Modelo de optimización para el Enrutamiento de un Recolector (Single Picker Routing Problem, por su sigla en inglés, SPRP) en un almacén de comercio electrónico y abordado resolutivamente a través de un algoritmo genético.

Edward Leonardo Rondón Barajas

Trabajo de Grado para Optar al Título de Ingeniero Industrial

Director:

Javier Eduardo Arias Osorio

Magister en Administración

Universidad Industrial de Santander

Facultad de ingenierías físico-mecánicas

Escuela de estudios industriales y empresariales

Ingeniería Industrial

Bucaramanga

2026

Agradecimientos

A Dios por su compañía en cada reto. A mi familia por su apoyo incondicional en todo mi proceso. A mi alma mater, la Universidad Industrial de Santander, por el crecimiento tanto académico como profesional. Al docente Javier Arias Osorio, por su apoyo a lo largo de la investigación.

Tabla de Contenido

Lista de tablas

Lista de figuras

Lista de Apéndices

Los apéndices están disponibles en el repositorio institucional

Apéndice A. Código algoritmo Genético

Apéndice B. Código Modelo exacto Gurobi

Apéndice C. Resultados Calibración

Apéndice D. Resultados Evaluación

Apéndice E. Resultados exacto vs. Algoritmo Genético

Apéndice F. Artículo científico de carácter publicable

Apéndice G. Código análisis estadístico

Apéndice H. Resultados Wilcoxon

Resumen

Título: Modelo de optimización para el Enrutamiento de un Recolector (Single Picker Routing Problem, por su sigla en inglés, SPRP) en un almacén de comercio electrónico y abordado resolutivamente a través de un algoritmo genético[*]

Autor: Edward Leonardo Rondón Barajas [**]

Palabras Clave: Enrutamiento de recolectores, Single Picker Routing Problem, Optimización Combinatoria, Investigación de operaciones.

Descripción: Los almacenes de comercio electrónico son parte fundamental de la cadena de suministro, conectan con proveedores y clientes, y se encargan de diversas operaciones como lo es la recepción, clasificación, almacenamiento, preparación y despacho de pedidos. En cuanto a la preparación de pedidos, esta constituye una de las actividades más críticas y repetitivas en la operación de los almacenes de comercio electrónico, despertando un gran interés en la comunidad académica. En este contexto, la determinación de la secuencia de recolección que debe seguir un recolector para atender una lista de pedidos de manera eficiente se conoce como el problema de enrutamiento de un Recolector (*Single Picker Routing Problem, SPRP).* Diferentes enfoques se han usado para abordar este problema, en este trabajo se propone un enfoque de optimización para el SPRP con almacenamiento disperso, basado en la formulación de un modelo matemático exacto y el desarrollo de un algoritmo genético híbrido como método de solución. La validación del enfoque propuesto se realiza mediante la resolución de instancias de prueba generadas a partir de datos sintéticos, permitiendo comparar la calidad de las soluciones y los tiempos de cómputo obtenidos por el modelo exacto y el algoritmo genético. Los resultados indican que el algoritmo genético híbrido logra generar soluciones de calidad en un tiempo computacional razonable. Este trabajo contribuye a ampliar la formulación compacta de Goeke & Schneider (2021) mediante la inclusión explícita de penalizaciones ergonómicas proxy relacionadas con la altura de picking.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Estudios Industriales y Empresariales. Director: Javier Eduardo Arias Osorio. MS.c en Administración.

Abstract

Title: Optimization Model for the Routing of a Picker (Single Picker Routing Problem, SPRP) in an e-commerce warehouse and addressed decisively through a genetic algorithm*

Author: Edward Leonardo Rondón Barajas[1]

Key Words: Picker Routing, Single Picker Routing Problem, Combinatorial Optimization, Operations Research.

Description: E-commerce warehouses are a fundamental part of the supply chain, connecting suppliers and customers, and handling various operations such as receiving, sorting, storing, picking, and shipping orders. Order picking is one of the most critical and repetitive activities in e-commerce warehouse operations, generating significant interest in the academic community. In this context, determining the picking sequence a picker should follow to efficiently fulfill a list of orders is known as the Single Picker Routing Problem (SPRP). Different approaches have been used to address this problem, including different warehouse layouts, storage distribution politics and different solution methods. This paper proposes an optimization approach for the SPRP with scattered storage, based on the formulation of an exact mathematical model and the development of a hybrid genetic algorithm as a solution method. The proposed approach is validated by solving test instances generated from synthetic data, allowing for a comparison of the solution quality and computation times obtained by the exact model and the genetic algorithm. The results indicate that the hybrid genetic algorithm successfully generates high-quality solutions in a reasonable computational time. This work contributes to expanding the compact formulation of Goeke & Schneider (2021) by explicitly including proxy ergonomic penalties related to picking height.

---

* Degree Work

[1] Faculty of Physicomechanical Engineering. Industrial and Business School. Industrial Engineering. Director: Javier Eduardo Arias Osorio. MS.c in Management.

Introducción

En las últimas décadas el comercio electrónico ha presentado un crecimiento significativo en la economía mundial. Se espera que para el año 2027 el comercio electrónico capture el 41% de las ventas minoristas a nivel mundial (Barthel, y otros, 2023). En cuanto a comercio electrónico se refiere la demanda se constituye en su mayoría de pedidos con pocos artículos y con una ventana de tiempo corta para ser procesados (Boysen, de Koster, & Weidinger, 2019), un escenario que incrementa significativamente la complejidad de las operaciones logísticas.

En este contexto, una de las actividades principales en los almacenes es la preparación de pedidos, que puede llegar a representar el 70% de los costos operacionales del almacén (Kulak, Sahin, & Taner, 2012), concentrando una proporción considerable de tiempo operativo y esfuerzo humano. En ese orden de ideas, la optimización de rutas de recolección se ha convertido en un tema de interés tanto académico como industrial.

El problema de ruteo de recolectores en almacenes ha sido ampliamente estudiado en la literatura, tradicionalmente buscando reducir la distancia recorrida o el tiempo total de desplazamiento. Existen muchos enfoques diferentes y cada uno aporta una aproximación a la realidad diferente, considerando factores como el peso de los elementos a recolectar, la energía consumida por un montacargas, la probabilidad de contagio de enfermedades, políticas de ruteo de montacargas para prevenir accidentes al interior del almacén, entre otras.

En el presente trabajo se aborda el problema de ruteo de un colector en un almacén de comercio electrónico con almacenamiento disperso, considerando explícitamente una penalización

proxy ergonómica asociada a la altura de los niveles de recolección. La **Tabla *1*** ilustra la estructura del trabajo y las partes relacionadas al cumplimiento de cada objetivo planteado.

Tabla 1

Cumplimiento de objetivos

Objetivos específicos	**Numerales relacionados**
Realizar una revisión de bibliografía sobre el problema de enrutamiento de un recolector (SPRPP) en un almacén de comercio electrónico.	Capítulo 2
Formular un modelo de optimización para el Enrutamiento de un Recolector (Single picker routing problem, por su sigla en inglés, SPRP) en un almacén de comercio electrónico.	Capítulo 3
Desarrollar un algoritmo genético como técnica de solución para el problema en estudio.	Capítulo 4
Validar y evaluar los resultados obtenidos a partir de diversas instancias, con datos	Capítulo 5

1. Generalidades

1.1 Planteamiento del problema

En la actualidad, en los almacenes de comercio electrónico, la preparación de pedidos constituye una de las operaciones más críticas y de mayor impacto en los costos y tiempos de respuesta del sistema logístico. En particular, el proceso de recolección de pedidos (*order picking*) concentra una parte significativa del esfuerzo operativo (Goeke & Schneider, 2021), dado que implica desplazamientos continuos del recolector y la manipulación manual de productos almacenados en múltiples ubicaciones y niveles verticales.

Tradicionalmente, se ha dado un enfoque netamente de minimización de la distancia recorrida o del tiempo total de desplazamiento cuando se aborda el problema de ruteo de recolectores en un almacén. Si bien estos enfoques demuestran efectividad en términos de logística, suelen simplificar el problema al no considerar el esfuerzo adicional asociado a la altura de recolección de los productos. Esta simplificación conduce a soluciones que pueden ser eficientes en términos de distancia recorrida, pero no necesariamente refleja de manera adecuada las condiciones físicas bajo las cuales se ejecuta la labor de recolección.

Desde el punto de vista computacional, la incorporación de criterios de ergonomía basados en las alturas de las estanterías añade al grado de dificultad de solución del problema de ruteo, ya que hay un factor adicional que tener en cuenta y se amplía el espacio de soluciones al ser más detallada la distribución de inventario al interior del almacén. En este trabajo, una penalización asociada a los distintos niveles de altura de las estanterías representa una aproximación operacional al esfuerzo físico (penalización proxy) durante la recolección, sin pretender modelar de manera detallada aspectos biomecánicos o fisiológicos del recolector.

En ese orden de ideas, el problema a abordar en este trabajo de investigación busca determinar una ruta de recolección óptima o de alta calidad en tiempo computacional razonable, para un recolector en un almacén de comercio electrónico con estanterías multi nivel, minimizando una función objetivo que contempla tanto los costos de desplazamiento horizontal y vertical con una penalización proxy ergonómica dependiente de la altura de los productos recolectados. Para ello, se propone la formulación de un modelo matemático exacto, y al desarrollo de un algoritmo genético híbrido como método de solución.

1.1.1 Pregunta de investigación

A partir del problema descrito en la sección anterior, se plantea la siguiente pregunta de investigación:

¿Cómo diseñar un modelo de ruteo de un recolector que integre los costos de desplazamiento y penalizaciones ergonómicas asociadas a la altura de recolección, manteniendo tiempos computacionales razonables para su aplicación en almacenes de comercio electrónico?

1.2 Justificación

Uno de los desafíos principales que enfrentan los sistemas logísticos modernos es la optimización del proceso de recolección de pedidos en almacenes de comercio electrónico. Dado que es una actividad que tiene un impacto significativo en los costos totales de operación y los tiempos de procesamiento de pedidos, las mejoras que puedan ser implementadas en este aspecto de la cadena logística tienen un gran impacto en la productividad, reducción de costos y tiempos de entrega de pedidos. Siendo que la actividad de *order picking* representa una porción significativa del tiempo total de operación, el modelamiento de diferentes factores que acerquen el problema más a la realidad brinda diferentes enfoques de solución útiles en diferentes contextos.

En este contexto, la mayoría de los enfoques tradicionales para el ruteo de recolectores se han centrado en la minimización de la distancia recorrida. Este trabajo se justifica en la necesidad de explorar formulaciones que integren una aproximación operacional al esfuerzo físico del recolector, agregando una penalización simple asociada a la altura de recolección, sin pretender modelar de manera detallada aspectos biomecánicos o fisiológicos.

Desde el punto de vista metodológico, el presente trabajo aporta una formulación matemática exacta que integra costos de desplazamiento horizontal y una penalización proxy

dependiente de la altura de recolección, lo que permite la generación de una ruta de recolección que integre de manera sencilla criterios ergonómicos simplificados en el problema de ruteo de recolectores.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar un modelo de optimización para el Enrutamiento de un Recolector (Single Picker Routing Problem, por su sigla en inglés, SPRP) en un almacén de comercio electrónico y abordado resolutivamente a través de un algoritmo genético híbrido.

1.3.2 Objetivos Específicos

Realizar una revisión de bibliografía sobre el problema de enrutamiento de un recolector (SPRP) en un almacén de comercio electrónico.

Formular un modelo de optimización para el Enrutamiento de un Recolector (Single picker routing problem, por su sigla en inglés, SPRP) en un almacén de comercio electrónico.

Desarrollar un algoritmo genético como técnica de solución para el problema en estudio.

Validar y evaluar los resultados obtenidos a partir de diversas instancias, con datos sintéticos.

2. Revisión De Literatura

2.1 Análisis Bibliométrico

2.1.1 Scopus

Con el objetivo de encontrar información pertinente respecto al enrutamiento de un recolector, se realizó una consulta en Scopus, una de las principales bases de datos de artículos científicos, buscando palabras clave por medio de la ecuación de búsqueda: "picker routing" AND (“order pick*" OR "warehous*”) AND (“optimization" OR "heuristics”), obteniendo 40 documentos que coinciden con la búsqueda al día 17/07/2025.

2.1.1.1 Indicadores Básicos. Por medio de los datos obtenidos con la búsqueda realizada, se desarrollaron los siguientes gráficos estadísticos.

2.1.1.1.1 Áreas De Investigación. Observando la **Figura *1***, se encuentra como principal área las ciencias de computación con un 23.5% de los resultados, seguida por ciencia de decisión con 22.5%, ingeniería con un 21.6%, matemáticas con 17.6%, y un 14.8% entre negocios, administración y contabilidad, ingeniería química, economía, econometría y finanzas, ciencia de materiales, y ciencias sociales.

2.1.1.1.2 Autores. En la **Figura *2*** se encuentran representados los autores con más publicaciones relevantes para el tema de investigación, liderando la lista Süral, H. con 4 publicaciones, Glock, C.H. y Celik, M con 3 publicaciones cada uno, seguidos por Beasley, J.E, Boysen, N, Cano, J.A, Cortés, P, Di Puglia Pugliese, L, Düzgit, Z, Guerriero, F, y Masae, M, con 2 publicaciones cada uno.

Figura 1

Áreas de investigación en Scopus.

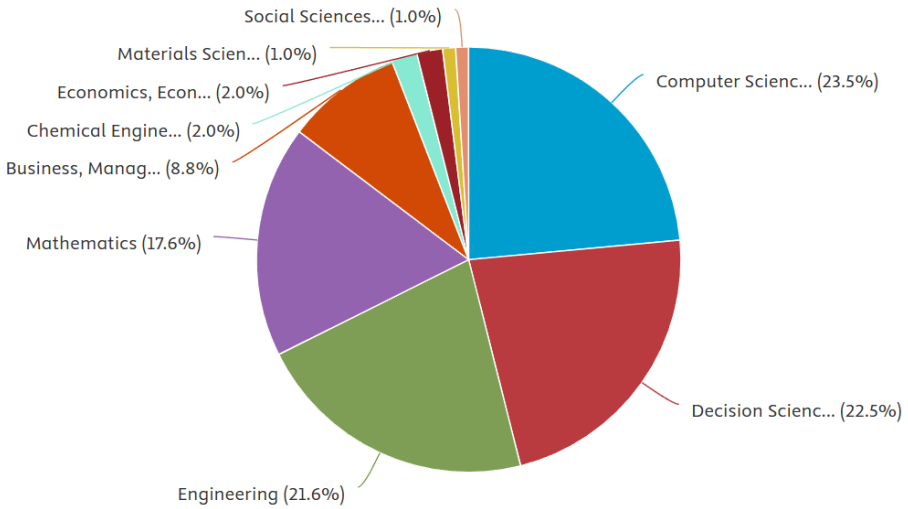
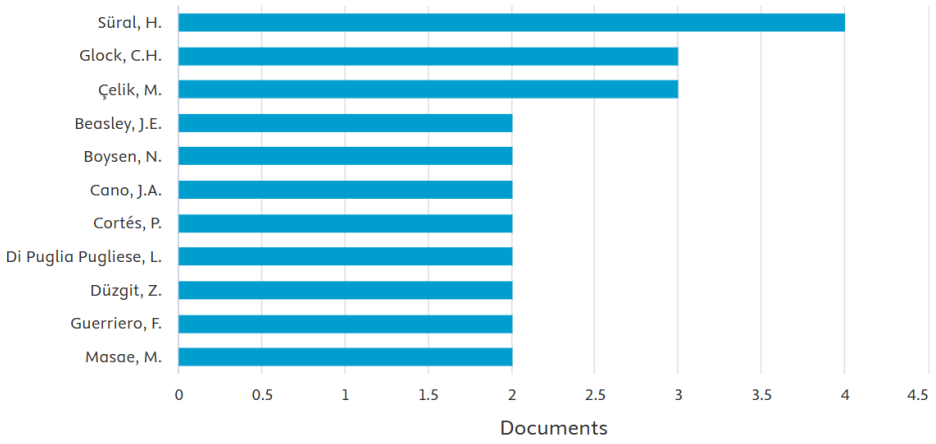


Figura 2

Autores con más publicaciones en Scopus

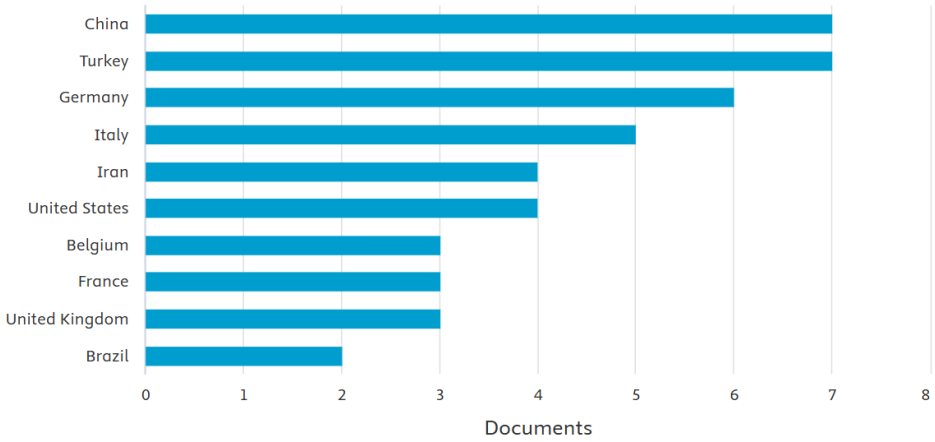


2.1.1.1.3 Países. La **Figura 3** muestra los países líderes en publicaciones científicas respecto al tema de investigación, liderando Turquía y China con7 documentos cada uno, seguido por Alemania con 6 documentos, Italia presenta 5 documentos, Irán y Estados Unidos contribuyen con 4 documentos cada uno, Reino Unido, Francia y Bélgica tienen 3 publicaciones cada uno y

países como, Brasil, Canadá, Colombia, Países Bajos, España, Taiwán, y Tailandia con 2 publicaciones cada uno.

Figura 3

Países destacados en Scopus

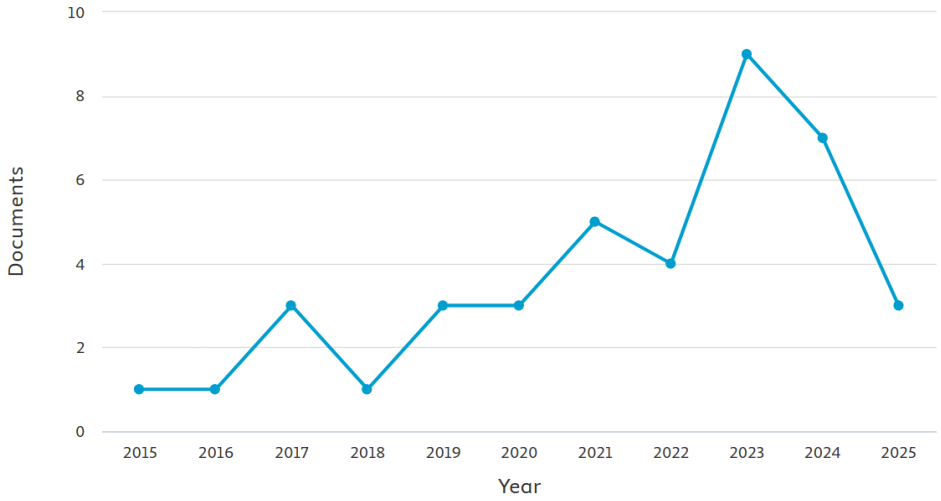


2.1.1.1.4 Documentos Por Año. En la **Figura 4** se puede observar que el año en que más artículos fueron publicados respecto al tema de investigación fue en el 2023, con 9 publicaciones. En el 2024 se encuentran 7 publicaciones referentes al tema, y en lo que va de 2025 se han publicado 3 documentos. Adicionalmente se puede evidenciar una tendencia creciente en cantidad de artículos publicados desde el 2015.

2.1.1.2 Indicadores De Relación. Para desarrollar un análisis de indicadores de relación se implementó la herramienta VOSviewer, que permite la construcción y visualización de redes bibliométricas a partir de citas, acoplamiento bibliográfico, cocitación o relaciones de coautoría.

Figura 4

Publicaciones a lo largo del tiempo



2.1.1.2.1 Palabras Clave. En la **Figura 5** se encuentra el mapa de palabras clave, para el cuál se realizó un tesauro con el objetivo de eliminar duplicidades, y se incluyeron las palabras clave que se mencionaron un mínimo de 4 veces en la totalidad de los documentos encontrados, agrupadas en 3 clústeres que representan algoritmos de agrupamiento por temáticas relevantes. Entre las palabras clave más representativas se encuentra picker routing, warehouse, order picking, heuristics y routing problems.

2.1.1.2.1 Citación De Autores. Para realizar el análisis de autores se realizó un tesauro con el fin de eliminar duplicidades en el nombre de los autores. Se tomó en cuenta autores con al menos un artículo publicado y que haya sido citado un mínimo de 20 veces. Se observan 3 clústeres, en los cuales los autores más influyentes son Beasley, Glock, y Süral. En la **Tabla *2*** se encuentran representados los 10 autores más citados y la cantidad de documentos de estos.

