EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación

y Desarrollo Tecnológico

Tesis de Maestría

Heurísticos con enfoque evolutivo y bioinspirados para el problema de carga en un solo contenedor (BPP3D)

presentada por

José Nicolás Agustín García Martínez

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dra. Andrea Magadán Salazar

Cuernavaca, Morelos, México. Febrero de 2025.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Agradecimientos

TecNM/CENIDET

- Agradezco al CONACYT por el apoyo económico brindado durante todo este proyecto y durante toda la situación de pandemia.
- Agradezco a la Dr. Andrea Magadan Salazar, por su gran apoyo y contribución en el desarrollo profesional, al ser quien continúo como mi asesora en este proyecto hasta poder terminarlo.
- Agradezco a mis sinodales Dr. Raúl Pinto Elías y Dr. Jonathan Villanueva Tavira, por sus enseñanzas y correcciones dadas para este proyecto.
- Agradezco al TecNM Cenidet por permitirme realizar mis estudios en un Centro de investigación de calidad en investigación.
- Agradezco a mis padres Rosalba Martínez Pérez y Antonio García Márquez por el apoyo incondicional durante mi programa completo de posgrado.
- Agradezco al Dr. Gerardo Reyes Salgado, por toda su ayuda y dirección brindada durante el proyecto.
- Agradezco a los compañeros del instituto por estar apoyándonos aún en tiempo de pandemia.

Centro Nacional de Investigación y Desarrollo tecnológico
Departamento de Ciencias Computacionales

Cuernavaca, Mor. 22/ENERO/2025
OFICIO/DCC/020/2025

Asunto: Aceptación de documento de tesis
CENIDET-AC-M14-OFICIO

CARLOS MANUEL ASTORGA ZAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes del Comité Tutorial de **José Nicolas Agustín García Martínez**, con número de control M21CE013, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"Heurísticos con enfoque evolutivo y bioinspirados para el problema de carga en un solo contenedor (BPP3D)"**, y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Conocimiento y Tecnología al Servicio de México"

Dra. Andrea Magadán Salazar
Directora de tesis

Dr. Raúl Pinto Elías
Revisor 1

Dr. Jonathan Villanueva Tavira
Revisor 2

2025 Año de La Mujer Indígena

Centro Nacional de Investigación y Desarrollo tecnológico
Subdirección Académica

Cuernavaca Mor, 22/enero/2025
Oficio No. SAC/024/2025
Asunto: Autorización de impresión de tesis

JOSÉ NICOLAS AGUSTÍN GARCÍA MARTÍNEZ
CANDIDATO AL GRADO DE MAESTRO
EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Heurísticos con enfoque evolutivo y bioinspirados para el problema de carga en un solo contenedor (BPP3D)"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Conocimiento y Tecnología al Servicio de México"

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

c.c.p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/lmz

Interior Internado Palmira S/N, Col. Palmira,
C. P. 62490, Cuernavaca, Morelos Tel. 01 (777) 3627770, ext. 4104,
e-mail: acad_cenidet@tecnm.mx tecnm.mx | cenidet.tecnm.mx

Resumen

El problema de carga en un solo contenedor de tres dimensiones, es un problema complejo logístico y combinatorio clasificado como NP-Duro. Principalmente se enfoca en optimizar el espacio del contenedor para maximizar el volumen de ocupación del contenedor, siguiendo las restricciones como la orientación de los objetos y que estos no pueden estar dentro del otro. En este trabajo se hace un análisis comparativo entre tres metaheuríticas: el Algoritmo Genético Híbrido (AGH), el Algoritmo de Luciérnagas Discreto (ALD) y el Algoritmo de Colonia de Abejas Discreto (ACAD). Estas metehurísticas implementarán técnicas evolutivas. El problema de carga en un solo contenedor de tres dimensiones, es un problema complejo logístico y combinatorio clasificado como NP-Duro. Principalmente se enfoca en optimizar el espacio del contenedor para maximizar el volumen de ocupación del contenedor, siguiendo las restricciones como la orientación de los objetos y que estos no pueden estar dentro del otro. En este trabajo se hace un análisis comparativo entre tres metaheuríticas: el Algoritmo Genético Híbrido (AGH), el Algoritmo de Luciérnagas Discreto (ALD) y el Algoritmo de Colonia de Abejas Discreto (ACAD). Estas metehurísticas implementarán técnicas evolutivas.

En la tesis se desarrolla un sistema generalizado para resolver problemas combinatorios, utilizando una codificación basada en orden para representar soluciones y operadores genéticos adaptados para trabajar en espacios discretos. Los resultados se evalúan en términos de aptitud, que mide la eficiencia del espacio utilizado, y el tiempo de procesamiento.

Además de las contribuciones metodológicas, el estudio proporciona un conjunto de datos que pueden servir como base para futuras investigaciones en problemas combinatorios relacionados. Se identifican oportunidades para explorar otros enfoques, como la incorporación de redes neuronales o la hibridación con otras heurísticas, para extender la aplicabilidad de los métodos desarrollados.

Abstract

The loading problem in a single three-dimensional container is a complex logistical and combinatorial problem classified as NP-Hard. It mainly focuses on optimizing the space of the container to maximize the occupation volume of the container, following restrictions such as the orientation of the objects and that they cannot be inside each other. In this work, a comparative analysis is made between three metaheuristics: the Hybrid Genetic Algorithm (HGA), the Discrete Firefly Algorithm (ALD) and the Discrete Bee Colony Algorithm (ACAD). These meteorologists will implement evolutionary techniques.

In the thesis a generalized system is developed to solve combinatorial problems, using order-based coding to represent solutions and genetic operators adapted to work in discrete spaces. The results are evaluated in terms of fitness, which measures the efficiency of the space used, and processing time.

In addition to the methodological contributions, the study provides a set of data that can serve as a basis for future research on related combinatorial problems. Opportunities are identified to explore other approaches, such as incorporating neural networks or hybridizing with other heuristics, to extend the applicability of the developed methods.

Índice general

Índice de tablas

Índice de figuras

ÍNDICE DE ALGORITMOS

CAPÍTULO 1
INTRODUCCIÓN

El problema de carga de contenedores es un desafío logístico que se refiere a la optimización del espacio dentro de un contenedor para maximizar la cantidad de mercancía que se puede transportar. Este problema se presenta en diversas formas, siendo el Container Loading Problem (CLP) uno de los más comunes, donde se busca acomodar un conjunto de cajas dentro de un contenedor con dimensiones específicas, cumpliendo ciertas restricciones como la no superposición de cajas y el alineamiento de sus caras con las del contenedor.

El problema de carga de contenedores (container loading problem) es probable el problema más importante en tres dimensiones [Bischoff and Marriott, 1990]. Es por ello que aun en la actualidad existen diversos estudios para resolver el problema de carga de contenedores con diferentes técnicas heurísticas [Delorme and Wagenaar, 2024] [Jiao et al., 2024][Bayraktar et al., 2021][Zhao et al., 2020].

El *problema de carga en un solo contenedor* es un problema particular del de carga de contenedores en donde sólo se enfoca en cargar objetos de diferentes tamaños en un sólo contenedor utilizando el máximo espacio posible. El desarrollo del sistema para resolver este problema se generaliza a cualquier problema combinatorio.

Como contribución al estudio de carga en un solo contenedor se desarrolló un análisis comparativo entre las tres metaheurísticas más utilizadas para el problema de carga en un solo contenedor, utilizando diferentes *técnicas evolutivas* en cada una de las metaheurísticas.

1.1 Objetivos

Objetivo general

Describir la utilización de técnicas evolutivas en el problema de carga en un solo contenedor, con y sin rotación, para la obtención de una configuración óptima para su posterior comparación usando diferentes algoritmos con enfoque evolutivo. El desarrollo del sistema para resolver este problema se generaliza a cualquier problema combinatorio.

Objetivos específicos

- Utilizar una heurística de empacado en conjunto con algoritmos metaheurísticos con un enfoque evolutivo.
- Modelo de las técnicas evolutivas para acomodarse al problema de carga en un solo contenedor.
- Aplicar el modelo en una instancia de pruebas generadas.
- Obtener resultados de 100 pruebas en cada técnica evolutiva.
- Interpretación de resultados obtenidos.

1.2 Alcances y limitaciones

1.2.1 Alcances

- El sistema generaliza los problemas combinatorios y puede ser aplicado a cualquiera de estos.
- El sistema utiliza diferentes algoritmos metaheurísticos con diferente enfoque y realiza una transformación del espacio original a un espacio de combinaciones.
- El sistema interpreta movimientos de un espacio tres-dimensional con permutaciones.
- El sistema describe un *dataset* que puede ser usado para experimentos posteriores.

1.2.2 Limitaciones

- El sistema solo utiliza una heurística de empacado, con complejidad temporal del orden O^3.
- El uso de metaheurísticas sobre un espacio discreto aumenta la complejidad temporal a ordenes mayores.
- La obtención de resultados puede tomar tiempo considerable dependiendo de la cantidad de objetos a empacar.
- Trabajar sobre un espacio discreto requiere interpretaciones de movimientos sobre un espacio tres-dimensional.
- La utilización de algoritmos metaheurísticos no garantiza el óptimo global.

1.3 Propuesta de solución

En este proyecto se realizó un análisis comparativo entre tres metaheurísticas bioinspiradas y evolutivas: *Algoritmo Híbrido Genético* [Wang and Chen, 2010a], *Algoritmo de Luciérnagas Discreto* [Ghaisani and Suyanto, 2019] y *Algoritmo de Colonia de Abejas Discreto* [Bayraktar et al., 2021].

Se desarrolló un sistema general para cualquier algoritmo metaheurístico que trabaje sobre un espacio de soluciones de permutaciones y con operaciones de combinación que se interpretan como operadores evolutivos.

1.4 Justificación

Dado que el problema de carga de contenedores es utilizado en amplias áreas de la industria, además de ser un problema combinatorio dentro del conjunto de problemas NP-duro cuya complejidad es exponencial, el desarrollo de algoritmos metaheurísticos con enfoque evolutivo contribuye tanto a la logística industrial como al desarrollo teórico y experimental de nuevas técnicas derivadas.

La técnica de la *clave aleatoria sesgada* (*biased random-key*) [Gonçalves and Resende, 2011] es la más utilizada para transformaciones de un espacio combinatorio a uno de números real, esto impide que se apliquen técnicas evolutivas por trabajar en un espacio diferente. No hay muchas investigaciones que interpreten metaheurísticas bioinspiradas que utilicen técnicas evolutivas a la vez, ya que el enfoque actual es utilizar redes neuronales [Chien et al., 2024][Jin et al., 2024].

Al trabajar con un problema combinatorio, el desarrollo de un sistema generalizado también contribuye a diferentes problemas, además del *Bin Packing 3D* como por ejemplo [Martello et al., 2000]:

- Problema de contenedores
- Problema del vendedor viajero.
- Problema mochila.
- Problemas de programación estocástica.
- Problema de secuenciación de trabajos.
- Entre otros problemas que tengan un espacio de soluciones definido con combinaciones.

1.5 Estructuración de la tesis

La tesis se estructura iniciando con el capítulo de introducción, mencionando el problema a abarcar y la propuesta de solución. El segundo capítulo presenta el estado del arte, donde se menciona a todos los trabajos relacionados que se estudiaron para llegar a la propuesta de solución. El tercer capítulo abarca el marco teórico, se menciona la teoría del problema del *Bin Packing 3D*, la representación del espacio de soluciones con rotaciones, los operadores genéticos a utilizar, la heurística de empacado y las metaheurísticas a utilizar. El cuarto capítulo detalla la metodología de la solución, se explica las modificaciones realizadas a los algoritmos bioinspirados seleccionados, el desarrollo del conjunto de datos y el sistema experimental a utilizar. El quinto capítulo muestra los resultados experimentales y el análisis de cada caso. Finalmente, el sexto capítulo se integra de las las conclusiones de la tesis, el logro de los objetivos, aportaciones y trabajo futuro.

CAPÍTULO 2
MARCO TEÓRICO

En este capítulo se detalla la formulación del problema de carga de contenedores, la representación del espacio de soluciones de este problema y como impacta el considerar o no, las rotaciones. Además, se muestra la codificación a utilizar para representar las soluciones y los operadores que se trabajan sobre este espacio. Finalmente, se menciona a la heurística de empacado a utilizar y las metaheurísticas bionspiradas tomadas del estado del arte.

2.1 Problema de carga de contenedores

El **problema de carga de un solo contenedor** (*Single Container Loading, SCL*) es un problema particular del CLP. Sin embargo, dentro del **problema de la carga de contenedores** *(Container Loading Problem, CLP)* es probable que el aspecto más importante sea el tratar el **bin packing** en tres dimensiones [Bischoff and Marriott, 1990] . Esto se debe a que es **problema combinatorio** de clase **NP-duro** [Garey and Johnson, 1990]. Además, el CLP, a partir de los objetos a empacar, tiene dos tipos de subproblema, que puede definirse como:

- **Fuertemente homogéneo**. Un mayor porcentaje de cajas con las mismas dimensiones.
- **Fuertemente heterogéneo**. Mayor cantidad de cajas de diferente tamaño.

En este trabajo, para el *problema de carga de un solo contenedor* se tiene un contenedor C de dimensiones (L, W, H), en donde L es el largo, W es el ancho y H el alto. El origen se encuentra en la esquina inferior. Los objetos a empacar son cajas rectangulares que definen el conjunto de cajas $B = \{b_1, b_2, ..., b_n\}$ con dimensiones (l_i, w_i, h_i), en donde l_i es el largo, w_i el ancho y h_i el

alto de la i-ésima caja. Se definen las direcciones del sistema cartesiano, donde la longitud del contenedor apunta en dirección x, el ancho en y y el alto en z, como se muestran en la Figura 2.1.

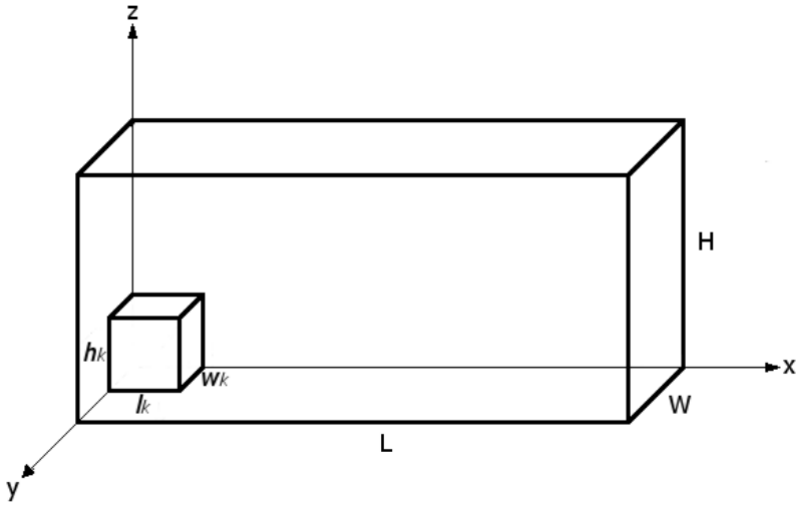


Figura 2.1: Sistema de referencia del contenedor [Karabulut and İnceoğlu, 2004].

De forma similar en [Karabulut and İnceoğlu, 2004], se define el **espacio útil** del contenedor, como la función objetivo a optimizar f, como la división de la suma de los volúmenes de las cajas cargadas en el contenedor entre el volumen del contenedor:

$$f = \frac{\sum_i l_i w_i h_i}{LVH} \tag{2.1}$$

Por lo tanto, se requiere obtener el máximo espacio útil del contenedor:

$$\text{máx } \frac{\sum_i l_i w_i h_i}{LVH} \tag{2.2}$$

Con las restricciones:

A. Cada caja debe entrar completamente al contenedor, paralelo a las paredes.

B. Las cajas no se pueden cruzar (una adentro de la otra).

C. En caso de existir rotaciones, sólo se admiten aquellas admisibles (orientación de la caja).

2.2 Espacio de soluciones

Se trabaja sobre un espacio de soluciones discreto donde las soluciones están representadas por una **secuencia de cajas** a empacar y cada caja tiene una única identificación, [Wang and Chen, 2010a]. Lo anterior implica que la propuesta utiliza **algoritmos fuera de línea (offline)**. En general, las cajas pueden tener dos tipos de rotaciones:

A. **2 - rotación**. Sólo se permite rotar la caja sobre el eje Z y la rotación de regreso a su posición original.

B. **6 - rotación**. Se permiten todas las rotaciones posibles en la caja.

Por lo tanto, el experimento tendrá tres tipos de problemas:

- **Problema sin rotación**.
- **Problema con 2 - rotación**.
- **Problema con 6 - rotación**.

2.2.1 Problema sin rotación ó 0 - rotación

El conjunto del espacio de soluciones es discreto y finito, se puede calcular la cardinalidad del espacio a través de métodos combinatorios. Considerando que el problema abarcado es de tipo **combinatorio**, implica que, si se requiere empacar n cajas rectangulares en un contenedor, el espacio de soluciones contiene todas las permutaciones sin repeticiones posibles representadas en **secuencia de cajas**. Es decir, si se tiene que las cajas no se pueden rotar de ninguna forma, la cardinalidad del espacio de soluciones S es:

$$||S|| = n! \tag{2.4}$$

2.2.2 Problema con 2 - rotación

En caso de que se permita la rotación sobre dos direcciones entonces se tendría, además de todas las permutaciones sin repeticiones posibles, las variaciones de cada caja. La cardinalidad del espacio de soluciones aumenta significativamente proporcional a 2^n. Por lo tanto, la cardinalidad del espacio de soluciones para dos rotaciones es S^{2Rot} es:

$$||S^{2Rot}|| = 2^n \cdot n! \tag{2.5}$$

2.2.3 Problema con 6 - rotación

De manera análoga, si se permite todas las rotaciones posibles sobre una caja, seis rotaciones, entonces la cardinalidad del espacio de soluciones S^{6Rot} es:

$$||S^{6Rot}|| = 6^n \cdot n! \tag{2.6}$$

Es decir, si se pasa de dos rotaciones a seis rotaciones, se aumenta el número de soluciones proporcional a 3^n:

$$||S^{6Rot}|| = 6^n \cdot n! = 3^n \cdot ||S^{2Rot}|| \tag{2.7}$$

En ese trabajo se utiliza cero rotación, dos rotaciones y seis rotaciones para comparar el desempeño de los algoritmos con respecto al volumen del total de cajas empacadas.

2.3 Método de codificación

La **codificación decimal basada en orden con representación diploide** [Wang and Chen, 2010a]. En la Figura 2.2, se muestra la representación diploide, en donde el primer cromosoma indica el orden en que las cajas serán empacadas, y el segundo cromosoma indica el tipo de rotación que tiene la caja. Por ejemplo, en este caso se tiene la posibilidad de *6 rotaciones* por cada caja, por lo tanto, toma valores en el intervalo entero $[0, 5]$.

2	4	5	1	3
0	4	3	1	5

Figura 2.2: Codificación decimal basada en orden

El tamaño de la codificación está dado por el número de cajas que se tiene que empacar en el contenedor.

La **codificación basada en orden** será utilizada para representar las soluciones del espacio de soluciones, las **secuencias de cajas** a empacar.

2.4 Operadores genéticos

Los operadores para utilizar en este trabajo operan sobre la *codificación decimal basada en orden*. En general, se tiene dos tipos de operadores con enfoque genético:

- **Operadores de cruza**: Utilizan dos codificaciones para generar una nueva.
- **Operadores de mutación**: Utiliza una codificación para obtener una nueva.

En esta sección se especifican los operadoses a utilizar en estre trabajo.

2.4.1 Operadores de cruza

En este trabajo se utilizará un solo operador de cruza, ver ejemplo en la Figura 2.3: el operador de **cruza por orden clásico modificado (cruza OX)**. Este operador es utilizado y descrito en [Wang and Chen, 2010b] para obtener dos resultados de resultados de la misma operación.

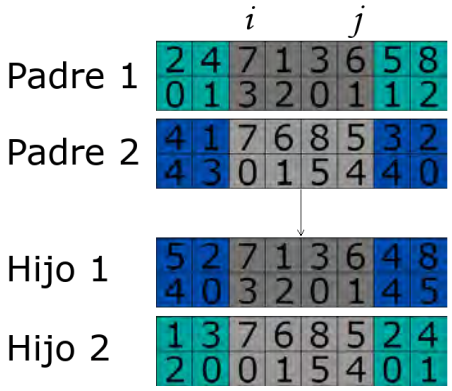


Figura 2.3: Cruza por orden clásico modificado.

En la Figura 2.3 se muestra un ejemplo de la *cruza OX*. El proceso consiste en seleccionar una subsecuencia $(i \rightarrow j)$ en los dos padres, estas subsecuencias se mantienen en cada uno de los hijos respectivamente, el hijo 1 tiene la subsecuencia del padre 1 y el hijo 2 la subsecuencia del padre 2. Después, se llena la secuencia del hijo 1, de manera circular e iniciando en $j+1$, con la secuencia del padre 2. Análogamente, se llena el hijo 2 iniciando en $j+1$ con la secuencia del padre 1

2.4.2 Operadores de mutación

2.4.2.1. Mutación por inversión de doble punto

El proceso consiste en eligir las posiciones i, j [Kıran et al., 2013] de manera aleatoria, se invierte la subsecuencia formada a partir de estas dos posiciones Figura 2.4.

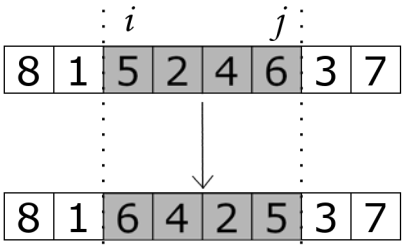


Figura 2.4: Mutación por inversión.

2.4.2.2. Grupo de intercambio (C1)

Es un grupo de tres operadores de mutación que operan con intercambios de subsecuencias. Los operadores están descritos en [Kıran et al., 2013]. Se elige uno de los siguientes operadores de manera aleatoria con igual probabilidad:

- **Intercambio aleatorio *(Random Swap, RS)***. Elije dos posiciones de la secuencia i, j, e intercambia los valores, ver Figura 2.5.

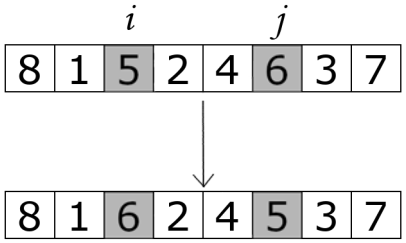


Figura 2.5: Mutación por intercambio aleatorio.

- **Intercambio aleatorio de subsecuencias *(Random Swap of Subsequences, RSS)***. Selecciona dos subsecuencias de manera aleatoria, $(i \rightarrow j)$ y $(k \rightarrow l)$, las intercambia de posición, ver Figura 2.6. En este trabajo, las subsecuencias deben tener el mismo tamaño.

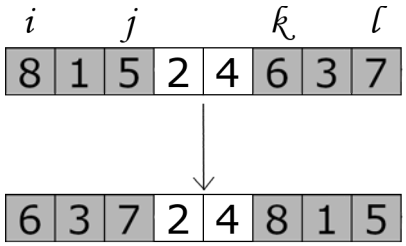


Figura 2.6: Mutación por intercambio aleatorio de subsecuencias.

- **Intercambio inverso aleatorio de subsecuencias *(Random Reversing Swap of Subsequences, RRSS)***. Selecciona dos subsecuencias de manera aleatoria, $(i \rightarrow j)$ y $(k \rightarrow l)$, las intercambia de posición con probabilidad del 0.5 (cada una) que la secuencia este invertida, ver Figura 2.7. En este trabajo, las subsecuencias deben tener tener el mismo tamaño.

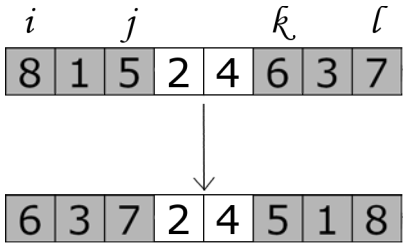


Figura 2.7: Mutación por intercambio inverso aleatorio de subsecuencias.

2.4.2.3. Grupo de inserción (C2)

Es un grupo de tres operadores de mutación que operan con inserciones de subsecuencias. Los operadores están descritos en [Kıran et al., 2013]. Se elige uno de los siguientes operadores de manera aleatoria con igual probabilidad:

- **Inserción aleatoria *(Random Insertion, RI)***. Selecciona aleatoriamente un punto i de la secuencia, después selecciona otro punto j , $j \neq i$. Inserta el punto j a una posición antes del punto i, ver Figura 2.8.

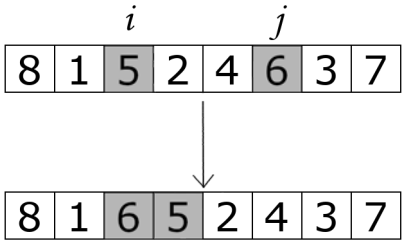


Figura 2.8: Mutación por inserción aleatoria.

- **Inserción aleatoria de subsecuencias *(Random Insertion of Subsequence, RIS)***. Selecciona aleatoriamente un punto k de la secuencia, después selecciona una subsecuencia aleatoriamente $(i \rightarrow j)$ y la inserta detrás del punto k, ver Figura 2.9.

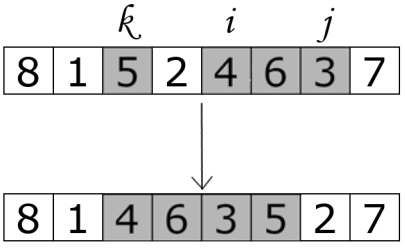


Figura 2.9: Mutación por inserción aleatoria de subsecuencias.

