modeling and Simulation 2*, 1 (2022), 18–34.

[109] Silva, M. M., Subramanian, A., and Ochi, L. S. An iterated local search heuristic for the split delivery vehicle routing problem. *Computers & Operations Research 53* (2015), 234–249.

[110] Socha, K., and Dorigo, M. Ant colony optimization for continuous domains. *European Journal of Operational Research 185*, 3 (2008), 1155–1173.

[111] Solano-Charris, E., Prins, C., and Santos, A. C. Local search based metaheuristics for the robust vehicle routing problem with discrete scenarios. *Applied Soft Computing 32* (2015), 518–531.

[112] Stützle, T., and Hoos, H. Max-min ant system.

[113] Sörensen, K. Metaheuristics – the metaphor exposed. *International Transactions of Operations Research In Press* (02 2013).

[114] Taillard, É. D. A heuristic column generation method for the heterogeneous fleet vrp. *RAIRO Oper. Res. 33* (1999), 1–14.

[115] Talarico, L., Springael, J., Sörensen, K., and Talarico, F. A large neighbourhood metaheuristic for the risk-constrained cash-in-transit vehicle routing problem. *Computers & Operations Research 78* (2017), 547–556.

[116] Talbi, and Talbi, E.-G. *Metaheuristics: From Design to Implementation*, vol. 74. 06 2009.

[117] Tarantilis, C., Stavropoulou, F., and Repoussis, P. A template-based tabu search algorithm for the consistent vehicle routing problem. *Expert Systems with Applications 39*, 4 (2012), 4233–4239.

[118] Velázquez Cruz, R. F., and Hernández Castellanos, C. I. A cooperative multi indicator-based ant colony optimization algorithm for the mogenconvrp. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA, 2024), GECCO '24 Companion, Association for Computing Machinery, p. 215–218.

[119] White, D. J. Epsilon efficiency. *J. Optim. Theory Appl. 49*, 2 (May 1986), 319–337.

[120] Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin 1*, 6 (1945), 80–83.

[121] Yang, M., Ni, Y., Yang, X., and Ralescu, D. A. The consistent vehicle routing problem under uncertain environment. *J. Intell. Fuzzy Syst. 41*, 2 (Jan. 2021), 2797–2812.

[122] Yu, B., and Yang, Z. Z. An ant colony optimization model: The period vehicle routing problem with time windows. *Transportation Research Part E: Logistics and Transportation Review 47*, 2 (2011), 166–181.

[123] Zhang, Q., and Li, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation 11*, 6 (2007), 712–731.

[124] Zitzler, E., and Künzli, S. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature - PPSN VIII* (Berlin, Heidelberg, 2004), X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, Eds., Springer Berlin Heidelberg, pp. 832–842.

[125] Zitzler, E., and Thiele, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation 3*, 4 (1999), 257–271.

[126] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., and da Fonseca, V. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation 7*, 2 (2003), 117–132.

[127] Çağri Koç, Bektaş, T., Jabali, O., and Laporte, G. A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. *Computers & Operations Research 64* (2015), 11–27.