Figura 5

Mapa de palabras clave representativas en Scopus

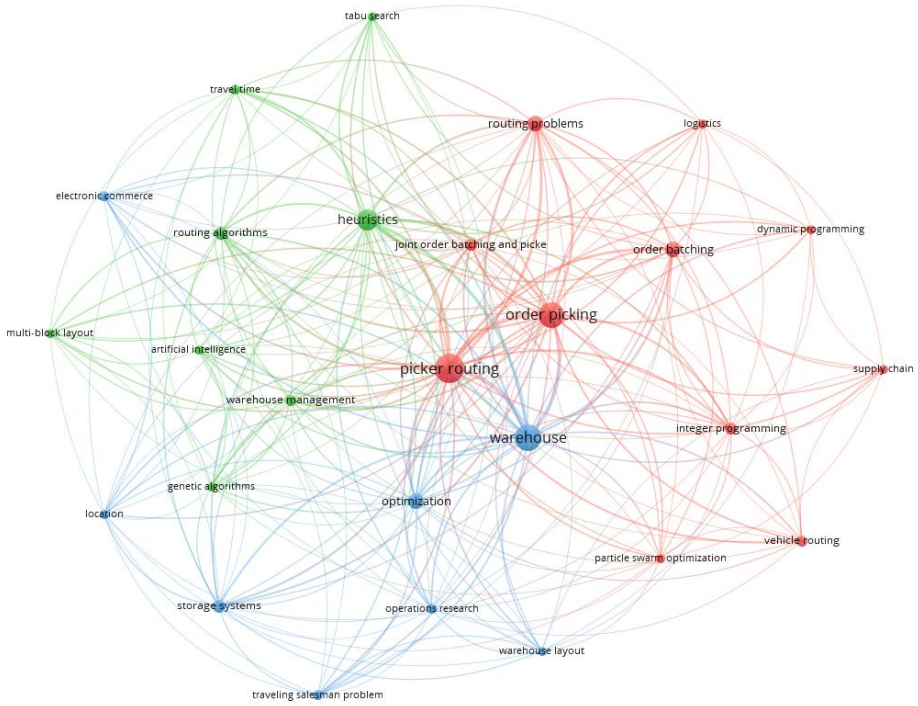


Figura 6

Autores con más citaciones en Scopus

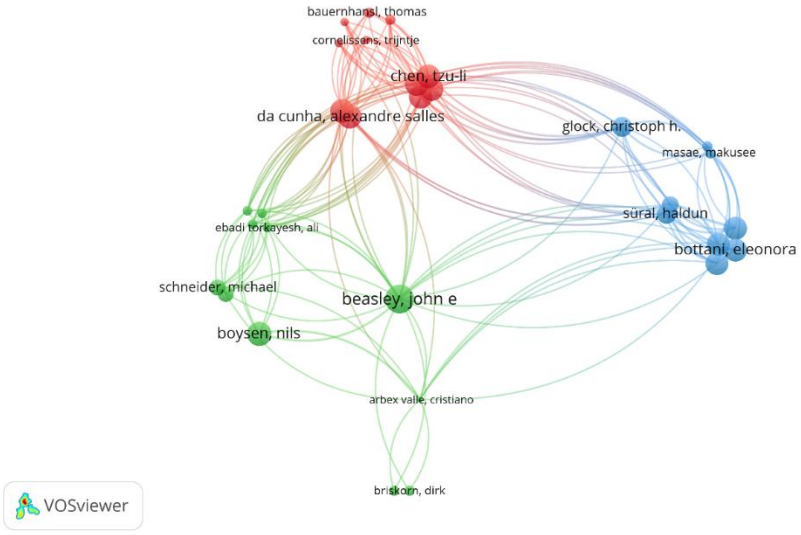


Tabla 2

Citaciones por autor en Scopus

Autores	Documentos	Citaciones	Total Link Strength
beasley, john e	2	133	29
glock, christoph h.	3	108	23
da cunha, alexandre salles	1	109	18
valle, cristiano arbex	1	109	18
masae, makusee	2	51	16
vichitkunakorn, panupong	2	51	16
bottani, eleonora	1	98	15
chen, tzu-li	1	108	15
chen, yin-yann	1	108	15
cheng, chen-yang	1	108	15

2.1.2 Web Of Science

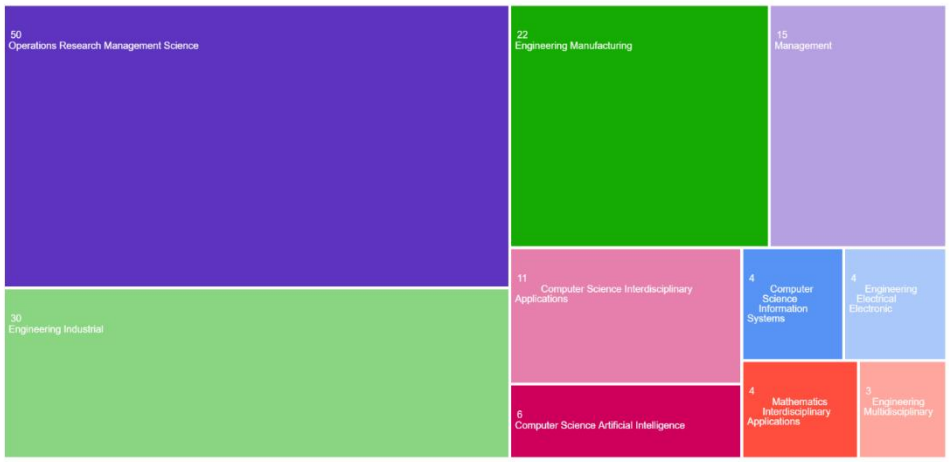
Para realizar la búsqueda en la base de datos de Web of science se implementó la siguiente ecuación de búsqueda: ("Picker Routing") OR ("Order picking" AND "rout*") AND ("heuristic" OR "metaheuristic" OR "optimization") con la cual se alcanzaron 75 resultados en inglés al día 17/07/2025.

2.1.2.1 Indicadores Básicos. Al analizar los resultados de la búsqueda se encontraron los siguientes resultados, representados en diferentes gráficas.

2.1.2.1.1 Áreas de investigación. La **Figura 7** permite ver las áreas de investigación más importantes según la búsqueda realizada en Web of Science, encontrando que el 66.6% de los artículos están relacionados con Operations Research Management Science, 40% Engineering Industrial, 29.3% Engineering manufacturing, y 20% acerca de Management,

Figura 7

Áreas de investigación en Web of Science



2.1.2.1.2 Documentos por año. En la **Figura 8** se encuentran manifestados la cantidad de documentos publicados respecto al tema de investigación en cada año. Se puede observar una tendencia creciente entre 2013 y 2022, año en el cual se publicaron mayor número de documentos, 11 en total. El segundo año con mayor número de publicaciones es el 2020 y 2024 con 10 publicaciones. En 2023 se realizaron 8 publicaciones y en lo que va de 2025 se han publicado 6 artículos.

2.1.2.1.3 Autores. En la **Figura 9** se encuentran plasmados los 10 autores con mayor número de publicaciones relacionadas con el tema de investigación. Se aprecia que en primer lugar se encuentra Glock CH, con un total de 6 publicaciones, seguido por Boysen N con 5 publicaciones, Braekers K, Masae M, Ramaekers K, Schneider M, y Süral H con 4 publicaciones cada uno. Cabe resaltar que los autores Glock CH, Süral H y Boysen N también se encuentran entre los 10 autores con mayor cantidad de publicaciones en la revisión realizada en Scopus.

Figura 8

Documentos por año

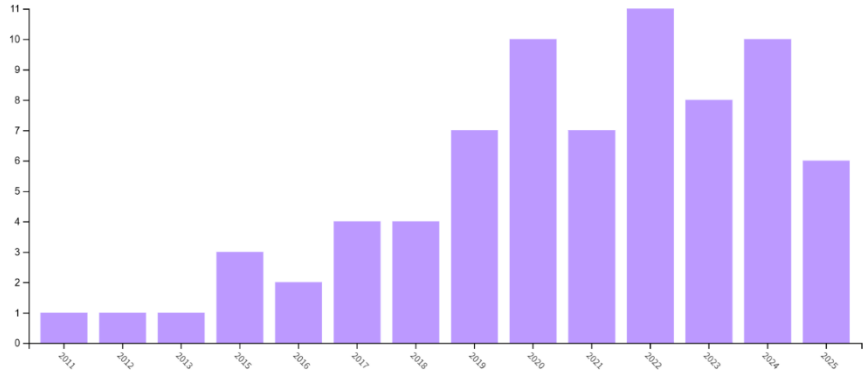
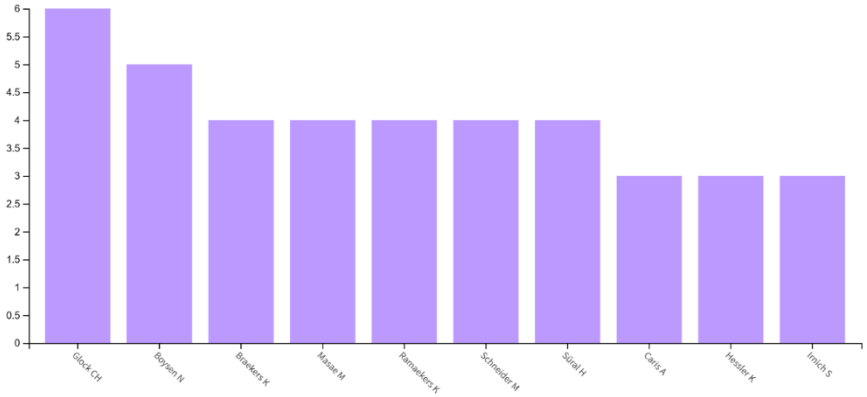


Figura 9

Autores Web of Science

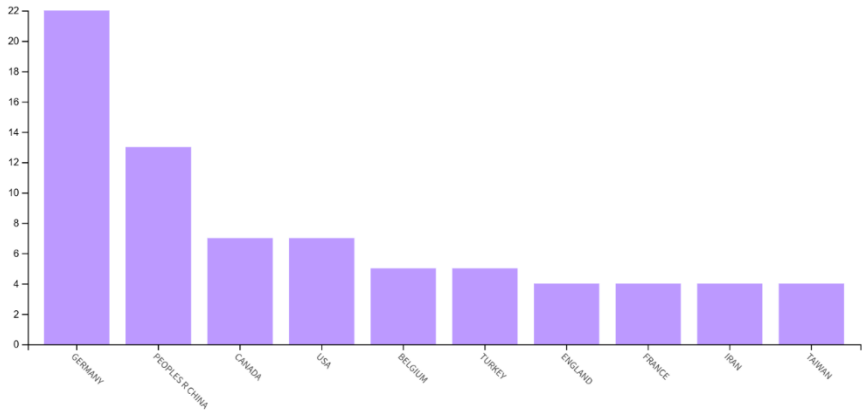


2.1.2.1.4 Países. En la **Figura 10** se resumen los países con más publicaciones respecto al tema de investigación. Se muestra una predominancia de parte de Alemania, con 22 publicaciones, seguido por china con 13 publicaciones, Canadá y USA con 6 publicaciones cada uno. Bélgica y Turquía contribuyen cada uno con 5 publicaciones, mientras que Inglaterra, Francia, Irán y Taiwán aportan 4 publicaciones cada uno. Es de resaltar que países como Turquía, China, Alemania, USA,

Bélgica, Taiwán, y Canadá estuvieron en el top 10 tanto en la búsqueda realizada en Scopus como la realizada en Web of Science.

Figura 10

Publicaciones por países en Web of Science

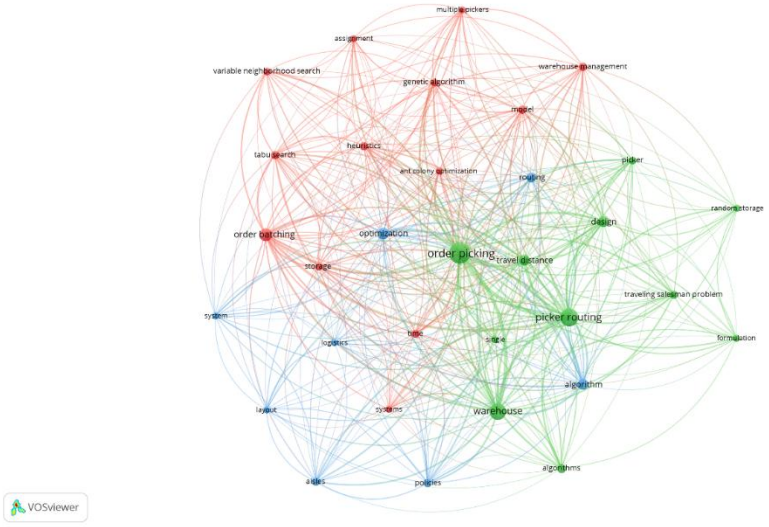


2.1.2.1 Indicadores De Relación. Los indicadores de relación fueron construidos nuevamente con la ayuda del software Vosviewer.

2.1.2.2.2 Palabras Clave. Se realizó un tesauro para eliminar la duplicidad de las palabras clave y se incluyeron en la gráfica aquellas palabras clave representativas que aparecieron un mínimo de 6 veces en la investigación. Como resultado se encuentra un mapa de correlación con 32 palabras, dividas en 3 clústers. Entre las más representativas se encuentran order picking, picker routing, y warehouse. Los subcampos con mayor parte de coocurrencias son, de mayor a menor, picker routing, seguido por order batrching.

Figura 11

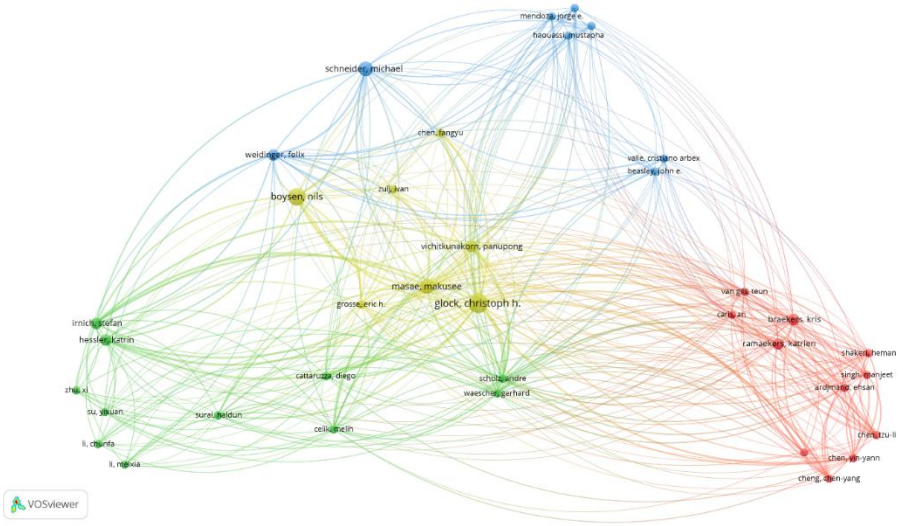
Mapa palabras clave Web of Science.



2.1.2.2.3 Citación De Autores. Con el objetivo de analizar los autores con mayor influencia encontrados en Web of Science, se realizó un tesauro para eliminar duplicidades y se incluyeron aquellos autores con un mínimo de 2 documentos publicados y 2 citaciones. En la **Figura 12** se refleja el núcleo con mayor fuerza de enlace total, articulado por Glock, Masae y Vichitkunakorn. En el clúster rojo se observa citación interna intensa. El mapa manifiesta comunidades definidas pero interconectadas por citaciones.

Figura 12

Mapa citación de autores Web Of Science.



2.1.2.2.4 Acoplamiento bibliográfico. Se construyó un mapa de acoplamiento bibliográfico de autores con al menos 2 documentos y 5 citas. En el mapa de la **Figura 13** se pueden encontrar 3 clústeres, y al contrastarlo con la visualización de la **Figura 14**, se puede concluir que los focos más recientes se concentran en la franja superior del grafo y en el clúster verde.

Figura 13

Acoplamiento bibliográfico autores

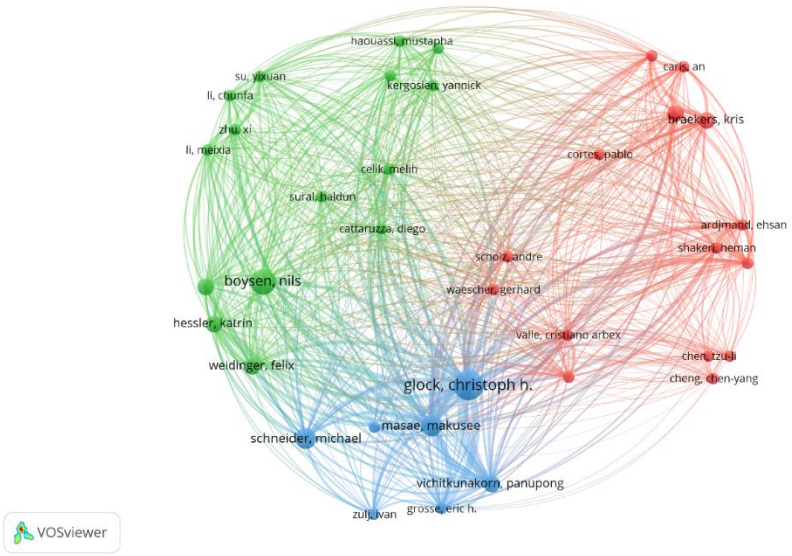
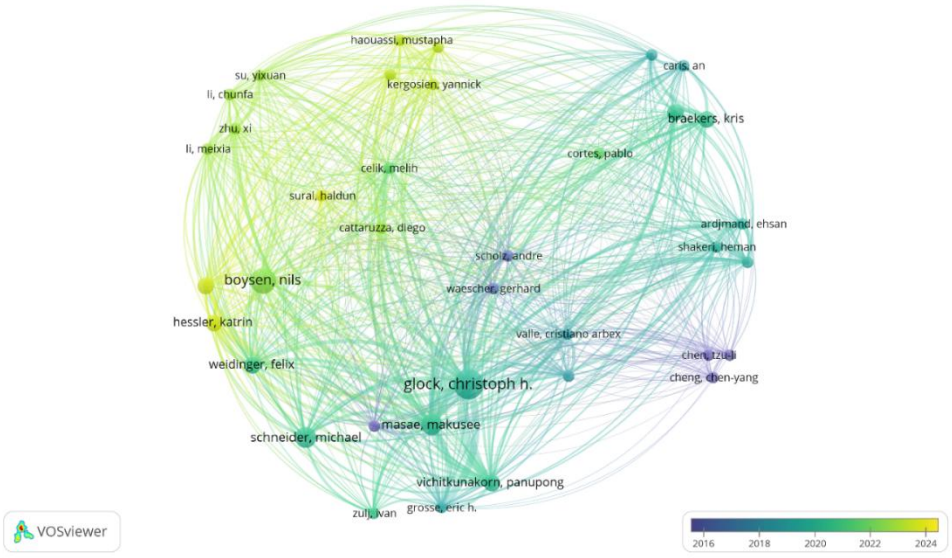


Figura 14

Acoplamiento bibliográfico en el tiempo



2.2. Análisis Preliminar De La Literatura

En la actualidad, donde las industrias y el comercio se mueven con rapidez, los tiempos de entrega de las órdenes a los clientes son bastante ajustados y pueden representar una ventaja

competitiva. Los almacenes son componentes esenciales de la cadena de suministro, ya que conectan a los proveedores, plantas de producción y sistemas de distribución: su desempeño y manejo afecta la eficiencia de la cadena de suministro, los costos de la logística y la percepción del servicio al cliente. Se estima que el almacenamiento puede representar hasta un 20% del costo de la cadena de suministro.

Dentro de los almacenes se llevan a cabo diferentes operaciones como la recepción de mercancías, almacenamiento, reabastecimiento, control de calidad, gestión de devoluciones, y preparación de pedidos (order picking). La preparación de pedidos es una actividad central en los almacenes que requiere una gran cantidad de mano de obra y recursos. La preparación de pedidos incluye loteo de órdenes, clasificación de productos y ruteo del recolector.

El ruteo del recolector (picker routing) es una operación clave para reducir tiempos y aumentar la eficiencia operativa, por tanto, a lo largo de los años ha despertado el interés en la comunidad académica y se han realizado investigaciones buscando la mejor manera de abordar este proceso. En la literatura se pueden identificar 3 enfoques metodológicos para el ruteo del colector los cuales son:

Algoritmos exactos que siempre buscan una solución óptima, en este caso, la ruta de recolección más corta. Suelen ser buenos para problemas no muy grandes ya que su complejidad computacional hace que requieran mucho tiempo para encontrar la solución óptima cuando se usan instancias demasiado grandes. Entre los algoritmos exactos más notables se encuentra el propuesto por Ratliff y Rosenthal (1983), el cual aborda el caso de un almacén con un solo bloque, donde el tour de recolección comienza y finaliza en el mismo lugar. Este algoritmo fue posteriormente extendido a un almacén con 2 bloques por Roodbergen y De Koster (2001).

Heurísticas que buscan soluciones factibles, no necesariamente optimas, pero útiles en situaciones donde se requiere un tiempo computacional muy elevado para encontrar una solución exacta. Hall (1993) propone y compara tres heurísticas para este problema, las cuales son trasversal (también conocida como S-Shape), punto medio, y brecha más grande (largest gap). Petersen (1997) agrega la heurística de retorno y compuesta, la cual combina retorno y S-Shape.

Metaheurísticas que son métodos iterativos para encontrar buenas soluciones a problemas de optimización con cierto grado de complejidad, brindando soluciones para problemas grandes en tiempos computacionales razonables. Chen et al. (2013) proponen un algoritmo de ruteo basado en colonia de hormigas que considera la congestión al tener dos recolectores. De Santis et al. (2018) adaptan el algoritmo de Floyd-Warshall con un algoritmo de optimización basado en colonia de hormigas para minimizar la distancia recorrida por recolectores en almacenes manuales.

Los problemas abordados con mayor frecuencia son el ruteo de un colector (Single picker routing), y el conjunto el loteo de órdenes y ruteo del recolector (joint order batching and picker routing). También se abordan problemas como loteo de órdenes, ruteo del recolector y ubicación de almacenamiento; ruteo de múltiples recolectores, y en ocasiones restricciones más realistas.

EL problema de ruteo de un recolector puede ser interpretado como un caso especial del clásico problema del agente viajero (TSP). Scholz, Henn, Stuhlmann y Wäscher (2016) presentan la primera formulación de programación matemática que tiene en cuenta las propiedades específicas del problema de ruteo de un recolector (single-picker routing problem) en un almacén de bloque único.

El problema de ruteo del colector ha sido abordado en diferentes configuraciones de almacenamiento, como la configuración de un único bloque (single block) (Scholz, Henn,

Stuhlmann, & Wäscher, 2016) (Weidinger, 2018) (Löffler, Schneider, & Žulj, Cost-neutral reduction of infection risk in picker-to-parts warehousing systems, 2022), almacenes multi-bloque (De Santis, Montanari, Vignali, & Bottani, 2018) (Chen, Xu, & Wei, 2019) (Çelik & Süral, 2019) (Masae, Glock, & Vichitkunakorn, Optimal order picker routing in a conventional warehouse with two blocks and arbitrary starting and ending points of a tour, 2020) (Cano, Cortés, Muñuzuri, & Correa-Espinal, 2023) (Su, Li, Zhu, & Li, 2022) (Su, et al., 2023) (Haouassi, Kergosien, Mendoza, & Rousseau, 2025). Para almacenes con diseño tipo Chevron, Masae et al. (2020) desarrollan un algoritmo utilizando programación dinámica y teoría de grafos, y también proponen heurísticas simples. Mas adelante, Masae et al. (2021) proponen un algoritmo exacto para el ruteo de colectores en almacenes con diseño tipo hoja (leaf warehouse) y desarrolla heurísticas simples para guiar al colector por el almacén.

Weidinger (2018) propone modelos y heurísticas teniendo en cuenta un almacenamiento en estanterías mixtas (mixed-shelves o Scattered Storage), una estrategia de almacenamiento en la que hay múltiples posiciones de almacenamiento para cada ítem.