- **Inserción inversa aleatoria de subsecuencias *(Random reversing insertion of subsequence, RRIS)***. Selecciona aleatoriamente un punto k de la secuencia, después selecciona una subsecuencia aleatoriamente $(i \rightarrow j)$, con probabilidad de 0.5 de ser invertida, y la inserta detrás del punto k, ver Figura 2.10.

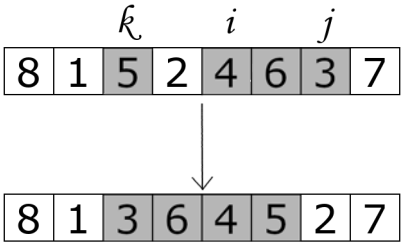


Figura 2.10: Mutación por inserción inversa aleatoria de subsecuencias.

2.4.2.4. Mutación por volteado *(Flip Mutation)*

Selecciona una posición aleatoriamente de la secuencia, elije un valor diferente al de la posición que esté dentro del rango de valores permitidos. En este trabajo, esta mutación sólo se implementa cuando se tienen rotaciones. En la Figura 2.11 se opera sobre las secuencias de 2 rotaciones que toma valores entre $[0, 1]$.

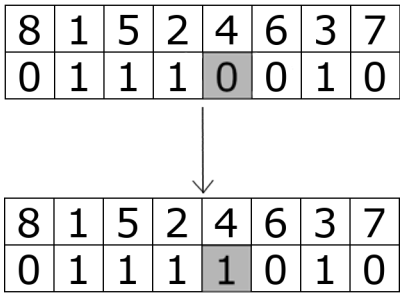


Figura 2.11: Mutación por volteado.

2.5 Heurística Deepest Bottom Left Fill

En el estudio del *bin-packing 3D* se han encontrado diversos métodos para resolverlo pero se pueden dividir en dos: métodos exactos [Martello and Toth, 1990] [Martello et al., 2000] y métodos aproximados [Lodi et al., 2002] [Karabulut and İnceoğlu, 2004]. Este trabajo utilizará métodos aproximados, en particular, un algoritmo genético y dos algoritmos bion-inspirados. Cada

algoritmo utiliza la heurística de empacado DBLF (*deepest bottom left fill*) [Karabulut and İnceoğlu, 2004] y se varían con tres versiones diferentes de cada uno utilizando diferentes operadores de mutación definido en Kıran et al. [2013] para los *algoritmos de colonia de abejas artificial*.

El DBLF sigue siendo una de las heurísticas de empacado más utilizadas en la actualidad [Kim et al., 2021][Bayraktar et al., 2021] para problemas de carga de contenedores donde el conjunto de objetos a utilizar no sobrepasa la cantidad de 150, ya que tiene una complejidad de $O(n^3)$, por lo que las instancias que se utilizarán para la experimentación no contienen más de 150 cajas a empacar.

El Algoritmo 1 es utilizado en el trabajo. Se diferencia de su versión original propuesta en [Karabulut and İnceoğlu, 2004] por la forma de obtener *puntos esquina*. En lugar de utilizar *espacios residuales* como se utiliza en [Karabulut and İnceoğlu, 2004], se ordenan todos los puntos esquinas en orden definido por el *Deepest Bottom Left* (DBL), es decir, el primer punto en la cola de prioridades es aquél que tenga la coordenada x más pequeña, seguido del que tenga la coordenada z más pequeña y finalmente con la coordenada y. Este paso se ve reflejado en la línea 4 del Algoritmo 1, en donde se obtiene el punto de la cola de prioridades y en la línea 10, la actualización de la cola de prioridades.

Algoritmo 1 Deepest Bottom Left with Fill

```
1:  Repetir
2:      Obtener siguiente caja
3:      Repetir
4:          Obtener siguiente posición de cola de prioridades ordenada DBL
5:          Si caja asignada entra en contenedor Entonces
6:              Probar intersecciones con todas las cajas en contenedor
7:              Si no intersección Entonces
8:                  Iterar por DBL
9:                  Insertar caja en posición
10:                 Actualizar cola de posiciones
11:                 Eliminar posiciones infactibles
12:                 Romper
13:     Hasta todas las posiciones han sido probadas
14: Hasta todas las cajas sean visitadas
```

2.6 Metaheurísticas

En esta sección se especifican las metaheurísticas a utilizar sin modificación para el desarrollo del experimento. Las dos últimas metaheurísticas son *bioinspiradas*, esto quiere decir que se basan en un sistema de optimización basado en la manera de organización de animales. Estos fueron inicialmente diseñados sobre un espacio de soluciones continuo, con números reales, a través de varias técnicas se puede llegar a una *discretización* de los algoritmos, y interpretando el significado de movimiento para un espacio de soluciones no continuo.

2.6.1 Algoritmo genético híbrido

Se utiliza un *algoritmo genético híbrido* basado en el trabajo de [Wang and Chen, 2010a], ver Figura 2.12.

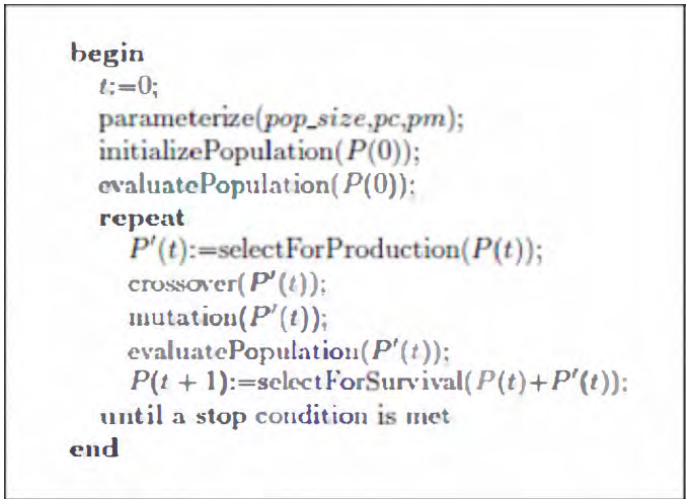


Figura 2.12: Algoritmo genético híbrido de Wang and Chen [2010a].

Se remarca en este algoritmo la manera de crear individuos para la siguiente generación: de los individuos creados en $P'(t)$ son agregados a la actual generación y se eligen los mejores individuos para la siguiente generación de tal forma que se mantenga la población establecida pop_size.

Para este trabajo, se utiliza el siguiente algoritmo sin modificación:

Algoritmo 2 Algoritmo Genético Híbrido

1: $Obtener\ población\ aleatoria,\ con\ 4\ primeros\ individuos\ ordenados \leftarrow P$
2: $Rank(P)$
3: $N \leftarrow |P|$
4: **Repetir**
5: $k \leftarrow Bin(N, \frac{N-1}{N})$
6: **Para** $i \leftarrow 1, k$ **Hacer**
7: $p_1 \leftarrow Torneo(P, pr_s)$, $p_2 \leftarrow Torneo(P, pr_s)$
8: **Si** pr_{cruza} **Entonces**
9: $c_1,\ c_2 \leftarrow CruzaOX(p_1, p_2)$
10: $c_1 \leftarrow Mutar(c_1, pr_{mut})$, $c_2 \leftarrow Mutar(c_2, pr_{mut})$
11: **Si** c_1 no ha sido visitado **Entonces**
12: $P \cup c_1$
13: **Si** c_2 no ha sido visitado **Entonces**
14: $P \cup c_2$
15: $Rank(P)$
16: $P \leftarrow P[: pop]$
17: **Hasta** $CondiciónTérmino()$

El Algoritmo 2 se puede seguir, a través de los siguientes pasos:

A. Se tiene una población P de tamaño N donde sus 4 primeros individuos están ordenados por: *volumen*, *largo*, *ancho* y *alto*.

B. Se ordena la población de la mejor aptitud al peor con $Rank(P)$.

C. En cada generación se elige el número de veces k que se efectuará la cruza por torneo, para decidir esto se optó por calcularlo con la distribución binomial para que en promedio se haga la cruza con la misma cantidad de individuos N en la población. La Figura 2.13 muestra un ejemplo para una población de 50.

D. Se obtiene dos crías de la *cruza OX*

E. Se mutan ambas crías y si estas no han sido visitadas anteriormente, se agregan a la población actual.

F. En la selección de padres mediante torneo no se toman en cuenta las nuevas crías agregadas a la población.

G. Se ordena la población con respecto a su aptitud y se eliminan los individuos sobrantes para

que la nueva población mantenga el número de individuos N.

H. Se itera hasta llegar a una *condición de término*.

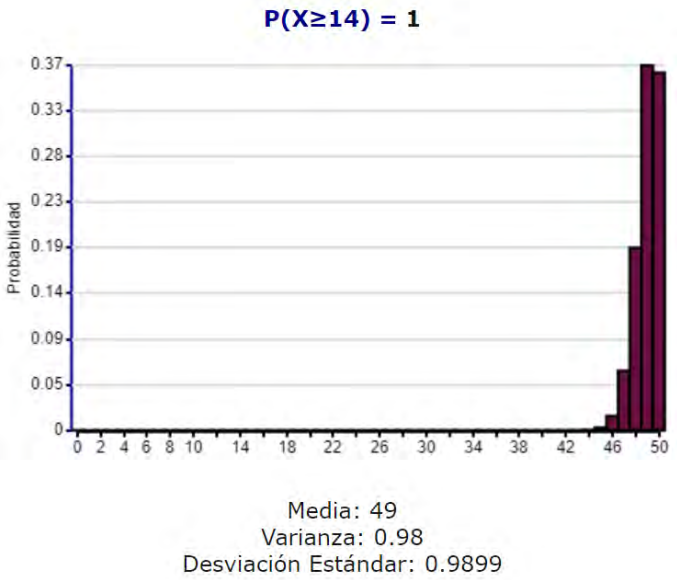


Figura 2.13: Distribución binomial para $N = 50$ y $p = 0.98$

2.6.1.1. Métrica para mutación

La probabilidad de mutación en los algoritmos genéticos en este trabajo es una probabilidad adaptativa Ecuación 2.8, definida en [Mokshin et al., 2020] y se demostró que se llega a mejores soluciones que optando por usar probabilidades de mutación fija. La probabilidad de mutación adaptativa p_m se compone principalmente de la aptitud de los dos padres que efectúan la cruza para el í-ésima cría, f_{1i} y f_{2i}.

$$p_m = 1 - \frac{f_{1i} - f_{2i}}{2}. \tag{2.8}$$

2.6.1.2. Métrica para la condición de término

La *condición de término* se cumple cuando se haya encontrado el individuo con aptitud 1.0 ó si el valor del *grado de convergencia* Ecuación 2.9 se cumple.

Se utiliza el **índice de diversidad** de la Ecuación 2.9 como grado de convergencia para medir

de la población de la actual iteración [Kim et al., 2021]. El índice mide el nivel de diferencia que tienen las soluciones en el conjunto, si su valor es cercano a 1, entonces el conjunto de soluciones está diversificado, en caso contrario implica que las soluciones tienen una aptitudes igual o muy cercanas entre sí.

$$\eta = \frac{f_{best} - f_{worst}}{f_{best}^2}. \tag{2.9}$$

2.6.2 Algoritmo de la colonia artificial de abejas

Se utiliza el algoritmo de abejas definido definido en [Kıran et al., 2013], y los operadores de mutación definido en este trabajo. En el estudio del estado del arte, se obtuvo una modificación del algoritmo de abejas para espacios discretos en [Dereli and Daş, 2011]. Teniendo nuevos parámetros para el algoritmo como: **número de buenos sitios** m, **número de sitios élite** e, **número de abejas élite** nep y **número de abejas no élite** nsp. Estos nuevos parámetros definen un nuevo esquema de movimiento para las abejas, como muestra el Algoritmo **??**.

Algoritmo 3 Algoritmo de Colonia de Abejas Discreto

Entrada m, e, nep, nsp

$Obtener\ población\ aleatoria,\ con\ 4\ primeros\ individuos\ ordenados \leftarrow P$
$pop \leftarrow |P|$
$Rank(P)$
Para $itr \leftarrow 1$, $MaxItr$ **Hacer**
    Para $sitioElite \leftarrow 1$, e **Hacer**
        Para $abejaElite \leftarrow 1$, nep **Hacer**
            $p' \leftarrow Mutar(P[sitioElite], 1)$
            Si $Aptitud(P[sitioElite]) < Aptitud(p')$ **Entonces**
                $P[sitioElite] \leftarrow p'$
    Para $buenSitio \leftarrow e$, m **Hacer**
        Para $abejaNoElite \leftarrow 1$, nsp **Hacer**
            $p' \leftarrow Mutar(P[buenSitio], 1)$
            Si $Aptitud(P[buenSitio]) < Aptitud(p')$ **Entonces**
                $P[buenSitio] \leftarrow p'$
    Para $k \leftarrow m$, pop **Hacer**
        $P[k] \leftarrow SoluciónAleatoria()$
    $Rank(P)$
    Si $Condición$ **Entonces**
        Romper

Al iniciar el algoritmo, se tiene que ordenar la población mediante *ranking*. Las iteraciones se hacen en base a la población actual de flores (sitios) encontradas. Es decir, se hace una búsqueda local (por operador de mutación) con iteración definido por el número de abejas élite. En seguida, se itera sobre las restantes flores que fueron seleccionadas como *mejores sitios* no élite, en donde el número de abejas no élite son las encargadas de explotar las soluciones. Finalmente, las flores restantes son encontradas de manera aleatoria. La solución aleatoria es una permutación aleatoria de la secuencia.

2.6.3 Algoritmo de luciérnagas

El Algoritmo 4 muestra el esquema de movimiento. Se hizo una modificación al algoritmo en donde se ordena al inicio la población de luciérnagas mediante *ranking*. En cada iteración, se encuentra la luciérnaga más *atractiva* sobre la libélula actual. Si se cumplen las condiciones, la libélula actual se aproxima a la libélula más atractiva mediante el operador de cruza, en caso contrario se aplica el operador de mutación, iterando **m** veces en ambos casos, donde m es el **factor de actualización**. La mejor solución sustituye antigua si su aptitud es mayor. En el trabajo [Ghaisani and Suyanto, 2019] cada nueva solución se anexa a una nueva generación olvidando las soluciones de la generación pasada, además sólo se utiliza el operador de mutación para la búsqueda aleatoria. En este trabajo, el movimiento de una libélula a otra se hace mediante un operador de cruza.

Algoritmo 4 Algoritmo de Luciérnagas Discreto

Entrada $\gamma, m, MaxItr$

$Obtener\ población\ aleatoria,\ con\ 4\ primeros\ individuos\ ordenados \leftarrow P$
$Rank(P)$
$pop \leftarrow |P|$
$n \leftarrow TotalCajas$
Para $itr \leftarrow 1, MaxItr$ **Hacer**
    Para $i \leftarrow 1, pop$ **Hacer**
        $p_{atr} \leftarrow EncontrarMásAtractiva(P,\ P[i])$
        Para $j \leftarrow 1, m$ **Hacer**
            Si p_{atr} existe **Entonces**
                $p' \leftarrow Cruza(P[i], p_{atr})$
            Sino
                $p' \leftarrow Mutar(P[i], 1)$
            Si $Aptitud(P[i]) < Aptitud(p')$ **Entonces**
                $P[i] \leftarrow p'$
    $Rank(P)$
    Si $Condición$ **Entonces**
        Romper

La intensidad de la libélula se sigue tomando de la forma:

$$I = \frac{f}{1 + \gamma d^2}, \tag{2.10}$$

en donde d es la distancia Hamming y f la ”aptitud"de la libélula.

Se generaliza γ para escribirla en función del tamaño de la codificación:

$$\gamma = \frac{1}{n}. \tag{2.11}$$

En los trabajos del estado del arte, para ”discretizar” el algoritmo se toma valores definido por el usuario de tal forma que se aproxime a la gaussiana que se utiliza en la formulación continua. Como se utiliza una aproximación a primer orden de la gaussiana, esta aproximación es válida para valores pequeños. Dado n el número de cajas a empacar en el contenedor, se obtiene una aproximación similar para cualquier n. La Figura 2.14 muestra el caso para $n = 100$.

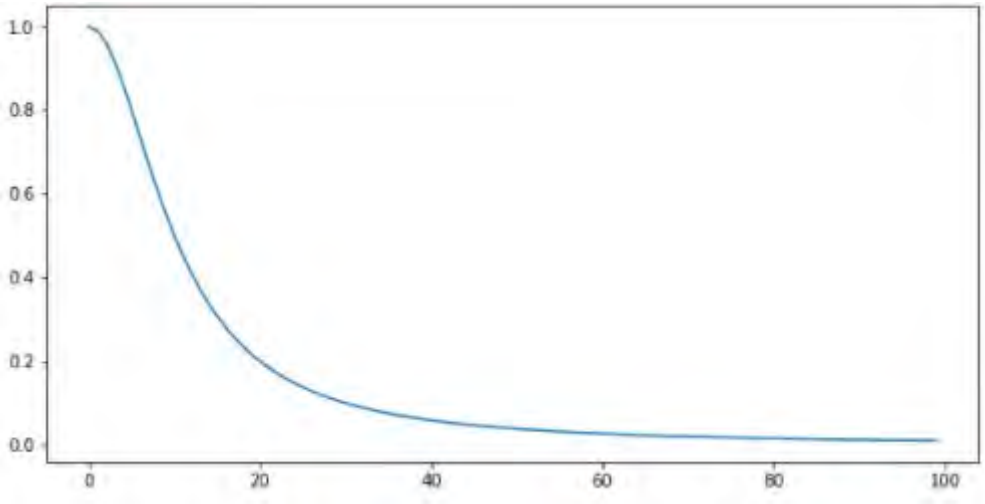


Figura 2.14: Gráfica para γ con $n = 100$

2.6.3.1. Métrica Hamming

En los algoritmos de luciérnagas de este trabajo, la distancia entre dos luciérnagas se mide con la **distancia Hamming**. Esta distancia mide la diferencia de lugares entre dos cadenas, en este trabajo, la diferencia de posiciones entre secuencias. Por ejemplo, en la Figura 2.15 se muestran dos secuencias diferentes, entonces su distancia Hamming es 4 ya que las posiciones $3, 4, 6$ y 7 de las secuencias son diferentes.

8	1	5	2	4	6	3	7

8	1	3	6	4	5	2	7

Figura 2.15: Dos secuencias diferentes.

CAPÍTULO 3
ESTADO DEL ARTE

En este capítulo se presenta el análisis de artículos relacionados con las metaheurísticas que se utilizan en la Tesis: Algoritmo Genéticos Híbridos, Algoritmo de Colonia de Abejas y Algoritmo de Luciérnagas. Además, se presenta la utilidad de los operadores evolutivos y métricas, que se utilizan en el proyecto.

3.1 Antecedentes

En el estudio del *bin packing 3D* se han encontrado diversos métodos para resolverlo pero se pueden dividir en dos: métodos exactos [Martello and Toth, 1990] [Martello et al., 2000] y métodos por heurísticas [Lodi et al., 2002] [Karabulut and İnceoğlu, 2004].

- En los métodos exactos se asegura una *solución óptima* al problema, pero este requiere la utilización de grandes cantidades de tiempo para problemas modernos [Duan et al., 2018].
- Hay alternativas para obtener *aproximación a soluciones óptimas* con las **heurísticas de empacado** se obtienen soluciones cerca al óptimo. Combinando heurísticas de empacado se puede mejorar la solución a una *solución factible* al problema.

Un ejemplo de ello es, desde los años 90's, se ha aplicado la metaheurística evolutiva de algoritmos genéticos al problema de «paletizado» de objetos rectangulares [Kröger et al., 1990]. Por lo que se encuentra una cantidad razonable de investigaciones e hibridaciones del GA aplicado al problema de carga en tres dimensiones. Actualmente, el GA ha pasado a ser utilizado en múltiples problemas, tanto para problemas continuos y discretos. Con el pasar de los años, se han modelado más

algoritmos evolutivos como el **Algoritmo de colonia de abejas artificial** ABC [Karaboga, 2005] y el **Algoritmo de libélulas** FFA [Yang, 2009]; estos algoritmos se basan en casos específicos de *inteligencia de enjambre*, que normalmente superan el rendimiento de los GA para problemas continuos.]

Hay un extenso estudio general del bin-packing 3D, en la literatura se siguen se siguen diversificado los métodos heurísticos e incluso se han combinado para resolver el problema de forma factible [Kanna et al., 2018] [Zhao et al., 2020] [Bayraktar et al., 2021]. Sin embargo, hay pocas investigaciones utilizando metaheurísticas evolutivas aplicadas al problema del bin-packing 3D en la región, esto es, Latinoamérica y México. Aunque para el problema en una dimensión, se pueden encontrar contriciones diversas aplicaciones de las metaheurísticas evolutivas y *frameworks* para ese problema en particular, en los trabajos de [Nieto, 2007], [Alvarado Lara, 2012], [Perez-Ortega et al., 2016], [Escamilla et al., 2017], entre otros.

3.2 Algoritmo Genético Híbrido

En esta sección se analizan los artículos relacionados con el Algoritmo Genético Híbrido que se utiliza en la Tesis, también se detalla la forma de obtención de poblaciones iniciales óptimas a través de 4 heurísticas.

3.2.1 A hybrid genetic algorithm for 3d bin packing problems

[Wang and Chen, 2010b] propone un Algoritmo Genético Híbrido para el problema de Bin Packing 3D, utilizando la codificación decimal en forma de diploide utilizando la heurística DBLF extendido. Además, diseña operaciones de algoritmos genéticos para mejorar el rendimiento del algoritmo en la búsqueda de soluciones óptimas.

Compara dos algoritmos genéticos híbridos con la heurística DBLF y la versión extendida del DBLF. Se muestra en la experimentación que el AGH con la heurística DBLF extendida obtiene mejores resultados que con la no extendida, probando en la experimentación para problemas con 0-rotación, 2-rotación y 6-rotación.

Conclusiones: Se define un AGH con esquema de diploide para la codificación del problema, extiende el DBLF para obtención de soluciones óptimas y se utiliza operador de cruza basado en orden modificado OX sobre diploides.

3.2.2 Heurísticas de agrupación híbridas eficientes para el problema de empacado de objetos en contenedores

[Cruz-Reyes et al., 2012] propone una solución al BPP de una dimensión utilizando un *algoritmo genético híbrido de agrupación*, HGGA-BP. Está inspirado en el esquema de representación de grupos de Falkenauer y aplica operadores evolutivos a nivel de contenedores.

El HGGA-BP contiene estrategias heurísticas eficientes, deterministas y aleatorias, para generar soluciones de calidad. En resultados experimentales se ha demostrado que el HGGA-BP obtiene buenas soluciones en tiempos cortos, superando los mejores algoritmos del estado del arte del trabajo.

Las heurísticas Primer Ajuste, Mejor Ajuste, Peor Ajuste y Best 3-Fit son la base del proceso de generación de individuos en el algoritmo HGGA-BP, las soluciones de estas heurísticas proporcionan individuos de buena calidad. Como se requiere obtener un grupo de individuos diversos, escribieron las técnicas heurísticas en versión aleatoria.

Conclusiones: Se desarrolló un nuevo algoritmo genético híbrido para el Bin Packing Problem de una dimensión, HGGA-BP. La metaheurística híbrida de HGGA-BP conjunta diferentes heurísticas de solución, creando un balance entre exploración y explotación del espacio de búsqueda.

3.2.3 3D heterogeneous bin packing framework for multi-constrained problems using hybrid genetic approach

[Kanna et al., 2018] utiliza algoritmos genéticos (GA) para optimizar el empaquetado tridimensional (3D) de contenedores prismáticos con objetos de formas y tamaños heterogéneos. Se aborda varias restricciones reales del empaquetado, como soporte de carga, estabilidad, orientación, solapamiento, peso y ubicación.

El objetivo principal es maximizar el uso del espacio dentro de contenedores comerciales estándar minimizando el espacio vacío no utilizado. Para lograr esto, se emplearon distintas variantes de GA, incluyendo codificación binaria y decimal, con y sin funciones de penalización, GA restringidos, GA heurísticos y GA híbridos. Además, se implementó un Algoritmo de Ajuste (TA) para mejorar los resultados genéticos rellenando espacios vacíos restantes y convirtiendo la solución en secuencias de empaquetado comprensibles.

Conclusión: [Kanna et al., 2018] propone un enfoque genético híbrido mejorado para resolver problemas bin packing (3D) de objetos con formas y tamaños variados, respetando restricciones reales de empaquetado. Los resultados experimentales muestran que mejoras heurísticas, como la cruza de doble punto, la mutación de orientación y las operaciones de intercambio, amplían el espacio de soluciones. Se logró un patrón de empaquetado factible cumpliendo las restricciones establecidas, y la efectividad de los resultados se optimizó aún más con el Algoritmo de Ajuste (TA). Además, se sugiere que futuras mejoras podrían incluir la consideración de objetos con formas no prismáticas.

3.2.4 Modified genetic algorithm as a new approach for solving the problem of 3d packaging

En [Mokshin et al., 2020] se utiliza una heurística con enfoque evolutivo para el problema del Bin Packing Problem 3D con un contenedor y cajas de diferente tamaño. Compara con 11 heurísticas con enfoque evolutivo con el algoritmo desarrollado *Modified Genetic Algorithm* (MGA). Utilizó un conjunto de 10 datos de entrada creados aleatoriamente para probar que el MGA obtiene mejores resultados que los comparados.

El algoritmo genético modificado (MGA) se basa en los principios del algoritmo genético, sin embargo, cuenta con patrones de comportamiento modificados como la **mutación adaptativa** Ecuación 2.8 y el modelo LBFL (Late-Botton-Front-Left, Late-Lower- Front-Left) para generar la población inicial. El LBFL es un algoritmo imperfecto de ordenación de bloques ordenando 4 niveles de prioridad: tiempo de descarga, altura, profundidad y ancho.

La experimentación consistió en utilizar 10 conjuntos de datos de entrada canónicamente independientes de un total de 1000 cajas con 50 tipos diferentes de cajas con un contenedor de tamaño constante, pero de diferentes dimensiones en cada conjunto de datos. Los resultados mostraron que el MGA obtuvo el mejor desempeño de los 11 algoritmos comparados, tardándose en promedio 1015 minutos con una ocupación del $94.56\,\%$.

Conclusiones: El MGA obtuvo el mejor rendimiento en este artículo por la utilización de la mutación adaptativa en conjunto con el LBFL.

3.2.5 Hybrid Genetic Algorithm for Packing Segments of Decommissioned Nuclear Reactor Components

En [Kim et al., 2021] se utiliza un Algoritmo Genético Híbrido para resolver el problema del acomodo de componente de desechos radioactivos, que tienen forma irregular, usando la heurística DBLf para la decodificación de soluciones.

Se utiliza la codificación decimal en diploide para la representación de soluciones con rotación y el DBLF para el acomodo de los componentes. En este trabajo se utilizó el *índice de diversidad* Ecuación 2.9 como condición de término para el AGH además de llegar al máximo de la capacidad de almacenamiento.

Conclusiones: El AGH propuesto demuestra que, con los datos obtenidos de entrada, los componentes nucleares, se puede llegar alcanzar la máxima capacidad de almacenamiento, $100\,\%$.

3.3 Algoritmo de Colonia de Abejas

En esta sección se presentan artículos relacionados con el Algoritmo de Abejas, que abarca desde la aplicación del algoritmo hasta la discretización de los movimientos de las abejas, incluyendo problemas con soluciones discretas.

3.3.1 An Artificial Bee Colony Algorithm for the 0–1 Multidimensional Knapsack Problem

[Sundar et al., 2010] presenta un algoritmo de colonia de abejas artificiales (ABC) para resolver el problema de la mochila multidimensional 0-1 (MKP 01). El objetivo del MKP 01 es seleccionar un subconjunto de objetos que maximice el beneficio total, respetando un conjunto de restricciones de capacidad. El algoritmo ABC incorpora operadores de reparación heurísticos y búsqueda local. Los resultados computacionales muestran que el algoritmo ABC no solo produce mejores resultados, sino que también converge más rápido en comparación con otros enfoques basados en enjambres.

Este artículo propone un enfoque basado en colonias de abejas artificiales (ABC MKP) para resolver el problema de la mochila multidimensional 0-1. El método se comparó con otros cuatro enfoques basados en enjambres, y los resultados computacionales demuestran su superioridad. Como trabajo futuro, se planea extender este enfoque al problema de mochilas múltiples.

Conclusiones: La utilización del ABC con operadores de reparación heurísticos y búsqueda local produce mejores resultados que los comparados en el artículo. Estos operadores se interpretan en la tesis como operadores genéticos ya que operan con permutaciones sobre secuencias.

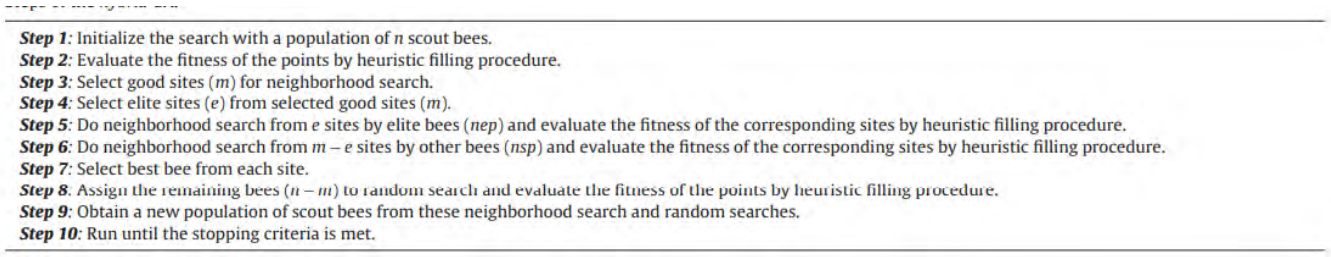
3.3.2 A hybrid 'bee(s) algorithm' for solving container loading problems

En [Dereli and Daş, 2011] se define un algoritmo de abejas híbrido para el problema de carga de contenedores, teniendo un enfoque discreto y definiendo nuevos parámetros clave de movimiento. Estos parámetros clave son: número de abejas exploradoras n, número de sitios seleccionados m, número de sitios de élite e elegidos entre m sitios, número de abejas reclutadas para buscar en e sitios de élite, número de abejas reclutadas para buscar en $m - e$ otros sitios nsp y los criterios de terminación. Estos parámetros del algoritmo se ajustan por ensayo y error, ya que no hay un procedimiento definido para ayudar a los usuarios a elegir el conjunto de parámetros más apropiado.

La Figura 3.1 muestra los pasos del algoritmo donde se implementan operadores de vecindad, operadores que implementan una búsqueda por vecindad, representando el movimiento de las

abejas. Las soluciones aleatorias se implementan generando nuevas perpetuaciones de las soluciones.

Conclusiones: Se desarrolló un algoritmo de abejas híbrido con nuevo esquema de movimiento, para resolver el problema de carga de un contenedor, definiendo una manera óptima y alternativa del algoritmo de abejas introduciendo nuevas variables.



Step 1: Initialize the search with a population of n scout bees.
Step 2: Evaluate the fitness of the points by heuristic filling procedure.
Step 3: Select good sites (m) for neighborhood search.
Step 4: Select elite sites (e) from selected good sites (m).
Step 5: Do neighborhood search from e sites by elite bees (nep) and evaluate the fitness of the corresponding sites by heuristic filling procedure.
Step 6: Do neighborhood search from $m-e$ sites by other bees (nsp) and evaluate the fitness of the corresponding sites by heuristic filling procedure.
Step 7: Select best bee from each site.
Step 8: Assign the remaining bees ($n-m$) to random search and evaluate the fitness of the points by heuristic filling procedure.
Step 9: Obtain a new population of scout bees from these neighborhood search and random searches.
Step 10: Run until the stopping criteria is met.

Figura 3.1: Algoritmo de abejas híbrido [Dereli and Daş, 2011].

3.3.3 The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem.

En [Kıran et al., 2013] se expone el problema del viajero (TSP) utilizando la metaheurística de algoritmo de colonia de abejas con operadores genéticos en lugar de las ecuaciones definidas originalmente. Estos operadores de vecindad se aplican en **codificación basada en orden**, y las agrupa en dos: *punto a punto* y *operación sobre subsecuencias*. Estos operadores se utilizan para hacer una exploración local (vecindario de la abeja). Además, elige una abeja obrera al azar, en lugar de una selección de por ruleta.

En los operadores de vecindad que se utilizaron son: Random swap (RS), Random insertion (RI), Random swap of subsequences (RSS), Random insertion of subsequence (RIS), Random reversing swap of subsequences (RRSS) y Random reversing insertion of subsequence (RRIS). Estos operadores los combina en dos grupos llamándolos “Combinación 1” y “CCombinación 2”.

Conclusiones: El algoritmo discretizado con operadores de vecindad obtuvo mejor rendimiento que los comparados en el trabajo, además de ser una buena solución para el problema del viajero y problemas discretos.

3.3.4 Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem

[Zhong et al., 2017] propone un ABC híbrido discreto adaptado al Problema del Viajante (TSP), que emplea un criterio de aceptación basado en el método de aceptación por umbral. Se introduce una nueva ecuación de actualización de soluciones que permite aprender tanto de otras "abejas" como de las características del problema, con el objetivo de evitar la convergencia prematura. Las abejas empleadas y observadoras usan el criterio de aceptación por umbral para decidir si aceptan nuevas soluciones, asegurando suficiente diversidad en la población.

Los experimentos realizados confirman que:

- La nueva ecuación mejora el rendimiento.
- Es crucial emplear estrategias no codiciosas para mantener la diversidad.
- Diferentes esquemas de selección para abejas observadoras influyen en los resultados.
- Las abejas exploradoras tienen un impacto significativo.

Comparaciones con instancias estándar del TSP muestran que este algoritmo supera a otros basados en ABC y es competitivo o superior a muchos algoritmos de vanguardia.

Conclusiones: Comprueba que el algoritmo de colonia de abejas artificial híbrido discreto con la nueva ecuación de movimiento con esquema discreto obtiene resultados de mayor rendimiento en el trabajo para el TSP.

3.3.5 Effects of Memory and Genetic Operators on Artificial Bee Colony Algorithm for Three-Dimensional Bin Packing Problem

[Bayraktar et al., 2021] utiliza el algoritmo de *Colonias de Abejas Artificial* (ABC, Artificial Bee Colony) para el problema de bin packing de tres dimensiones (3DBPP). Se utiliza mecanismos de memoria y *operadores genéticos* para desarrollar dos algoritmos híbridos ABC.

Se resalta que el algoritmo diversifica la búsqueda aleatoria usando operadores de mutación de

cruza. Se utilizó cuatro tipos de operadores para la selección de camino en la abeja empleada y la abeja en fase espectadora:

A. Operador de cruza.

B. Operador de mutación.

C. Operador de cruza y mutación juntos.

D. Operador de búsqueda aleatoria.

Conclusiones: Se compara dos metaheurísticas: *memory integrated artificial bee colony algorithm (MIACB)* y *genetic operator based artificial bee colony algorithm (GABC)* . En donde el primer algoritmo utiliza una lista tabú para que las abejas tomen aminos óptimos a y no repetir los mismos que puedan llevar a soluciones no factibles, utilizando las ecuaciones para problemas binarios. El segundo algoritmo utiliza operadores genéticos como: cruza, mutación, cruza y mutación; y un operador de búsqueda aleatoria para seleccionar el camino que deben ir las abejas obreras y las espectadoras. Se concluye que el MIACB supera en rendimiento al GABC, y ambos al ABC

3.4 Algoritmo de Luciérnagas

En esta sección se analiza los artículos relacionados a la formulación del Algoritmo de Luciérnagas Discreto, así como la utilización de operadores evolutivos para representar el movimiento de las luciérnagas y el nuevo esquema de movimiento.

3.4.1 A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems

[Sayadi et al., 2010] propone un nuevo método metaheurístico discreto basado en el comportamiento de luciérnagas para minimizar el “makespan” en este tipo de problemas. Los resultados preliminares muestran que este método supera a la técnica de optimización por colonias de hormigas en algunos problemas de referencia conocidos.

Se propone un algoritmo discreto basado en luciérnagas para resolver el problema de

programación en talleres de flujo permutacional, con el objetivo de minimizar el “makespan” (tiempo total de producción). El método fue probado en problemas de referencia de tamaños pequeños y medianos, comparándose con una técnica alternativa de optimización por colonias de hormigas. Los resultados preliminares muestran que el método propuesto ofrece un mejor desempeño. Además, su esquema de solución es fácil de aplicar a otros algoritmos y problemas.

Conclusiones: El algoritmo de luciérnagas discreto ofrece mejor desempeño que los comparados en el artículo. Además, muestra los pasos del algoritmo de libélulas para resolver problemas discretos .

3.4.2 Discrete Firefly Algorithm for Traveling Salesman Problem: A New Movement Scheme

[Jati et al., 2013] desarrolla un Algoritmo de libélulas discreto evolutivo (EDFA) que propone un nuevo esquema de movimiento para resolver el problema de este esquema de movimiento en EDFA. Este nuevo esquema de movimiento garantiza que después de que una luciérnaga se mueva hacia la más brillante, la distancia entre ellas disminuye. Aplica el algoritmo de libélulas discreto evolutivo al problema del viajero.

Se menciona que al no tener una dirección en dicho espacio discreto, utiliza estrategias evolutivas para el movimiento de las libélulas, en donde aplica el operador genético de *mutación por inversión* sobre una misma libélula m veces, y elige la de mayor aptitud.

Conclusiones: Aunque el nuevo movimiento garantiza que un individuo se acerque más a otro más brillante, no garantiza que mejore la aptitud de ese individuo. Por lo tanto, el nuevo DFA debe diseñarse para tener una absorción de luz adecuada de modo que tenga algunos movimientos aleatorios para que la evolución funcione bien.

3.4.3 Firefly Algorithm for Discrete Optimization Problems: A Survey

[Tilahun and Ngnotchouye, 2017] hace una revisión de métodos del algoritmo de luciérnagas para casos discretos, en donde se puede hacer una transformación del espacio discreto al continuo o una formulación discreta. En la formulación discreta, toma la **distancia hamming** como la

distancia entre dos libélulas. Además, divide el movimiento de las libélulas en dos: *paso alpha* y *paso beta*. El paso beta es la atracción de una libélula a otra, en donde

$$\beta = \frac{1}{1 + \gamma \cdot d^2}, \tag{3.1}$$

se interpreta como la probabilidad que una libélula compartirá las mismas entradas con la libélula que se le acerca. El paso alpha es el movimiento aleatorio, en donde sólo se redondea la ecuación que originalmente se usa para problemas continuos. Además, el *parámetro de aleatoriedad* α se escribe en función de la iteración itr y la dimensión del problema n:

$$\alpha = \left\lfloor n - \frac{Itr}{MaxItr} n \right\rfloor. \tag{3.2}$$

Conclusiones: Discretiza los parámetros más importantes del algoritmo de libélulas como la luminosidad discreta de una libélula y el movimiento discreto.

3.4.4 A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy

En [Osaba et al., 2017] se desarrolla un Algoritmo de Libélulas Discreto (Discrete Firefly Algorithm, DFFA) para aplicarlo al Problema de rutas de vehículos asimétricas y agrupadas con recogida y entrega simultáneas de costos variables y caminos prohibidos (AC-VRP-SPDVCFP). Esta aplicación es la primera aplicación del algoritmo de Libélulas al problema de enrutamiento de vehículos enriquecido.

Utiliza la *distancia hamming* para calcular la intensidad de cada libélula, donde la distancia hamming representa la distancia entre dos libélulas. Para el movimiento aleatorio utiliza un operador evolutivo de inserción sobre el valor $n = Rand(2, d_{ij} \cdot \gamma^g)$ que define las veces que se aplicara el operador evolutivo sobre la libélula, en donde d_{ij} es la distancia hamming entre la libélula i y la libélula j, g es el número de iteración y γ la constante que define la intensidad del brillo de una luciérnaga.

El rendimiento del DFA fue comparado con dos técnicas clásicas: el algoritmo evolutivo y el recocido simulado evolutivo. Los resultados muestran que el DFA supera a estas dos metaheurísticas tradicionales, destacándose como una técnica prometedora.

Conclusiones: La iteración del algoritmo del DFFA propuesto en este artículo es el mismo que el FFA original, como se muestra en la Figura 3.2, la diferencia radica cuando se evalúan las intensidades de brillo de las luciérnagas, en caso de que se cumpla la condición de acercamiento se modifica lo siguiente:

A. La distancia se calcula con la distanca hamming en lugar de la distancia cartesiana.

B. El movimiento se define con un operador genético en lugar de una ecuación numérica.

```
1  Initialize the firefly population X = x_1, x_2, ..., x_n;
2  γ = 0.95;
3  for each firefly x_i in the population do
4  |  I_i = fitness(x_i);
5  end
6  repeat
7  |  for each firefly x_i in the swarm do
8  |  |  for each other firefly x_j in the swarm do
9  |  |  |  if I_j < I_i then
10 |  |  |  |  r_ij = HammingDistance(x_i, x_j);
11 |  |  |  |  n = Random(2, r_ij · γ^g);
12 |  |  |  |  x_i = InsertionFunction (x_i, n);
13 |  |  |  end
14 |  |  |  Evaluate new solutions and update light intensity;
15 |  |  end
16 |  end
17 |  Rank the fireflies and find the current best;
18 until termination criterion reached;
19 Rank the fireflies and return the best one;
```

Figura 3.2: Algoritmo de libélulas discreto [Osaba et al., 2017].

3.4.5 Discrete firefly algorithm for an examination timetabling

[Ghaisani and Suyanto, 2019] desarrolla un algoritmo de libélulas discreto alternativo para problemas donde se involucren secuencias de tareas ejecutadas en t tiempo. Modifica la manera de iterar del algoritmo de libélulas discreto introduciendo una nueva variable m de movimiento.

Se diferencia del algoritmo FFA y del DFFA definido en otros trabajos donde sólo se recurre una iteración sobre todas las libélulas, y se encuentra siempre la libélula que más brille para que la libélula en donde se está iterando se acerque a la otra, en caso contrario hace un movimiento aleatorio, mostrado en la Figura 3.3 No hace un *ranking* de las libélulas y al final de cada ciclo, la mejor libélula se queda inmóvil.

Conclusiones: El esquema de movimiento de las luciérnagas en este trabajo no garantiza que después de que una luciérnaga se mueva hacia luciérnagas más brillantes, la distancia entre ellas disminuirá porque la luciérnaga no tiene dirección cuando se mueve. Por lo tanto, el esquema de movimiento de las luciérnagas debe modificarse para garantizar que la distancia entre las dos luciérnagas disminuya después de que una luciérnaga se mueva hacia otras luciérnagas más brillantes.

Input: Initialize population of n fireflies x_i (I = 1, 2,..., n), light absorption coefficient (γ), number of moves (m) and the maximum generation (MaxGeneration).
Output: x_i

```
for i = 1 to n do
    x_i ← Initialize solution
end for
while generation ≠ MaxGeneration do
    for i = 1 to n do
        x_j ← find brightest firefly from all firefly
        Calculate intensity x_i(I_i) and x_j(I_j)
        if (I_j > I_i) then
            Move_firefly(x_i, x_j) m times
            Choose the brightest firefly for the next generation
        else
            Move_random(x_i) m times
            Choose the brightest firefly for the next generation
        end if
    end for
end while
return x_i
```

Figura 3.3: Algoritmo de libélulas discreto modificado [Ghaisani and Suyanto, 2019].

3.5 Tablas de los artículos del Estado del Arte

En la Tabla 3.1 se condensan los artículos del Estado del Arte donde se mencionando el objetivo del trabajo, las técnicas, los resultados y la utilidad para la tesis.

Tabla 3.1: Tabla de los artículos del Estado del Arte.

Artículo	Objetivo	Técnicas	Resultados	Utilidad para la Tesis
Wang and Chen [2010b]	Desarrollar nuevo AGH para BPP3D.	Utilización del DBLF extendido y DBLF. Codificación diploide.	AGH con DBLF extendido da mejores resultados	Codificación diploide para 2-rotación y 6-rotación
Cruz-Reyes et al. [2012]	Desarrollar un algoritmo genético híbrido de agrupación denominado HGGA-BP.	Las heurísticas Primer Ajuste, Mejor Ajuste, Peor Ajuste y Best 3-Fit. Mutación y cruzamiento de grupos.	Resultados óptimos que los comparados.	Utilización de las heurísticas de ordenamiento para generar poblaciones de calidad.
Kanna et al. [2018]	Desarrollar un AGH para el empaquetado tridimensional (3D) de contenedores prismáticos con objetos de formas y tamaños heterogéneos.	Codificación Decimal. GA híbridos. Cruza de doble punto. Mutación por rotación.	Las mejoras heurísticas amplían el espacio de soluciones.	Utilización de los operadores de cruza y mutación.
Mokshin et al. [2020]	Desarrollar un algoritmo genético modificado para resolver el problema del bin packing 3D.	Mutación adaptativa. Modelo LBFL.	MGA obtiene los mejores resultados de 11 algoritmos comparados.	Formulación de la mutación adaptativa.
Kim et al. [2021]	Desarrollar un algoritmo genético híbrido para resolver el problema del acomodo de componente de desechos radioactivos.	Deepest Bottom Left Fill. Codificación diploide. Rotación de componentes. Índice de diversidad.	Se alcanza la ocupación del $100\,\%$.	Utilización del índice de diversidad como condición de paro.
Sundar et al. [2010]	Seleccionar un subconjunto de objetos que maximice el beneficio total utilizando ABC.	Operadores de reparación heurísticos. Operadores de búsqueda local.	La utilización del ABC con los operadores produce mejores resultados que los comparados.	Utilización de los operadores de búsqueda.
Dereli and Daş [2011]	Desarrollar un algoritmo de abejas híbrido discreto para el problema de carga de contenedores.	Nuevo esquema de movimiento. Definición de nuevos parámetros.	Comprueba que el algoritmo obtiene resultados de mayor rendimiento.	Uso del nuevo esquema de movimiento y parámetros.
Kıran et al. [2013]	Desarrollo de un algoritmo de colonia de abejas con operadores genéticos para el TSP.	Operadores de vecindad. Codificación decimal.	El algoritmo obtuvo mejor rendimiento que los comparados en el artículo.	El uso de los operadores de vecindad y la representación de soluciones como codificación decimal.
Zhong et al. [2017]	Desarrollo de un ABC híbrido discreto adaptado al Problema del Viajante (TSP).	Mejora en ecuación de movimiento. Estrategias no codiciosas.	Este algoritmo supera a otros basados en ABC y es competitivo o superior a muchos algoritmos de vanguardia.	Modificación en la ecuación de movimiento con los puntos recomendados en el artículo.
Bayraktar et al. [2021]	Desarrollo de un ABC para el BPP3D.	Mecanismo de memoria. Operadores genéticos.	MIACB supera en rendimiento al GABC, y ambos al ABC.	Utilización de los operadores genéticos.
Sayadi et al. [2010]	Propone nuevo algoritmo de libélulas discreto para minimizar el "makespan".	Discretización de pasos en el algoritmo de libélulas.	El método propuesto en el artículo ofrece un mejor desempeño	Pasos discretos del algoritmo de libélulas.
Jati et al. [2013]	Desarrollar un algoritmo de libélulas discreto evolutivo para el TSP.	Nuevo esquema de movimiento. Mutación por inversión.	El nuevo movimiento garantiza que un individuo se acerque más a uno más brillante.	Uso del nuevo esquema de movimiento con operadores evolutivos.
Tilahun and Ngnotchouye [2017]	Revisión de métodos del algoritmo de luciérnagas para casos discretos.	Discretización de pasos alfa y beta. Distancia hamming.	Discretiza los parámetros más importantes del algoritmo y el movimiento.	Uso de la discretización de pasos y distancia hamming.
Osaba et al. [2017]	Desarrollar un DFFA para aplicarlo al AC-VRP-SPDVCFP.	Distancia hamming. Operador de inserción.	El DFFA supera en rendimiento al algoritmo genético y a algoritmo de recocido simulado.	Uso de la distancia hamming y el operador de inserción en problemas discretos.
Ghaisani and Suyanto [2019]	Desarrollo de un algoritmo libélulas discreto alternativo para problemas de secuencia de tareas.	Nuevo esquema de movimiento. Movimiento con operadores genéticos.	No se garantiza que distancia entre luciérnagas se disminuya d y se debe hacer modificaciones.	Uso del nuevo esquema de movimiento con operadores.

CAPÍTULO 4
METODOLOGÍA DE LA SOLUCIÓN

4.1 Sistema general

El sistema consiste en generalizar el uso de cualquier algoritmo metaheurístico que trabaje sobre el espacio de soluciones de **secuencias de cajas**. Para este trabajo se utiliza los algoritmos genéticos, los de abejas y los de luciérnagas. Aunque los algoritmos de abejas y luciérnagas son *bio-inspirados*, se adoptó un enfoque *evolutivo* con los *operadores genéticos* que son interpretados como movimientos de las luciérnagas y las posiciones del polen de las abejas.

En general, los *operadores genéticos* u *operadores de combinación* tienen que operar a través de permutaciones sobre las soluciones, este sería el requisito para poder incorporarlos al sistema. Inicialmente, siempre se ordena la la lista de soluciones de acuerdo con su *aptitud*, *polen* o *intensidad de luz*, teniendo siempre a la mejor solución al principio y la peor al final.

Finalmente, se procede a “evolucionar” la lista de soluciones en cada iteración. En este punto se debe remarcar que los algoritmos metaheurísticos a utilizar en este trabajo, como se desarrollaron en el marco teórico, sus soluciones evolucionan en cada iteración, esto quiere decir que, en promedio, las soluciones pasadas tienen un menor rendimiento que la obtenida por la actual iteración. Este no es un requisito para la aplicación en el sistema.

La figura 4.1 representa el sistema generalizado en diagrama de flujo:

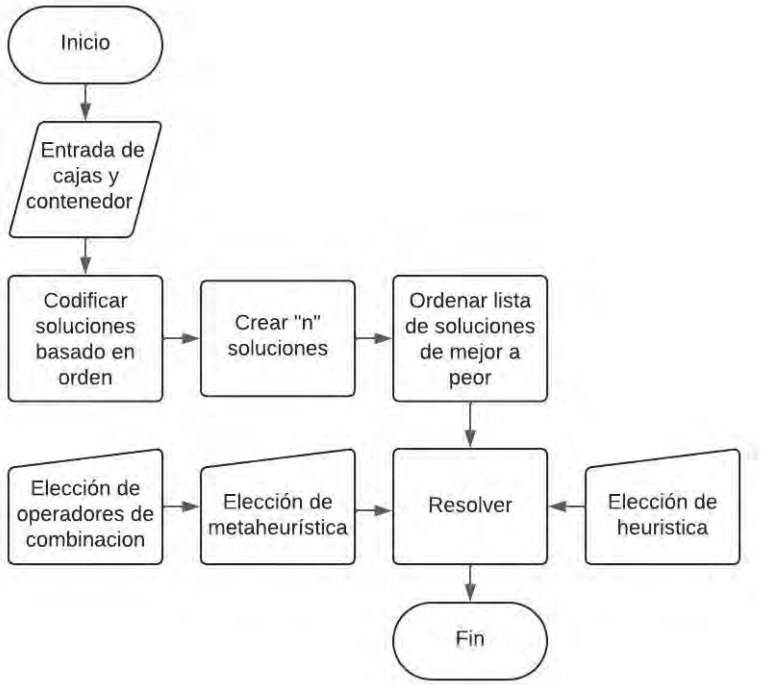


Figura 4.1: Diagrama de sistema generalizado.