En cuanto a la cantidad de colectores, Chen et al. (2013) abordan el problema con dos colectores, teniendo en cuenta la congestión que puede surgir cuando dos colectores atraviesan el mismo pasillo al tiempo. Así mismo, Löffler et al. (2023) abordan el problema con más de un colector, y teniendo en cuenta la pandemia por el Covid-19, proponen un modelo para mitigar el riesgo de infección minimizando el tiempo de superposición de los colectores en los pasillos, sin aumentar la distancia recorrida.

En grandes centros de distribución se pueden implementar montacargas, carros de recolección y vehículos guiados automáticamente, Atashi Khoei et al. (2023) presentan y estudian el ruteo de montacargas recolectores con el objetivo de reducir el consumo de energía, incluyendo

cálculos del consumo de energía tanto en movimientos horizontales como movimientos verticales; Bock et al. (2025) integran regulaciones de tráfico interno en un algoritmo exacto de ruteo de recolectores, y evalúan la eficiencia operativa y el impacto de las diferentes regulaciones.

A partir de la revisión de literatura realizada, se identifican vacíos en investigación relevantes tanto a nivel conceptual como metodológico. Estudios tanto clásicos como recientes, como los de Roodbergen y De Koster, han enfocado las soluciones al problema de ruteo de un recolector desde una perspectiva predominantemente logística, concentrándose en la minimización de la distancia recorrida o del tiempo total de desplazamiento mediante estrategias de ruteo y políticas de recorrido eficientes. Más recientemente, trabajos como los de Goeke y Schneider (2021) han aportado significativamente al desarrollo de formulaciones exactas y métodos de solución avanzados para el SPRP, reafirmando el problema como un caso relevante dentro de la optimización combinatoria. No obstante, la integración directa de criterios ergonómicos dentro de la formulación del SPRP continúa siendo limitada,

En ese orden de ideas, el presente trabajo se posiciona como una contribución que busca cerrar esta brecha al proponer una formulación del SPRP que integra explícitamente una penalización ergonómica proxy dependiente de la altura de recolección, agregada directamente en la función objetivo de un modelo matemático exacto. De igual manera, es propuesto un algoritmo genético híbrido que busca soluciones de alta calidad en tiempos computacionales razonables, contribuyendo así a cerrar la brecha existente entre modelos teóricos y aplicaciones operacionales en sistemas order picking.

2.3. Marco Teórico

A continuación, se relacionan las técnicas, métodos y herramientas comúnmente utilizadas para dar solución al problema de enrutamiento de un recolector (SPRP):

2.3.1 Problema de Optimización

Un problema de optimización consiste en encontrar la solución óptima a un problema entre un grupo de soluciones factibles que cumplan todas las restricciones del problema. Normalmente está compuesto de una función objetivo, variables de decisión y restricciones. Busca identificar los valores de las variables de decisión que ayudaran a minimizar o maximizar el resultado de la función objetivo.

2.3.2 Modelo De Optimización

Un modelo de optimización es la representación matemática de un problema de optimización, comprende las variables de decisión, la función objetivo y una serie de restricciones propias del problema de optimización.

2.3.2.1 Programación Lineal. Un problema es de programación lineal cuando todas sus restricciones y su función objetivo son lineales. Son comúnmente resueltos mediante el método simplex.

2.3.2.2 Programación Lineal Entera. Es un caso especial de un problema de programación lineal, en el cual las variables de decisión no pueden tomar valores enteros.

2.3.2.3 Programación Lineal Entera Mixta. Permite que algunas variables sean continuas y otras sean enteras.

2.3.3 Logística

Según la Real Academia Española (s.f.), “logística es el conjunto de medios y métodos necesarios para llevar a cabo la organización de una empresa, o de un servicio, especialmente de distribución” (https://dle.rae.es/logística). La logística es la parte de la cadena de suministro que planifica, gestiona y controla el flujo y el almacenamiento de los bienes, servicios e información generada, con fin último el satisfacer la demanda de los consumidores (Escudero Serrano, 2013).

2.3.4 Almacén

Un almacén es “el edificio o lugar donde se guardan o depositan mercancías o materiales” (Escudero Serrano, 2013). En los almacenes se llevan a cabo funciones de almacenamiento, como lo son recepción, almacenaje, recolección de órdenes y despacho (Gu, Goetschalckx, & McGinnis, 2007).

Los almacenes pueden ser clasificados en dos categorías según la manera en que los ítems y los recolectores interactúan. Un almacén picker-to-parts es aquel en el que los colectores (humanos o robots) se mueven hacía la ubicación de los ítems requeridos, mientras que en un almacén parts-to-picker son los ítems los cuales son transportados, por un sistema de transporte de materiales, hasta los colectores que permanecen estáticos. (Boysen & de Koster, 2025).

2.3.4.1 Almacén de comercio electrónico. Un almacén de comercio electrónico es una instalación logística que se encarga de la recepción, almacenamiento, preparación y despacho de pedidos individuales generados a partir de canales digitales. Se caracterizan por manejar ordenes individuales compuestas de pocos elementos, al tiempo que cuentan con una gran variedad de productos, tiempos de entrega bastante ajustados, en ocasiones de menos de 24 horas, y cargas de trabajo diversas debido a la volátil demanda (Boysen, de Koster, & Weidinger, 2019).

2.3.5 Gestión De Almacenes

La gestión de almacenes es definida por ten Hompel y Schmidt (2007) como “control y optimización de sistemas complejos de almacenamiento y distribución”, es la operación eficiente de un sistema de almacenamiento y distribución. Entre las operaciones que se desarrollan en un almacén se encuentran la recepción de mercancías, almacenamiento, reabastecimiento, control de calidad, gestión de devoluciones, planificación de la demanda y pronóstico, y recolección de órdenes.

2.3.6 Problema Del Vendedor Viajero

El problema del vendedor viajero es un problema de optimización sencillo de describir, pero no tan sencillo de resolver. Formulado por primera vez en 1930, consiste en encontrar la ruta más corta para visitar una serie de ciudades, de las que se conoce la distancia entre ellas, y regresar al punto de partida. El problema del vendedor viajero no puede ser resuelto en tiempo polinomial, ya que es NP-Hard (Dahiya & Sangwan, 2018).

Las aplicaciones del problema del vendedor viajero son variadas, incluyendo el análisis de estructuras de cristal, manufactura de microchips, manejo de material en almacenes, e incluso se encuentra como un sub-problema en problemas combinatorios más complejos, como el ruteo de vehículos (Goyal, 2010).

Para resolver el problema del vendedor viajero se encuentran dos enfoques, el determinista que busca una solución exacta, y el no determinista que busca una solución cercana a la óptima. El enfoque no determinista suele ser más útil en contextos donde se prioriza el tiempo que toma resolver el problema sobre la calidad del resultado.

2.3.7 Single Picker Routing Problem

El objetivo del picker routing problem es definir la ruta que un colector debe ejecutar dentro de un almacén para recolectar todos los elementos contenidos en una lista de recolección, de sus respectivas posiciones en el almacén, buscando la ruta de costo mínimo iniciando y finalizando en un depósito, entendiendo el costo comúnmente como la distancia recorrida por el recolector. Suele ser modelado como una variante del problema del agente viajero (Ratliff & Rosenthal, 1983), o como un problema de ruteo de vehículos con restricción de capacidad.

2.3.7 Técnicas Exactas

Las técnicas exactas hacen uso de procedimientos matemáticos que les permiten explorar sistemáticamente el espacio de soluciones factibles, y así llegar a la solución óptima a un problema de optimización. Sin embargo, se ven limitadas por la capacidad computacional disponible, ya que a medida que el problema se hace más grande se requieren más recursos para poder barrer completamente el espacio de soluciones y encontrar la solución óptima, por tanto, las técnicas exactas suelen ser utilizadas en problemas pequeños donde la complejidad computacional no limita demasiado.

2.3.8 Heurísticas

El término heurística proviene del griego heurískein, que significa “encontrar” o “Descubrir”, se deriva de la exclamación atribuida a Arquímedes “eureka”. Puede denotar el arte o disciplina del descubrimiento o técnicas prácticas de resolución de problemas por medio de métodos no rigurosos (Real Academia Española, s.f.). Resolver tareas inteligentemente utilizando la información disponible.

En la investigación de operaciones, se define heurístico como “un procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un coste computacional razonable” (Melian, Moreno-Pérez, & Moreno-Vega, 2003). Sin embargo, una solución encontrada por un método heurístico no garantiza ser óptima. Las heurísticas suelen ser empleadas para resolver un problema de optimización cuando, debido a la naturaleza del problema no existe método exacto para resolverlo, cuando el método exacto es conocido pero su costo computacional es muy elevado, o cuando le flexibilidad del método heurístico permite contemplar situaciones difíciles de reflejar en un modelo matemático (de Antonio Suárez, 2011)

2.3.8.1 S-Shape. Es una heurística de enrutamiento que consiste en atravesar completamente los pasillos que contienen producto que se debe recolectar, iniciando desde los pasillos transversales, se ignoran los pasillos que no contienen artículos necesarios y posteriormente el recolector regresa al depósito. (Chen, Xu, & Wei, 2019)

2.3.8.2 La Brecha más Grande. Esta heurística consiste en evitar recorrer el mayor espacio vacío, definiendo este como la separación entre dos productos a recoger adyacentes, entre el pasillo transversal superior y el primer producto a recoger, o del último producto a recoger presente en el pasillo y el pasillo transversal inferior. De modo que solo se recorre completamente el primer y el último pasillo que se visitan, a los demás se ingresa por el pasillo transversal superior para recolectar los productos de la mitad delantera, y se ingresa por el pasillo transversal inferior para recolectar los productos de la mitad trasera. (Chen, Xu, & Wei, 2019)

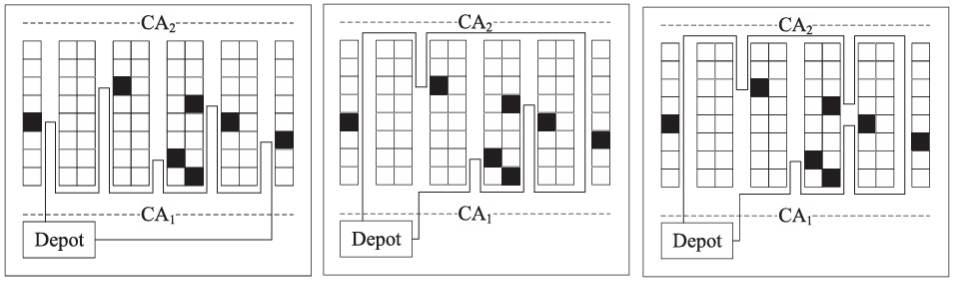
2.3.8.3 Punto Medio. Consiste en dividir por la mitad los pasillos que contienen producto para la recolección, y posteriormente recolectar los productos de la mitad superior ingresando al pasillo por el pasillo transversal superior, y los productos de la mitad inferior por medio del pasillo transversal inferior. Usando esta heurística se puede presentar pasillos que se accedan dos veces,

y solo el primer y último pasillo son atravesados completamente para regresar al depósito. (Chen, Xu, & Wei, 2019)

2.3.8.4 Retorno. Con esta política de enrutamiento, el colector ingresa y sale del pasillo por la misma entrada, visitando solo pasillos con productos a recolectar, y finalmente regresando al depósito. (Chen, Xu, & Wei, 2019)

Figura 15

Heurísticas de enrutamiento en almacenes



(a) *Return* for a single block (b) *Largest Gap* for a single block (c) *Mid-point* for a single block

Nota. Adaptado de *Heuristic routing methods in multiple-block warehouses with ultra-narrow aisles and access restriction,* por F. Chen, G. Xu, y Y. Wei, 2019, *International Journal of Production Research* (núm. 57), p. 236

2.3.9 Metaheurística

Etimológicamente, “metaheurística” está compuesta por el griego “meta” y “heurística”, de modo que el prefijo “meta” añade a la definición de heurística “más allá de”, y “un nivel superior”. Las metaheurísticas son “estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento” (Melian, Moreno-Pérez, & Moreno-Vega, 2003), son definidas como un “proceso de generación iterativo que guía una heurística subordinada combinando inteligentemente diferentes conceptos para explorar y explotar los espacios de

búsqueda utilizando estrategias de aprendizaje para estructurar la información con el fin de encontrar soluciones eficientemente cercanas a las óptimas" (Osman & Kelly, 1996)

Las metaheurísticas son métodos aproximados que abordan problemas combinatorios complejos, permitiendo traer a la mesa herramientas de solución basadas en heurísticas clásicas, inteligencia artificial, evolución biológica, sistemas neuronales, y mecánica estadística. A pesar de que con el uso de metaheurísticas no es posible demostrar la optimalidad de las soluciones obtenidas, las metaheurísticas han sido ampliamente exitosas encontrando soluciones casi óptimas, haciéndolo incluso mejor que sus heurísticas subordinadas (Osman & Kelly, 1996).

2.3.9.1 Recocido Simulado. Considerada un algoritmo de búsqueda, es una metaheurística inspirada en la física, más exactamente en el proceso de calentamiento y enfriamiento de un metal (de Antonio Suárez, 2011).

2.3.9.2 Optimización Basada en Colonias de Hormigas. Es una metaheurística inspirada en la biología, de modo que busca simular computacionalmente la manera en que las hormigas se comunican entre ellas para establecer el camino más adecuado entre su nido y alguna fuente de alimentos, dejando rastros de feromonas cuando los individuos repiten cierto camino, siendo el camino con el rastro de feromonas más fuerte el más adecuado (de Antonio Suárez, 2011).

2.3.9.3 Búsqueda Tabú. Es una metaheurística que, por medio de un procedimiento de búsqueda local, explora el espacio de soluciones más allá de los óptimos locales. La búsqueda tabú ha encontrado muy buenas soluciones frente a diferentes problemas de optimización combinatoria, con aplicaciones en variados campos como las telecomunicaciones, programación, y reconocimiento de caracteres. Parte de la efectividad de la búsqueda tabú está sustentada por el uso de memoria flexible, memoria de diferentes periodos de tiempo, y condiciones que restringen

y liberan estratégicamente el espacio de búsqueda para una basta exploración del espacio de búsqueda (Glover, 1990).

2.3.9.4 Enjambre de Partículas. La metaheurística de optimización por enjambre de partículas consiste en un algoritmo iterativo, inspirada por la dinámica social de especies de individuos que se mueven en grupos, como peces, insectos y aves. En dicha dinámica cada individuo comunica al grupo su experiencia, y en base a eso el grupo se mueve en cierta dirección. (Lima M. & Barán C, 2006). En la optimización por enjambre de partículas, cada partícula es una posible solución al problema, y cada partícula recuerda su mejor posición . También se tiene acceso a la mejor posición encontrada por todo el grupo de partículas, y por medio de estas dos fuentes de información, el enjambre de partículas explora el espacio de soluciones eficiente y cooperativamente.

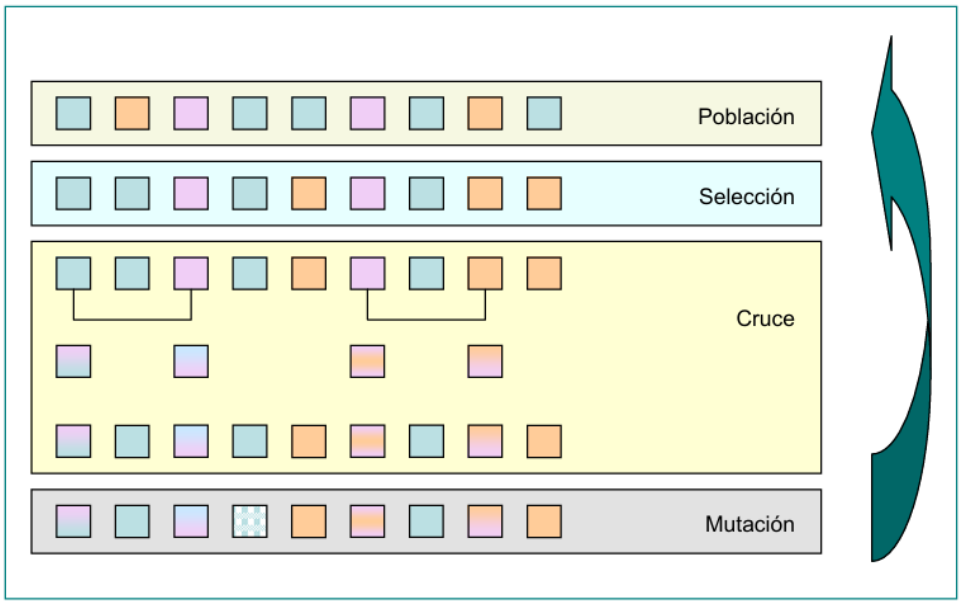
2.3.9.5 Algoritmo Genético. Considerada un algoritmo evolutivo, es una metaheurística inspirada en la evolución (de Antonio Suárez, 2011). El algoritmo genético imita el comportamiento de una población de individuos que evoluciona según los principios de la evolución natural: reproducción por medio de operadores genéticos y selección de los individuos más fuertes, entendiendo la población de individuos como las soluciones al problema de optimización y los individuos más fuertes como las mejores soluciones al problema. (Melián Batista, Moreno Pérez, & Moreno Vega, 2009).

Para entender cómo la evolución natural se relaciona con los problemas de optimización, Melián Batista et al. (2009) plantean las siguientes relaciones: Los cromosomas, que se encargan de transmitir información genética de un ser vivo, equivalen a una posible solución al problema de optimización; así como en la naturaleza algunos individuos están mejor adaptados al entorno, esto se mide por medio de una función fitness que nos deja saber qué tan buena es una solución

para el problema (menor costo, mayores ganancias, menor tiempo, etc.); en biología las especies evolucionan por medio de cruce (combinando genes) y mutaciones (cambiando genes al azar), en un algoritmo genético el cruce no es más que combinar partes de dos soluciones para generar nuevas, y la mutación es la combinación aleatoria de partes de una solución.

Figura 16

Esquema de funcionamiento de un algoritmo genético



Nota. Tomado de *Algoritmos Genéticos. Una visión práctica,* por M. B. Melián Batista, J. A. Moreno Pérez, y J. M. Moreno Vega, 2009, *Números: Revista de Didáctica de las Matemáticas* (núm. 71), p. 33.

3. Modelo de Optimización

3.1 Descripción del problema

El problema abordado es el enrutamiento de un recolector en un almacén de un único bloque, un único depósito y una configuración de almacenamiento disperso, en la cual un mismo SKU puede estar disponible en más de una posición de recolección a lo largo de los estantes disponibles a la izquierda y derecha de los pasillos de un almacén, un problema que Weidinger (2018) demostró ser NP-Hard. El objetivo es determinar un tour de recolección que permita reunir todas las unidades de mantenimiento de existencias (SKU por su sigla en inglés) de una lista de recolección, asumiendo que el recolector puede cargar todos los elementos pertenecientes a la lista, minimizando el costo total del tour. El costo del tour está determinado por (i) la distancia total recorrida al interior del almacén y (ii) una penalización proxy dependiente de la altura de recolección del SKU recolectado. Las SKUs están almacenadas en estanterías a ambos lados de los pasillos longitudinales del almacén, que a su vez son adjuntos a un pasillo transversal superior y uno inferior sin almacenamiento. En cada pasillo longitudinal se encuentra un número de posiciones de recolección, siendo irrelevante el estante del cual se requiera retribuir un artículo (estante derecho, izquierdo, o ambos) ya que solo se considera el costo de desplazamiento para llegar a las posiciones de recolección.

En cuanto a la penalización proxy de altura, se asume que la dificultad de recolección depende del nivel vertical del SKU en la estantería, bajo el supuesto que la dificultad para retribuir un objeto según la altura en que este se encuentra aumenta de la siguiente manera: altura media (zona entre el hombro y la cadera) dificultad baja, altura superior (sobre el nivel del hombro) dificultad media y altura inferior (entre la cadera y los pies) dificultad alta. Adicionalmente, se asume que al visitar una posición de recolección el recolector puede extraer cualquier cantidad hasta el inventario disponible en dicha posición.

La formulación compacta propuesta por Goeke y Schneider (2021) hace uso de dos propiedades de un tour óptimo propuestas por Ratliff y Rosenthal (1983), siendo que existen solo 4 formas de conectar dos pasillos consecutivos en un tour de recolección y, para evitar la generación de sub tours aislados, basta con asegurar que el tour siempre está conectado y el grado de las conexiones en el punto superior e inferior de cada pasillo sea par. Sin embargo, esta formulación no tiene en cuenta la altura en la que se encuentra ubicada el SKU en el estante. En consecuencia, en este trabajo se propone una extensión del modelo de Goeke y Schneider (2021) que incorpora la penalización por ergonomía según la altura a la que se encuentre el SKU.

3.2 Modelo matemático

Parámetros:

m: Cantidad de pasillos, indexados de izquierda a derecha

J: Conjunto de pasillos

n: Cantidad de posiciones de recolección en cada pasillo, indexados de arriba abajo

l: pasillo en el cual se encuentra posicionado el depósito.

$\dot{\theta}$: Posición del depósito (1 si es en pasillo superior, 0 si es en pasillo inferior inferior)

C: Costos del recorrido cuando cada variable asociada vale 1 (Hay un costo por cada variable de decisión X).

H: SKUs que deben ser recolectados

b_h: Cantidad de ítems del SKU h que se requieren

I_j: Conjunto de posiciones de recolección i en el pasillo j donde hay SKUs de la lista de recolección en alguna altura k

I_{jh}: Conjunto de posiciones de recolección i donde el SKU h está disponible en el pasillo j en alguna altura k

r: Cantidad de niveles verticales.