En las siguientes secciones se presentan las modificaciones realizadas a cada una de las metaheurísticas bioinspiradas, así como la propuesta de generación de instancias para la creación del conjunto de datos propio.

Es importante volver a mencionar que se trabaja sobre un espacio de soluciones discreto, donde las soluciones están representadas por **una secuencia de cajas** a empacar y cada caja tiene una única identificación. Para este proyecto se considera que las cajas pueden presentar tres tipos de problemas:

- **Problema sin rotación**.
- **Problema con 2 - rotación**. Sólo se permite rotar la caja sobre el eje Z y la rotación de regreso a su posición original.
- **Problema con 6 - rotación**. Se permiten todas las rotaciones posibles de la caja.

4.2 Metaheurísticas modificadas

4.2.1 Algoritmo genético híbrido modificado

La modificación del Algoritmo Genético Híbrido (AGH) se lleva a cabo en la forma de obtener nuevas soluciones, o nuevas crías, al sustituir a un *individuo no élite* en cada iteración. Un individuo no élite es aquel que no pertenece al subconjunto de individuos élite. Los individuos élite se eligen seleccionando el número de individuos élite N_{best} de la población P y son aquellos con la mejor aptitud de toda la población. Si la población P está ordenada por aptitud, el primer individuo es el mejor individuo, entonces los individuos élite se seleccionan tomando los primeros N_{best} de la población P, denotado como P_{best} .

Análogamente, si se requiere saber el *número de individuos no élite* N_{worst} entonces este se obtiene mediante la resta del N es el número total individuos de la población P y el número de individuos élite N_{best}:

$$N_{worst} = N - N_{best}. \tag{4.1}$$

El Algoritmo Genético Híbrido Modificado (AGHM), con cambios significativos marcados en rojo en Algoritmo 5, consiste en los siguientes pasos:

A. Ordenar la población P por aptitud, línea 2.

B. Elegir el número *individuos no élite* a sustituir, línea 5.

C. Iterar sobre la población la *población no élite*, línea 7. El número de individuos no élite, línea 6, se elige con la distribución binomial parametrizado con el número total de individuos N, tomando casi el total de su población.

D. En la elección de padres, líneas $8 - 9$, se eligen con probabilidad pr_s del conjunto de mejores individuos P_{best} mediante torneo. Además, se va almacenando todos los individuos o soluciones que han sido ya visitadas.

E. Si la probabilidad de cruza pr_{cruza} se satisface, entonces cruzar los padres p_1, p_2 para generar dos crías c_1, c_2 y perpetuar la mutación estas con *probabilidad adaptativa*, Ecuación 2.8,

pr_{mut}, línea $10 - 12$.

F. Si las crías c_1, c_2 no han sido visitadas y si alguna de estas crías tienen la aptitud mayor que al individuo no élite seleccionado $P[i]$, entonces el individuo no élite es sustituido. Líneas $13 - 16$.

G. Al final de cada iteración se ordena la población, línea 17.

Algoritmo 5 Algoritmo Genético Híbrido Modificado (por sustitución)

1: $Obtener\ población\ aleatoria,\ con\ 4\ primeros\ individuos\ ordenados \leftarrow P$
2: $Rank(P)$
3: $N \leftarrow |P|$
4: **Repetir**
5: $N_{worst} \leftarrow Bin(N, \frac{N-1}{N})$
6: $i_0 \leftarrow N - N_{worst} + 1$
7: **Para** $i \leftarrow i_0, N$ **Hacer**
8: $p_1 \leftarrow Torneo(P_{best}, pr_s)$
9: $p_2 \leftarrow Torneo(P_{best}, pr_s)$
10: **Si** pr_{cruza} **Entonces**
11: $c_1,\ c_2 \leftarrow CruzaOX(p_1, p_2)$
12: $c_1 \leftarrow Mutar(c_1, pr_{mut}),\ c_2 \leftarrow Mutar(c_2, pr_{mut})$
13: **Si** c_1 no ha sido visitado y $Aptitud(c_1) > (Aptitud(c_2) \& Aptitud(P[i]))$ **Entonces**
14: $P[i] \leftarrow c_1$
15: **Si** c_2 no ha sido visitado y $Aptitud(c_2) > (Aptitud(c_1) \& Aptitud(P[i]))$ **Entonces**
16: $P[i] \leftarrow c_2$
17: $Rank(P)$
18: **Hasta** $CondiciónTérmino()$

4.2.2 Algoritmo de la colonia artificial de abejas discreto modificado

El Algoritmo de Colonia de Abejas Discreto Modificado (ACADM) se diferencia del ACAD, Algoritmo **??**, en que las nuevas soluciones encontradas por las abejas élite, y las no élite, se anexan al final de la población de flores si la "aptitud" o "néctar" de las nuevas soluciones no han sido visitadas, en comparación del trabajo [Dereli and Daş, 2011] que se elige la mejor solución por cada sitio. Estos pasos se muestran en $8 - 9$ y $13 - 14$ del Algoritmo 6. En los pasos $17 - 18$, se implementa una analogía del AGHM, 5, al final de cada iteración se ordenan (en este caso) las flores a través del *ranking* y se recorta con el número inicial de soluciones N.

Algoritmo 6 Algoritmo de Colonia de Abejas Discreto Modificado (por anexión)

Entrada m, e, nep, nsp

1: $Obtener\ población\ aleatoria,\ con\ 4\ primeros\ individuos\ ordenados \leftarrow P$
2: $N \leftarrow |P|$
3: $Rank(P)$
4: **Para** $itr \leftarrow 1, MaxItr$ **Hacer**
5: **Para** $sitioElite \leftarrow 1,\ e$ **Hacer**
6: **Para** $abejaElite \leftarrow 1,\ nep$ **Hacer**
7: $p' \leftarrow Mutar(P_e, 1)$
8: **Si** p' no ha sido visitada **Entonces**
9: $P \cup p'$
10: **Para** $buenSitio \leftarrow e,\ m$ **Hacer**
11: **Para** $abejaNoElite \leftarrow 1,\ nsp$ **Hacer**
12: $p' \leftarrow Mutar(P_m, 1)$
13: **Si** p' no ha sido visitada **Entonces**
14: $P \cup p'$
15: **Para** $k \leftarrow m,\ N$ **Hacer**
16: $P[k] \leftarrow SoluciónAleatoria()$
17: $Rank(P)$
18: $P \leftarrow P[: N]$
19: **Si** $Condición$ **Entonces**
20: **Romper**

4.2.3 Algoritmo de luciérnagas discreto modificado

Para que cada generación de las luciérnagas sea mejor o igual que la pasada, teniendo un enfoque evolutivo, se modifica el algoritmo de tal forma que las antiguas soluciones se consideran y cada nueva solución, se anexa al final de la población actual si la solución no ha sido visitada, líneas $13 - 14$. Estas nuevas soluciones generadas no participan cuando se busca la libélula más atractiva. Al final de cada iteración se vuelve a ordenar la población y se acota esta para regresar al número inicial de luciérnagas, línea 16, descartando las menos aptas. Esto se representa en el Algoritmo 7.

Algoritmo 7 Algoritmo de luciérnagas Discreto Modificado (por anexión)

Entrada $\gamma, m, MaxItr$

1: $Obtener\ población\ aleatoria,\ con\ 4\ primeros\ individuos\ ordenados \leftarrow P$
2: $Rank(P)$
3: $N \leftarrow |P|$
4: $n \leftarrow TotalCajas$
5: **Para** $itr \leftarrow 1, MaxItr$ **Hacer**
6: **Para** $i \leftarrow 1, N$ **Hacer**
7: $p_{atr} \leftarrow EncontrarMásAtractiva(P, P[i])$
8: **Para** $j \leftarrow 1, m$ **Hacer**
9: **Si** p_{atr} existe **Entonces**
10: $p' \leftarrow Cruza(P[i], p_{atr})$
11: **Sino**
12: $p' \leftarrow Mutar(P[i], 1)$
13: **Si** p' no ha sido visitado **Entonces**
14: $P \cup p'$
15: $Rank(P)$
16: $P \leftarrow P[: N]$
17: **Si** $Condición$ **Entonces**
18: **Romper**

4.3 Datasets

4.3.1 Generador de instancias heterogéneas

El generador de instancias aleatorias tomado se describen en [Karabulut and İnceoğlu, 2004]. En cada instancia se tomaron diferentes semillas aleatorias, para $p = \{1, \ldots, 100\}$ se utiliza la fórmula en [Bischoff and Ratcliff, 1995]:

$$s = 250250 + 100(p - 1) \tag{4.1}$$

Las instancias se generaron con el algoritmo 2 definido en [Karabulut and İnceoğlu, 2004].

Tabla 4.1: 100 instancias por problema usando Alg. 2.

	Dim. del contenedor	Total de cajas	Prom. de cajas diferentes	Máx. de cajas diferentes	Mín. de cajas diferentes
Problema 1	20x20x20	16	14 (87.5 %)	16 (100 %)	9 (56 %)
Problema 2	50x50x50	25	22 (88 %)	25 (100 %)	15 (60 %)
Problema 3	100x100x100	52	46 (88.4 %)	52 (100 %)	34 (65 %)
Problema 4	200x200x200	100	88 (88 %)	100 (100 %)	73 (73 %)
Problema 5	300x300x300	151	134 (88.7 %)	148 (98 %)	114 (75 %)

La Tabla 4.1 describe las instancias por los 5 problemas que fueron generados. La primera columna describe el tipo de problema. La segunda columna describe dos parámetros: *número de cajas* y las *dimensiones del contenedor*; por ejemplo, el problema 1 tiene 16 cajas con un contenedor de $20 \times 20 \times 20$. La tercera, cuarta y quinta columna muestran respectivamente el promedio, el máximo y el mínimo del *número de tipos de cajas diferentes* de las 100 instancias creadas por cada problema. El **porcentaje de cajas de diferente tipo** es el porcentaje entre paréntesis a lado de los datos de la cuarta, quinta y sexta columnas. Esto último indica que las instancias generadas son fuertemente heterogéneas.

4.3.2 Explicación de lectura de instancias

A continuación, se muestra un ejemplo, Tabla 4.2, de la instancia generada usando el problema 2 (P2A2) con $p = 1$ de la Ecuación 4.1.

Tabla 4.2: Ejemplo de la primera instancia de P2A2.

100	25						
1	2502505						
50	50	50					
23							
1	1	1	50	1	18	1	11
2	1	1	50	1	18	1	39
3	1	1	7	1	32	1	33
...	...	...	...	...	...	...	...
23	1	1	3	1	2	1	1

La primera línea se lee una única vez, Tabla 4.3. Se tienen dos columnas: la primera columna indica el número de instancias creadas y la segunda columna indica el número de cajas de todas las instancias.

Tabla 4.3: Primera línea del archivo de datos.

Número de instancias	**Número de cajas**
100	25

La segunda línea, Tabla 4.4, del archivo de datos contiene dos columnas: la primera columna indica *el valor de la semilla* para el valor de p que igualmente se interpreta como el *número de*

instancia, y la segunda columna indica el valor de generado de s de la Ecuación 4.1. Esta línea se repite para cada instancia en el archivo.

Tabla 4.4: Segunda línea del archivo de datos.

Número de instancia	**Valor de s**
1	2502505

La tercera línea, Tabla 4.5, indica las dimensiones del contenedor en forma. Esta información ser repite en el archivo por cada instancia creada.

Tabla 4.5: Tercera línea del archivo de datos. Dimensiones del contenedor.

Largo	**Ancho**	**Largo**
50	50	50

La cuarta línea, Tabla 4.6, indica el *número de tipos de cajas* que hay en la instancia creada. Esta información se irá repitiendo en el archivo por cada instancia creada.

Tabla 4.6: Cuarta línea del archivo de datos. Número de tipo de cajas

Tipos de cajas
23

Finalmente, las líneas siguientes, son el mismo número al número de tipos de cajas, cada una contiene ocho columnas que representan respectivamente: *número de tipo de caja*, *número total de cajas por tipo*, *indicador de rotación sobre X* ($X\,0/1$), *largo*, *indicador de rotación sobre Y* ($Y\,0/1$), *ancho*, *indicador de rotación sobre Z* ($Z\,0/1$) y *alto*. Los **indicadores de rotación** sólo pueden tomar valores de $\{0, 1\}$, si su valor es 0 la rotación no es permitida y si su valor es 1 la rotación es permitida; para este trabajo todas las rotaciones son permitidas.

Las instancias pueden ser consultadas públicamente en el siguiente repositorio: https://github.com/M21CE13Cenidet/HeterogeneousSingleContainerDataSet/blob/main

4.4 Herramientas experimentales

En este trabajo se utilizó una computadora con procesador marca *AMD* modelo *Ryzen 7 5700G* de 3.80 GHz con 8 núcleos, y 2×8 GB de RAM DDR4 a 3600 Mhz. Se utilizó el lenguaje de programación *C++* versión 14. Las metaheurísticas fueron programadas sin la utilización de ninguna librera externa al estándar.

CAPÍTULO 5
EXPERIMENTACIÓN Y RESULTADOS

En este capítulo se presenta la experimentación hecha para evaluar el desempeño las metaheurísticas utilizando 100 instancias por cada problema listado en la Tabla 4.1. A continuación, se listan los casos que se llevan a cabo en la experimentación, además del número total de soluciones por rotación que se deben analizar:

A. **P1A2** - Problema 1 Algoritmo 2. 16 cajas y contenedor de $20 \times 20 \times 20$

   a) 0-rotación: 2.0922×10^{13} soluciones

   b) 2-rotación: 1.37120×10^{18} soluciones

   c) 6-rotación: 5.9025×10^{25} soluciones

B. **P2A2** - Problema 2 Algoritmo 2. 25 cajas y contenedor de $50 \times 50 \times 50$

   a) 0-rotación: 1.5511×10^{25} soluciones

   b) 2-rotación: 5.2046×10^{32} soluciones

   c) 6-rotación: 4.4098×10^{44} soluciones

C. **P3A2** - Problema 3 Algoritmo 2. 52 cajas y contenedor de $100 \times 100 \times 100$

   a) 0-rotación: 8.0658×10^{67} soluciones

   b) 2-rotación: 3.6325×10^{83} soluciones

   c) 6-rotación: 2.3470×10^{108} soluciones

D. **P4A2** - Problema 4 Algoritmo 2. 100 cajas y contenedor de $200 \times 200 \times 200$

   a) 0-rotación: 9.3326×10^{157} soluciones

   b) 2-rotación: 1.1830×10^{188} soluciones

   c) 6-rotación: 6.0971×10^{235} soluciones

E. **P5A2** - Problema 5 Algoritmo 2. 151 cajas y contenedor de $300 \times 300 \times 300$

   a) 0-rotación: 8.6272×10^{264} soluciones

   b) 2-rotación: 2.4625×10^{310} soluciones

   c) 6-rotación: 7.8022×10^{427} soluciones

Cada uno de los problemas tiene 100 instancias, además, se prueban 6 metaheurísticas con 3 tipos de mutaciones diferentes. Por lo tanto, en total se tienen $(100)(6)(3) = 1800$ experimentos en este trabajo.

5.1 Descripción del experimento

Se presenta en la Figura 5.1 el diagrama de flujo de los experimentos para promediar las mejores soluciones de las instancias por problema, los pasos son los siguientes:

A. Se inicia eligiendo el problema a resolver: **P1A2**, **P2A2**, **P3A2**, **P4A2** o **P5A2**.

B. Se Se elige el tipo de rotación: **0 - rotación**, **2 - rotación** o **6 - rotación**.

C. Se elige los operadores de mutación: **inversión**, **C1**, **C2**

D. Se elige la metaheurística a utilizar.

E. Después, se obtiene de las 100 instancias generadas, la primera para obtener los datos del contenedor y las cajas a empacar para codificarlas y posteriormente resolver el problema, obteniendo así el valor del **máximo espacio útil** encontrado

F. Se itera sobre las 100 instancias para posteriormente promediar los resultados. Por cada instancia se utiliza una única heurística, DBLF

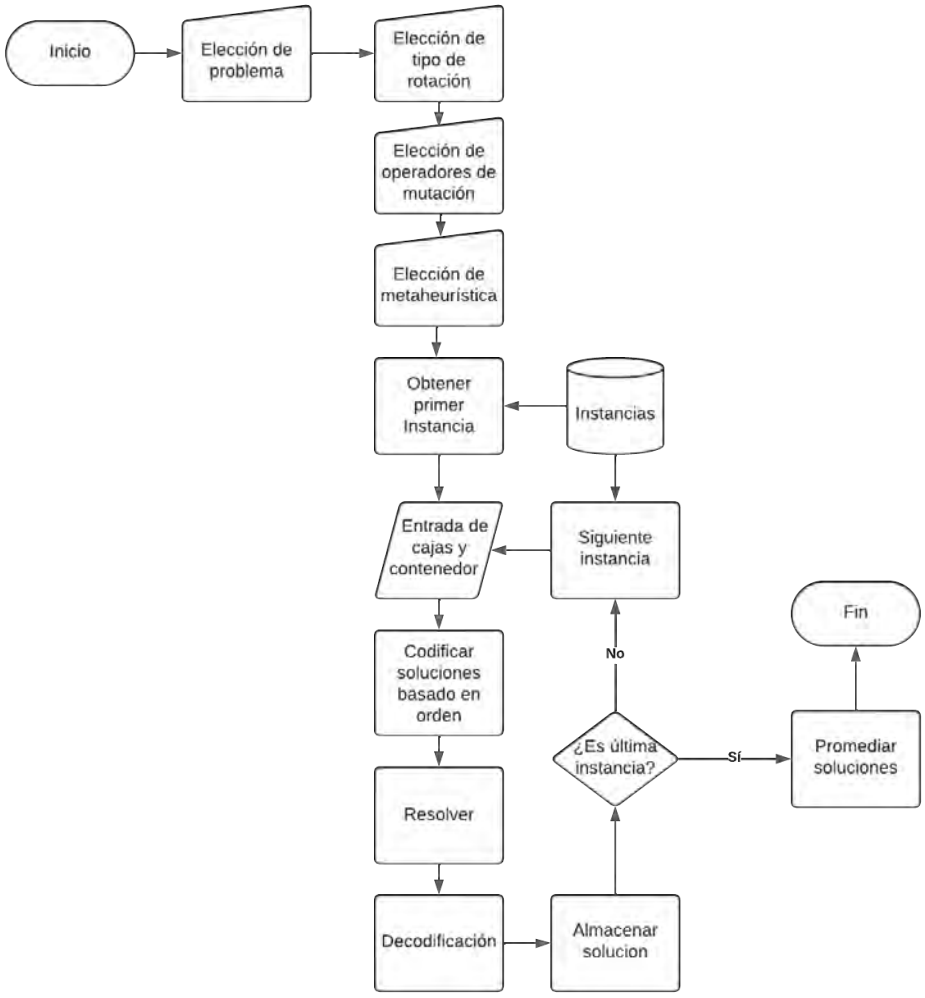


Figura 5.1: Diagrama de flujo de experimento por instancia.

5.1.1 Versiones de metaheurísticas en el experimento

En general, cada metaheurística tiene 2 **versiones**: una que sustituye soluciones de la población y otra que agrega soluciones a la población. Además, por cada metaheurística se utiliza **3 tipos de mutaciones diferentes**, por lo tanto, cada metaheurística tendrá 6 versiones. A continuación, se listan las metaheurísticas con sus respectivas versiones, en donde una versión fue tomada del estado del arte y la otra fue modificada en este trabajo, se agrega la palabra “modificado” a cada metaheurística:

- Algoritmo genético:

    - Por sustitución: **Algoritmo genético híbrido modificado (AGHM)**, Algoritmo 5.
    - Por anexión: **Algoritmo genético híbrido (AGH)**, Algoritmo 2.
- Algoritmo de colonia de abejas:
    - Por sustitución: **Algoritmo de Colonia de Abejas Discreto (ACAD)**, Algoritmo 3.
    - Por anexión: **Algoritmo de Colonia de Abejas Discreto Modificado (ACADM)**, Algoritmo 6.
- Algoritmo de luciérnagas:
    - Por sustitución: **Algoritmo de luciérnagas Discreto (ALD)**, Algoritmo 4.
    - Por anexión: **Algoritmo luciérnagas Discreto Modificado (ALDM)**, Algoritmo 7.

5.2 Parámetros experimentales

Los parámetros de experimentación se tienen que definir de tal forma que la comparativa de resultados entre las metaheurísticas sea proporcional al número de veces que se explota una solución.

5.2.1 Parámetros generales

Para todas las metaheurísticas:

- El **número de iteraciones máximas** es de 1000
- Para **la mutación sobre la codificación de rotaciones** se utiliza el operador de **mutación por volteado (Flip mutation)** con una probabilidad del 0.01 que se cambie el valor de una posición en la codificación.

5.2.2 Elección del número de población

Para elegir el número de población de las metaheurísticas se debe considerar que este número será las veces que se llama la heurística DBLF. En este experimento el número de individuos

en el algoritmo genético n_{AG} definirá el número de las poblaciones del algoritmo de abejas y luciérnagas.

(a) En este trabajo, el número de individuos para los algoritmos genéticos es 50.

$$n_{AG} = 50. \tag{5.1}$$

(b) Para los algoritmos de abejas, sus parámetros deben ser elegidos de tal forma que se llame a la heurística DBLF n_{AG} veces. En esta metaheurística se tienen 5 parámetros que la define: **número de buenos sitios** m, **número de sitios élite** e, **número de abejas élite** nep y **número de abejas no élite** nsp. Haciendo un análisis del algoritmo se llega a la conclusión que, el número total de sitios (o flores) n_{sitios} y los parámetros de la metaheurísitca se debe elegir tal que se cumpla la siguiente ecuación:

$$n_{AG} = e \cdot nep + (m - e) \cdot nsp + n_{sitios} - m. \tag{5.2}$$

Reorganizando la ecuación quedaría como:

$$n_{AG} = e \cdot (nep - nsp) + m \cdot (nsp - 1) + n_{sitios}. \tag{5.3}$$

(c) De forma análoga en la elección del número de soluciones para el algoritmo de luciérnagas, es decir, número de luciérnagas, se tiene que cumplir la siguiente igualdad:

$$n_{AG} = n_{libelulas} \cdot m \tag{5.4}$$

Las ecuaciones Ecuación 5.3 y Ecuación 5.4 puede definir varios números naturales que cumplen dichas ecuaciones. En este trabajo no se toma en cuenta qué valores se les asignan tal que sean óptimos.

5.3 Resultados

Se promedia la mejor **aptitud** f_i **(espacio útil)** de cada metaheurística por cada problema y se calcula el **intervalo de confianza** al $95\,\%$. Por ejemplo, si se elige la metaheurística GAH para resolver el problema P1A2, se promedia las 100 aptitudes obtenidas y se calcula el intervalo de confianza.

Se hace una comparación en una tabla las metaheurísticas por problema. Las columnas de las tablas representan la metaheurística con su operador de mutación, con motivo de simplificar la mutación de inversión se abrevia como *"Inv"*, el grupo C1 como *"C1"* y el grupo C2 como *"C2"*. Las filas de las tablas representan los promedios obtenidos los problemas, la última fila representa el promedio de todas las aptitudes de la metaheurística con su operador de mutación. Se remarcará en negrita las mejores soluciones promediadas por metaheurística.

Una vez obtenido todos los resultados, a través de gráficas, se compara las metaheurísticas modificadas con las no modificadas, y aquellas con el mejor desempeño en el experimento. En las gráficas se representarán los intervalos de confianza para cada medición.

Los datos mostrados en tabla son transformados de **aptitud** f_i a **porcentaje del espacio utilizado** P_i para un mejor entendimiento e interpretación de estos valores:

$$\bar{P}_i = \bar{f}_i \cdot 100\,\%. \tag{5.5}$$

De igual forma se agregan gráficos de las aptitudes $(Aptitud \times Numero\,de\,Cajas)$ encontradas por metaheurísticas con todas las mutaciones usadas. Además, se agrega un gráfico del tiempo promedio por número de cajas.

5.4 Resultados de los algoritmos genéticos

5.4.1 Parámetros del algoritmo genético híbrido

- Número de libélulas: $n_{AG} = 50$.

- Probabilidad de cruza: $p_c = 0.75$.
- Probabilidad de selección $p_s = 0.85$.
- Probabilidad de mutación: **probabilidad de mutación adaptativa** Ecuación 2.8.
- Condición de término: si un individuo tiene aptitud de 1.0 ó si el índice de diversidad η Ecuación 2.9 es menor a 0.01.

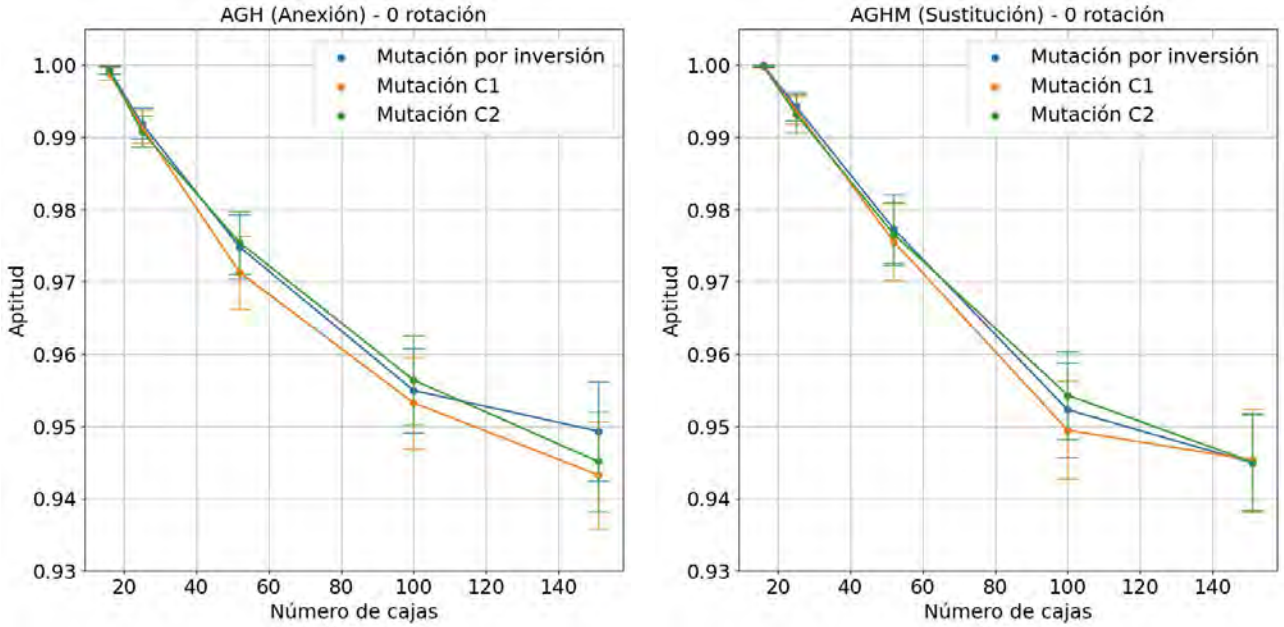
5.4.2 Resultados para 0-rotación

(a) **Aptitud promedio**

La Tabla 5.1 muestra los resultados del Algoritmo Genético Híbrido (por Anexión) y el Algoritmo Genético Híbrido Modificado (por Sustitución) en términos del porcentaje de ocupación del contenedor. La metaheurística con el mejor rendimiento promedio se remarca en negrita. La Figura 5.2 muestra la comparación gráfica de las metaheurísticas con sus versiones de mutación, se relaciona aptitud contra el número de cajas a empacar.

Tabla 5.1: Resultados del AGH con 0-rotación

	AGH (Inv)	AGHM (Inv)	AGH (C1)	AGHM (C1)	GAH (C2)	AGHM (C2)
P1A2	$99.9380 \pm 0.0578\,\%$	$99.9950 \pm 0.0098\,\%$	$99.8839 \pm 0.0913\,\%$	$99.9805 \pm 0.0220\,\%$	$99.9271 \pm 0.0514\,\%$	$99.9875 \pm 0.0175\,\%$
P2A2	$99.1911 \pm 0.2177\,\%$	$99.4301 \pm 0.1961\,\%$	$99.1492 \pm 0.2249\,\%$	$99.3789 \pm 0.1942\,\%$	$99.0830 \pm 0.2190\,\%$	$99.3253 \pm 0.2671\,\%$
P3A2	$97.4808 \pm 0.4472\,\%$	$97.7321 \pm 0.4719\,\%$	$97.1299 \pm 0.5086\,\%$	$97.5590 \pm 0.5288\,\%$	$97.5416 \pm 0.4348\,\%$	$97.6601 \pm 0.4364\,\%$
P4A2	$95.4951 \pm 0.5876\,\%$	$95.2285 \pm 0.6557\,\%$	$95.3240 \pm 0.6374\,\%$	$94.9465 \pm 0.6805\,\%$	$95.6386 \pm 0.6147\,\%$	$95.4305 \pm 0.6047\,\%$
P5A2	$94.9313 \pm 0.6865\,\%$	$94.4949 \pm 0.6800\,\%$	$94.3253 \pm 0.7357\,\%$	$94.5321 \pm 0.7129\,\%$	$94.5097 \pm 0.6872\,\%$	$94.5035 \pm 0.6619\,\%$
	$\mathbf{97.4073 \pm 0.3994\,\%}$	$97.3760 \pm 0.4027\,\%$	$97.1624 \pm 0.4396\,\%$	$97.2794 \pm 0.4277\,\%$	$97.3400 \pm 0.4014\,\%$	$97.3814 \pm 0.3975\,\%$



(a) Aptitud vs. Número de cajas para el AGH con 0-rotación.

(b) Aptitud vs. Número de cajas para el AGHM con 0-rotación.

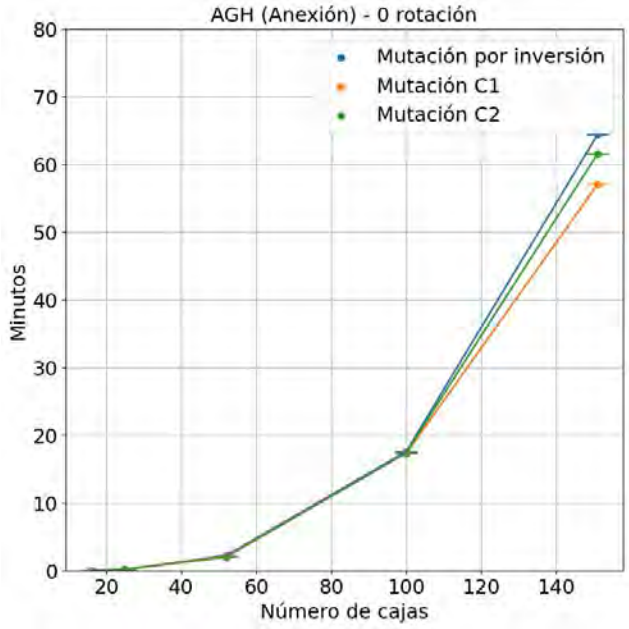
Figura 5.2: Comparativa de la aptitud promedio de los algoritmos genéticos con 0-rotación.

(b) **Tiempo promedio**

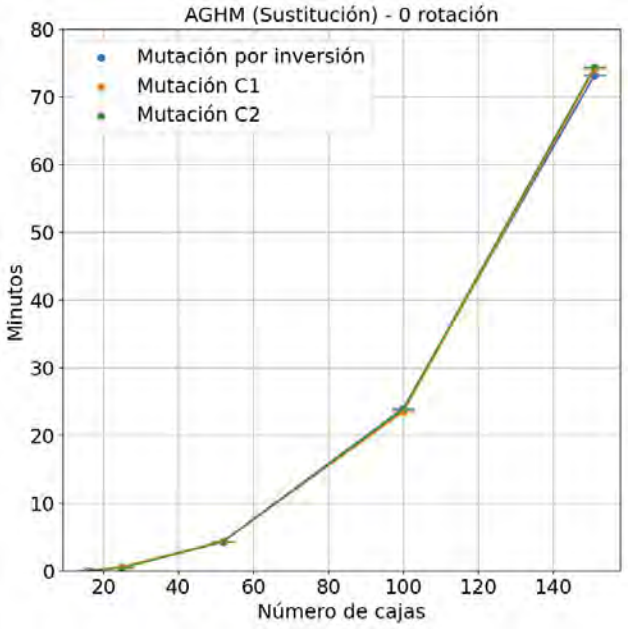
La Tabla 5.1 muestra los tiempos promedios de ejecución de las metahueristicas por cada problema y mutación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.3 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.2: Tiempos en minutos del AGH con 0-rotación

	AGH (Inv)	AGHM (Inv)	AGH (C1)	AGHM (C1)	GAH (C2)	AGHM (C2)
P1A2	0.0149 min	0.0082 min	0.0184 min	0.0163 min	0.0119 min	0.0121 min
P2A2	0.1821 min	0.5450 min	0.2828 min	0.5961 min	0.2599 min	0.4232 min
P3A2	2.2473 min	4.3183 min	2.1016 min	4.3516 min	2.0081 min	4.2768 min
P4A2	17.6037 min	23.9748 min	17.3468 min	23.4634 min	17.4264 min	23.8130 min
P5A2	64.4192 min	73.2107 min	57.0740 min	74.0765 min	61.5779 min	74.4115 min
	16.8934 min	20.4114 min	**15.3647 min**	20.5008 min	16.2569 min	20.5873 min



(a) Minutos vs. Número de cajas para el AGH con 0-rotación.



(b) Minutos vs. Número de cajas para el AGHM con 0-rotación.

Figura 5.3: Comparativa de los tiempos en minutos de los algoritmos genéticos con 0-rotación.

(c) **Análisis de resultados**

La Tabla 5.1 muestra los resultados de los algoritmos genéticos híbridos usando los tres tipo de mutaciones respectivamente. Los resultados representan el promedio del porcentaje del espacio utilizado junto con su intervalo de confianza al $95\,\%$. El valor promedio más algo fue obtenido por el algoritmo genético híbrido por anexión (sin modificar), sin embargo, las demás variaciones de los algoritmos genéticos se mantienen dentro del intervalo de confianza entre sí, esto se muestra gráficamente en las figuras Figura 5.2a y Figura 5.2b, donde el algoritmo con menor dispersión fue el AGHM, aunque con un menor rendimiento al AGH con mutación por inversión. Si se promedian todos los resultados del AGH y del AGHM para compararlos se obtiene los valores de 97.3032 ± 0.4135 y 97.3456 ± 0.4093 respectivamente, se observa que sus promedios son muy cercanos entre sí y caen dentro del intervalo de confianza del otro.

La Tabla 5.2 muestra los tiempos promedios, en minutos, de cada problema con cada algoritmo genético híbrido. Se muestra que el AGH ó el algoritmo genético híbrido con

enfoque por anexión, tarda aproximadamente $20\,\%$ menos que el AGHM ó el algoritmo genético híbrido con enfoque por sustitución, los resultados son mas evidentes en las figuras Figura 5.3a y Figura 5.3b

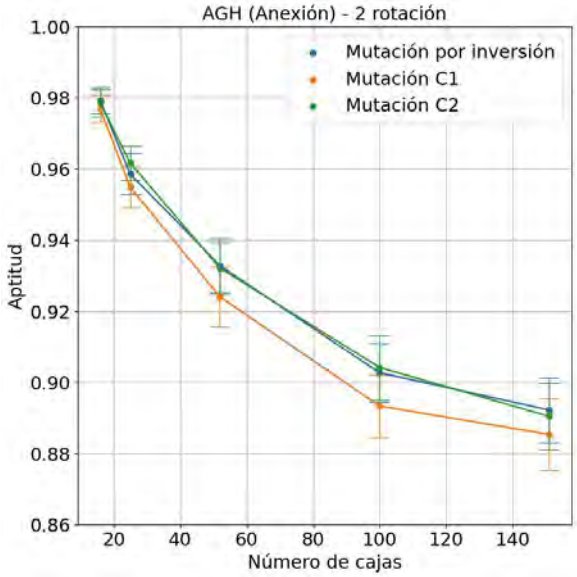
5.4.3 Resultados para 2-rotación

(a) **Aptitud promedio**

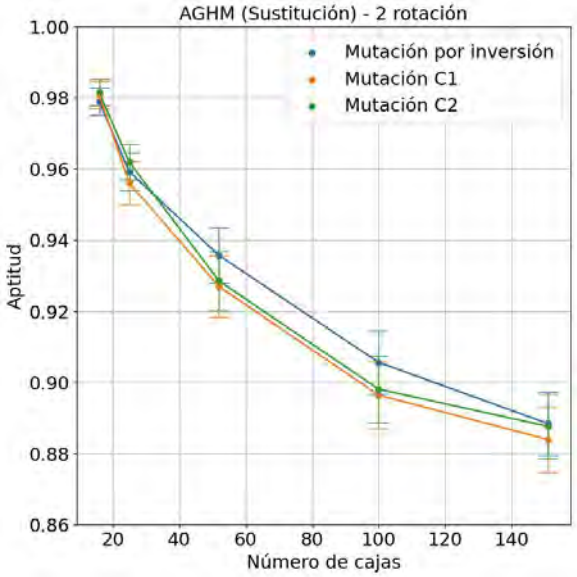
La Tabla 5.3 muestra los resultados las metaheurísticas en términos del porcentaje de ocupación del contenedor considerando que las cajas pueden tener 2 tipos de rotación. La metaheurística con el mejor rendimiento promedio se remarca en negrita. La Figura 5.4 muestra la comparación gráfica de las metaheurísticas con sus versiones de mutación, se relaciona aptitud contra el número de cajas a empacar.

Tabla 5.3: Resultados del AGH con 2-rotación

	AGH (Inv)	AGHM (Inv)	AGH (C1)	AGHM (C1)	AGH (C2)	AGHM (C2)
P1A2	$97.9241 \pm 0.3728\,\%$	$97.8919 \pm 0.3885\,\%$	$97.6931 \pm 0.3734\,\%$	$98.0726 \pm 0.3814\,\%$	$97.8560 \pm 0.3773\,\%$	$98.1485 \pm 0.3770\,\%$
P2A2	$95.8554 \pm 0.5784\,\%$	$95.9244 \pm 0.5430\,\%$	$95.4830 \pm 0.5941\,\%$	$95.6065 \pm 0.6046\,\%$	$96.1631 \pm 0.4819\,\%$	$96.1931 \pm 0.5012\,\%$
P3A2	$93.2781 \pm 0.7587\,\%$	$93.5660 \pm 0.7764\,\%$	$92.4032 \pm 0.8498\,\%$	$92.6927 \pm 0.8577\,\%$	$93.2074 \pm 0.7219\,\%$	$92.8507 \pm 0.8372\,\%$
P4A2	$90.2664 \pm 0.8272\,\%$	$90.5599 \pm 0.8862\,\%$	$89.3307 \pm 0.8702\,\%$	$89.6387 \pm 0.9410\,\%$	$90.4153 \pm 0.9073\,\%$	$89.8056 \pm 0.9320\,\%$
P5A2	$89.2200 \pm 0.9005\,\%$	$88.8351 \pm 0.8996\,\%$	$88.5335 \pm 1.0046\,\%$	$88.3920 \pm 0.9300\,\%$	$89.0414 \pm 0.9336\,\%$	$88.7598 \pm 0.9024\,\%$
	$93.3088 \pm 0.6875\,\%$	$\mathbf{93.3555 \pm 0.6987\,\%}$	$92.6887 \pm 0.7376\,\%$	$92.8805 \pm 0.7429\,\%$	$93.3366 \pm 0.6844\,\%$	$93.1515 \pm 0.7100\,\%$



(a) Aptitud vs. Número de cajas para el AGH con 2-rotación.



(b) Aptitud vs. Número de cajas para el AGHM con 2-rotación.

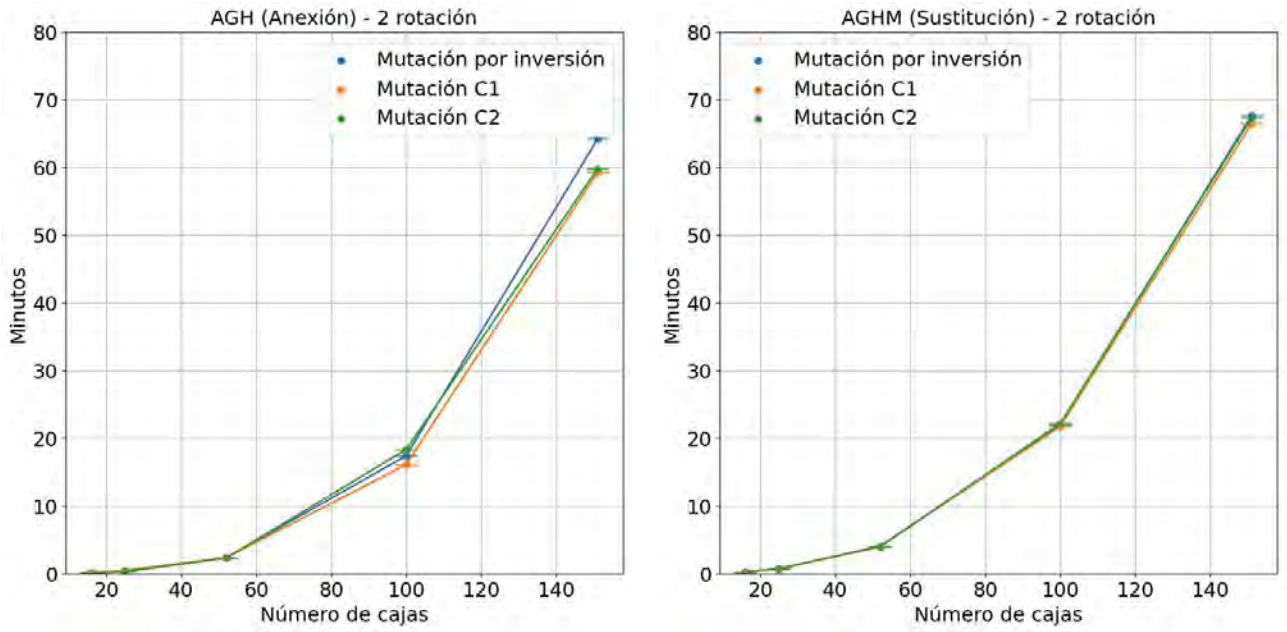
Figura 5.4: Comparativa de los tiempos en minutos de los algoritmos genéticos con 2-rotación.

(b) **Tiempo promedio**

La Tabla 5.4 muestra los tiempos promedios, para 2-rotación, de ejecución de las metahueristicas por cada problema y mutación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.5 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.4: Tiempos en minutos del AGH con 2-rotación

	AGH (Inv)	AGHM (Inv)	AGH (C1)	AGHM (C1)	GAH (C2)	AGHM (C2)
P1A2	0.1265 min	0.2861 min	0.2055 min	0.3107 min	0.1191 min	0.2808 min
P2A2	0.3293 min	0.7699 min	0.5254 min	0.7518 min	0.3946 min	0.7625 min
P3A2	2.3602 min	4.0692 min	2.3712 min	4.0211 min	2.3217 min	3.9945 min
P4A2	17.4336 min	21.9901 min	16.1298 min	21.7546 min	18.3530 min	22.1885 min
P5A2	64.2128 min	67.6375 min	59.2955 min	66.4322 min	59.8212 min	67.3020 min
	16.8925 min	18.9506 min	**15.7055 min**	18.6541 min	16.2019 min	18.9057 min



(a) Minutos vs. Número de cajas para el AGH con 2-rotación.

(b) Minutos vs. Número de cajas para el AGHM con 2-rotación.

Figura 5.5: Comparativa de los tiempos en minutos de los algoritmos genéticos con 2-rotación.

(c) **Análisis de resultados**

La Tabla 5.3 muestra los promedios del porcentaje del espacio útil para 2-rotación. Similar a

los resultados del 0-rotación, los algoritmos genéticos caen dentro del intervalo de confianza de cada una de sus versiones de mutación y de la versión modificada, AGHM.

El promedio más alto obtenido fue por el AGHM con mutación por inversión, mostrado en la Figura 5.4b, aunque la diferencia con el AGH no sea significativamente grande, como se muestra en la Figura 5.4a. Los promedios totales para el AGH y AGHM son, respectivamente: $93.1114 \pm 0.7030\,\%$ y $93.1292 \pm 0.7172\,\%$. El mayor valor lo obtuvo el AGHM pero su diferencia es mínima con respecto al AGH, su diferencia es de $0.0178\,\%$.

La Tabla 5.4 muestra los tiempos promedios, donde se puede observar una mínima diferencia entre el AGH, Fig 5.5a, y el AGHM, Fig 5.5b, respecto a la versión de 0-rotación; el AGH mantiene los menores tiempos con respecto al AGHM con un aproximado de $15\,\%$ de menor tiempo.

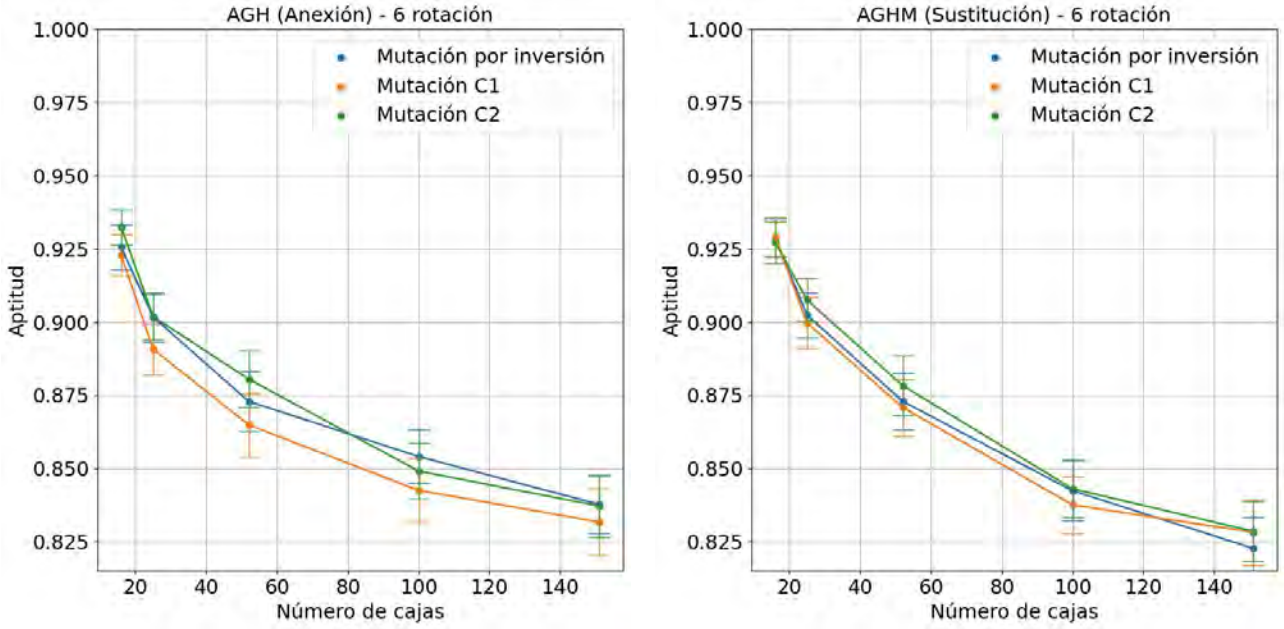
5.4.4 Resultados para 6-rotación

(a) **Aptitud promedio**

La Tabla 5.5 muestra los resultados las metaheurísticas en términos del porcentaje de ocupación del contenedor considerando que las cajas pueden tener todos los tipos de rotación. La metaheurística con el mejor rendimiento promedio se remarca en negrita. La Figura 5.6 muestra la comparación gráfica de las metaheurísticas con sus versiones de mutación, se relaciona aptitud contra el número de cajas a empacar.

Tabla 5.5: Tabla de resultados con 6-rotación AGH

	AGH (Inv)	AGHM (Inv)	AGH (C1)	AGHM (C1)	AGH (C2)	AGHM (C2)
P1A2	$92.5480 \pm 0.7637\,\%$	$92.8750 \pm 0.6611\,\%$	$92.2818 \pm 0.6882\,\%$	$92.9064 \pm 0.6624\,\%$	$93.2344 \pm 0.6042\,\%$	$92.7144 \pm 0.7108\,\%$
P2A2	$90.1639 \pm 0.8448\,\%$	$90.2324 \pm 0.7775\,\%$	$89.0634 \pm 0.8720\,\%$	$89.9630 \pm 0.8680\,\%$	$90.1782 \pm 0.7774\,\%$	$90.7424 \pm 0.7332\,\%$
P3A2	$87.2848 \pm 1.0256\,\%$	$87.2855 \pm 0.9686\,\%$	$86.4820 \pm 1.0752\,\%$	$87.0769 \pm 0.9628\,\%$	$88.0440 \pm 0.9598\,\%$	$87.8214 \pm 1.0179\,\%$
P4A2	$85.4032 \pm 0.9136\,\%$	$84.2318 \pm 1.0243\,\%$	$84.2493 \pm 1.0879\,\%$	$83.7502 \pm 0.9668\,\%$	$84.9060 \pm 0.9632\,\%$	$84.2987 \pm 0.9743\,\%$
P5A2	$83.7664 \pm 0.9823\,\%$	$82.2671 \pm 1.0640\,\%$	$83.1684 \pm 1.1517\,\%$	$82.8186 \pm 1.1202\,\%$	$83.7111 \pm 1.0544\,\%$	$82.8518 \pm 1.0269\,\%$
	$87.8333 \pm 0.9060\,\%$	$87.3784 \pm 0.8991\,\%$	$87.0482 \pm 0.9750\,\%$	$87.3030 \pm 0.9160\,\%$	$\mathbf{88.0147 \pm 0.8718\,\%}$	$87.6857 \pm 0.8926\,\%$



(a) Aptitud vs. Número de cajas para el AGH con 6-rotación.

(b) Aptitud vs. Número de cajas para el AGHM con 6-rotación.

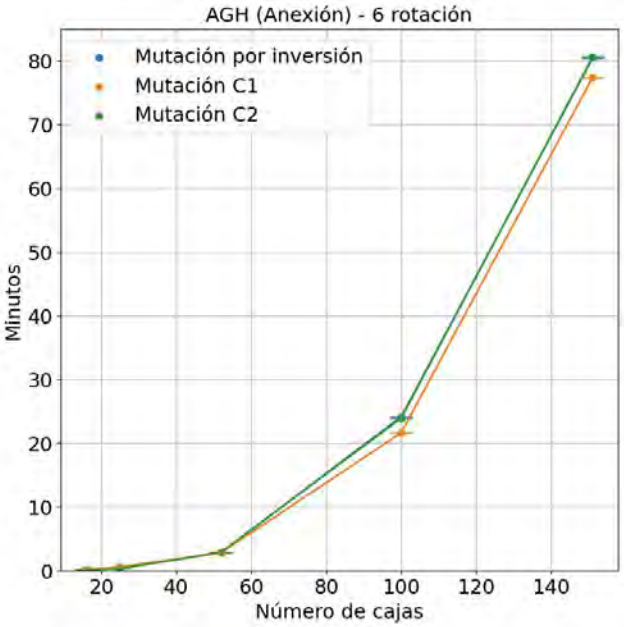
Figura 5.6: Comparativa de la aptitud promedio de los algoritmos genéticos con 6-rotación.