K: Conjunto de posiciones verticales

q_{jikh}: Cantidad de ítems del SKU h disponible en el pasillo j, posición de recolección i, altura k

C_v: Costo de recolectar según la posición vertical del SKU

Variables de decisión:

$$\overline{\overline{x_j}} \begin{cases} 1 \ si\ se\ recorre\ del\ pasillo\ j\ a\ j+1\ dos\ veces\ por\ el\ pasillo\ superior \\ 0\ \ de\ lo\ contrario \end{cases}$$

$$\underline{\underline{x_j}} \begin{cases} 1 \ si\ se\ recorre\ del\ pasillo\ j\ a\ j+1\ dos\ veces\ por\ el\ pasillo\ inferior \\ 0\ \ de\ lo\ contrario \end{cases}$$

$$\underline{\overline{x_j}} \begin{cases} 1 \ si\ se\ recorre\ del\ pasillo\ j\ a\ j+1\ dos\ veces\ por\ el\ pasillo\ superior\ e\ inferior \\ 0\ \ de\ lo\ contrario \end{cases}$$

$$\underline{\overline{x_j}} \begin{cases} 1 \ si\ se\ recorre\ del\ pasillo\ j\ a\ j+1\ una\ vez\ por\ el\ pasillo\ superior\ e\ inferior \\ 0\ \ de\ lo\ contrario \end{cases}$$

$$x'_j \begin{cases} 1 \ si\ se\ recorre\ del\ pasillo\ j\ por\ completo\ 1\ vez \\ 0\ \ de\ lo\ contrario \end{cases}$$

$$x''_j \begin{cases} 1 \ si\ se\ recorre\ del\ pasillo\ j\ por\ completo\ 2\ veces \\ 0\ \ de\ lo\ contrario \end{cases}$$

$$\tilde{x}_{ji} \begin{cases} 1 \ si\ se\ ingresa\ al\ pasillo\ j\ por\ arriba\ y\ se\ recorre\ hasta\ posición\ i \\ 0\ \ de\ lo\ contrario \end{cases}$$

$$\underset{\sim}{x}_{ji} \begin{cases} 1\ si\ se\ ingresa\ al\ pasillo\ j\ por\ abajo\ y\ se\ recorre\ hasta\ posición\ i \\ 0\ de\ lo\ contrario \end{cases}$$

$$\tilde{\pi}_j\ \ Variable\ entera, grado\ de\ conexiones\ en\ el\ pasillo\ superior\ j\ dividido\ en\ 2$$

$$\underset{\sim}{\pi}_j\ Variable\ entera, grado\ de\ conexiones\ en\ el\ pasillo\ inferior\ j\ dividido\ en\ 2$$

$$\tau_j \begin{cases} 0\ si\ la\ ruta\ desde\ el\ pasillo\ más\ a\ la\ izquierda\ hasta\ el\ j\ está\ conectada \\ 1\ si\ la\ ruta\ está\ dividida\ en\ dos\ componentes \end{cases}$$

$$x_{ji} \begin{cases} 1\ si\ se\ visita\ la\ posición\ de\ recolección\ i\ en\ el\ pasillo\ j \\ 0\ de\ lo\ contrario \end{cases}$$

$$\tilde{x}_j \begin{cases} 1\ si\ el\ pasillo\ j\ es\ alcanzado\ por\ el\ recolector\ en\ el\ tour \\ 0\ de\ lo\ contrario \end{cases}$$

$$xl_{jik} \begin{cases} 1\ si\ se\ recolecta\ en\ la\ posición\ de\ recolección\ i\ en\ el\ pasillo\ j\ en\ la\ altura\ k \\ 0\ de\ lo\ contrario \end{cases}$$

Los parámetros empleados en el modelo se resumen en la **Tabla 3**. Las variables de decisión se resumen en la **Tabla 4**.

Función Objetivo

$$\textbf{mi n} \sum_{j\in J} \underline{\underline{c_j}}.\underline{\underline{x_j}} + \overline{\overline{c_j}}.\overline{\overline{x_j}} + \underline{c_j}.\underline{x_j} + \overline{\overline{c_j}}.\overline{\overline{x_j}} + c'_j.x'_j + c''_j.x''_j + \sum_{j\in J}\sum_{i\in I_j}\left(\underset{\sim}{c}_{ji}.\underset{\sim}{x}_{ji} + \tilde{c}_{ji}.\tilde{x}_{ji}\right)$$

$$+ \sum_{j\in J}\sum_{i\in I_j}\sum_{k\in K} C_v * xl_{jik} \qquad (1)$$

Tabla 3

Resumen parámetros del modelo

Parámetro	Descripción	Unidad
m	Cantidad de pasillos, indexados de izquierda a derecha	-
J	Conjunto de pasillos	Lista
n	Cantidad de posiciones de recolección en cada pasillo, indexados de arriba abajo	-
l	pasillo en el cual se encuentra posicionado el depósito.	-
$\dot{\theta}$	Posición del depósito (1 si es en pasillo superior, 0 si es en pasillo inferior inferior)	-
C	Costos del recorrido cuando cada variable asociada vale 1 (Hay un costo por cada variable de decisión X)	Unidades de costo
H	SKUs que deben ser recolectados	Lista
b_h	Cantidad de ítems del SKU h que se requieren	Unidades
I_j	Conjunto de posiciones de recolección i en el pasillo j donde hay SKUs de la lista de recolección en alguna altura k	-
I_{jh}	Conjunto de posiciones de recolección i donde el SKU h está disponible en el pasillo j en alguna altura k	-
r	Cantidad de niveles verticales del sistema de almacenamiento	-
K	Conjunto de posiciones verticales	Lista
q_{jikh}	Cantidad de ítems del SKU h disponible en el pasillo j, posición de recolección i, altura k	Unidades
C_v	Costo de recolectar según la posición vertical del SKU	Unidades de costo

Tabla 4

Variables de decisión del modelo

Variable	Tipo	Interpretación
$\overline{\overline{x_j}}$	Binaria	Toma valor 1 si se recorre el pasillo ja $j+1$dos veces por el pasillo superior; 0 en caso contrario
$\underline{\underline{x_j}}$	Binaria	Toma valor 1 si se recorre el pasillo ja $j+1$dos veces por el pasillo inferior; 0 en caso contrario
$\overline{\overline{x_j}}$ $\underline{\underline{\ }}$	Binaria	Toma valor 1 si se recorre el pasillo ja $j+1$dos veces utilizando ambos pasillos, superior e inferior; 0 en caso contrario
$\overline{\underline{x_j}}$	Binaria	Toma valor 1 si se recorre el pasillo j a $j+1$una vez utilizando ambos pasillos, superior e inferior; 0 en caso contrario
x'_j	Binaria	Toma valor 1 si el pasillo j es recorrido completamente una vez; 0 en caso contrario
x''_j	Binaria	Toma valor 1 si el pasillo j es recorrido completamente dos veces; 0 en caso contrario
$\tilde{\tilde{x}}_{ji}$	Binaria	Toma valor 1 si se ingresa al pasillo j por arriba y se recorre hasta la posición de recolección i; 0 en caso contrario
$\underset{\sim}{x}_{ji}$	Binaria	Toma valor 1 si se ingresa al pasillo $j$por abajo y se recorre hasta la posición de recolección i; 0 en caso contrario
$\tilde{\tilde{\pi}}_j$	Entera	Grado de conexiones en el pasillo superior j, dividido entre 2
$\underset{\sim}{\pi}_j$	Entera	Grado de conexiones en el pasillo inferior j, dividido entre 2
τ_j	Binaria	Toma valor 1 si la ruta desde el pasillo más a la izquierda hasta el pasillo $j$está dividida en dos componentes; 0 si es conexa
x_{ji}	Binaria	Toma valor 1 si se visita la posición de recolección $i$en el pasillo j; 0 en caso contrario
$\tilde{\tilde{x}}_j$	Binaria	Toma valor 1 si el pasillo $j$es alcanzado por el recolector durante el tour; 0 en caso contrario
xl_{jik}	Binaria	Toma valor 1 si se recolecta en la posición de recolección $i$del pasillo $j$en el nivel vertical k; 0 en caso contrario

Restricciones:

Tabla 5

Restricciones modelo matemático

$$\tilde{x}_j \geq x_{ji} \qquad j \in J, i \in I_j \qquad (2)$$

$$\tilde{x}_l = 1 \qquad (3)$$

$$\underline{\overline{x_j}} + \overline{\overline{x_j}} + \underline{x_j} + \overline{x_j} = \tilde{x}_{j+1} \qquad j \in J \setminus \{m-1\} : j \geq l \qquad (4)$$

$$\underline{\overline{x_j}} + \overline{\overline{x_j}} + \underline{x_j} + \overline{x_j} = \tilde{x}_j \qquad j \in J : j < l \qquad (5)$$

$$\tilde{x}_j \geq \tilde{x}_{j+1} \qquad j \in J \setminus \{m-1\} : j \geq l \qquad (6)$$

$$\tilde{x}_j \leq \tilde{x}_{j+1} \qquad j \in J : j < l \qquad (7)$$

$$\tilde{x}_j \in \{0,1\} \qquad j \in J \qquad (8)$$

$$\sum_{j \in J} \sum_{i \in I_{jh}} \sum_{k \in K} q_{jikh} . xl_{jik} \geq b_h \qquad h \in H \qquad (9)$$

$$x'_j + x''_j + \sum_{i' \in I_j : i' \geq i} \tilde{x}_{ji'} + \sum_{i' \in I_j : i' \geq i} \underset{\sim}{x}_{ji} \geq x_{ji} \qquad j \in J, i \in I_j \qquad (10)$$

$$x_{ji} \in \{0,1\} \qquad j \in J, i \in I_j \tag{11}$$

$$\left[\overline{\overline{x_{j-1}}} + \underline{\underline{x_{j-1}}} + \overline{\underline{x_{j-1}}}\right] + \underline{\overline{x_j}} + \underline{\underline{x_j}} + \overline{\underline{x_j}} \geq \underline{x}_{ji} \qquad Si\,(\dot{\theta} = 1)\{j \in J\}\ else\ \{j \in J \setminus \{l\}\},\ i \in I_j \tag{12}$$

$$\left[\overline{\overline{x_{j-1}}} + \overline{\overline{x_{j-1}}} + \overline{\underline{x_{j-1}}}\right] + \underline{\overline{x_j}} + \overline{\overline{x_j}} + \overline{\underline{x_j}} \geq \tilde{\pi}_{ji} \qquad Si\,(\dot{\theta} = 0)\{j \in J\}\ else\ \{j \in J \setminus \{l\}\},\ i \in I_j \tag{13}$$

$$\overline{\overline{x_{j-1}}} + \underline{x_j} \leq x''_j + 1 \qquad j \in J \setminus \{0\} \tag{14}$$

$$\underline{\underline{x_{j-1}}} + \overline{\overline{x_j}} \leq x''_j + 1 \qquad j \in J \setminus \{0\} \tag{15}$$

$$2x''_l + x'_l + \left[\overline{\overline{x_{l-1}}} + \underline{\overline{x_{l-1}}}\right]_{l>0} + \overline{\overline{x_l}} + \underline{\overline{x_l}} \geq \left[\underline{\underline{x_{l-1}}}\right]_{l>0} + \underline{\underline{x_l}} \qquad Si\,(\dot{\theta} = 1) \tag{16}$$

$$2x''_l + x'_l + \left[\underline{\underline{x_{l-1}}} + \underline{\overline{x_{l-1}}}\right]_{l>0} + \underline{\underline{x_l}} + \underline{\overline{x_l}} \geq \left[\overline{\overline{x_{l-1}}}\right]_{l>0} + \overline{\overline{x_l}} \qquad Si\,(\dot{\theta} = 0) \tag{17}$$

$$\left[\overline{\underline{x_{j-1}}} + 2\overline{\underline{x_{j-1}}} + 2\overline{\underline{x_{j-1}}}\right]_{j>0} + \overline{\underline{x_j}} + 2\underline{\overline{x_j}} + 2\overline{\underline{x_j}} + x'_j + 2x''_j = 2\,\tilde{\pi}_j \qquad j \in J \tag{18}$$

$$\left[\underline{\overline{x_{j-1}}} + 2\underline{\underline{\overline{x_{j-1}}}} + 2\underline{\underline{x_{j-1}}}\right]_{j>0} + \underline{\overline{x_j}} + 2\underline{\underline{\overline{x_j}}} + 2\underline{\underline{x_j}} + x'_j + 2x''_j = 2\,\underset{\sim}{\pi}_j \qquad j \in J \qquad (19)$$

$$\underline{\underline{\overline{x_j}}} + \underline{\underline{\overline{x_{j-1}}}} + \underline{\underline{x_{j-1}}} - x''_j \leq \tau_j + 1 \qquad j \in J\backslash\{0\} \qquad (20)$$

$$\underline{\underline{\overline{x_j}}} + \left[-\underline{\underline{\overline{x_{j-1}}}} - \underline{\underline{\overline{x_{j-1}}}} - \underline{\underline{x_{j-1}}}\right]_{j>0} - x''_j - x'_j \leq \tau_j \qquad j \in J \qquad (21)$$

$$\tau_{j-1} - x'_j - x''_j \leq \tau_j \qquad j \in J\,\{0\} \qquad (22)$$

$$\tau_j \leq \underline{\underline{\overline{x_j}}} \qquad i \in J \qquad (23)$$

$$\underline{\underline{\overline{x_j}}}, \underline{\underline{x_j}}, \underline{\underline{\overline{x_j}}}, \underline{\overline{x_j}}, \tau_j \in \{0,1\} \qquad j \in J\backslash\{m-1\} \qquad (24)$$

$$x'_j, x''_j \in \{0,1\} \qquad j \in J \qquad (25)$$

$$\tilde{x}_{ji}, \underset{\sim}{x}_{ji} \in \{0,1\} \qquad j \in J \qquad (26)$$

$$\tilde{\pi}_j, \underset{\sim}{\pi}_j \in \{0,1,2,3\} \qquad j \in J \qquad (27)$$

$$\overline{\overline{x_{m-1}}}, \underline{\underline{x_{m-1}}}, \overline{\overline{\overline{x_{m-1}}}}, \underline{\underline{\underline{x_{m-1}}}}, \tau_{m-1} = 0 \tag{28}$$

$$x_{ji} \geq xl_{jik} \qquad j \in J, i \in I_j, k \in K \tag{29}$$

La función objetivo (1) describe el costo total del tour, el cual se busca minimizar. Las restricciones (2) y (3) aseguran que tanto el pasillo donde está el depósito como los pasillos que contienen las posiciones de recolección de los SKUs requeridos sean visitados. La restricción (4) define la conexión para alcanzar pasillos que se encuentren a la derecha del depósito, mientras la restricción (5) hace lo mismo para pasillos a la izquierda del depósito. La restricción (6) asegura que si se visita el pasillo j+1 a la derecha del depósito primero se visitó el pasillo j, y (7) asegura que si se visita el pasillo j a la izquierda del depósito primero se visitó el pasillo j+1. La restricción (9) asegura que la cantidad del SKU h disponible en las posiciones de recolección visitadas por la ruta de recolección cumpla con la cantidad de ítems requeridos del SKU h. La restricción (10) asegura que las posiciones de recolección seleccionadas sean visitadas por el tour de recolección. Las restricciones (12) y (13) aseguran que el ingreso al pasillo j por el pasillo transversal inferior (superior) solo sucede si el pasillo j está conectado al pasillo anterior o siguiente con una configuración que emplee el pasillo transversal inferior (superior); los paréntesis cuadrados excluyen el pasillo anterior cuando se restringe el primer pasillo. Las restricciones (14) y (15) aseguran que los cambios entre pasillo inferior y superior estén definidos de una manera factible. Restricciones (16) y (17) aseguran que el depósito sea incluido en el tour de recolección. Restricciones (18) y (19) aseguran que el grado de todas las conexiones arriba y debajo de cada pasillo sea par, ya que cada posición debe ser abandonada tantas veces como es visitada. La

restricción (20) ajusta el número de componentes a 2 si hay una transición de $\overline{\overline{x_{j-1}}} = 1$ o $\underline{\underline{x_{j-1}}} = 1$ a $\overline{\overline{x_j}} = 1$ sin conectar los pasillos transversales superior e inferior por medio de $x''_j = 1$. La restricción (21) establece el número de componentes como 2 si el pasillo transversal superior e inferior no están conectados por un recorrido completo del pasillo j y la sección a la izquierda del pasillo no es visitada por el tour. Restricción (22) se encarga de propagar la desconexión entre el pasillo superior e inferior y eventualmente corregirla. La restricción (23) asegura que la configuración $\overline{\overline{x_j}}$ solo sea usada si hay dos componentes del tour. Por último, las restricciones (24)-(28) definen las variables de decisión. La restricción (29) asegura que solo se acceda a una altura k en una posición i, j si la posición es visitada por el tour de recolección.

3.3 validación del modelo

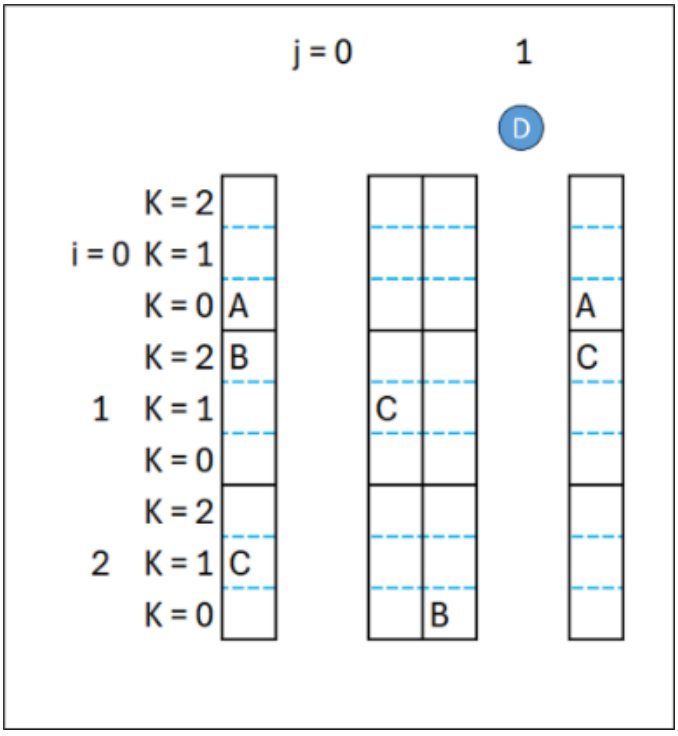
Una vez formulado el modelo matemático, se procede a programar y resolver en Python mediante el solver Gurobi. Se formula una instancia pequeña que pueda ser fácilmente resuelta a optimalidad con el objetivo de contrastar la solución encontrada por el modelo.

Se aborda un problema con m = 2 pasillos, n = 3 posiciones de recolección en cada pasillo cada posición con sus respectivas r = 3 categorías de altura, y una lista de recolección que cuenta con tres elementos, siendo demandada una sola unidad de cada elemento. El depósito se encuentra en el pasillo 2 (j = 1). La distribución del almacén se puede ver en la **Figura 17**, a modo de ejemplo el elemento A esta disponible en el pasillo 0, posición de recolección 0 y altura 0 (entre el suelo y la altura de la cadera). El costo de trasladarse una vez de un pasillo j a j+1 equivale a 3 unidades, y cada desplazamiento al interior de un pasillo tiene un costo de 1 unidad. Respecto a la recolección

en los diferentes niveles, se modela el nivel r = 0 con un costo de 1.6 unidades, r = 1 costo de 1unidad, y r = 2 costo de 1.3 unidades.

Figura 17

Distribución almacén estancia



Al existir diferentes puntos donde cada producto está disponible, se crean diferentes grupos de puntos de recolección que pueden satisfacer la demanda, cada grupo con sus respectivas rutas de recolección. En la **Figura 18** se muestran algunas de las diferentes rutas de recolección que representan soluciones factibles al problema en cuestión. Tanto con el modelo resuelto en Gurobi como manualmente, se encontró como solución óptima la solución número 4, teniendo un costo de 9.9 unidades, para la cuál las variables de decisión $\overline{\overline{x_0}}, \tilde{\tilde{x}}_{01}, x_{00}, \ x_{01}, \ xl_{000}, xl_{011}, xl_{012}, \tilde{x}_0, \tilde{x}_1$ toman el valor de 1, y $\tilde{\tilde{\pi}}_0 = 2, \tilde{\tilde{\pi}}_1 = 1$. La **Tabla *6*** refleja los costos asociados a cada variable de decisión.

Figura 18

Soluciones factibles para la instancia propuesta

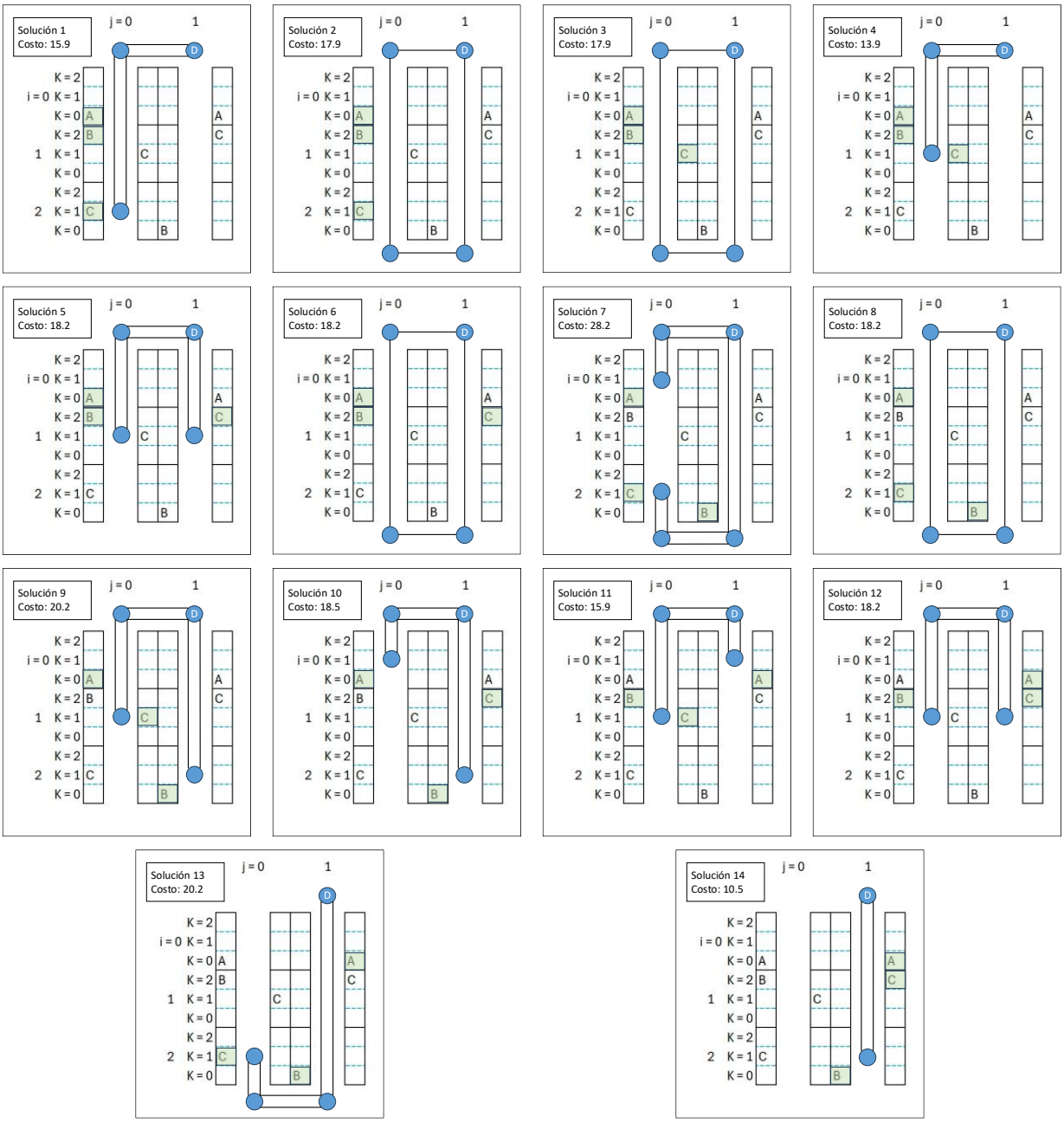


Tabla 6

Costos asociados a variables de decisión de la solución óptima

Variable	Costo Asociado	Variable	Costo Asociado
$\tilde{\tilde{x}}_{12}$	6	xl_{112}	1.3
xl_{100}	1.6	xl_{120}	1.6
Total		9.9	

En la **Tabla** *7* se manifiesta un equivalente entre las variables de decisión del modelo y las variables programadas en el script.

Tabla 7

Nombre de variables en Script

Variable Modelo	Variable en Script	Variable Modelo	Variable en Script
$\overline{\overline{x_j}}$	Xsup2[j]	$\underset{\sim}{x}_{ji}$	Xcheck [j, i]
$\underline{\underline{x_j}}$	Xinf2[j]	$\tilde{\tilde{\pi}}_j$	Pihat[j]
$\underline{\overline{x_j}}$	Xsupinf2[j]	$\underset{\sim}{\pi}_j$	Picheck[j]
$\overline{\underline{x_j}}$	Xinf_sup1[j]	τ_j	Tau[j]
x'_j	Xvert1[j]	x_{ji}	Xpos [j, i]
x''_j	Xvert2[j]	$\tilde{x}_j$	Xtilde[j]
$\tilde{\tilde{x}}_{ji}$	Xhat [j, i]	xl_{jik}	Xlevel [j, i, k]

4. Algoritmo Genético

4.1 Diseño algoritmo genético

El problema abordado es el Single Picker Routing Problem (SPRP) en un almacén con estanterías multinivel, donde adicionalmente al costo del desplazamiento horizontal, se integra un costo ergonómico asociado a la altura de recolección. Como método de solución se propone un algoritmo genético híbrido, que combina búsqueda evolutiva, con una intensificación local a través de una búsqueda local.

El algoritmo genético explora el espacio de soluciones relacionado a la selección de posiciones de recolección que permitan satisfacer la demanda de cada SKU, y una heurística determinista de dos pasillos de cruce construye una ruta factible que inicia y finaliza en el depósito mediante:

- Selección de dos pasillos para ser recorridos por completo
- Clasificación los demás pasillos como ‘recolección desde arriba’ o ‘recolección desde abajo.

La construcción de las posiciones de recolección posibles se encarga de respetar la cantidad de cada SKU que está en inventario, evitando la creación de cromosomas no viables, y asegurando suplir la demanda para cada SKU. La ***Tabla 8*** muestra el seudocódigo bajo el cual opera la lógica del algoritmo genético híbrido planteado.

Tabla 8

Seudocódigo algoritmo

Algoritmo GA_Híbrido_SPRP_Shelves

Entradas: q[j][i][k][h]: inventario del almacén; bh[h]: demanda de cada SKU; J, I, K: pasillos, posiciones, niveles; l_depot: ubicación del depósito; θ: posición del depósito (arriba / abajo); tam_pob: tamaño de la población n_gen: número máximo de generaciones; p_cruce: probabilidad de cruce; p_mut: probabilidad de mutación; k_torneo; tamaño del torneo; elitismo: porcentaje o número de élites; erg_cost[k]: costo ergonómico por nivel

Salida: Mejor_solucion, mejor_costo, tiempo_total, tiempo_mejor

```
1. PREPROCESAMIENTO
   H_req ← {h ∈ H | bh[h] > 0}
   Para cada h ∈ H_req:
       Posiciones[h] ← {(j, i, k) | q[j][i][k][h] > 0}
2. INICIALIZACIÓN DE LA POBLACIÓN
   Para p = 1 hasta tam_pob:
      Para cada h ∈ H_req:
         Asignar aleatoriamente ubicaciones hasta satisfacer bh[h]
      Construir cromosoma p
      (Opcional) Aplicar búsqueda local al cromosoma
3. EVALUACIÓN INICIAL
   Para cada individuo p:
      Construir ruta con heurística S-Shape con dos pasillos de cruce
      Calcular:
         costo_ruta
         costo_ergonómico
      fitness[p] ← costo_total
   Guardar mejor individuo global
4. EVOLUCIÓN
   sin_mejora ← 0
   patience ← 0.2 × n_gen
   Para gen = 1 hasta n_gen:
      4.1 ELITISMO
         Copiar los mejores individuos a la nueva población
      4.2 REPRODUCCIÓN
         Mientras nueva_población < tam_pob:
            padre1 ← selección por torneo
            padre2 ← selección por torneo
            (hijo1, hijo2) ← CRUCE (padre1, padre2)
            MUTAR (hijo1)
            MUTAR (hijo2)
            Agregar hijos a la nueva población
      4.3 EVALUACIÓN
         Evaluar todos los individuos de la nueva población
      4.4 ACTUALIZAR MEJOR SOLUCIÓN
         Si existe mejora:
            actualizar mejor  solución
            sin_mejora ← 0
         Si no:
            sin_mejora ← sin_mejora + 1

      4.5 CRITERIO DE PARADA
         Si sin_mejora ≥ patience:
            Terminar algoritmo
   5. DEVOLVER RESULTADOS
      Retornar mejor_solución, mejor_costo, tiempo_total, tiempo_mejor
```

4.1.1 Representación del individuo

Cada individuo perteneciente a la población representa una solución factible al problema, codificado a partir de una estructura basada en los SKUs requeridos por la orden de recolección. Cada gen corresponde a un grupo de posiciones de recolección que suple la demanda de un único SKU sin exceder la disponibilidad del inventario, permitiendo la modificación independiente de las ubicaciones de selección para cada SKU.

Para cada SKU h el cromosoma almacena una lista de asignaciones de la siguiente forma:

$$(\,(j, i, k), q)$$

Donde j representa el pasillo, i es la posición de recolección, k es el nivel en el rack y q es la cantidad recolectada del SKU h en esta ubicación. De modo que la suma de las cantidades q recolectadas en cada posición de recolección permite suplir la demanda del SKU h. A modo ilustrativo, para una instancia con $I = [0, 1]$ pasillos, $J = [0, 1, 2]$ posiciones de recolección, $K = [0, 1, 2]$ niveles de los racks, $H = [0, 1, 2]$ elementos a recolectar, y con una demanda $b_0 = 4, b_1 = 2, b_2 = 2$, un cromosoma se presenta a continuación:

$$C = \{0 : [((0, 1, 1), 1), ((0, 2, 0), 3)], 1 : [((1, 2, 1), 2)], 2 : [((0, 2, 2), 1), ((1, 0, 2), 1)]\}$$

4.1.2 Población inicial

Para cada SKU en la lista de recolección se asigna su demanda total a un conjunto aleatorio de posiciones de recolección factibles, seleccionadas entre las ubicaciones donde existe inventario disponible, garantizando que los individuos de la población inicial cumplen con las restricciones de demanda del problema. La asignación de posiciones de recolección aleatorias asegura la factibilidad al seleccionar únicamente posiciones con inventario disponible y cumplir con las cantidades demandadas.

4.1.3 Fitness

La evaluación de cada individuo es desarrollada en dos etapas. Inicialmente, a partir de la información de los puntos de recolección brindados por el cromosoma, se identifican los pasillos a visitar, y se construye una ruta factible utilizando una heurística de ruteo tipo S-shape, que considera explícitamente dos pasillos de cruce completos. Esta selección se realiza según la cantidad de posiciones de recolección que contiene cada pasillo, priorizando aquellos con mayor número de posiciones a visitar.

Posteriormente, se calcula el costo total de la solución como la suma del costo de desplazamiento horizontal – asociado a la ruta generada – y el costo ergonómico, calculado como la suma de penalizaciones asociadas a los niveles verticales k visitados en cada punto de recolección. En la ***Tabla 9*** se aprecia el seudocódigo responsable de esta operación.

Tabla 9

Seudocódigo evaluación del individuo

Evaluación de ruta: evaluate_individual
Entrada: Cromosoma, dimensiones de almacén, localización del depósito, costos ergonómicos
Salida: Costo Total

```
Identificar los pasillos usados por el cromosoma
Seleccionar dos pasillos de cruce completo
Clasificar pasillos (arriba/abajo) según costo mínimo
Construir ruta dependiente del depósito: info_ruta
Calcular:
   Costo_ruta
   Costo_ergonomia = suma erg_cost[k] para cada nivel k visitado
Retornar costo_total, info_ruta
```

4.1.4 Selección

Un torneo binario se encarga de seleccionar los individuos para la reproducción, eligiendo aleatoriamente dos individuos de la población y seleccionando aquel con menor fitness, buscando

favorecer la selección de soluciones de mejor calidad sin descuidar la diversidad genética. Este proceso está documentado en la ***Tabla 10***.

Tabla 10

Seudocódigo torneo

Selección por torneo: torneo
Entradas: Población, fitness, k
Salida: Individuo seleccionado
Seleccionar k individuos al azar
Retornar individuo con menor fitness

4.1.5 Cruce

Con el fin de garantizar que los descendientes conserven el mismo conjunto de SKUs requeridos y evitar soluciones no factibles, se implementa un cruce uniforme por SKU según una probabilidad predefinida, de modo que para cada SKU requerido el gen correspondiente del padre se hereda con igual probabilidad, como lo refleja el pseudocódigo de la ***Tabla 11***.

4.1.6 Mutación

La mutación se desarrolla como muestra la ***Tabla 12*** seleccionando aleatoriamente un SKU requerido y volviendo a generar su asignación de ubicaciones y cantidades, introduciendo variabilidad a la población y permitiendo explorar el campo de soluciones.

4.1.7 Búsqueda Local

En aras de intensificar la búsqueda, se incorpora una búsqueda local aplicada a los individuos generados. Se selecciona un SKU aleatorio y se propone una nueva asignación factible de posiciones de recolección que permitan suplir la demanda de dicho SKU, esta solución es

aceptada únicamente cuando produce una mejora en el valor fitness, permitiendo refinar soluciones prometedoras. El pseudocódigo se puede encontrar en la ***Tabla 13***.

Tabla 11

Seudocódigo cruce

Cruce: crossover

Entradas: padre1, padre2, H_req, probabilidad de cruce

Salidas: hijo1, hijo2

```
if random () > probabilidad de cruce:
    Retornar padre1 y padre2
Else:
        Para cada SKU h en H_req:
            Con probabilidad 0.5:
                hijo1[h] ← padre1[h]
                hijo2[h] ← padre2[h]
            Si no:
                hijo1[h] ← padre2[h]
                hijo2[h] ← padre1[h]
    Retornar (hijo1, hijo2)
```

Tabla 12

Seudocódigo mutación

Mutación: mutate

Entradas: cromosoma, positions_per_sku, demanda bh, H_req, probabilidad mutar

Salidas: Cromosoma mutado

```
Con probabilidad p_mut:
    Seleccionar un SKU h al azar
Reasignar aleatoriamente ubicaciones que satisfagan bh[h]
```

Tabla 13

Seudocódigo búsqueda local

Búsqueda local: local_search
Entradas: cromosoma, positions_per_sku, demanda bh, parámetros almacén, iteraciones máximas
Salidas: Cromosoma mejorado

```
Repetir hasta iteraciones máximas:
   Seleccionar SKU h
   Generar nueva asignación factible
   Si mejora el fitness:
      Aceptar cambio
   Si no:
      Detener
```

4.1.8 Criterios de parada

El algoritmo se ejecuta hasta cumplir un número máximo de generaciones o no observar mejoras en la solución encontrada durante un numero predefinido de iteraciones consecutivas.

4.2 Calibración del algoritmo genético.

Buscando identificar la configuración de parámetros que ofrece el mejor desempeño promedio equilibrando entre eficiencia computacional y calidad de solución, se desarrolló una calibración del algoritmo. En esta sección no se persigue evaluar el desempeño final del método de solución propuesto, sino elegir una mezcla de parámetros robusta para posteriormente abordar la validación del algoritmo.

Cabe resaltar que las instancias tanto de calibración como de prueba empleadas en este trabajo se generan siguiendo el procedimiento propuesto por Goeke & Schneider (2021). Este procedimiento define el número de SKUs distintos, clasificación en clases de popularidad y la

distribución de inventario en el almacén de modo que cada SKU se encuentre en al menos una posición de recolección, y la demanda de cada SKU pueda ser suplida con el inventario disponible en el almacén. Para este trabajo, dicho procedimiento se adaptó para incorporar las estanterías multinivel en el almacén.

4.2.3 Instancias de calibración.

En esta sección se dispuso de un conjunto de seis demandas representativas del problema, denominadas pequeñas, medianas y grandes según sus valores, y generadas según el procedimiento descrito en la sección 5.1. Cada una de las instancias fue resuelta a optimalidad con el modelo exacto implementado en Gurobi dentro del tiempo límite establecido, permitiendo usar el valor óptimo de la función objetivo como referencia para el cálculo del gap de optimalidad.

Formalmente, cada instancia es descrita por un conjunto de parámetros estructurales (m, n, a, α) y una semilla de generación para garantizar su reproducibilidad. La **Tabla *14*** conglomera el conjunto de instancias empleadas en esta fase y sus respectivos valores óptimos.

Tabla 14

Instancias resueltas por Gurobi

ID	Tamaño	Seed instancia	(m, n, alpha, a)	Mejor Gurobi	Gap Gurobi (%)	Status	Tiempo límite (s)
I1	Pequeña	101	(5, 30, 10, 15)	143	0.00%	Optimo	1800
I2	Pequeña	202	(5, 60, 40, 30)	247.99	0.00%	Optimo	1800
I3	Mediana	303	(25, 30, 10, 15)	149.89	0.00%	Optimo	1800
I4	Mediana	404	(25, 60, 40, 30)	361.99	0.00%	Optimo	1800
I5	Grande	505	(100, 30, 10, 15)	364.9	0.00%	Optimo	1800
I6	Grande	606	(100, 60, 40, 30)	278.8	0.00%	Optimo	1800

4.2.4 Diseño del experimento de Calibración

Mediante una búsqueda en grilla sobre un conjunto acotado de valores para los principales parámetros del algoritmo genético, se consolidaron distintas combinaciones de tamaño de población, probabilidades de cruce y mutación, tamaño del torneo y tasa de elitismo, manteniendo la búsqueda local activada como parte integral del enfoque híbrido. Los parámetros considerados están consolidados en la **Tabla *15***.

Manteniendo un presupuesto computacional uniforme por medio del número de generaciones, permitió asegurar la comparabilidad entre configuraciones. Cada configuración fue evaluada múltiples veces utilizando diferentes semillas para el generador aleatorio. Cada una de las 108 combinaciones de parámetros fue ejecutada sobre las 6 instancias de calibración y 5 semillas distintas, para un total de 30 corridas por configuración, y un total global de 3240 corridas, cuyos resultados se pueden ver en el Apéndice C. De cada corrida se guardó el valor de la mejor solución encontrada, tiempo total de ejecución y el tiempo requerido para alcanzar dicha solución.

Tabla 15

Parámetros de calibración algoritmo genético.

Parámetro	**Variable en código**	**Niveles**
Tamaño de población	tam_pob	{50, 100, 150}
Probabilidad de cruce	prob_cruce	{0.70, 0.90}
Probabilidad de mutación	prob_mutar	{0.05, 0.10, 0.20}
Tamaño de torneo	tam_torneo	{2, 3, 5}
Elitismo (porcentaje)	elitismo	{1%, 5%}
Búsqueda local	usar_local_search	Fijo {ON}
Numero de generaciones	n_generaciones	Fijo {500}

Semillas GA	seed_GA	{101, 202, 303, 404, 505}

4.2.5 Métricas de evaluación.

La evaluación principal del desempeño de cada configuración se realizó mediante el gap promedio de optimalidad, que se define como:

$$gap(\%) = 100 * \frac{Z_{GA} - Z^*}{Z^*}$$

Z_{GA} : Valor de la función objetivo encontrado por el algoritmo genético.

Z^*: Valor óptimo encontrado con el modelo exacto.

Adicionalmente, el gap mediano, tiempo promedio para alcanzar la mejor solución y la tasa de éxito (porcentaje de corridas con un gap inferior o igual al 1%) también fueron considerados.

Un criterio jerárquico dictó la selección final de la configuración, tomando en primer lugar la configuración con gap promedio mínimo; en caso de presentarse un empate, se tomó el tiempo promedio para alcanzar la mejor solución como medida de desempate.

4.2.6. Resultados de la calibración.

Entre todas las configuraciones evaluadas, el mejor gap promedio se encontró con una configuración formada por un tamaño de población de 150 individuos lo cual asegura una diversidad inicial, probabilidad de cruce 0.7, probabilidad de mutación e 0.2 que ayuda a evitar el estancamiento, tamaño de torneo de 5 que acelera la convergencia, y una tasa de elitismo del 5% que preserva soluciones buenas sin congelar la población.

Esta configuración, dominante en cuanto a calidad promedio de solución se refiere, fue seleccionada como la configuración final del algoritmo genético y se utilizó posteriormente en la fase de validación del algoritmo, sin someterse a modificaciones. La **Tabla *16*** presenta las cinco mejores configuraciones obtenidas durante la calibración, ordenadas de manera ascendente respecto al gap promedio.

Es valido resaltar que obtener en este punto unos valores del gap que pueden resultar elevados, es consistente con el objetivo de la calibración, y resalta la importancia de esta. La evaluación del desempeño final del algoritmo es presentada en la fase de validación.

Tabla 16

Top 5 configuraciones.

Rank	Tam_pob	Prob_cruce	Prob_mutar	Tam_torneo	Elitismo	Gap_mean (%)	Gap_median (%)	Tbest_mean (s)	SR1 %
1	150	0.7	0.2	5	0.05	62.10	46.91	6.29	5.54
2	150	0.9	0.2	2	0.01	63.03	40.24	6.59	5.10
3	150	0.7	0.2	2	0.01	63.21	42.12	6.50	5.71
4	150	0.9	0.2	5	0.05	63.26	46.45	6.13	4.90
5	150	0.9	0.2	2	0.05	63.86	41.11	6.32	4.82

5. Validación y evaluación

Los experimentos computacionales desarrollados en el presente trabajo fueron ejecutados en un computador con procesador AMD Ryzen 5 5600H con Radeon Graphics, con una frecuencia base de 3.3 GHz, 6 núcleos y 12 hilos de procesamiento, y 16 GB de memoria RAM. El sistema

operativo empleado fue Microsoft Windows 11 Home de 64 bits. El modelo matemático exacto fue implementado y resuelto mediante el software Gurobi Optimizer, mientras que el algoritmo genético híbrido fue desarrollado en el lenguaje de programación Python. Todos los tiempos computacionales y resultados reportados corresponden a ejecuciones realizadas en este entorno computacional.

5.1 Generación de instancias

Las instancias de prueba generadas con el objetivo de realizar la evaluación del desempeño del modelo matemático exacto y el algoritmo genético propuesto fueron generadas siguiendo el procedimiento descrito por Goeke y Schneider, replicado en la literatura relacionada con el tema de investigación.

El procedimiento considera tres cantidades de pasillos $m \in \{5, 25, 100\}$ y tres diferentes posiciones de recolección por pasillo $n \in \{30, 60, 180\}$. El trabajo base asume inicialmente la presencia de un único SKU en cada posición de recolección, para efectos de este trabajo dicho supuesto es extendido para considerar estanterías con múltiples niveles, de modo que cada posición de recolección puede tener un único SKU en cada uno de sus r niveles de la estantería.

Para analizar la influencia del grado de duplicación de los SKUs en el almacén, se varía el número de SKUs distintos ξ almacenados, según la capacidad total del almacén, definida como $m * n * r$, y de un factor de duplicación $\alpha \in \{1, 5, 10, 40\}$, determinando el número de SKUs según la siguiente expresión:

$$\xi = \max\left(\mathrm{a}, \left[\frac{m * n * r}{\alpha}\right]\right)$$

Donde a $\epsilon \{3, 7, 15, 30\}$ representa el numero de SKUs diferentes incluidos en la lista de recolección. Es importante mencionar que cuando se tiene un factor de duplicación $\alpha = 1$, es un almacen donde cada SKU esta ubicado en una única posición de recoleccion, por lo tanto no se tiene en cuenta en el presente trabajo, ya que el problema de estudio son almacenes con almacenamiento disperso.

Posteriormente, los SKUs son divididos en 3 clases según su rotación y popularidad, asignando 20% de los SKUs a la clase A, 30% a la clase B, y el 50% a la clase C. Cada producto es asignado inicialmente a una posición aleatoria, con el fin de asegurar al menos una posición de recolección por SKU. Posteriormente se asigna el resto de las posiciones disponibles así: 80% SKUs de la clase A, 15% clase B y 5% clase C, con una cantidad de unidades almacenadas generada aleatoriamente entre 1 y 3.

La lista de recolección es generada seleccionando a skus distintos siguiendo las mismas probabilidades asociadas a las clases. La demanda de cada SKU h es generado aleatoriamente en el intervalo $\mathbb{N} \cap [1, min\ (6, \bar{q}_h)]$, donde $\bar{q}_h$ corresponde a la cantidad total disponible del SKU h en el almacén.

5.2 Validación del algoritmo genético.

El objetivo de este capítulo es validar el correcto funcionamiento y la calidad de las soluciones obtenidas por el algoritmo genético híbrido propuesto para el SPRP con estanterías de múltiples niveles y penalización ergonómica.

Para realizar dicha validación, se lleva a cabo (i) un chequeo de la factibilidad operativa de las soluciones generadas, verificando que una ruta generada por la heurística propuesta cumpla las restricciones del problema y (ii) una comparación directa de las soluciones obtenidas mediante el

modelo matemático resuelto por Gurobi y las soluciones obtenidas mediante el algoritmo genético híbrido propuesto. Para esto, se emplearon instancias de tamaño reducido donde el modelo exacto puede asegurar optimalidad en tiempos computacionales despreciables.

Cabe mencionar que esta validación no busca evaluar el desempeño del algoritmo en términos de la escalabilidad, más bien busca verificar que el algoritmo produce soluciones factibles y de buena calidad, y que adicionalmente su comportamiento es coherente con la estructura del problema planteado.