(b) **Tiempo promedio**

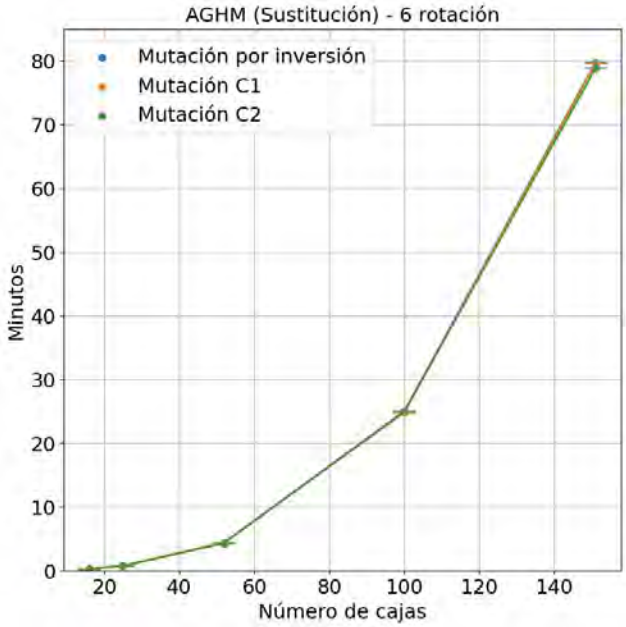
La Tabla 5.6 muestra los tiempos promedios, para 6-rotación, de ejecución de las metaheurísticas por cada problema y mutación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.7 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.6: Tiempos en minutos del AGH con 6-rotación

	AGH (Inv)	AGHM (Inv)	AGH (C1)	AGHM (C1)	GAH (C2)	AGHM (C2)
P1A2	0.0963 min	0.3216 min	0.2349 min	0.3517 min	0.1372 min	0.3217 min
P2A2	0.3810 min	0.7746 min	0.6132 min	0.8076 min	0.3847 min	0.7548 min
P3A2	2.8626 min	4.4376 min	2.8454 min	4.4411 min	2.9021 min	4.3308 min
P4A2	24.1631 min	25.0096 min	21.6453 min	24.7798 min	23.9390 min	24.9772 min
P5A2	80.4455 min	79.7778 min	77.4110 min	79.6272 min	80.6338 min	78.8615 min
	21.5897 min	22.0643 min	**20.5500 min**	22.0015 min	21.5994 min	21.8492 min



(a) Minutos vs. Número de cajas para el AGH con 6-rotación.



(b) Minutos vs. Número de cajas para el AGHM con 6-rotación.

Figura 5.7: Comparativa de los tiempos en minutos de los algoritmos genéticos con 6-rotación.

(c) **Análisis de resultados**

En 6-rotación, se observa que que los algoritmos genéticos híbridos por anexión tuvieron un mejor desempeño para el problema P5A2 que los de sustitución, como se muestra en las Figura 5.6a y 5.6b. En la Tabla 5.5 se muestra en negrita que el algoritmo genético genético híbrido (por sustitución) usando el grupo de mutación C2 obtuvo el mejor desempeño que los demás. Sin embargo, el intervalo de confianza de todos los algoritmos genéticos están dentro del otro.

Las Tabla 5.6 muestra los tiempos promedio por cada algoritmo, mostrando que ambas versiones tanto por anexión como por sustitución duran aproximadamente lo mismo, sin tener una mejora alguna. Las gráficas Figura 5.7a y Figura 5.7b se aprecia mejor la similitud de la duración promedio de los algoritmos.

5.5 Resultados de los algoritmos de colonia de abejas

5.5.1 Parámetros del algoritmo de colonia de abejas

- Número de sitios élite: $e = 8$
- Número de abejas élite: $nep = 4$
- Número de buenos sitios: $m = 11$
- Número de abejas no élite: $nsp = 3$
- Número total de sitios: $n_{sitios} = 20$
- Probabilidad de mutación $p_m = 1.0$
- Condición de término: si una solución (lugar o flor) tiene aptitud de 1.0.

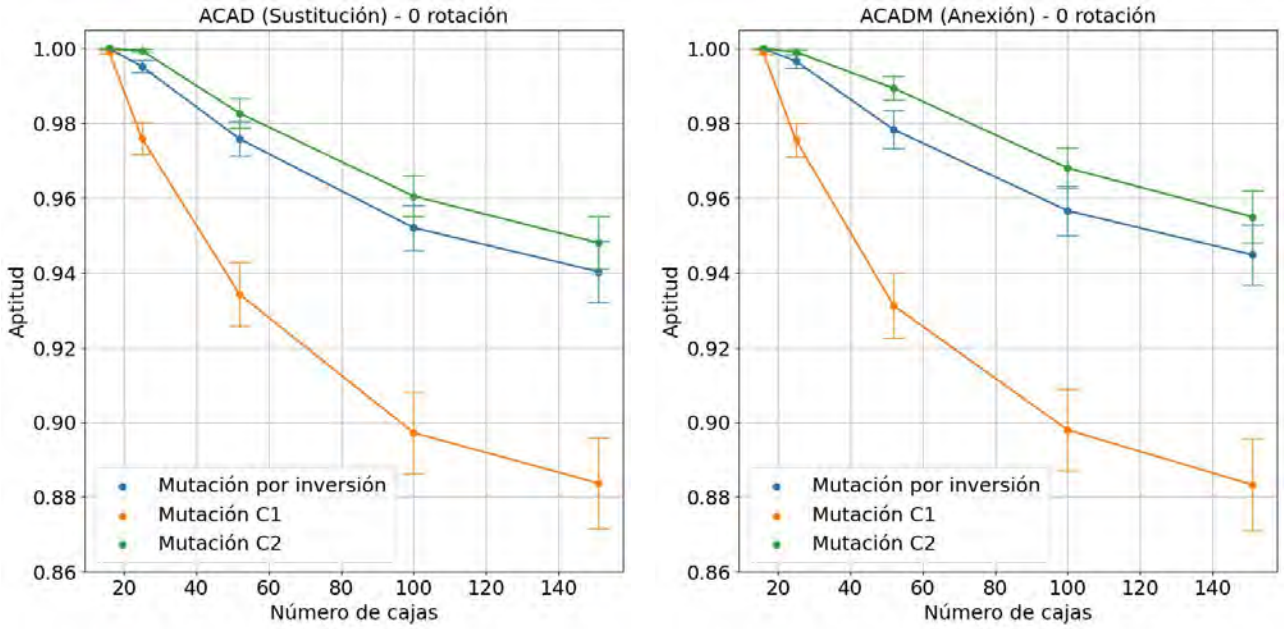
5.5.2 Resultados para 0-rotación

(a) **Aptitud promedio**

La Tabla 5.7 muestra los resultados de las metaheurísticas: Algoritmo de Colonia de Abejas Discreto (por Sustitución) y el Algoritmo de Colonia de Abejas Discreto Modificado (por Anexión). Los resultados se presentan en términos del porcentaje de ocupación del contenedor. La metaheurística con el mejor rendimiento promedio se remarca en negrita. La Figura 5.8 muestra la comparación gráfica de las metaheurísticas con sus versiones de mutación, se relaciona aptitud contra el número de cajas a empacar.

Tabla 5.7: Resultados de los algoritmos de colonia de abejas discreto con 0-rotación

	ACAD (Inv)	ACADM (Inv)	ACAD (C1)	ACADM (C1)	ACAD (C2)	ACADM (C2)
P1A2	$100.0000 \pm 0.0000\,\%$	$99.9900 \pm 0.0195\,\%$	$99.9139 \pm 0.0610\,\%$	$99.9028 \pm 0.0554\,\%$	$100.0000 \pm 0.0000\,\%$	$100.0000 \pm 0.0000\,\%$
P2A2	$99.5211 \pm 0.1614\,\%$	$99.6650 \pm 0.1804\,\%$	$97.6020 \pm 0.4202\,\%$	$97.5591 \pm 0.4560\,\%$	$99.9437 \pm 0.0479\,\%$	$99.9059 \pm 0.0601\,\%$
P3A2	$97.5820 \pm 0.4571\,\%$	$97.8323 \pm 0.5140\,\%$	$93.4226 \pm 0.8515\,\%$	$93.1213 \pm 0.8792\,\%$	$98.2699 \pm 0.3983\,\%$	$98.9394 \pm 0.3213\,\%$
P4A2	$95.2064 \pm 0.6082\,\%$	$95.6631 \pm 0.6611\,\%$	$89.7196 \pm 1.0845\,\%$	$89.7969 \pm 1.0818\,\%$	$96.0523 \pm 0.5422\,\%$	$96.8047 \pm 0.5471\,\%$
P5A2	$94.0294 \pm 0.8230\,\%$	$94.4815 \pm 0.7957\,\%$	$88.3710 \pm 1.2227\,\%$	$88.3286 \pm 1.2461\,\%$	$94.8019 \pm 0.7055\,\%$	$95.4975 \pm 0.6891\,\%$
	$97.2678 \pm 0.4099\,\%$	$97.5264 \pm 0.4341\,\%$	$93.8058 \pm 0.7280\,\%$	$93.7417 \pm 0.7437\,\%$	$97.8136 \pm 0.3388\,\%$	$\mathbf{98.2295 \pm 0.3235\,\%}$



(a) Aptitud vs. Número de cajas para el ACAD con 0-rotación.

(b) Aptitud vs. Número de cajas para el ACADM con 0-rotación.

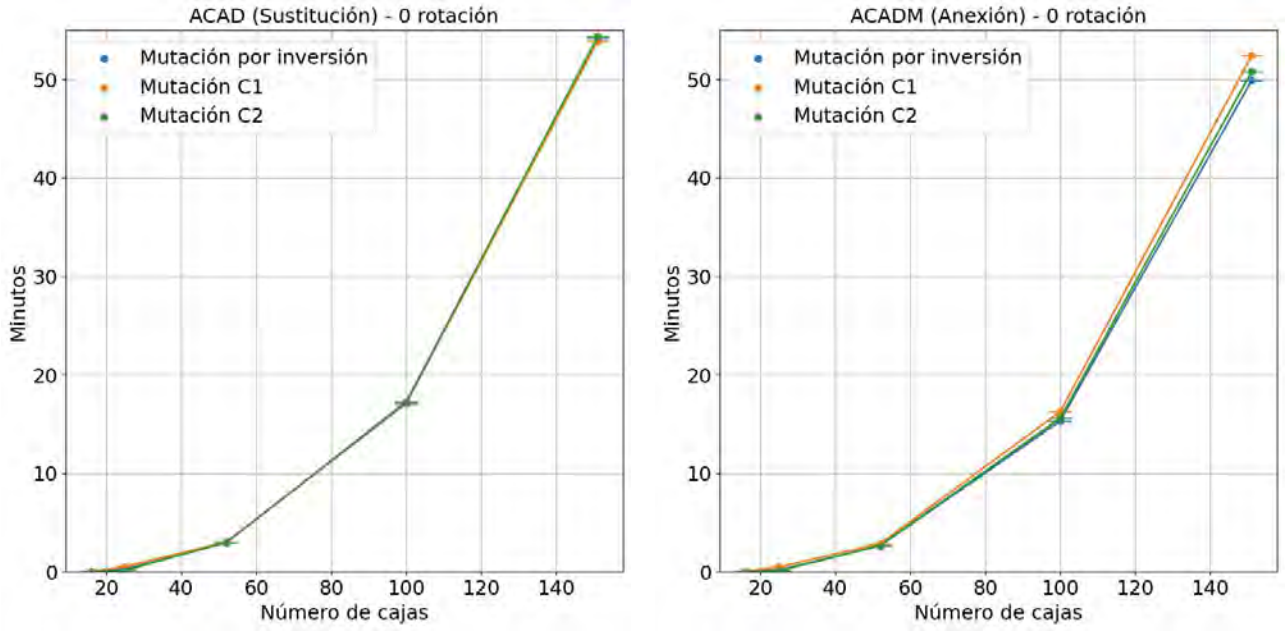
Figura 5.8: Comparativa de la aptitud promedio de los algoritmos de colonia de abeja con 0-rotación.

(b) **Tiempo promedio**

La Tabla 5.8 muestra los tiempos promedios, sin rotación, de ejecución de las metaheurísticas por cada problema y mutación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.9 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.8: Resultados de los tiempos de los algoritmos de colonia de abejas discreto con 0-rotación

	ACAD (Inv)	ACADM (Inv)	ACAD (C1)	ACADM (C1)	ACAD (C2)	ACADM (C2)
P1A2	0.0014 min	0.0027 min	0.0391 min	0.0451 min	0.0008 min	0.0028 min
P2A2	0.2692 min	0.1913 min	0.5345 min	0.4971 min	0.1456 min	0.0962 min
P3A2	2.9359 min	2.6366 min	3.0008 min	2.8511 min	2.9821 min	2.6701 min
P4A2	17.2536 min	15.3128 min	17.1090 min	16.2514 min	17.1088 min	15.6049 min
P5A2	54.2081 min	49.8696 min	53.8961 min	52.3975 min	54.3321 min	50.7722 min
	14.9336 min	**13.6026 min**	14.9159 min	14.4085 min	14.9139 min	13.8292 min



(a) Minutos vs. Número de cajas para el ACAD con 0-rotación.

(b) Minutos vs. Número de cajas para el ACADM con 0-rotación.

Figura 5.9: Resultados de los tiempos de los algoritmos de colonia de abejas discreto con 0-rotación

(c) **Análisis de resultados**

La tabla 5.7 muestra que el ACADM con mutación C2 obtuvo el mejor desempeño entre todos y el ACAD con mutación C1 obtuvo el peor desempeño. Además, para el problema P1A2, el ACAD con mutación por inversión, el ACAD con mutación C2 y el ACADM con mutación C2 obtuvieron la mejor solución en las 100 instancias.

La Figura 5.8a y la Figura 5.10a muestran que la utilización de grupo de mutación C1 da el peor rendimiento en comparación del grupo de mutación C2 y con la mutación por inversión.

En general, el grupo de mutación C2 siempre obtuvo el mejor desempeño en ACAD y ACADM, resaltando que ACADM con mutación C2 superó en rendimiento al ACAD.

La Tabla 5.8, la Figura 5.9a y la Figura 5.9b muestran que la utilización de un enfoque por anexión, es decir, usando la metahurística ACADM, tarda aproximadamente $0.93\,\%\,\%$ menos que el ACAD.

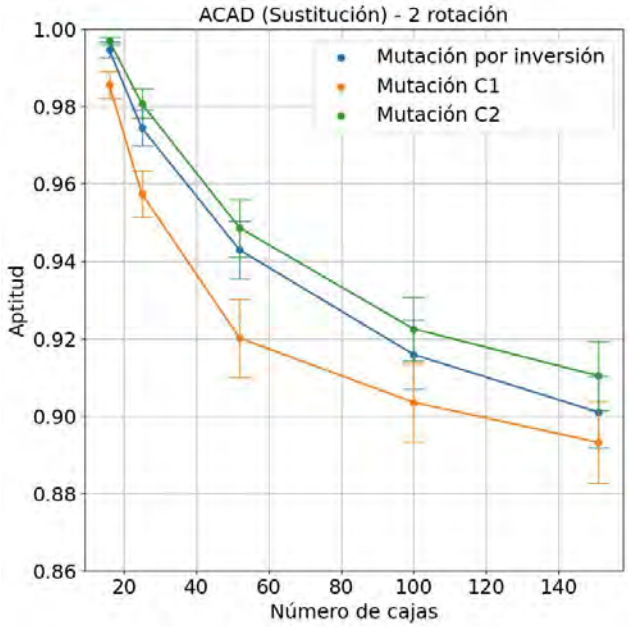
5.5.3 Resultados para 2-rotación

(a) **Aptitud promedio**

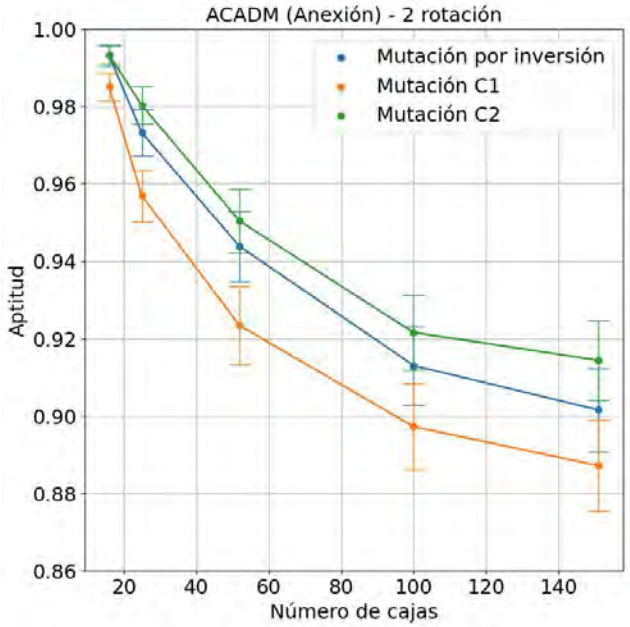
La Tabla 5.9 muestra los resultados de las metaheurísticas por cada problema y mutación. Los resultados se presentan en términos del porcentaje de ocupación del contenedor. La metaheurística con el mejor rendimiento promedio se remarca en negrita. La Figura 5.10 muestra la comparación gráfica de las metaheurísticas con sus versiones de mutación, se relaciona aptitud contra el número de cajas a empacar.

Tabla 5.9: Resultados de los algoritmos de colonia de abejas discreto con 2-rotación

	ACAD (Inv)	ACADM (Inv)	ACAD (C1)	ACADM (C1)	ACAD (C2)	ACADM (C2)
P1A2	$99.4654 \pm 0.2053\,\%$	$99.3141 \pm 0.2732\,\%$	$98.5586 \pm 0.3547\,\%$	$98.5102 \pm 0.3582\,\%$	$99.6995 \pm 0.1025\,\%$	$99.3220 \pm 0.2373\,\%$
P2A2	$97.4398 \pm 0.4687\,\%$	$97.3180 \pm 0.6004\,\%$	$95.7422 \pm 0.5962\,\%$	$95.6933 \pm 0.6660\,\%$	$98.0703 \pm 0.3787\,\%$	$98.0272 \pm 0.4854\,\%$
P3A2	$94.2887 \pm 0.7412\,\%$	$94.3793 \pm 0.9037\,\%$	$92.0110 \pm 1.0018\,\%$	$92.3362 \pm 1.0097\,\%$	$94.8586 \pm 0.7402\,\%$	$95.0343 \pm 0.8226\,\%$
P4A2	$91.5871 \pm 0.8992\,\%$	$91.3032 \pm 1.0100\,\%$	$90.3525 \pm 1.0115\,\%$	$89.7234 \pm 1.1145\,\%$	$92.2498 \pm 0.8178\,\%$	$92.1550 \pm 0.9730\,\%$
P5A2	$90.0982 \pm 0.9266\,\%$	$90.1559 \pm 1.0843\,\%$	$89.3194 \pm 1.0531\,\%$	$88.7166 \pm 1.1773\,\%$	$91.0373 \pm 0.8881\,\%$	$91.4380 \pm 1.0313\,\%$
	$94.5758 \pm 0.6482\,\%$	$94.4941 \pm 0.7743\,\%$	$93.1967 \pm 0.8034\,\%$	$92.9960 \pm 0.8651\,\%$	$95.1831 \pm 0.5855\,\%$	$\mathbf{95.1953 \pm 0.7099\,\%}$



(a) Aptitud vs. Número de cajas para el ACAD con 2-rotación.



(b) Aptitud vs. Número de cajas para el ACADM con 2-rotación.

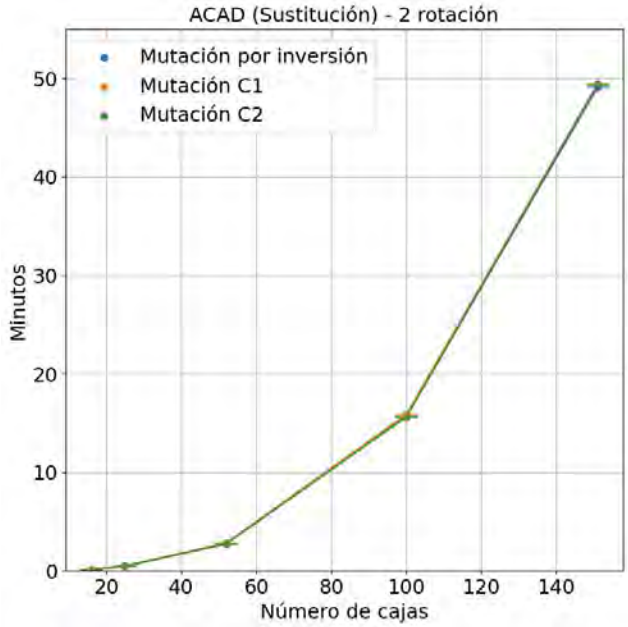
Figura 5.10: Comparativa de la aptitud promedio de los algoritmos de colonia de abeja con 2-rotación.

(b) **Tiempo promedio**

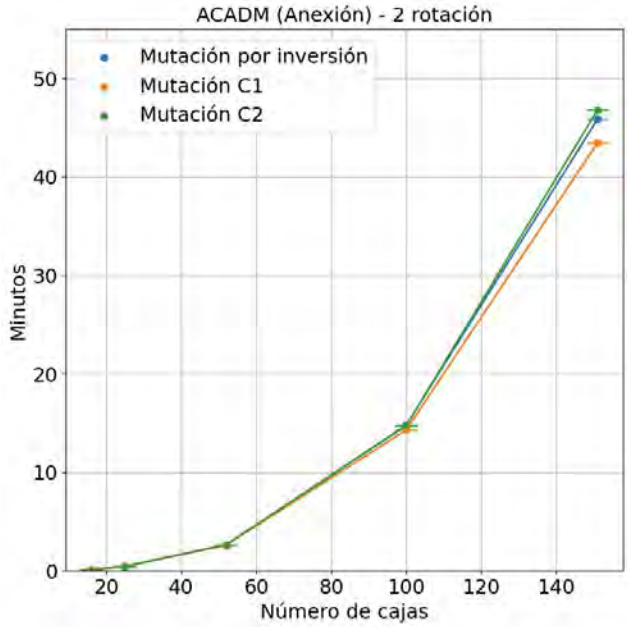
La Tabla 5.10 muestra los tiempos promedios, con 2-rotación, de ejecución de las metaheurísticas por cada problema y mutación en 2-rotación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.11 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.10: Resultados de los tiempos de los algoritmos de colonia de abejas con 2-rotación

	ACAD (Inv)	ACADM (Inv)	ACAD (C1)	ACADM (C1)	ACAD (C2)	ACADM (C2)
P1A2	0.1136 min	0.1061 min	0.1507 min	0.1514 min	0.1243 min	0.1202 min
P2A2	0.4776 min	0.4464 min	0.4904 min	0.4706 min	0.4798 min	0.4613 min
P3A2	2.7384 min	2.6200 min	2.8102 min	2.6052 min	2.7452 min	2.6341 min
P4A2	15.6552 min	14.7749 min	15.8448 min	14.3019 min	15.6230 min	14.7341 min
P5A2	49.1174 min	45.8703 min	49.4041 min	43.4589 min	49.3505 min	46.7931 min
	13.6204 min	12.7635 min	13.7400 min	**12.1976 min**	13.6646 min	12.9486 min



(a) Minutos vs. Número de cajas para el ACAD con 2-rotación.



(b) Minutos vs. Número de cajas para el ACADM con 2-rotación.

Figura 5.11: Resultados de los tiempos de los algoritmos de colonia de abejas con 2-rotación

(c) **Análisis de resultados**

La Tabla 5.9 muestran la aptitud (porcentual) promedio utilizando 2-rotación muestra que el ACADM con mutación C2 obtuvo el mejor rendimiento; mientras que el ACADM con mutación C1 obtuvo el peor rendimiento, con una diferencia promedio aproximada de $3\,\%$.

La Figura 5.8a y la Figura 5.8b muestran que la utilización de la mutación C1 da el peor rendimiento entre ambos algoritmos; mientras que la utilización de la mutación C2 dio los mejores resultados. Este comportamiento es similar para 0-rotación, Figura 5.8, aunque las curvas de los algoritmos que usan la mutación C2 y, por inversión están más cercanas a la mutación C1.

La Tabla 5.8, la Figura 5.9a y la Figura 5.9 muestran que el ACADM tarda, en promedio aproximado, $0.95\,\%$ menos. Este resultado es similar al obtenido con 0-rotación.