5.2.1. Factibilidad de la ruta generada.

El algoritmo genético no construye directamente una ruta en términos de movimientos secuenciales, sino que determina un conjunto de posiciones de recolección y decisiones de recorrido de los pasillos del almacén, la cual posteriormente es traducida en una secuencia de desplazamientos mediante una heurística de ruteo propuesta, permitiendo interpretar la solución final como una ruta ejecutable por un recolector al interior del almacén.

Se resuelve una instancia pequeña con 10 pasillos, 5 posiciones de recolección por pasillo, una duplicidad de 30 y 10 SKUs en la lista de recolección, la **Figura *19*** muestra una captura de pantalla de la terminal con la salida generada por el algoritmo, donde se detalla paso a paso la secuencia de movimientos a realizar por el recolector, incluyendo el costo de cada movimiento. Esta salida corresponde al individuo final seleccionado por el algoritmo genético y permite verificar de forma explícita las decisiones tomadas durante el recorrido, incluyendo movimientos al interior del almacén y en que posiciones recolectar SKU.

Para facilitar la interpretación espacial del recorrido la **Figura *20*** presenta un diagrama esquemático del almacén, y se dibuja la ruta correspondiente al ejemplo analizado. Esta

representación permite visualizar de manera intuitiva como el recolector se desplaza al interior del almacén y como están agrupadas las recolecciones dentro de cada pasillo. Se puede verificar que el tour inicia y finaliza en el depósito, los movimientos al interior del almacén son factibles y se evidencia coherencia entre la solución del algoritmo y la estructura física del sistema.

Figura 19

Ruta construida por algoritmo genético

```
=== RUTA EN SECUENCIA (PASO A PASO) ===
Inicio en DEPÓSITO: pasillo j=1, cross-aisle SUPERIOR
01) Recolectar en pasillo 1 entrando por ARRIBA: avanzar hasta i=2, recolectar en [(i=1,k=1), (i=2,k=1)]
 y regresar por ARRIBA (costo=6.00)
02) Moverse por cross-aisle SUPERIOR: pasillo 1 -> 2 (costo=3.00)
03) Moverse por cross-aisle SUPERIOR: pasillo 2 -> 3 (costo=3.00)
04) Moverse por cross-aisle SUPERIOR: pasillo 3 -> 4 (costo=3.00)
05) Moverse por cross-aisle SUPERIOR: pasillo 4 -> 5 (costo=3.00)
06) Recolectar en pasillo 5 entrando por ARRIBA: avanzar hasta i=1, recolectar en [(i=1,k=2)] y regresar
 por ARRIBA (costo=4.00)
07) Moverse por cross-aisle SUPERIOR: pasillo 5 -> 6 (costo=3.00)
08) Moverse por cross-aisle SUPERIOR: pasillo 6 -> 7 (costo=3.00)
09) Moverse por cross-aisle SUPERIOR: pasillo 7 -> 8 (costo=3.00)
10) Moverse por cross-aisle SUPERIOR: pasillo 8 -> 9 (costo=3.00)
11) Cruzar completo el pasillo 9: SUPERIOR -> INFERIOR, y recolectar en [(i=1,k=2), (i=2,k=2), (i=4,k=0)
] (costo=6.00)
12) Moverse por cross-aisle INFERIOR: pasillo 9 -> 8 (costo=3.00)
13) Recolectar en pasillo 8 entrando por ABAJO: avanzar hasta i=4, recolectar en [(i=4,k=2)] y regresar
osto=2.00)
14) Moverse por cross-aisle INFERIOR: pasillo 8 -> 7 (costo=3.00)
15) Cruzar completo el pasillo 7: INFERIOR -> SUPERIOR, y recolectar en [(i=0,k=1), (i=1,k=1), (i=2,k=1)
, (i=3,k=2), (i=4,k=0), (i=4,k=1)] (costo=6.00)
16) Moverse por cross-aisle SUPERIOR: pasillo 7 -> 6 (costo=3.00)
17) Moverse por cross-aisle SUPERIOR: pasillo 6 -> 5 (costo=3.00)
18) Moverse por cross-aisle SUPERIOR: pasillo 5 -> 4 (costo=3.00)
19) Moverse por cross-aisle SUPERIOR: pasillo 4 -> 3 (costo=3.00)
20) Moverse por cross-aisle SUPERIOR: pasillo 3 -> 2 (costo=3.00)
21) Moverse por cross-aisle SUPERIOR: pasillo 2 -> 1 (costo=3.00)
=== FIN DE RUTA ===

Costo total del tour de recoleccion es de:  87.7
```

Con el objetivo de rectificar el costo del tour de recolección, en la **Tabla *17*** se encuentra un desglose del costo de cada movimiento realizado por el tour de recolección, especificando cuantas unidades de movimiento se computan en cada configuración y el costo de esta, para el costo ergonómico se calcula mediante la penalización por recoger en un nivel k multiplicado por el número de veces que se recolecta en ese nivel k. Al realizar la suma de todos los costos, se evidencia que el costo total es el mismo reportado por el algoritmo genético.

Figura 20

Representación visual de la ruta

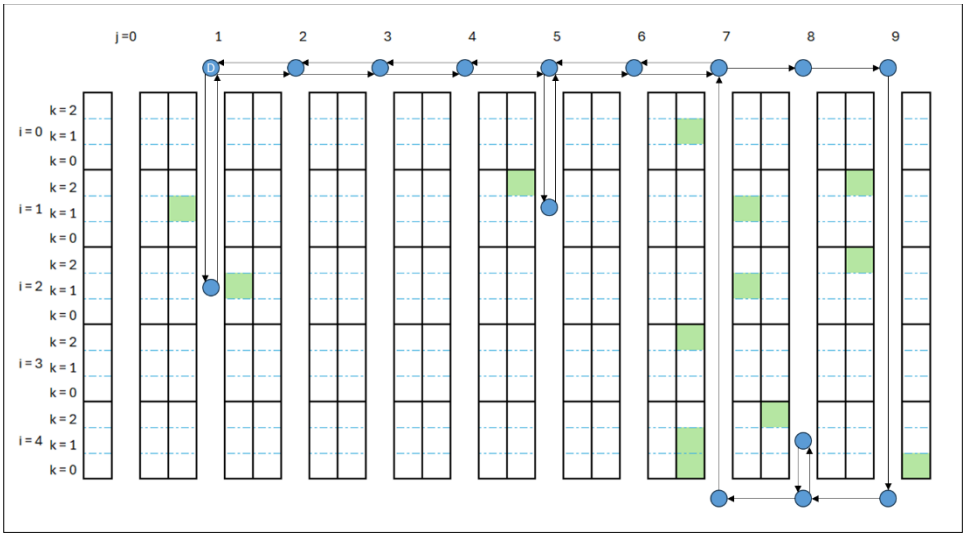


Tabla 17

Desglose costos tour de recolección

Movimientos en pasillo superior	Movimientos en pasillo inferior	Movimientos en pasillos completos	Movimientos en pasillos abordados por superior	Movimientos en pasillos abordados por inferior	Costo por ergonomía
3 x 14	2 x 3	2 x 6	(2 x 2) + (3 x 2)	(2 x 1)	(1.6 x 2) + (1.3 x 5) + (1.0 x 6)
42	6	12	10	2	15.7
	Costo Total			87.7	

5.2.2 Instancias de validación.

Para esta etapa, las instancias fueron generadas siguiendo el mismo procedimiento descrito previamente, utilizado también por los autores. Se consideraron instancias pequeñas descritas en la **Tabla *18***.

Tabla 18

Parámetros para instancias de validación

Parámetro	**Valores**
Numero de pasillos m	{3, 4}
Numero de posiciones de recolección por pasillo n	{6, 10}
Numero de SKUs en lista de recolección a	{3, 7}
Factor de duplicación de SKUs α	{1, 5}

Cabe resaltar que en cuanto a los niveles de las estanterías se mantuvo siempre 3 niveles acorde a lo descrito en la descripción del problema. Para cada combinación de parámetros fueron generadas múltiples instancias independientes utilizando diferentes semillas aleatorias, con el fin de evaluar la robustez del algoritmo frente a cambios en la disposición del inventario y la demanda. Cada configuración fue evaluada sobre cinco instancias diferentes, y para cada instancia el algoritmo fue ejecutado cinco veces con diferentes semillas.

5.2.3 Configuración del algoritmo.

La configuración del algoritmo genético empleada durante todo el proceso de validación fue la configuración que mejor desempeño demostró en la etapa de calibración de parámetros. Esto en aras de evitar sesgos y garantizar coherencia metodológica. Los parámetros pueden encontrarse en la **Tabla *19***. El elitismo es empleado como una proporción de la población, garantizando que por lo menos un individuo de la mejor calidad se preserve en cada generación.

Tabla 19

Configuración del algoritmo para la validación.

Parámetro	Valor	Parámetro	Valor
Tamaño de población	150	Probabilidad de mutación	0.2
Numero de generaciones	500	Tamaño del torneo	5
Probabilidad de cruce	0.7	Elitismo	5%

5.2.5 Resultados de la validación.

La **Tabla *20*** muestra los resultados de la validación del algoritmo genético propuesto, comparando las soluciones obtenidas con los valores óptimos calculados mediante el modelo exacto. Se reportan métricas de calidad de solución, como el gap promedio, gap mediano y gap máximo, métricas de estabilidad como la tasa de alcance del optimo, y una métrica de desempeño computacional tiempo promedio del algoritmo genético.

Los resultados obtenidos permiten ver que el algoritmo genético propuesto genera soluciones factibles en el 100% de las ejecuciones, ya que la representación de los individuos y los operadores genéticos fueron diseñados de forma que el algoritmo genético explora únicamente el espacio de soluciones factibles, corroborado en la etapa de validación, donde no se observaron ejecuciones con soluciones inviables.

En cuanto a la calidad de las soluciones, los resultados presentados en la **Tabla *20*** muestran que el algoritmo alcanza soluciones de alta calidad en todas las configuraciones evaluadas. Mientras las instancias se mantienen más simples, con menor número de SKUs y menor factor de duplicación, el gap promedio de optimalidad se mantiene bajo, logrando incluso reproducir el valor optimo en una gran parte de las ejecuciones. Sin embargo, incluso a medida que la complejidad

del problema aumenta – mayor número de SKUs y mayor grado de duplicación significan un mayor espacio de búsqueda – los valores del gap permanecen en rangos moderados.

Tabla 20

Resultados de validación.

m	n	α	a	Gap promedio	Gap mediano	Gap máximo	Tasa de alcance máximo	Tiempo promedio (s) AG
3	6	1	3	15.81 %	6.45 %	34.78 %	20%	0.34
3	6	1	7	12.33 %	0.00 %	35.56 %	60%	0.41
3	6	5	7	0.00 %	0.00 %	0.00 %	100%	0.48
3	10	1	3	0.00 %	0.00 %	0.00 %	100%	0.30
3	10	1	7	6.10 %	0.00 %	27.12 %	60%	0.41
3	10	5	3	29.13 %	35.71 %	55.56 %	0%	0.34
3	10	5	7	5.46 %	2.44 %	11.32 %	28%	0.46
4	6	1	3	11.64 %	0.00 %	40.00 %	60%	0.35
4	6	1	7	9.00 %	10.17 %	12.50 %	0%	0.65
4	6	5	3	18.98 %	13.04 %	62.50 %	20%	0.46
4	6	5	7	0.18 %	0.00 %	4.55 %	96%	0.66
4	10	1	3	8.78 %	0.00 %	24.39 %	60%	0.38
4	10	1	7	4.01 %	0.00 %	14.49 %	60%	0.46
4	10	5	3	43.36 %	38.10 %	80.00 %	0%	0.40
4	10	5	7	4.61 %	0.00 %	34.21 %	52%	0.60
6	6	1	3	7.46 %	3.92 %	17.14 %	40%	0.44
6	6	1	7	21.50 %	21.82 %	49.23 %	20%	0.55
6	6	5	3	4.60 %	2.78 %	13.33 %	20%	0.48
6	6	5	7	6.33 %	2.27 %	28.57 %	48%	0.66
6	10	1	3	4.71 %	0.00 %	23.53 %	80%	0.38
6	10	1	7	12.50 %	2.53 %	52.17 %	40%	0.55
6	10	5	3	11.87 %	8.70 %	47.22 %	24%	0.51
6	10	5	7	8.86 %	1.52 %	34.09 %	48%	0.70

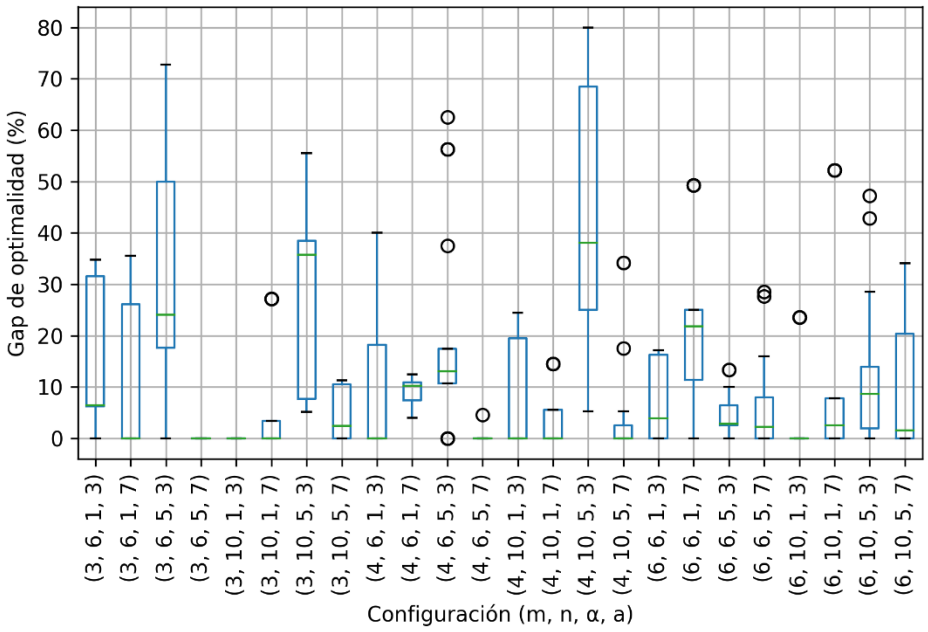
La tasa de alcance del optimo y la distribución del gap de optimalidad, ilustrada en la **Figura 21** , permiten analizar la estabilidad del algoritmo genético. Para las configuraciones de menor complejidad, el algoritmo alcanza el óptimo en una proporción importante de las ejecuciones, mostrando que el algoritmo genera soluciones de buena calidad de una manera

consistente. En configuraciones más complejas la tasa de alcance del optimo se ve reducida, sin embargo, en este contexto resulta relevante ver que la mayoría de las ejecuciones se concentran en valores de gap reducidos, mostrando estabilidad.

Respecto al tiempo computacional, se encuentra que el algoritmo tiene tiempos de ejecución promedio inferiores a medio segundo por ejecución en todas las configuraciones de validación, significativamente mayor que el tiempo requerido para estas instancias pequeñas por el modelo exacto. Este comportamiento resulta relevante considerando que el algoritmo ejecuta un numero fijo de generaciones y que, en cada ejecución, incorpora una búsqueda local.

Figura 21

Distribución del gap de optimalidad del AG

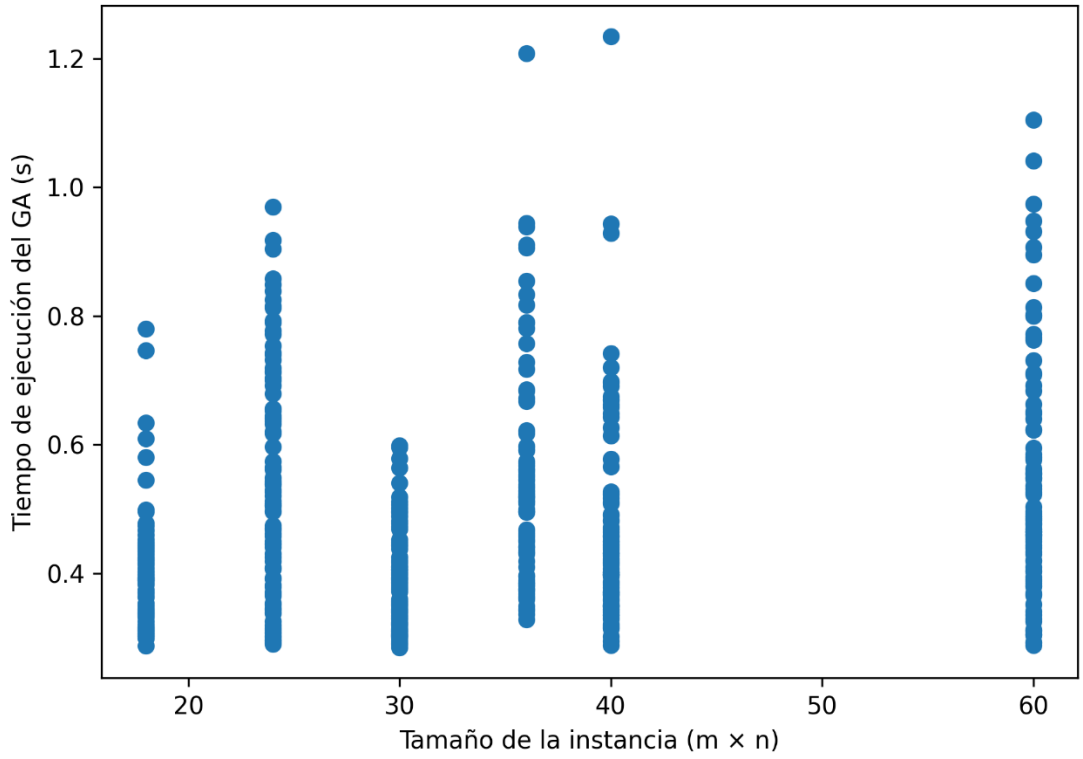


La **Figura 22** permite ver crecimiento moderado entre el tamaño de la instancia – número de pasillos por número de posiciones por pasillo – y el tiempo de ejecución del algoritmo. Por tanto, el algoritmo presenta un comportamiento computacional adecuado, ya que su tiempo de ejecución es coherente con el aumento de la complejidad estructural del problema.

Los resultados de la validación confirman que el algoritmo genético propuesto se encuentra correctamente implementado y calibrado. El algoritmo también muestra capacidad de generar soluciones factibles y de alta calidad de manera consistente, con gaps controlados y tiempos de ejecución reducida. Una disminución en la tasa de alcance optimo respecto a un incremento en la complejidad del problema es coherente con la naturaleza del método de solución. A partir de este análisis, se concluye que el algoritmo genético constituye una estrategia valida y robusta para abordar el problema de ruteo del recolector considerado.

Figura 22

Tiempo de ejecución del AG vs tamaño de la instancia



5.3 Evaluación del algoritmo genético

En esta sección se busca evaluar el desempeño del algoritmo genético propuesto para resolver el problema de ruteo de un recolector en un almacén de comercio electrónico con

almacenamiento disperso y penalizaciones por ergonomía, considerando tanto el modelo exacto como el algoritmo genético híbrido desarrollado. Particularmente, se busca analizar:

- La calidad de las soluciones obtenidas por el algoritmo genético respecto al modelo exacto.
- El tiempo computacional empleado por ambos enfoques para resolver diferentes configuraciones de instancias.
- La estabilidad del algoritmo genético frente a la variabilidad inducida por el uso de semillas aleatorias.
- El comportamiento de ambos métodos de solución según la complejidad del problema.

5.3.1 Diseño experimental

Para el desarrollo de la evaluación se caracterizó la configuración del almacén, la distribución de SKUs, y la lista de recolección, generando un total de 108 instancias diferentes definidas por variaciones en los parámetros expuestos en la **Tabla *21***. Cada combinación de estos parámetros define una instancia única, y en aras de garantizar la reproducibilidad de los experimentos, a cada instancia se asignó una semilla fija empleada en la generación aleatoria de los datos.

Las instancias implementadas en el diseño experimental siguen la misma lógica de generación propuesta por Goeke & Schneider (2021), adaptadas para incluir los niveles de recolección y los costos ergonómicos, tal como se describe en la sección 5.1.

Tabla 21

Parámetros instancias evaluación

Parámetro	Valores
Pasillo (m)	{20, 50, 100}
Posiciones por pasillo (n)	{60, 120, 180}
SKUs distintos en lista de recolección	3, 7, 15, 30
Factor de duplicidad Alpha	5, 10, 40
Niveles k	3

5.3.2 Métodos de solución evaluados

Cada una de las instancias construidas fueron resueltas desde dos enfoques diferentes: (i) el enfoque exacto, mediante la implementación del modelo de optimización y su resolviendo este mediante el optimizador Gurobi, con un límite de 30 minutos por instancia, y (ii) el enfoque metaheurístico que busca soluciones aproximadas en tiempos computacionales racionales mediante el algoritmo genético híbrido propuesto; dicho algoritmo aborda cada instancia en cinco corridas utilizando diferentes semillas aleatorias.

5.3.3 Métricas de evaluación

Ambos métodos de solución fueron evaluados según las siguientes métricas:

- Valor mínimo de la función objetivo encontrado por cada método, que incluye tanto el costo total de los desplazamientos como las penalizaciones por ergonomía.
- Tiempo computacional.
- GAP relativo, calculando respecto a la solución óptima.

- Promedio del valor de la función objetivo encontrado a través del algoritmo genético sobre las cinco corridas de cada instancia.
- Desviación estándar del valor de la función objetivo obtenida por el algoritmo genético.

5.3.4 Resultados Modelo exacto

Los resultados del costo y tiempo obtenido en cada instancia mediante el modelo de optimización resuelto en Gurobi se ven reflejados en la **Tabla *22***. Es de resaltar que el modelo exacto encontró la solución óptima en todas las instancias bajo el límite de 30 minutos de ejecución, lo cual permite usar el modelo exacto como referencia de optimalidad para la evaluación del algoritmo genético.

En cuanto al tiempo computacional, el análisis de los tiempos de ejecución revelo que el modelo es rápido en cuanto a las instancias pequeñas, con tiempos de ejecución menores a 1 segundo en todas las instancias pequeñas e incluso algunas instancias medianas. Sin embargo, es evidente un decrecimiento en el desempeño computacional a medida que las características de las instancias incrementan el espacio de soluciones y la complejidad del problema. En la instancia más grande el tiempo de ejecución se elevó hasta los 800 segundos, evidenciando las limitaciones prácticas de este enfoque para instancias grandes y coherencia con la naturaleza NP-hard del problema.

Tabla 22

Resultados método exacto

	a = 3		**a = 7**		**a = 15**		**a = 30**	
(m, n)	**Costo**	**Tiempo (s)**	**Costo**	**Tiempo (s)**	**Costo**	**Tiempo (s)**	**Costo**	**Tiempo (s)**
α = 5								
(5, 30)	65.1	0.19	93.7	0.07	101.6	0.14	190.7	0.10
(5, 60)	85.8	0.19	159.5	0.14	272.9	0.12	231	0.13
(5, 180)	108.5	0.31	176.7	0.43	325.1	1.51	555.4	0.50

(25, 30)	77.8	0.27	189.2	0.56	243.8	0.75	420.8	0.62
(25, 60)	169.2	1.06	265.2	0.76	361.5	1.21	542.3	0.77
(25, 180)	106.8	1.55	509	1.82	898.8	1.63	1102.7	2.57
(100, 30)	219.1	0.95	316.5	3.26	357.5	1.00	808.9	1.70
(100, 60)	340.5	2.18	522.4	1.94	729.8	8.69	1051	11.97
(100, 180)	1293.4	12.91	710.7	26.19	1855.9	12.43	1779.8	87.98

	a = 3		**a = 7**		**a = 15**		**a = 30**	
(m, n)	**Costo**	**Tiempo (s)**	**Costo**	**Tiempo (s)**	**Costo**	**Tiempo (s)**	**Costo**	**Tiempo (s)**
α = 10								
(5, 30)	42.5	0.14	62.5	0.09	101	0.08	205.6	0.06
(5, 60)	60.5	0.23	111.3	0.43	196	0.10	209.7	0.15
(5, 180)	183.2	0.30	472.8	0.35	359.4	2.57	427.2	0.50
(25, 30)	144.8	0.20	89.5	0.29	242.2	1.10	339.9	0.98
(25, 60)	61.8	0.38	133.7	2.29	251.8	1.24	400.2	0.56
(25, 180)	210.6	2.13	440.6	4.58	497.2	3.18	994.3	2.60
(100, 30)	137.9	2.96	241.4	2.03	245.2	2.35	631.4	9.18
(100, 60)	390.2	4.15	284.4	5.23	500.7	10.47	979	19.46
(100, 180)	238.5	7.94	887.9	9.96	1031.4	40.44	1087	236.24

	a = 3		**a = 7**		**a = 15**		**a = 30**	
(m, n)	**Costo**	**Tiempo (s)**	**Costo**	**Tiempo (s)**	**Costo**	**Tiempo (s)**	**Costo**	**Tiempo (s)**
α = 40								
(5, 30)	15.2	0.09	56.8	0.15	161.6	0.06	197.9	0.06
(5, 60)	37.3	0.26	186.3	0.12	185.4	0.17	358.3	0.19
(5, 180)	13	0.42	75.5	0.61	493.2	0.63	500.1	0.95
(25, 30)	13.5	0.26	102.5	2.12	188.3	0.55	289.6	0.78
(25, 60)	116.3	2.85	139.6	2.14	188.1	2.48	307.8	7.56
(25, 180)	74.8	2.50	88.4	2.24	231.9	17.15	527.6	57.29
(100, 30)	72.6	2.81	139.3	6.27	206.5	1.22	410.6	9.31
(100, 60)	89.9	4.98	114.3	9.13	371.5	3.48	401.9	70.90
(100, 180)	60.9	9.06	214.6	39.69	629.6	197.56	665.8	800.56

5.3.5 Resultados algoritmo genético

En la **Tabla *27*** se visualiza un resumen de los resultados obtenidos mediante el algoritmo genético. Se reporta el valor promedio de la función objetivo, la desviación estándar, la mejor solución obtenida y el tiempo promedio de ejecución. Cabe mencionar que el valor promedio representa el desempeño típico del algoritmo, a diferencia de la mejor solución que muestra el

potencial máximo del algoritmo. La desviación estándar proporciona una medida de la robustez, ya que evidencia la estabilidad frente a semillas aleatorias. Los resultados completos pueden ser revisados en el Apéndice D.

Debido a que la magnitud del costo del tour de recolección crece significativamente con el tamaño de la instancia, se complementa el análisis de desviación estándar absoluta con el coeficiente de variación CV, que permite comprar la robustez entre instancias de distinta escala.

$$CV = \frac{Desviacion\ estandar\ AG}{Costo\ promedio\ AG}$$

Tabla 23

Estadísticos globales de variabilidad del algoritmo genético

Métrica	Promedio	Mediana	Máximo
Desviación estándar	28.75	17.68	140.54
Coeficiente de variación	5.76%	4.20%	33.96%

La **Tabla *23*** resume la variabilidad global del algoritmo genético. Si bien la desviación estándar absoluta es elevada, la medida de 5.76% en el coeficiente de variación indica que el algoritmo presenta un comportamiento estable frente a diferentes semillas aleatorias, en términos relativos. Analizando la variabilidad promedio según diferentes parámetros (**Tabla *24***, **Tabla *25***, **Tabla *26***), se observa:

- Un aumento en la cantidad de pasillos refleja una mayor sensibilidad al algoritmo a la inicialización, debido a que el coeficiente de variación incrementa.

- Ante un aumento en el número de posiciones por pasillo el coeficiente de variación se mantiene relativamente estable, indicando que el algoritmo genético escala de manera controlada.
- Incrementos en el tamaño de la lista de recolección aumentan la variabilidad del algoritmo genético, No obstante, el coeficiente de variación se mantiene moderado.

Tabla 24

Variabilidad promedio según número de pasillos

m	Promedio desv estándar	CV promedio
5	11.85	3.11%
25	26.88	6.11%
100	47.53	8.05%

Tabla 25

Variabilidad promedio según posiciones por pasillo

n	Promedio desv estándar	CV promedio
30	19.15	6.15%
60	22.37	5.35%
180	44.73	5.77%

Tabla 26

Variabilidad promedio según tamaño de la pick list

A	Promedio desv estándar	CV promedio
3	7.41	3.68%
7	28.81	7.75%
15	36.82	6.11%
30	41.96	5.49%

El análisis de variabilidad muestra que el algoritmo genético tiene un comportamiento robusto en términos relativos, con una variabilidad controlada incluso en las instancias de gran tamaño.

Tabla 27

Resumen resultados algoritmo Genético (5 corridas independientes)

	a = 3				a = 30			
(m,n)	**Costo promedio**	**Desv. Estandar**	**Mejor Costo**	**Tiempo (s)**	**Costo promedio**	**Desv. Estandar**	**Mejor Costo**	**Tiempo (s)**
α = 40								
(5, 30)	74.8	0.00	74.8	0.35	211.26	6.63	204.4	2.00
(5, 60)	132.6	0.00	132.6	0.32	298.06	20.98	282	2.22
(5, 180)	368.9	2.74	366.9	0.31	742.3	17.45	711.1	2.95
(25, 30)	113.08	4.74	104.6	0.49	451.54	25.24	427.1	5.26
(25, 60)	169.2	0.00	169.2	0.41	634.22	21.84	610.5	6.74
(25, 180)	365.9	0.00	365.9	0.33	1460.26	76.07	1363.9	6.12
(100, 30)	275.18	31.35	219.1	0.95	837.86	10.38	828.4	8.16
(100, 60)	426.2	0.00	426.2	1.25	1274.46	52.22	1210.2	12.09
(100, 180)	1717.1	0.00	1717.1	3.18	1932.38	49.64	1877.4	12.75
	a = 3				**a = 30**			
(m,n)	**Costo promedio**	**Desv. Estandar**	**Mejor Costo**	**Tiempo (s)**	**Costo promedio**	**Desv. Estandar**	**Mejor Costo**	**Tiempo (s)**
α = 10								
(5, 30)	74.2	2.68	73	0.42	208.76	6.08	205.6	2.28
(5, 60)	129.74	2.38	128.6	0.50	272.62	20.05	257.6	2.70
(5, 180)	380.6	0.00	380.6	0.36	501.66	70.52	428.4	2.93
(25, 30)	175.62	0.16	175.5	0.73	411.42	22.59	376.2	5.96
(25, 60)	146.68	8.38	132.6	0.45	494.84	38.28	452.2	5.57
(25, 180)	414.32	26.91	387.4	0.62	1334.3	104.28	1207.9	6.79
(100, 30)	137.9	0.00	137.9	0.90	828.54	37.36	777.6	15.31
(100, 60)	406.9	0.00	406.9	1.51	1081.32	42.22	1046.5	14.25
(100, 180)	480.74	29.38	467.6	0.93	1551.18	140.54	1401.7	14.82
	a = 3				**a = 30**			
(m,n)	**Costo promedio**	**Desv. Estandar**	**Mejor Costo**	**Tiempo (s)**	**Costo promedio**	**Desv. Estandar**	**Mejor Costo**	**Tiempo (s)**
α = 40								
(5, 30)	66.3	0.00	66.3	0.60	206.68	0.16	206.5	1.75

(5, 60)	128.64	3.07	126.3	0.53	389.44	8.83	382.8	3.15
(5, 180)	365	0.00	365	0.43	574.76	39.55	534.8	2.79
(25, 30)	69.6	5.37	66	0.77	396.32	19.72	374.9	7.17
(25, 60)	158.52	5.63	156	0.60	408.44	23.98	383.5	6.61
(25, 180)	419.28	12.72	403.9	1.05	880.44	86.76	776.9	8.23
(100, 30)	107.1	34.31	78.6	1.13	721.58	67.94	651.7	13.99
(100, 60)	215.62	12.62	205.3	1.71	885.62	54.60	792.5	14.90
(100, 180)	389.46	17.65	372.6	0.96	1335.18	68.88	1259.1	15.47

5.3.6 Comparación modelo exacto vs algoritmo genético

Mediante una comparación directa entre las soluciones obtenidas por el modelo exacto y el algoritmo genético, se realizó la evaluación de la calidad de las soluciones del algoritmo genético**.** La **Tabla *28*** muestra los resultados de las instancias con valor de duplicidad α de 5 y 3 elementos en la lista de recolección, donde costo exacto es el costo encontrado por Gurobi, AG promedio es el promedio de las corridas del algoritmo genético, AG mejor es la mejor solución encontrada con el algoritmo genético, el GAP promedio y GAP mejor se calculan comparando la solución exacta y la solución promedio y mejor del algoritmo genético respectivamente, El tiempo exacto y el tiempo algoritmo es el tiempo que tomo a Gurobi y al algoritmo genético encontrar sus soluciones . Los resultados completos de todas las instancias evaluadas se presentan en el Apéndice E.

Tabla 28

Comparación soluciones algoritmo genético vs modelo exacto, ejemplo 1

	a = 3							
(m,n)	**Costo exacto**	**AG promedio**	**AG mejor**	**Desv. AG**	**GAP (%) promedio**	**GAP (%) mejor**	**Tiempo exacto**	**Tiempo algoritmo**
α = 5								
(5, 30)	65.1	74.8	74.80	0.00	14.90	14.90	0.19	0.35
(5, 60)	85.8	132.6	132.60	0.00	54.55	54.55	0.19	0.32
(5, 180)	108.5	368.9	366.90	2.74	240.00	238.16	0.31	0.31
(25, 30)	77.8	113.08	104.60	4.74	45.35	34.45	0.27	0.49

(25, 60)	169.2	169.2	169.20	0.00	0.00	0.00	1.06	0.41
(25, 180)	106.8	365.9	365.90	0.00	242.60	242.60	1.55	0.33
(100, 30)	219.1	275.18	219.10	31.35	25.60	0.00	0.95	0.95
(100, 60)	340.5	426.2	426.20	0.00	25.17	25.17	2.18	1.25
(100, 180)	1293.4	1717.1	1717.10	0.00	32.76	32.76	12.91	3.18

El algoritmo genético presenta una diferencia significativa entre el promedio de las soluciones encontradas para una misma instancia y su mejor solución, evidenciando la importancia de realizar múltiples ejecuciones para encontrar soluciones de calidad.

Para valores de duplicidad α de 5 y 10 los GAPS del algoritmo son moderados, incluso cercanos a cero en algunas instancias, un ejemplo de ello se puede ver en la **Tabla *29***

. En cuanto a los casos en que el valor de duplicidad α es 40, aparecen gaps bastante elevados, de modo que en instancias con una longitud de pasillos alta y con alta demanda concentrada la heurística del ruteo eleva el costo de la solución al hacer 2 recorridos completos de un pasillo y el modelo exacto puede encontrar soluciones mucho mejores.

Tabla 29

Comparación soluciones algoritmo genético vs modelo exacto, ejemplo 2

	a = 30							
(m,n)	**Costo exacto**	**AG promedio**	**AG mejor**	**Desv AG**	**GAP (%) promedio**	**GAP mejor**	**Tiempo exacto**	**Tiempo algoritmo**
α = 10								
(5, 30)	205.6	208.76	205.60	6.08	1.54	0.00	0.06	2.28
(5, 60)	209.7	272.62	257.60	20.05	30.00	22.84	0.15	2.70
(5, 180)	427.2	501.66	428.40	70.52	17.43	0.28	0.50	2.93
(25, 30)	339.9	411.42	376.20	22.59	21.04	10.68	0.98	5.96
(25, 60)	400.2	494.84	452.20	38.28	23.65	12.99	0.56	5.57
(25, 180)	994.3	1334.3	1207.90	104.28	34.19	21.48	2.60	6.79
(100, 30)	631.4	828.54	777.60	37.36	31.22	23.15	9.18	15.31
(100, 60)	979	1081.32	1046.50	42.22	10.45	6.89	19.46	14.25
(100, 180)	1087	1551.18	1401.70	140.54	42.70	28.95	236.24	14.82

La **Figura 23** muestra la desviación porcentual entre la mejor solución generada por el algoritmo genético y la solución óptima, según diferentes tamaños de la lista de recolección. La grafica revela desviaciones elevadas, especialmente para tamaños pequeños. Sin embargo, el algoritmo demuestra un comportamiento más estable en instancias más complejas, mostrando la capacidad de generar soluciones competitivas en escenarios de mayor escala. Este comportamiento, aunque parezca contraintuitivo, es coherente con la naturaleza del problema y el funcionamiento del algoritmo.

Estos valores elevados del gap porcentual en instancias pequeñas se deben a que el espacio de soluciones factibles puede ser limitado, y en este contexto, los operadores de cruce y mutación probabilísticos empleados por el algoritmo genético para explorar el espacio de soluciones, combinado con el marco definido por la heurística de construcción de ruta, limita la capacidad del algoritmo de corregir decisiones desfavorables. En contraste, el modelo exacto tiene más libertad en cuando al diseño de la ruta de recolección y también evalúa exhaustivamente el espacio de soluciones factibles.

La **Figura *24*** compara el tiempo de resolución del modelo matemático exacto y del algoritmo genético propuesto, según el tamaño de la lista de recolección. El tiempo de cómputo requerido por el modelo exacto para encontrar la solución óptima se ve aumentado según aumenta la cantidad de elementos a recolectar, mientas que el algoritmo genético mantiene un comportamiento más estable y tiempos de cómputo considerablemente menores, lo que resalta su viabilidad para resolver instancias de gran escala.

Figura 23

GAP porcentual del GA vs tamaño de instancia

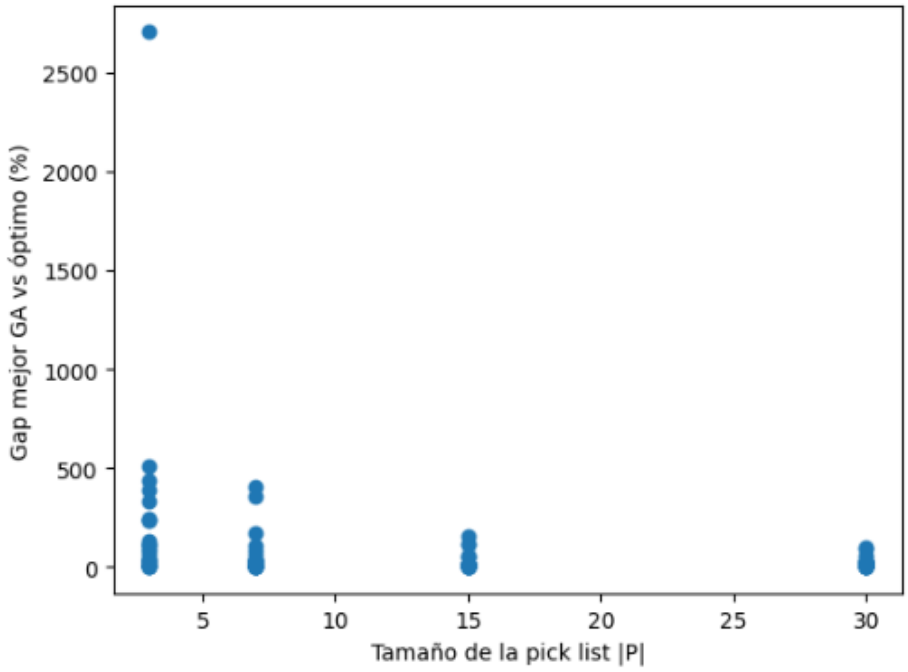
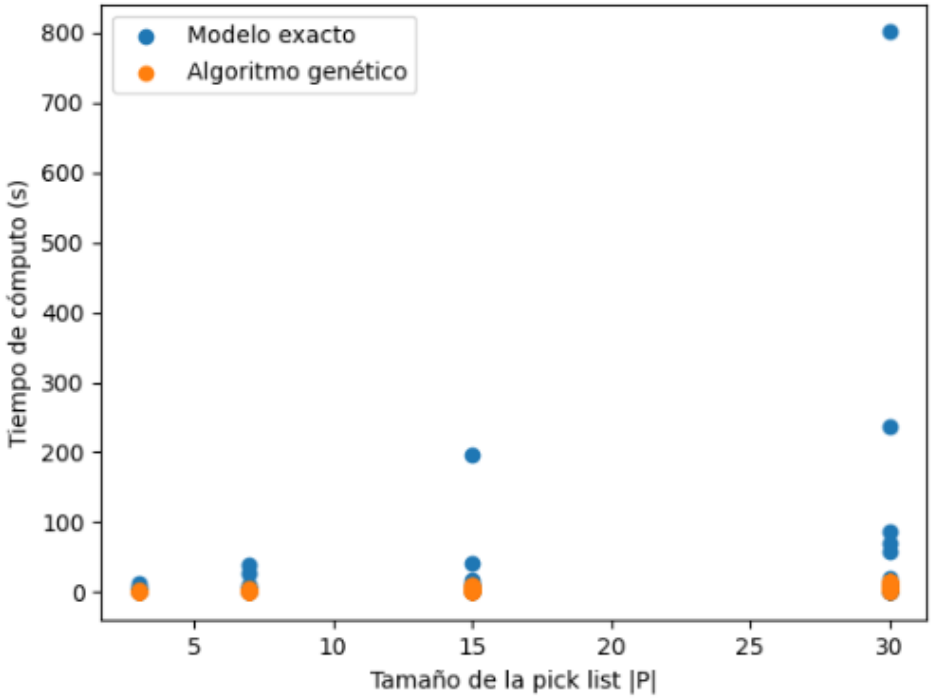


Figura 24

Tiempos de cómputo algoritmo genético vs modelo exacto



5.3.7 Análisis estadístico del desempeño del algoritmo genético

Con el fin de profundizar en el análisis del comportamiento del algoritmo genético, se llevó a cabo un estudio estadístico utilizando diagramas de caja y pruebas de Wilcoxon para datos emparejados. Este enfoque facilita el análisis de la consistencia del algoritmo, la variación en los resultados y su comportamiento según diferentes características del problema. Los análisis se centraron en tres dimensiones principales del experimento: las dimensiones del almacén, la cantidad de SKUs en la orden de recolección y la diversidad de ubicaciones de los SKUs. En el Apéndice G se encuentra el código empleado para realizar las gráficas y el análisis estadístico con Python, y el Apéndice H contiene los resultados del Wilcoxon.

La **Figura 25** presenta el diagrama de caja del GAP promedio en función del tamaño de la instancia (expresado como mxn). Se observa que el algoritmo muestra un comportamiento estable, con GAPs bajos y poca variación, en instancias pequeñas. Sin embargo, según el tamaño del almacén aumenta, se incrementa gradualmente la variabilidad y aparecen datos atípicos, particularmente en configuraciones con pasillos más largos.

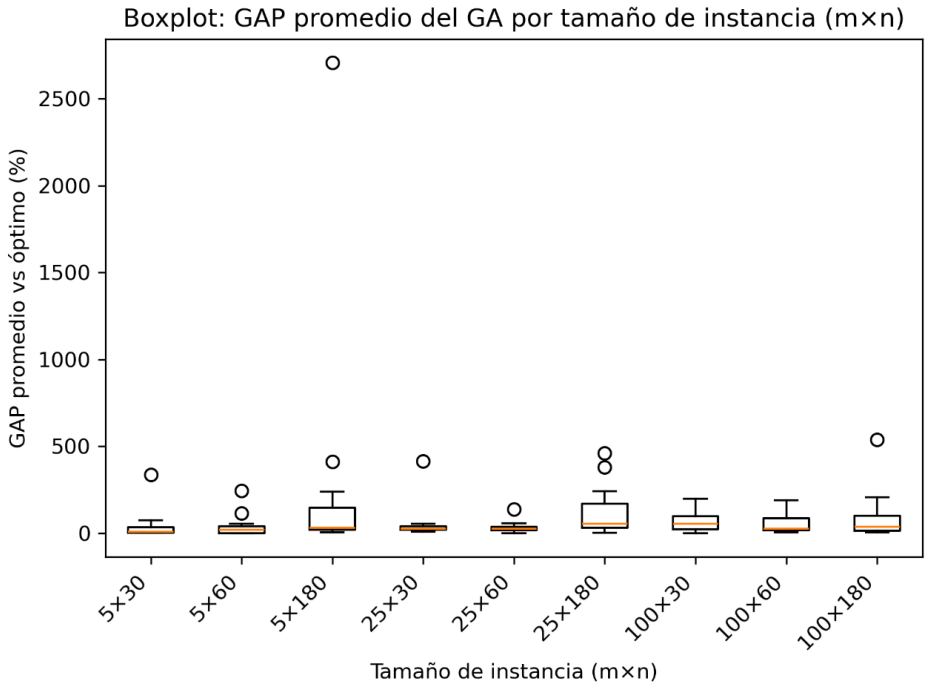
Este comportamiento indica que el crecimiento del tamaño del almacén amplía el espacio de soluciones factibles y dificulta la exploración eficiente del algoritmo genético. Aún así, la mediana del GAP se mantiene en niveles razonables, sugiriendo que el algoritmo conserva una buena calidad promedio de solución incluso en instancias de mayor tamaño.

La **Figura 26** expone el diagrama de caja del GAP promedio del algoritmo genético para distintos números de SKUs diferentes incluidos en la lista de recolección (a). Se evidencia que con listas pequeñas ($a = 3$) hay una alta dispersión del GAP y valores atípicos extremos, sugiriendo una gran sensibilidad a decisiones específicas de ruteo. Según el número de SKUs en

la lista aumenta, el GAP promedio es más bajo y el algoritmo se comporta de manera más estable. Esto puede explicarse en el hecho que las listas de recolección de mayor tamaño imponen una estructura más rígida al recorrido del recolector, reduciendo la influencia relativa de decisiones locales no óptimas.