5.5.4 Resultados para 6-rotación

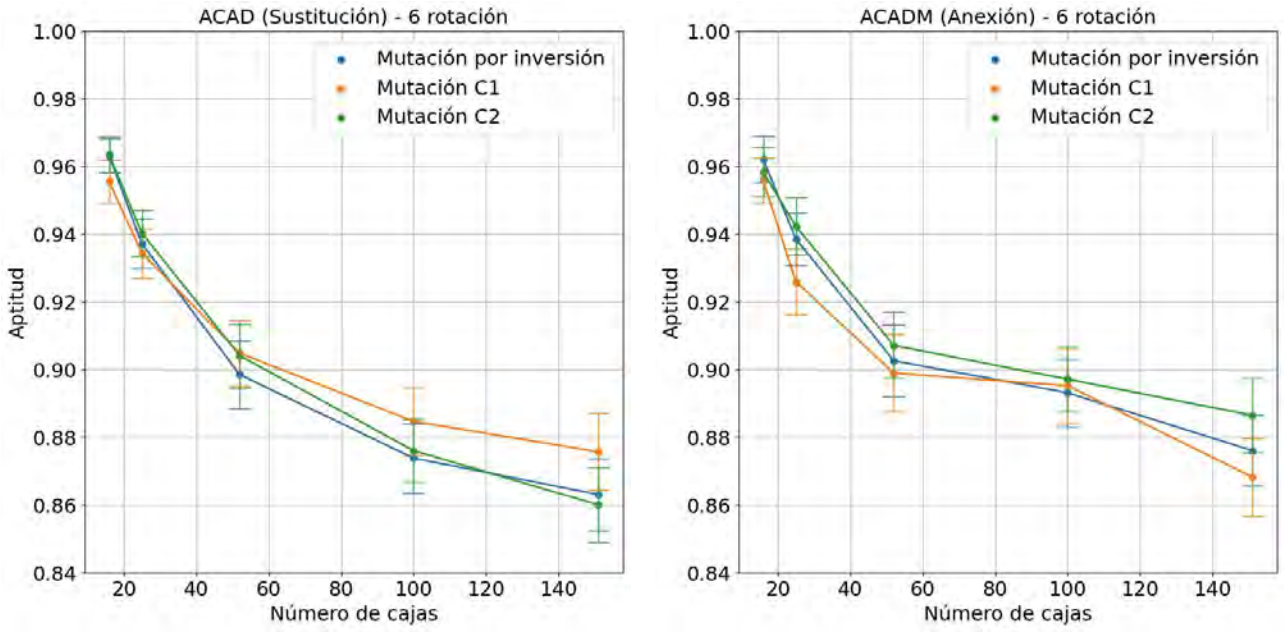
(a) **Aptitud promedio**

La Tabla 5.11 muestra los resultados de las metaheurísticas por cada problema y mutación

en 6-rotación. Los resultados se presentan en términos del porcentaje de ocupación del contenedor. La metaheurística con el mejor rendimiento promedio se remarca en negrita. La Figura 5.12 muestra la comparación gráfica de las metaheurísticas con sus versiones de mutación, se relaciona aptitud contra el número de cajas a empacar.

Tabla 5.11: Tabla de resultados con 6 rotación

	ACAD (Inv)	ACADM (Inv)	ACAD (C1)	ACADM (C1)	ACAD (C2)	ACADM (C2)
P1A2	$96.3178 \pm 0.5044\,\%$	$96.2106 \pm 0.6757\,\%$	$95.5558 \pm 0.6296\,\%$	$95.5910 \pm 0.6684\,\%$	$96.3542 \pm 0.5073\,\%$	$95.8305 \pm 0.7241\,\%$
P2A2	$93.7220 \pm 0.7215\,\%$	$93.8645 \pm 0.7840\,\%$	$93.4451 \pm 0.7300\,\%$	$92.6010 \pm 0.9740\,\%$	$94.0311 \pm 0.6826\,\%$	$94.2362 \pm 0.8501\,\%$
P3A2	$89.8514 \pm 0.9901\,\%$	$90.2635 \pm 1.0616\,\%$	$90.4864 \pm 0.9810\,\%$	$89.8965 \pm 1.1508\,\%$	$90.4122 \pm 0.9435\,\%$	$90.7151 \pm 0.9816\,\%$
P4A2	$87.3853 \pm 1.0280\,\%$	$89.3132 \pm 0.9938\,\%$	$88.4682 \pm 1.0096\,\%$	$89.5238 \pm 1.0993\,\%$	$87.6062 \pm 0.9210\,\%$	$89.7153 \pm 0.9626\,\%$
P5A2	$86.3118 \pm 1.0677\,\%$	$87.6108 \pm 1.0516\,\%$	$87.5744 \pm 1.1282\,\%$	$86.8247 \pm 1.1454\,\%$	$86.0047 \pm 1.1012\,\%$	$88.6505 \pm 1.0866\,\%$
	$90.7176 \pm 0.8623\,\%$	$91.4525 \pm 0.9133\,\%$	$91.1060 \pm 0.8957\,\%$	$90.8874 \pm 1.0076\,\%$	$90.8817 \pm 0.8311\,\%$	$\mathbf{91.8295 \pm 0.9210\,\%}$



(a) Aptitud vs. Número de cajas para el ACAD con 6-rotación.

(b) Aptitud vs. Número de cajas para el ACADM con 6-rotación.

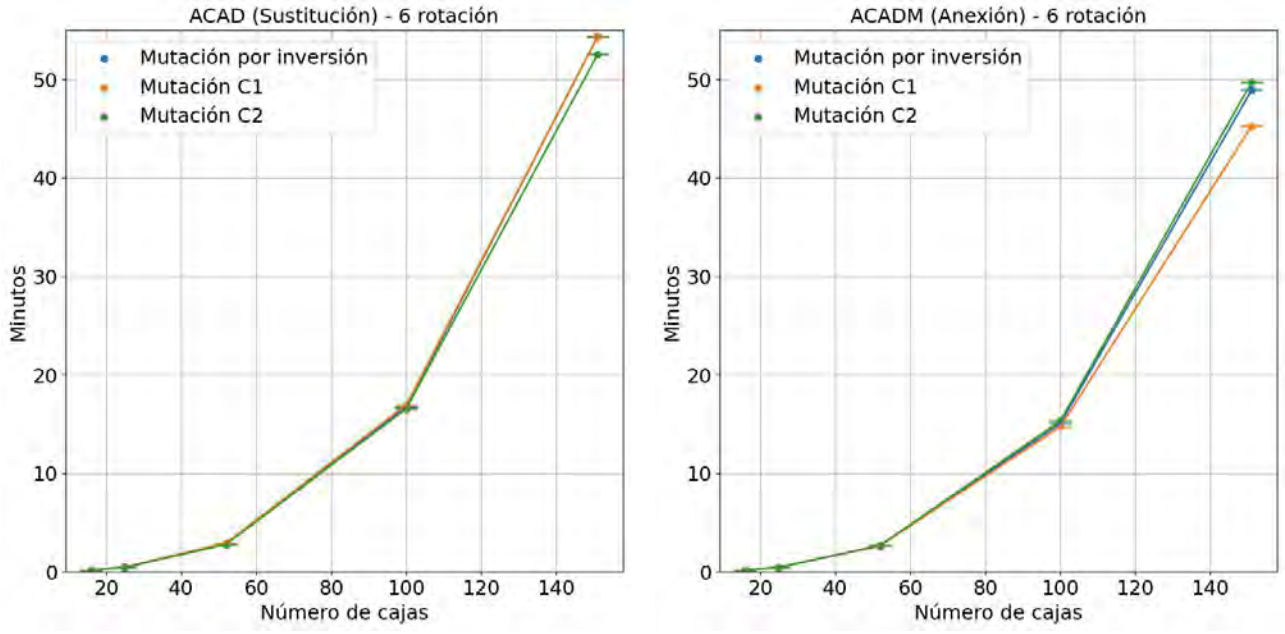
Figura 5.12: Comparativa de la aptitud promedio de los algoritmos de colonia de abeja con 6-rotación.

(b) **Tiempo promedio**

La Tabla 5.12 muestra los tiempos promedios, con 6-rotación, de ejecución de las metaheurísticas por cada problema y mutación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.13 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.12: Resultados de los tiempos de los algoritmos de colonia de abejas discreto con 6-rotación

	ACAD (Inv)	ACADM (Inv)	ACAD (C1)	ACADM (C1)	ACAD (C2)	ACADM (C2)
P1A2	0.1816 min	0.1774 min	0.1902 min	0.1784 min	0.1839 min	0.1850 min
P2A2	0.4606 min	0.4606 min	0.4908 min	0.4609 min	0.4640 min	0.4592 min
P3A2	2.8132 min	2.6355 min	2.9209 min	2.6488 min	2.7887 min	2.6771 min
P4A2	16.7133 min	15.0290 min	16.9435 min	14.7049 min	16.5485 min	15.3031 min
P5A2	54.3355 min	48.9073 min	54.2905 min	45.2528 min	52.5237 min	49.7059 min
	14.9008 min	13.4419 min	14.9672 min	**12.6492 min**	14.5018 min	13.6660 min



(a) Minutos vs. Número de cajas para el ACAD con 6-rotación.

(b) Minutos vs. Número de cajas para el ACADM con 6-rotación.

Figura 5.13: Resultados de los tiempos de los algoritmos de colonia de abejas discreto con 6-rotación

(c) **Análisis de resultados**

La Tabla 5.11 muestra que, para 6-rotación, el ACADM con mutación C2 obtuvo el mejor desempeño de todos los evaluados. Sin embargo, esta superioridad no es tan significativa como en 0-rotación, y 2-rotación. Además, en la mayoría de los casos, los puntos generados por cada problema caen dentro del intervalo de confianza del rendimiento medido de algoritmos.

La Figura 5.12a muestra que usando la mutación C1 para el problema P5A2, obtiene el

mejor rendimiento que las demás mutaciones, contradiciendo la expectativa que usando la mutación C2 tendría el mejor desempeño ya que para el caso de 0-rotación y 2-rotación mostrada el mejor desempeño. Sin embargo, para el ACADM, Figura 5.12b, se mantiene la expectativa que la mutación C2 obtiene los mejores resultados.

Para 6-rotación, las curvas generadas por ambos algoritmos con mutación C2 y mutación por inversión están más cercanas entre sí, un comportamiento similar a los algoritmos genéticos híbridos donde estas curvas siempre se mantuvieron cercanas entre sí.

La Tabla 5.12, la Figura 5.13a y la Figura 5.13b muestran un comportamiento similar con 0-rotación y 2-rotación, el ACADM tarda un $0.94\,\%$ menos que el ACAD.

5.6 Resultados de los algoritmos de luciérnaga

5.6.1 Parámetros del algoritmo de luciérnagas

Los parámetros del algoritmo de luciérnagas son:

- Número de luciérnagas: $n_{lucirnagas} = 25$.
- Factor de actualización: $m = 2$
- Probabilidad de cruza: $p_r = 1.0$.
- Probabilidad de mutación $p_m = 1.0$
- Condición de término: si una libélula tiene aptitud de 1.0.

5.6.2 Resultados para 0-rotación

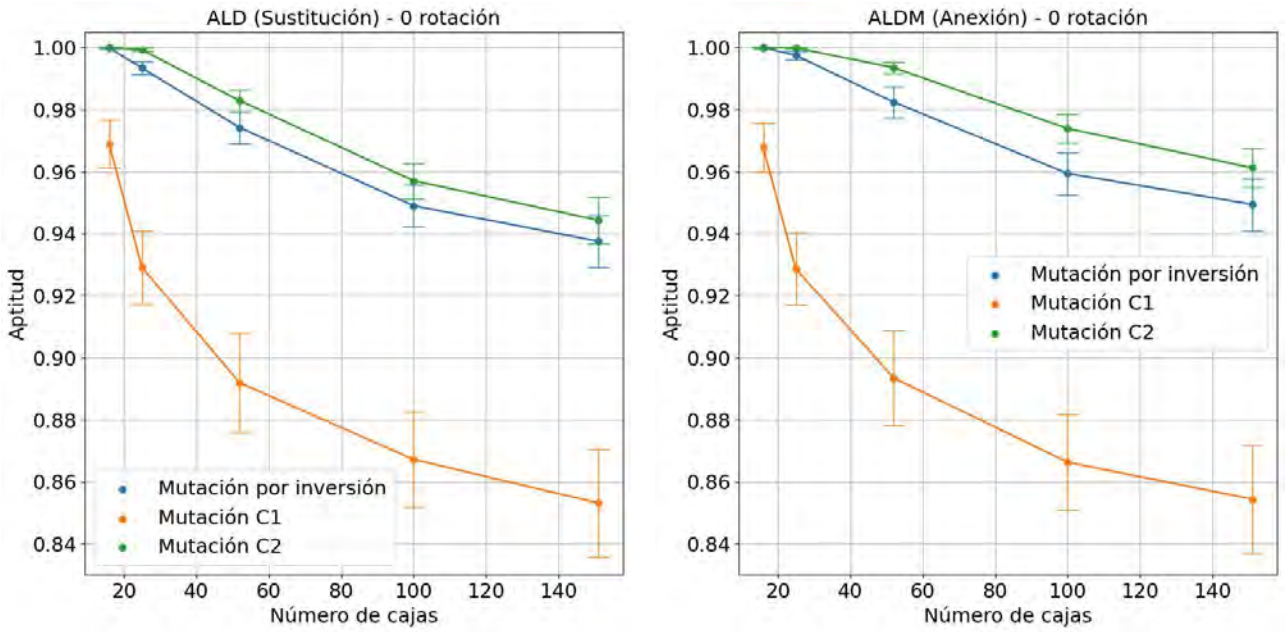
(a) **Aptitud promedio**

La Tabla 5.14 muestra los resultados de las metaheurísticas: Algoritmo de Luciérnagas Discreto (por Sustitución) y el Algoritmo de Luciérnagas Discreto Modificado (por Anexión). Los resultados se presentan en términos del porcentaje de ocupación del contenedor. La metaheurística con el mejor rendimiento promedio se remarca en negrita. La Figura 5.15

muestra la comparación gráfica de las metaheurísticas con sus versiones de mutación, se relaciona aptitud contra el número de cajas a empacar.

Tabla 5.14: Resultados de los algoritmos de luciérnagas discreto con 0-rotación

	ALD (Inv)	ALDM (Inv)	ALD (C1)	ALDM (C1)	ALD (C2)	ALDM (C2)
P1A2	$99.9900 \pm 0.0195\,\%$	$100.0000 \pm 0.0000\,\%$	$96.8979 \pm 0.7826\,\%$	$96.7722 \pm 0.7834\,\%$	$100.0000 \pm 0.0000\,\%$	$99.9825 \pm 0.0341\,\%$
P2A2	$99.3384 \pm 0.2040\,\%$	$99.7421 \pm 0.1416\,\%$	$92.9197 \pm 1.1817\,\%$	$92.8677 \pm 1.1629\,\%$	$99.9252 \pm 0.0490\,\%$	$99.9864 \pm 0.0163\,\%$
P3A2	$97.4170 \pm 0.5186\,\%$	$98.2391 \pm 0.5061\,\%$	$89.1867 \pm 1.5973\,\%$	$89.3443 \pm 1.5293\,\%$	$98.2837 \pm 0.3486\,\%$	$99.3460 \pm 0.1857\,\%$
P4A2	$94.8970 \pm 0.6717\,\%$	$95.9355 \pm 0.6841\,\%$	$86.7178 \pm 1.5328\,\%$	$86.6265 \pm 1.5396\,\%$	$95.7006 \pm 0.5660\,\%$	$97.3872 \pm 0.4591\,\%$
P5A2	$93.7503 \pm 0.8410\,\%$	$94.9402 \pm 0.8380\,\%$	$85.3109 \pm 1.7301\,\%$	$85.4367 \pm 1.7393\,\%$	$94.4292 \pm 0.7560\,\%$	$96.1148 \pm 0.6194\,\%$
	$97.0785 \pm 0.4510\,\%$	$97.7714 \pm 0.4340\,\%$	$90.2066 \pm 1.3649\,\%$	$90.2095 \pm 1.3509\,\%$	$97.6677 \pm 0.3439\,\%$	$\mathbf{98.5634 \pm 0.2629\,\%}$



(a) Aptitud vs. Número de cajas para el ALD con 0-rotación.

(b) Aptitud vs. Número de cajas para el ALDM con 0-rotación.

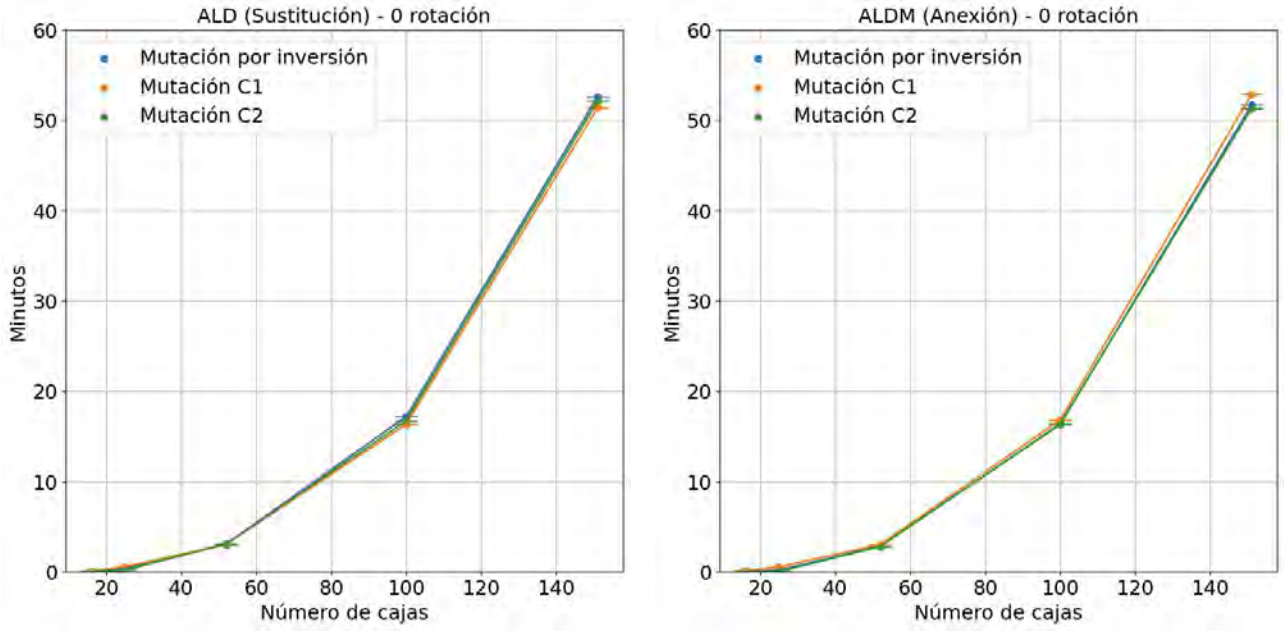
Figura 5.14: Comparativa de la aptitud promedio de los algoritmos de luciérnaga discreto con 0-rotación.

(b) **Tiempo promedio**

La Tabla 5.15 muestra los tiempos promedios de ejecución de las metaheurísticas por cada problema y mutación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.15 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.15: Resultados de los tiempos de los algoritmos de luciérnagas discreto con 0-rotación

	ALD (Inv)	ALDM (Inv)	ALD (C1)	ALDM (C1)	ALD (C2)	ALDM (C2)
P1A2	0.0053 min	0.0026 min	0.1398 min	0.1523 min	0.0024 min	0.0038 min
P2A2	0.3590 min	0.1788 min	0.5502 min	0.5438 min	0.1939 min	0.0567 min
P3A2	3.0660 min	2.8360 min	3.0395 min	3.0135 min	3.0608 min	2.7798 min
P4A2	17.1898 min	16.3164 min	16.3354 min	16.8088 min	16.7271 min	16.3804 min
P5A2	52.6225 min	51.7083 min	51.4069 min	52.8920 min	52.1233 min	51.3088 min
	14.6485 min	14.2084 min	14.2944 min	14.6821 min	14.4215 min	**14.1059 min**



(a) Minutos vs. Número de cajas para el ALD con 0-rotación.

(b) Minutos vs. Número de cajas para el ALDM con 0-rotación.

Figura 5.15: Resultados de los tiempos de los algoritmos de luciérnagas discreto con 0-rotación

(c) **Análisis de resultados**

Los resultados sin rotación, Tabla 5.14 y Figura 5.16a, para ambas versiones con la mutación C1 obtuvieron el peor desempeño. Mientras que el uso del operador C2 obtuvo el mejor desempeño. El ALDM con mutación C2 obtuvo la menor *varianza* y mayor aptitud promedio que las demás mutaciones.

Las gráficas Figura 5.14a y Figura 5.16a se muestra que la distancia entre curvas con la mutación C1 es significativamente grande. Este comportamiento es similar al de abejas, ver

Figura 5.8, con la diferencia que para el problema P1A2, usando la mutación C1, no daba un pésimo desempeño, en comparación de las otras mutaciones, como el de luciérnagas.

Los tiempos promedios, los resultados de la Tabla 5.15, la Figura 5.15a y la Figura 5.15b, se observan que duraron aproximadamente lo mismo en todos los problemas.

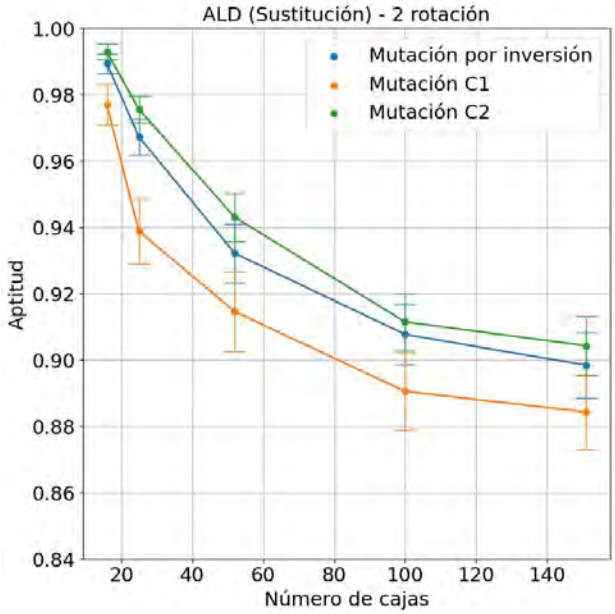
5.6.3 Resultados para 2-rotación

(a) **Aptitud promedio**

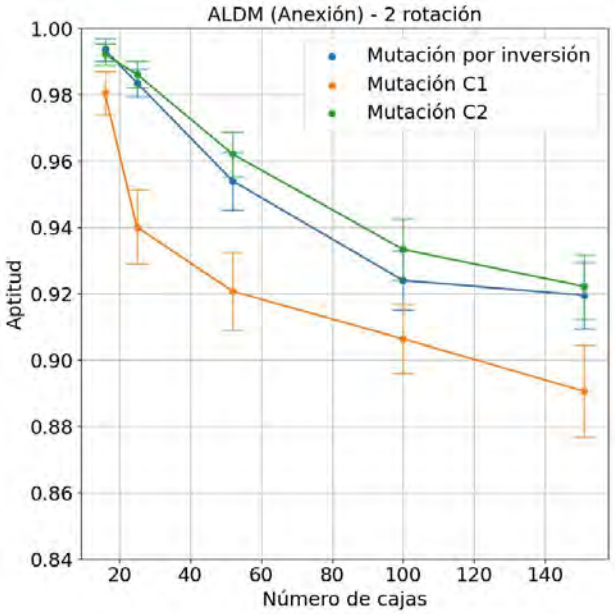
La Tabla 5.16 muestra los resultados de las metaheurísticas para 2-rotación. Los resultados se presentan en términos del porcentaje de ocupación del contenedor. La metaheurística con el mejor rendimiento promedio se remarca en negrita. La Figura 5.17 muestra la comparación gráfica de las metaheurísticas con sus versiones de mutación, se relaciona aptitud contra el número de cajas a empacar.

Tabla 5.16: Resultados de los algoritmos de luciérnagas discreto con 2-rotación

	ALD (Inv)	ALDM (Inv)	ALD (C1)	ALDM (C1)	ALD (C2)	ALDM (C2)
P1A2	$98.9355 \pm 0.2965\,\%$	$99.3536 \pm 0.3415\,\%$	$97.6906 \pm 0.6078\,\%$	$98.0415 \pm 0.6587\,\%$	$99.2849 \pm 0.2364\,\%$	$99.2175 \pm 0.3141\,\%$
P2A2	$96.7299 \pm 0.5496\,\%$	$98.3575 \pm 0.4302\,\%$	$93.8818 \pm 0.9791\,\%$	$94.0082 \pm 1.1230\,\%$	$97.5578 \pm 0.3940\,\%$	$98.6125 \pm 0.3896\,\%$
P3A2	$93.2162 \pm 0.8772\,\%$	$95.3964 \pm 0.8734\,\%$	$91.4601 \pm 1.1988\,\%$	$92.0703 \pm 1.1697\,\%$	$94.3032 \pm 0.7226\,\%$	$96.2063 \pm 0.6663\,\%$
P4A2	$90.7725 \pm 0.8987\,\%$	$92.3970 \pm 0.8907\,\%$	$89.0548 \pm 1.1638\,\%$	$90.6370 \pm 1.0436\,\%$	$91.1472 \pm 0.8586\,\%$	$93.3322 \pm 0.9220\,\%$
P5A2	$89.8404 \pm 0.9851\,\%$	$91.9492 \pm 0.9988\,\%$	$88.4304 \pm 1.1354\,\%$	$89.0560 \pm 1.3865\,\%$	$90.4306 \pm 0.9010\,\%$	$92.2178 \pm 0.9623\,\%$
	$93.8989 \pm 0.7214\,\%$	$95.4907 \pm 0.7069\,\%$	$92.1036 \pm 1.0170\,\%$	$92.7626 \pm 1.0763\,\%$	$94.5447 \pm 0.6225\,\%$	$\mathbf{95.9173 \pm 0.6509\,\%}$



(a) Aptitud vs. Número de cajas para el ALD con 2-rotación.



(b) Aptitud vs. Número de cajas para el ALDM con 2-rotación.