Figura 25

GAP promedio del GA por tamaño de instancia (mxn)



La prueba de Wilcoxon entre $a = 3$ y $a = 30$, reportada en la **Tabla *30***, confirma diferencias estadísticamente significativas ($p < 0.05$), lo que indica que el número de SKUs en la lista de recolección tiene un impacto relevante en el desempeño del algoritmo genético.

Figura 26

GAP promedio del GA por número de SKUs en la lista de recolección

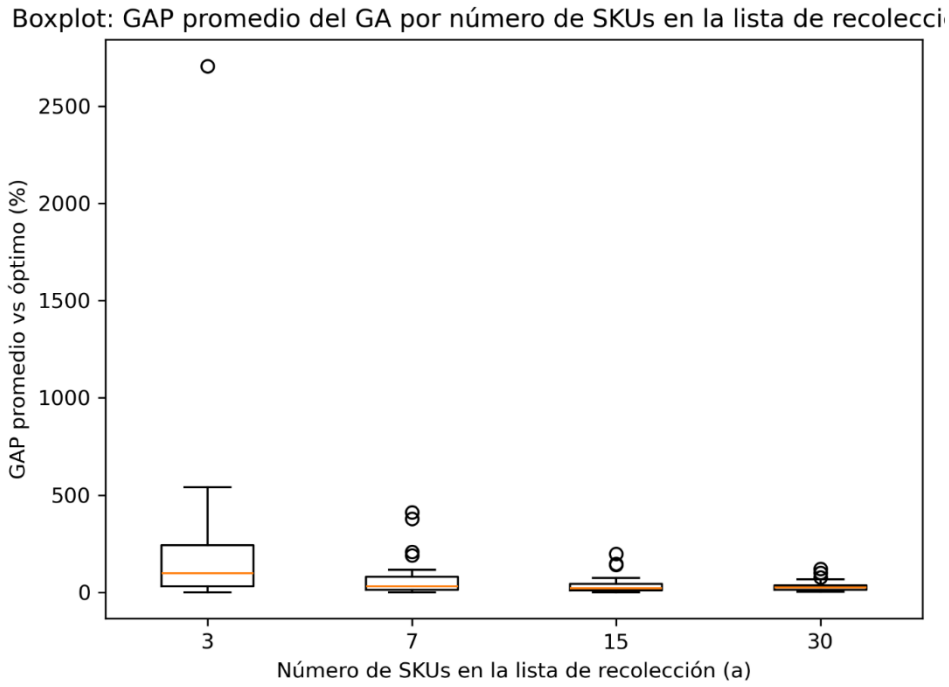


Tabla 30

Resultados Wilcoxon impacto del número de SKUs en la lista (a) sobre el GAP promedio

Comparación	Métrica	Hipótesis nula	Estadístico	p-valor	Conclusión
a = 3 vs a = 30	GAP (%)	No hay diferencia en la mediana del GAP	W = 30	5.49×10^{-5}	Diferencia significativa

El diagrama de caja del GAP promedio en función del factor de duplicación de los SKUs (α) se muestra en la **Figura 27**. Se observa que con valores bajos de α, los GAPs son reducidos y el algoritmo se comporta de manera estable. Sin embargo, al aumentar el factor de duplicación, hay un incremento significativo tanto en la mediana del GAP como en su dispersión.

Este hallazgo indica que la multiplicidad de ubicaciones incrementa notablemente la complejidad combinatoria, ya que el algoritmo debe decidir tanto el orden de visita como la ubicación específica de cada SKU entre varias opciones. En la **Tabla 31** se manifiesta la prueba de Wilcoxon entre valores extremos de α, la cual confirma diferencias estadísticamente significativas, identificando el factor de duplicidad como determinante clave del rendimiento del algoritmo.

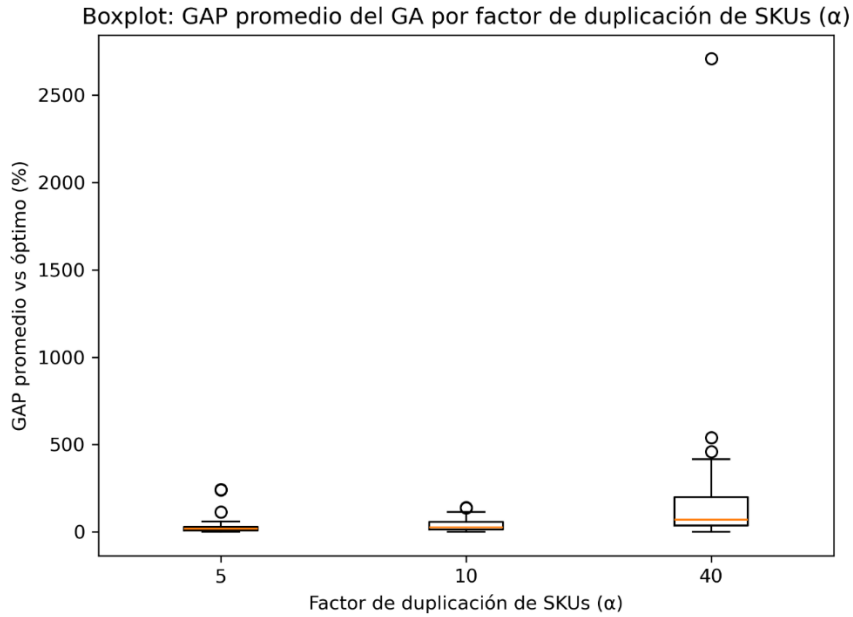
Tabla 31

Resultados Wilcoxon impacto del factor de duplicación (α) sobre el GAP promedio

Comparación	Métrica	Hipótesis nula	Estadístico	p-valor	Conclusión
$\boldsymbol{\alpha}$ = 5 vs $\boldsymbol{\alpha}$ = 40	GAP (%)	No hay diferencia en la mediana del GAP	W = 40	5.92×10^{-8}	Diferencia significativa

Figura 27

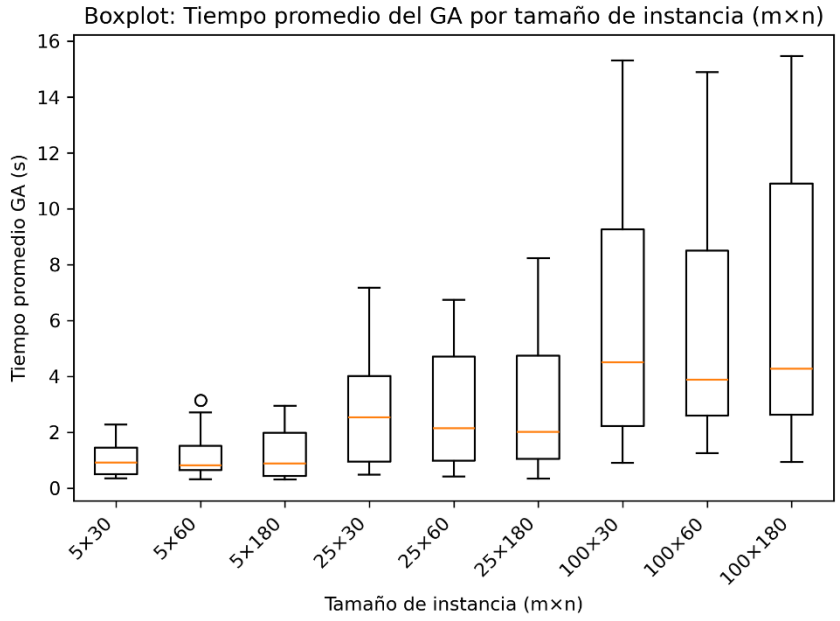
GAP promedio del GA por factor de duplicación de SKUs



La **Figura 28** muestra el tiempo computacional promedio del algoritmo genético en función del tamaño de la instancia. El tiempo computacional promedio aumenta según crece el tamaño del problema, como era de esperarse. Aun así, incluso en las instancias más grandes analizadas, los tiempos promedio permanecen dentro de márgenes operativamente aceptables. Este comportamiento confirma que el algoritmo genético escala adecuadamente y representa una opción viable para instancias de tamaño real.

Figura 28

Tiempo promedio del GA por tamaño de instancia (mxn)



6. Conclusiones

El presente trabajo abordó el problema de ruteo de un recolector en un almacén de comercio electrónico con un solo bloque y depósito, integrando explícitamente consideraciones ergonómicas

asociadas a la recolección en niveles verticales en estanterías multinivel. A lo largo de la investigación se desarrolló un modelo matemático exacto y un algoritmo genético híbrido con el objetivo de estudiar el compromiso entre calidad de solución, eficiencia computacional y aplicabilidad practica en escenarios reales.

Inicialmente, se formuló un modelo matemático con una extensión del problema clásico de ruteo de recolectores, incluyendo penalizaciones ergonómicas asociadas a los niveles de recolección. Esta formulación permite una aproximación más realista a las condiciones operativas de un almacén, al integrar, mediante penalizaciones asociadas a la altura de recolección, una aproximación al esfuerzo físico del recolector dentro de la función objetivo. Los resultados encontrados al buscar soluciones al modelo exacto mediante Gurobi, validan su capacidad de encontrar soluciones óptimas.

En segundo lugar, un algoritmo genético híbrido fue desarrollado y adaptado según la estructura específica del problema, incorporando mecanismos de inicialización heurística, operadores de cruce y mutación especializados, y criterios de selección orientados a la exploración eficiente del espacio de soluciones. La integración de estos componentes permitió brindar soluciones factibles en tiempos computacionales menores que el modelo exacto, especialmente para instancias de mayor tamaño.

En términos del análisis experimental, los hallazgos estadísticos obtenidos a partir de los diagramas de caja y las pruebas no paramétricas de Wilcoxon revelan que el desempeño del algoritmo genético depende considerablemente de la cantidad de SKUs diferentes en la lista de recolección. Puntualmente, se identificó que órdenes con pocos SKUs presentan una mayor fluctuación en cuanto a la calidad de las soluciones, y valores de GAP más dispersos, en contraste con órdenes extensas donde el algoritmo se comporta de una manera más consistente. Este

fenómeno puede explicarse por el hecho que, según aumenta el número de ítems a recolectar, la ruta de recolección adquiere una estructura más definida, reduciendo la sensibilidad del problema a decisiones locales no óptimas.

Complementariamente, el estudio estadístico corroboró que la variabilidad de los SKUs dentro del almacén, expresada mediante el factor de duplicación α, representa un determinante fundamental de la dificultad del problema. El espacio de soluciones crece considerablemente a medida que un mismo SKU puede encontrarse en más posiciones de recolección, reflejándose en un crecimiento tanto del GAP promedio como de la dispersión de los resultados del algoritmo genético. Las pruebas de Wilcoxon validaron la significancia estadística de estas diferencias, permitiendo establecer que la diversidad de ubicaciones de los SKUs influye más intensamente en el rendimiento del algoritmo que las dimensiones físicas del almacén de manera aislada.

Por consiguiente, los resultados indican que el algoritmo genético desarrollado es especialmente apropiado para configuraciones de almacenes con listas de recolección amplias, mientras que su desempeño se ve más desafiado en escenarios donde la variabilidad de ubicaciones de los SKUs es alta.

Cabe mencionar que el algoritmo genético propuesto como método de solución no solo ofrece soluciones en tiempos computacionales razonables, también genera explícitamente una ruta de recolección detallada y ordenada, la cual facilita su interpretación como instrucciones operativas para el recolector, indicando orden de visita de los pasillos, posiciones de recolección a visitar, y los niveles de los estantes correspondientes. Esta característica facilita la implementación práctica del enfoque propuesto en sistemas de gestión de almacenes, permitiendo transformar la solución computacional en acciones claras y ejecutables en el entorno operativo.

7. Recomendaciones

A partir del desarrollo del presente trabajo y de los resultados obtenidos, las siguientes recomendaciones son planteadas, orientadas tanto a la aplicación del enfoque propuesto como a futuras líneas de investigación.

Inicialmente, se recomienda que futuros trabajos incorporen al modelo de optimización extensiones que permitan considerar configuraciones de almacén más complejas, como almacenes multi bloque, múltiples depósitos, múltiples colectores, incluso distintos diseños de almacén. Estas extensiones permitirían analizar el comportamiento del modelo y el algoritmo genético híbrido en escenarios más próximos a la operación real de los almacenes de comercio electrónico.

En segundo lugar, se recomienda evaluar el enfoque propuesto frente a diferentes metaheurísticas reportadas en la literatura, como el recocido simulado, algoritmos de colonia de hormiga o incluso búsqueda tabú. Esto con el fin de realizar comparaciones más amplias en términos de calidad de solución y eficiencia computacional.

Finalmente, se recomienda ampliar el proceso de validación mediante la incorporación de instancias de prueba adicionales o datos basados en escenarios reales, lo cual permitiría realizar un análisis con mayor profundidad sobre la aplicabilidad del enfoque propuesto y su comportamiento bajo diferentes condiciones operativas.

Referencias bibliográficas

Aparicio Macias, L., & Chaparro Quintero, A. V. (2023). *Modelo de Enrutamiento para la recolección de pedidos en un almacén convencional con líneas de dos bloques iguales, mediante los algoritmos heurísticos S-shape y la brecha más grande.* Tesis de grado, Universidad Industrial de Santander, Bucaramanga.

Atashi Khoei, A., Süral, H., & Tural, M. (2023). Energy minimizing order picker forklift routing problem. *European Journal of Operational Research, 307*(2), 604-626. doi:10.1016/j.ejor.2022.08.038

Barthel, M., Faraldi, A., Robnett, S., Darpö, O., Lellouche Tordjman, K., Derow, R., & Ernst, C. (31 de Octubre de 2023). *Winning Formulas for W-Commerce Growth.* The Boston Consulting Group. Obtenido de The Boston Consulting Group.

Bock, S., Boysen, N., Braken, R., & Kroll, T. (2025). The price of safety: Order picking in warehouses with in-house traffic regulations. *IISE Transactions*, 1-15. doi:10.1080/24725854.2024.2434922

Boysen, N., & de Koster, R. (2025). 50 years of warehousing research - An operations research perspective. *European Journal of Operationnal Research, 320*(3), 449-464. doi:10.1016/j.ejor.2024.03.026

Boysen, N., de Koster, R., & Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 396-411. doi:10.1016/j.ejor.2018.08.023.

Cano, J. A., Correa-Espinal, A. A., Gómez-Montoya, R. A., & Cortés, P. (2019). Genetic Algorithms for the Picker Routing Problem in Multi-block Warehouses. *Lecture Notes in*

Business Information Processing. 353, págs. 313-322. Cham: Springer. doi:https://doi-org.bibliotecavirtual.uis.edu.co/10.1007/978-3-030-20485-3_24

Cano, J. A., Cortés, P., Muñuzuri, J., & Correa-Espinal, A. (2023). Solving the picker routing problem in multi-block high-level storage systems using metaheuristics. *Flexible Services and Manufacturing Journal, 35*, 376-415. doi:10.1007/s10696-022-09445-y

Çelik, M., & Süral, H. (2019). Order picking in parallel-aisle warehouses with multiple blocks: complexity and a graph theory-based heuristic. *International Journal of Production Research, 57*(3), 888-906. doi:10.1080/00207543.2018.1489154

Chen, F., Wang, H., Qi, C., & Xie, Y. (2013). An ant colony optimization routing algorithm for two order pickers with congestion consideration. *Computers and industrial engineering, 66*(10.1016/j.cie.2013.06.013), 77-85.

Chen, F., Wang, H., Qi, C., & Xie, Y. (2013). An Ant Colony Optimization Routing Algorithm for Two Order Pickers with Congestion Consideration. *Computers & Industrial Engineering, 66*, 77-85. doi:10.1016/j.cie.2013.06.013

Chen, F., Xu, G., & Wei, Y. (2019). Heuristic routing methods in multiple-block warehouses with ultra-narrow aisles and access restriction. *International Journal of Production Research, 57*(1), 228-249. doi:10.1080/00207543.2018.1473657

Dahiya, C., & Sangwan, S. (2018). Literature Review on Travelling Salesman Problem. *International Journal of Research*, 1152-1155.

de Antonio Suárez, O. (2011). Una aproximación a la heuristica y metaheuristicas. *INGE@UAN, 1*(1), 44-51. Obtenido de https://revistas.uan.edu.co/index.php/ingeuan/article/view/217

De Santis, R., Montanari, R., Vignali, G., & Bottani, E. (2018). An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses. *European Journal of Operational Research, 267*(1), 120-137. doi:10.1016/j.ejor.2017.11.017

De Santis, R., Montanari, R., Vignali, G., & Bottani, E. (2018). An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses. *European Journal of Operational Research, 267*(1), 120-137. doi:10.1016/j.ejor.2017.11.017

Escudero Serrano, M. (2013). *Gestión logística y comercial.* Madrid: McGraw-Hill.

Glover, F. (1990). Tabu Search: A Tutorial. *Interfaces, 20*(4), 74-94. doi:10.1287/inte.20.4.74

Goeke, D., & Schneider, M. (2021). Modeling Single-Picker Routing Problems in Classical and Modern Warehouses. *INFORMS Journal on Computing, 33*(2), 436-451. doi:10.1287/ijoc.2020.1040

Goyal, S. (2010). ASurvey on Travelling Salesman Problem. *Midwest instruction and computing symposium*, 1-9.

Grosse, E. H., Glock, C. H., & Neumann, W. (2017). Human factors in order picking: a content analysis of the literature. *International Journal of Production Research*, 1260-1276. doi:10.1080/00207543.2016.1186296

Gu, J., Goetschalckx, M., & McGinnis, L. F. (16 de Febrero de 2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research, 177*(1),

1-21. Obtenido de https://www-sciencedirect-com.bibliotecavirtual.uis.edu.co/science/article/pii/S0377221706001056?via%3Dihub

Hall, R. (1993). DISTANCE APPROXIMATIONS FOR ROUTING MANUAL PICKERS IN A WAREHOUSE. *IIE Transactions*, 76-87.

Haouassi, M., Kergosien, Y., Mendoza, J. E., & Rousseau, L.-M. (2025). The picker routing problem in mixed-shelves, multi-block warehouses. *International Journal of Production Research, 63*(4), 1304–1325. doi:10.1080/00207543.2024.2374845

Hompel, M., & Schmidt, T. (2007). *Warehouse Management.* Berlin: Springer-Verlag. doi:10.1007/978-3-540-35220-4

Kulak, O., Sahin, Y., & Taner, M. (2012). Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, 52-80. doi:10.1007/s10696-011-9101-8

Lima M., J. Q., & Barán C, B. (2006). Optimización de Enjambre de Partículas aplicada al Problema del Cajero Viajante Bi-objetivo. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, 10*(32), 67-76.

Löffler, M., Schneider, M., & Žulj, I. (2022). Cost-neutral reduction of infection risk in picker-to-parts warehousing systems. *OR Spectrum*, 151-179. doi:10.1007/s00291-022-00695-8

Löffler, M., Schneider, M., & Žulj, I. (2023). Cost-neutral reduction of infection risk in picker-to-parts warehousing systems. *OR Spectrum, 45*, 151-179. doi:10.1007/s00291-022-00695-8

Masae, M., Glock, C. H., & Vichitkunakorn, P. (2020). Optimal order picker routing in a conventional warehouse with two blocks and arbitrary starting and ending points of a tour. *International Journal of Production Research, 58*(17), 5337–5358. doi:10.1080/00207543.2020.1724342

Masae, M., Glock, C. H., & Vichitkunakorn, P. (2020). Optimal order picker routing in the chevron warehouse. *IISE TRANSACTIONS, 52*(6), 665-687. doi:10.1080/24725854.2019.1660833

Masae, M., Glock, C. H., & Vichitkunakorn, P. (2021). A method for efficiently routing order pickers in the leaf warehouse. *International Journal of Production Economics, 234*. doi:10.1016/j.ijpe.2021.108069

Melián Batista, B., Moreno Pérez, J., & Moreno Vega, J. (2009). Algoritmos Genéticos. Una visión práctica. *Números*, 29-47.

Melian, B., Moreno-Pérez, J. A., & Moreno-Vega, J. (2003). Metaheuristicas: Una visión global. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artifical*, 7-28.

Merlano Canoles, K. L., & Castellanos Pico, K. L. (2021). *Un algoritmo de búsqueda tabú para el problema de enrutamiento de un recolector (SPRP) en un almacén de comercio electrónico con almacenamiento disperso y múltiples depósitos.* Tesis de grado, Universidad Industrial de Santander, Bucaramanga.

Osman, I., & Kelly, J. (1996). *Meta-Heuristics: Theory and Applications.* New York: Springer New York. doi:10.1007/978-1-4613-1361-8

Petersen, C. (1997). An evaluation of order picking routeing policies. *International Journal of Operations and Production Management, 17*(11). doi:10.1108/01443579710177860

Ratliff, D., & Rosenthal, A. (1983). Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research, 31*(3), 507-521. doi:10.1287/opre.31.3.507

Real Academia Española. (s.f.). *Heurística*. Obtenido de Diccionario de la lengua española: https://dle.rae.es/heur%C3%ADstico

Real Academia Española. (s.f.). *Logística*. Obtenido de Diccionario de la lengua española: https://dle.rae.es/log%C3%ADstica

Roodbergen, K., & De Koster, R. (2001). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research, 39*(9), 32 - 43. doi:10.1016/S0377-2217(00)00199-8

Sáenz Hoyos, E. S. (2020). *Desarrollo de un modelo matemático para el problema de ruteo e inventario con múltiles depósitos (MDIRP).* Tesis de grado, Universidad Industrial de Santander, Bucaramanga.

Scholz, A., Henn, S., Stuhlmann, M., & Wäscher, G. (2016). A new mathematical programming formulation for the single-picker routing problem. *European Journal of Operational Research, 253*(1), 68-84. doi:10.1016/j.ejor.2016.02.018

Su, Y., Li, M., Zhu, X., & Li, C. (2022). Steiner TSP based on aisle as a unit for order picking. *Computers & Industrial Engineering*. doi:10.1016/j.cie.2022.108026

Su, Y., Zhu, X., Yuan, J., Teo, K. L., Li, M., & Li, C. (2023). An extensible multi-block layout warehouse routing optimization model. *European Journal of Operational Research, 305*(1), 222-239. doi:10.1016/j.ejor.2022.05.045

Weidinger, F. (2018). Picker routing in rectangular mixed shelves warehouses. *Computers and Operations research, 95*, 139-150. doi:10.1016/j.cor.2018.03.012