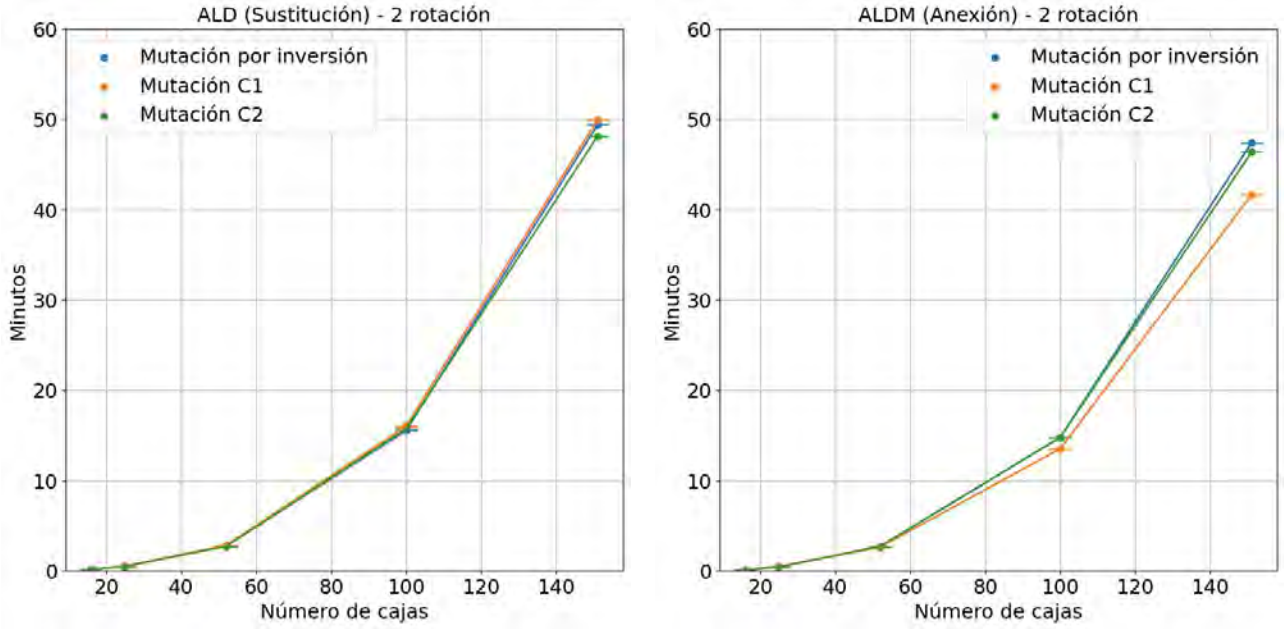
Figura 5.16: Comparativa de la aptitud promedio de los algoritmos de luciérnaga discreto con 2-rotación.

(b) **Tiempo promedio**

La Tabla 5.17 muestra los tiempos promedios de ejecución de las metaheurísticas por cada problema y mutación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.17 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.17: Resultados de los tiempos de los algoritmos de luciérnagas discreto con 2-rotación

	ALD (Inv)	ALDM (Inv)	ALD (C1)	ALDM (C1)	ALD (C2)	ALDM (C2)
P1A2	0.1665 min	0.0843 min	0.1783 min	0.1230 min	0.1701 min	0.1035 min
P2A2	0.5024 min	0.4347 min	0.5081 min	0.4807 min	0.5004 min	0.4471 min
P3A2	2.7375 min	2.6921 min	2.8192 min	2.6164 min	2.7005 min	2.6954 min
P4A2	15.5397 min	14.7716 min	16.0371 min	13.5175 min	15.7191 min	14.7662 min
P5A2	49.3842 min	47.4171 min	49.9308 min	41.6575 min	48.1302 min	46.3875 min
	13.6660 min	13.0800 min	13.8947 min	**11.6790 min**	13.4441 min	12.8799 min



(a) Minutos vs. Número de cajas para el ALD con 2-rotación.

(b) Minutos vs. Número de cajas para el ALDM con 2-rotación.

Figura 5.17: Resultados de los tiempos de los algoritmos de luciérnagas discreto con 2-rotación

(c) **Análisis de resultados**

Los resultados 2-rotación, mostrados en la Tabla 5.16, la Figura 5.16a y la Figura 5.14a, para ambas versiones con la mutación C1 obtienen el peor desempeño, pero la diferencia entre los demás operadores de mutación es menor; comportamiento similar al de las abejas, ver Figura 5.10a. El uso del operador C2 obtuvo el mejor desempeño, aunque con la mutación por inversión entran en los *intervalos de confianza*.

En los tiempos, mostrados en la Figura 5.15a y la Figura 5.17a, se observa que el ALDM duro menos que el ALD, aproximadamente $0.96\,\%$ menos. El uso del operador C1 con el algoritmo modificado obtuvo el menor tiempo promedio, aunque también obtuvo el peor rendimiento.

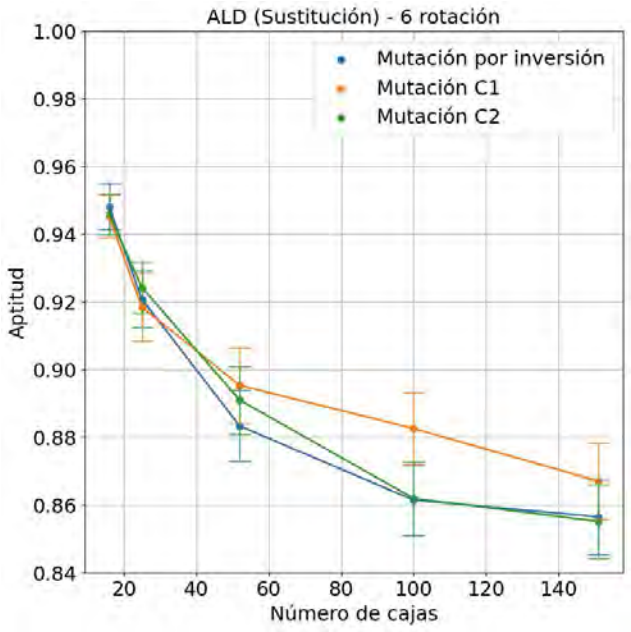
5.6.4 Resultados para 6-rotación

(a) **Aptitud promedio**

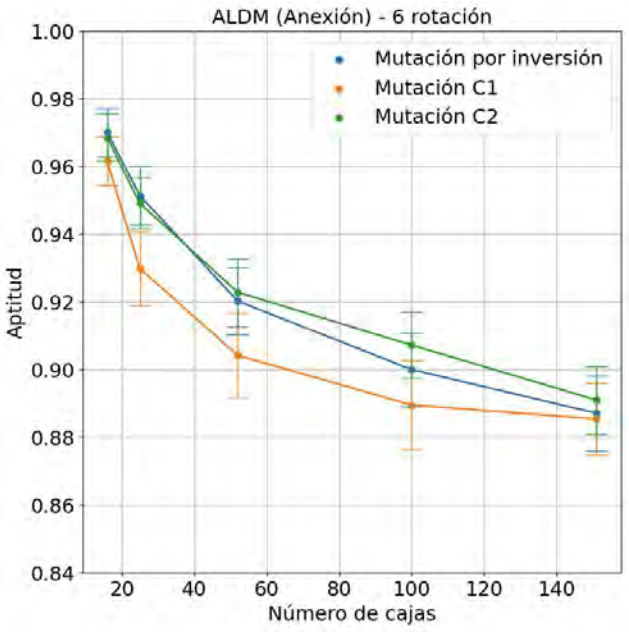
La Tabla 5.18 muestra los resultados de las metaheurísticas para 6-rotación. Los resultados se presentan en términos del porcentaje de ocupación del contenedor. La metaheurística con el mejor rendimiento promedio se remarca en negrita. La Figura 5.18 muestra la comparación gráfica de las metaheurísticas con sus versiones de mutación, se relaciona aptitud contra el número de cajas a empacar.

Tabla 5.18: Resultados de los algoritmos de luciérnagas discreto con 6-rotación

	ALD (Inv)	ALDM (Inv)	ALD (C1)	ALDM (C1)	ALD (C2)	ALDM (C2)
P1A2	$94.8108 \pm 0.6709\,\%$	$97.0019 \pm 0.7157\,\%$	$94.5301 \pm 0.6290\,\%$	$96.1606 \pm 0.7132\,\%$	$94.5894 \pm 0.5992\,\%$	$96.8592 \pm 0.6903\,\%$
P2A2	$92.0827 \pm 0.8381\,\%$	$95.1366 \pm 0.8562\,\%$	$91.8497 \pm 1.0089\,\%$	$92.9859 \pm 1.1022\,\%$	$92.4321 \pm 0.7558\,\%$	$94.9112 \pm 0.7550\,\%$
P3A2	$88.3370 \pm 1.0503\,\%$	$92.0236 \pm 0.9941\,\%$	$89.5348 \pm 1.1135\,\%$	$90.4208 \pm 1.2486\,\%$	$89.0990 \pm 1.0002\,\%$	$92.2823 \pm 0.9997\,\%$
P4A2	$86.1480 \pm 1.0441\,\%$	$89.9994 \pm 1.0759\,\%$	$88.2592 \pm 1.0528\,\%$	$88.9565 \pm 1.3097\,\%$	$86.1969 \pm 1.0771\,\%$	$90.7231 \pm 0.9799\,\%$
P5A2	$85.6524 \pm 1.0962\,\%$	$88.7133 \pm 1.1075\,\%$	$86.7089 \pm 1.1374\,\%$	$88.5393 \pm 1.0613\,\%$	$85.5191 \pm 1.0932\,\%$	$89.1002 \pm 1.0080\,\%$
	$89.4062 \pm 0.9399\,\%$	$92.5750 \pm 0.9499\,\%$	$90.1765 \pm 0.9883\,\%$	$91.4126 \pm 1.0870\,\%$	$89.5673 \pm 0.9051\,\%$	$\mathbf{92.7752 \pm 0.8866\,\%}$



(a) Aptitud vs. Número de cajas para el ALD con 6-rotación.



(b) Aptitud vs. Número de cajas para el ALDM con 6-rotación.

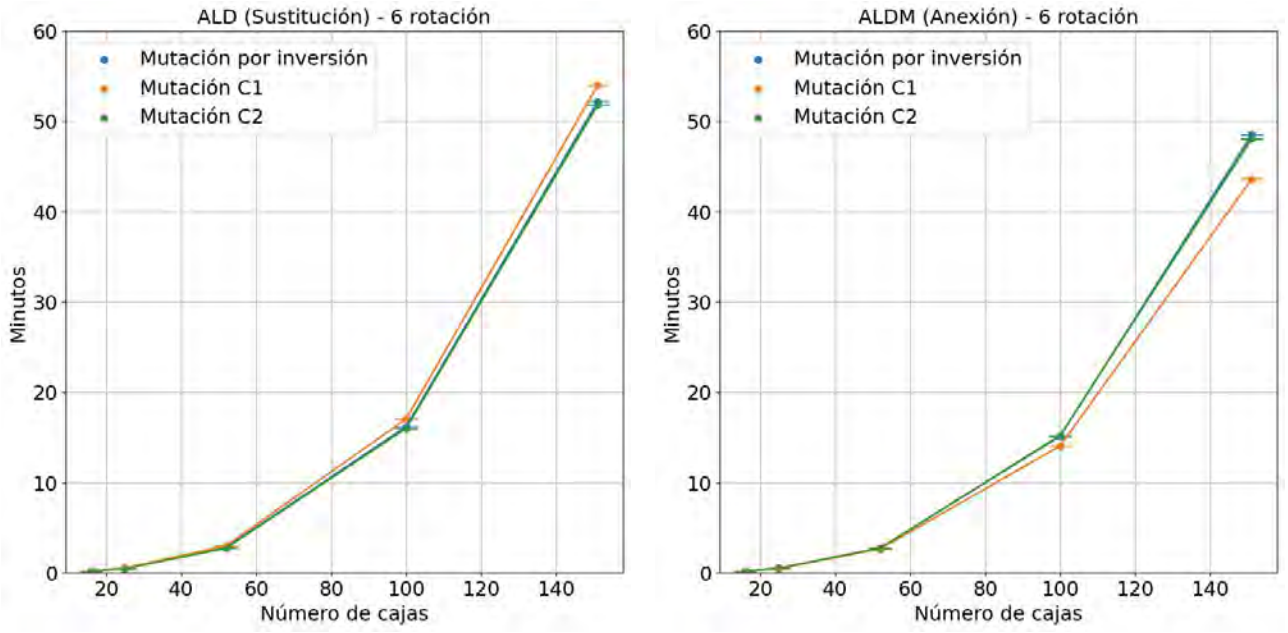
Figura 5.18: Comparativa de la aptitud promedio de los algoritmos de luciérnaga discreto con 6-rotación.

(b) **Tiempo promedio**

La Tabla 5.19 muestra los tiempos promedios de ejecución de las metaheurísticas por cada problema y mutación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.19 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.19: Resultados de los tiempos de los algoritmos de luciérnagas discreto con 6-rotación

	ALD (Inv)	ALDM (Inv)	ALD (C1)	ALDM (C1)	ALD (C2)	ALDM (C2)
P1A2	0.2018 min	0.1865 min	0.2184 min	0.1951 min	0.2005 min	0.2075 min
P2A2	0.4873 min	0.5071 min	0.5319 min	0.4929 min	0.4721 min	0.5174 min
P3A2	2.7906 min	2.7296 min	2.9799 min	2.6445 min	2.7238 min	2.7072 min
P4A2	16.1896 min	15.1372 min	17.0408 min	14.0524 min	16.0114 min	15.2160 min
P5A2	52.1581 min	48.4974 min	54.0218 min	43.5986 min	51.7937 min	48.0527 min
	14.3655 min	13.4116 min	14.9586 min	**12.1967 min**	14.2403 min	13.3402 min



(a) Minutos vs. Número de cajas para el ALD con 6-rotación.

(b) Minutos vs. Número de cajas para el ALDM con 6-rotación.

Figura 5.19: Resultados de los tiempos de los algoritmos de luciérnagas discreto con 6-rotación

(c) **Análisis de resultados**

Los resultados de 6-rotación solo para el ALD, ver Tabla 5.18 y Figura 5.18a, muestran que

usando la mutación C1, supera en rendimiento a las demás rotaciones. Este comportamiento también es visto con el algoritmo de abejas discreto (ACAD), ver Figura 5.12a, donde usando la mutación C1 también obtuvo el mejor desempeño promedio que las demás mutaciones, mientras que para 0-rotación y 2-rotación este comportamiento no se presenta.

El algoritmo modificado ALDM, ver Tabla 5.18 y Figura 5.18b, con mutación C1 sólo obtuvo valores más cercados a las demás mutaciones en el P5A2, el uso del operador C2 supera el rendimiento de las demás mutaciones. Comparando con el ALD, el ALDM supera el rendimiento del ALD.

Los tiempos, ver Figura 5.19a, muestran que el ALDM tardó menos que el ALD, aproximadamente $0.94\,\%$ menos. El uso del operador de mutación C1 obtuvo menores tiempos.

5.7 Comparativa de metatheurísticas por anexión con mutación C2

De acuerdo a los resultados anteriores, se observa que las metaheurísticas que tienen un enfoque donde las nuevas soluciones se anexan al conjunto actual de soluciones obtuvieron, en promedio, el mejor desempeño que, su contra parte el enfoque por sustitución. Además, en este este mismo conjunto se observa que la utilización del grupo de mutación C2 obtuvo, en promedio, el mejor desempeño entre mutaciones por cada metaheurística.

A continuación, se hace un análisis comparando las metaheurísticas con enfoque por anexión y usando el grupo de mutación C2, estas son: algoritmo genético híbrido no-modificado (GAH), algoritmo de colonia de abejas discreto modificado (ACADM) y el algoritmo de luciérnagas discreto modificado (ALDM).

5.7.1 Resultados para 0-rotación

(a) **Aptitud promedio**

En la Tabla 5.20 se muestran los promedios del porcentaje de ocupación en el contenedor sin considerar rotaciones. Se remarca en negrita la metaheurística

con el mejor rendimiento. La Figura 5.20 muestra la gráfica Aptitud vs Número de Cajas en donde cada curva representa el uso de una metaheurística.

Tabla 5.20: Tabla de aptitud porcentual promedio entre metaheurísticas con 0-rotación

	GAH (C2)	ACADM (C2)	ALDM (C2)
P1A2	$99.9271 \pm 0.0514\,\%$	$100.0000 \pm 0.0000\,\%$	$99.9825 \pm 0.0341\,\%$
P2A2	$99.0830 \pm 0.2190\,\%$	$99.9059 \pm 0.0601\,\%$	$99.9864 \pm 0.0163\,\%$
P3A2	$97.5416 \pm 0.4348\,\%$	$98.9394 \pm 0.3213\,\%$	$99.3460 \pm 0.1857\,\%$
P4A2	$95.6386 \pm 0.6147\,\%$	$96.8047 \pm 0.5471\,\%$	$97.3872 \pm 0.4591\,\%$
P5A2	$94.5097 \pm 0.6872\,\%$	$95.4975 \pm 0.6891\,\%$	$96.1148 \pm 0.6194\,\%$
	$97.3400 \pm 0.4014\,\%$	$98.2295 \pm 0.3235\,\%$	$\mathbf{98.5634 \pm 0.2629\,\%}$

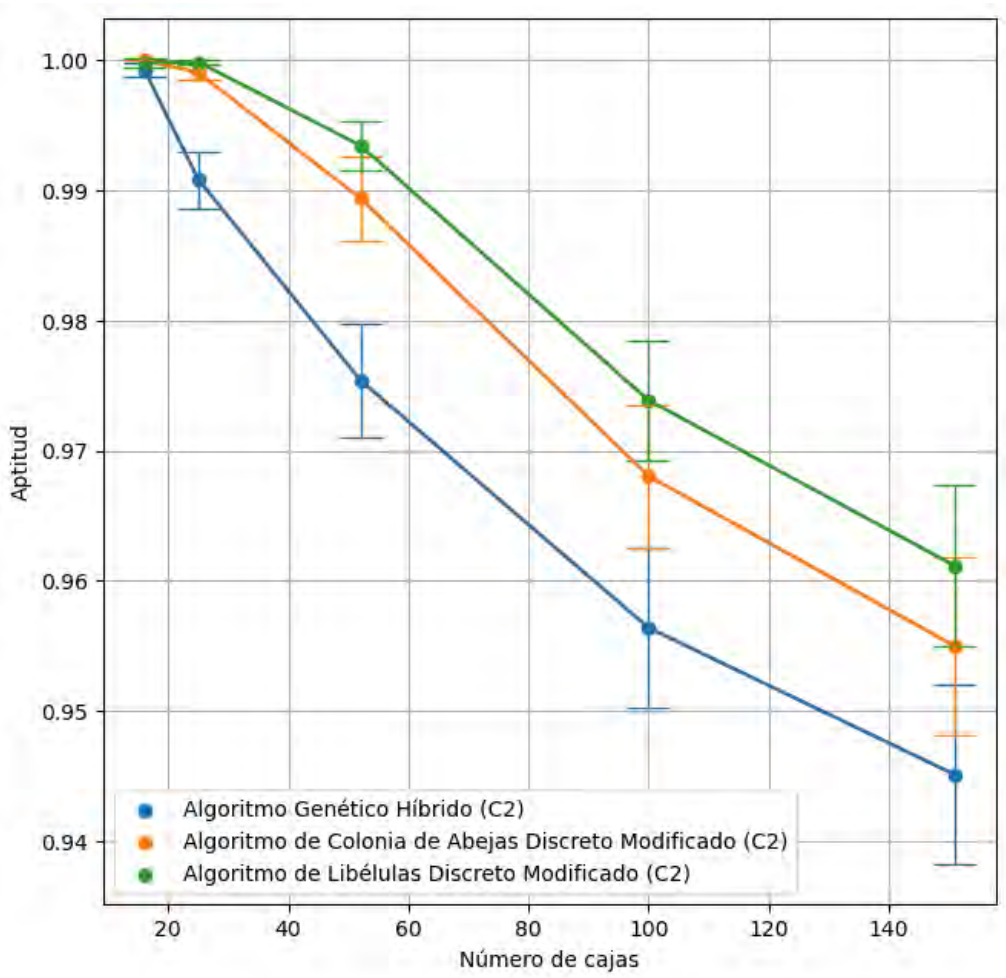


Figura 5.20: Comparativa de aptitud promedio de metaheurísticas por anexión con 0-rotación.

(b) **Tiempo promedio**

La Tabla 5.21 muestra los tiempos promedios de ejecución de las metaheurísticas en minutos.

Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.21 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.21: Tabla de tiempo promedio entre metaheurísticas con 0-rotación

	GAH (C2)	ACADM (C2)	ALDM (C2)
P1A2	0.0119 min	0.0028 min	0.0038 min
P2A2	0.2599 min	0.0962 min	0.0567 min
P3A2	2.0081 min	2.6701 min	2.7798 min
P4A2	17.4264 min	15.6049 min	16.3804 min
P5A2	61.5779 min	50.7722 min	51.3088 min
	16.2569 min	**13.8292 min**	14.1059 min

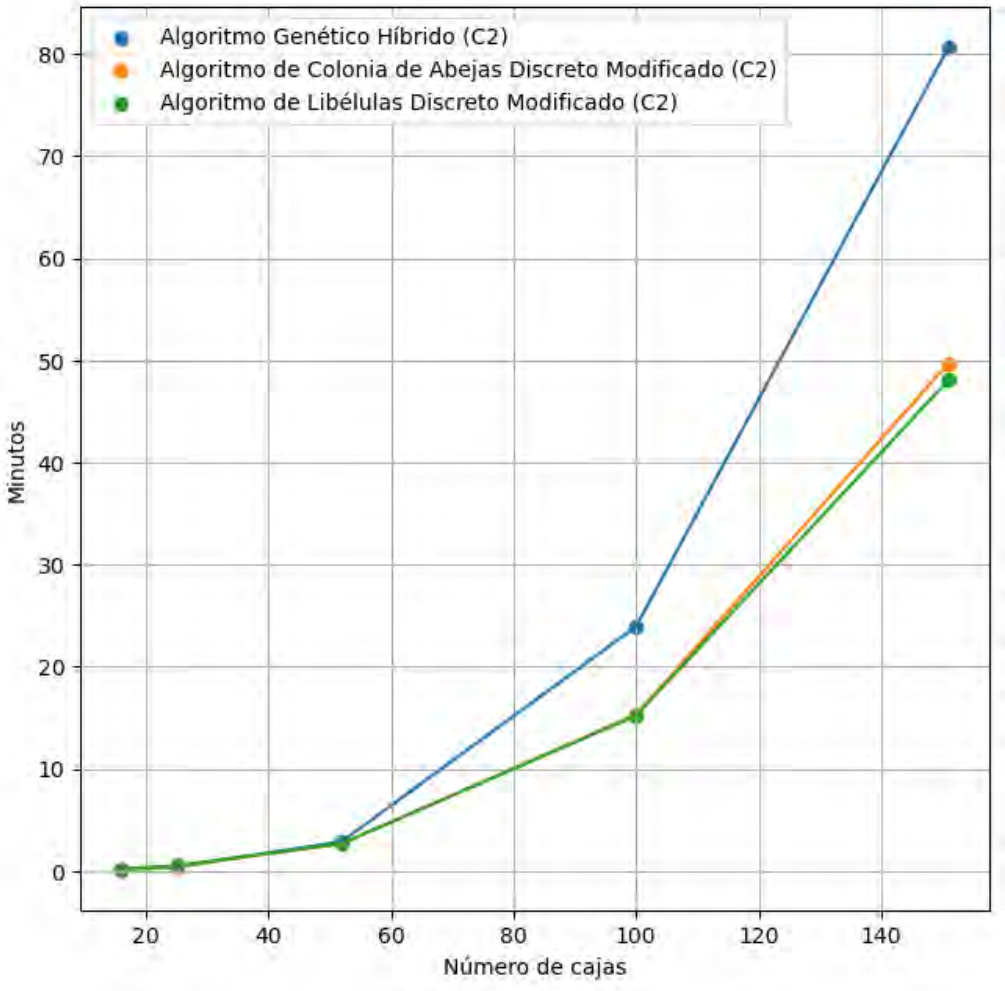


Figura 5.21: Comparativa del tiempo promedio de metaheurísticas por anexión con 0-rotación.

(c) **Análisis de resultados**

La Figura 5.20 y la Tabla 5.21, muestra que el ALDM obtuvo, en promedio, el mejor

rendimiento entre las 3 metaheurísticas para todos los problemas. La metaheurística ACADM obtuvo valores que caen dentro del intervalo de confianza del ALDM, como se muestra en la Figura 5.20. A pesar de que obtiene valores promedios menores que el ALDM, por ello es el segundo con mejor desempeño. Finalmente, el peor desempeño lo obtuvo el AGH, mostrándose una mayor diferencia entre las otras dos metaheurísticas.

La Figura 5.21 y la Tabla 5.21 muestran que el AGH fue el que más se tardó en resolver el problema en comparación del ACADM y el ALDM, aproximadamente $86\,\%$ más, es decir, casi el doble. Mientras que no hay diferencia significativa entre el ACADM y el ALDM, aproximadamente $0.98\,\%$.

5.7.2 Resultados para 2-rotación

(a) **Aptitud promedio**

En la Tabla 5.22 se muestran los promedios del porcentaje de ocupación en el contenedor con 2-rotación. Se remarca en negrita la metaheurística con el mejor rendimiento. La Figura 5.22 muestra la gráfica Aptitud vs Número de Cajas en donde cada curva representa el uso de una metaheurística.

Tabla 5.22: Tabla de aptitud porcentual promedio entre metaheurísticas con 2-rotación

	GAH (C2)	ACADM (C2)	ALDM (C2)
P1A2	$97.8560 \pm 0.3773\,\%$	$99.3220 \pm 0.2373\,\%$	$99.2175 \pm 0.3141\,\%$
P2A2	$96.1631 \pm 0.4819\,\%$	$98.0272 \pm 0.4854\,\%$	$98.6125 \pm 0.3896\,\%$
P3A2	$93.2074 \pm 0.7219\,\%$	$95.0343 \pm 0.8226\,\%$	$96.2063 \pm 0.6663\,\%$
P4A2	$90.4153 \pm 0.9073\,\%$	$92.1550 \pm 0.9730\,\%$	$93.3322 \pm 0.9220\,\%$
P5A2	$89.0414 \pm 0.9336\,\%$	$91.4380 \pm 1.0313\,\%$	$92.2178 \pm 0.9623\,\%$
	$93.3366 \pm 0.6844\,\%$	$95.1953 \pm 0.7099\,\%$	$\mathbf{95.9173 \pm 0.6509\,\%}$

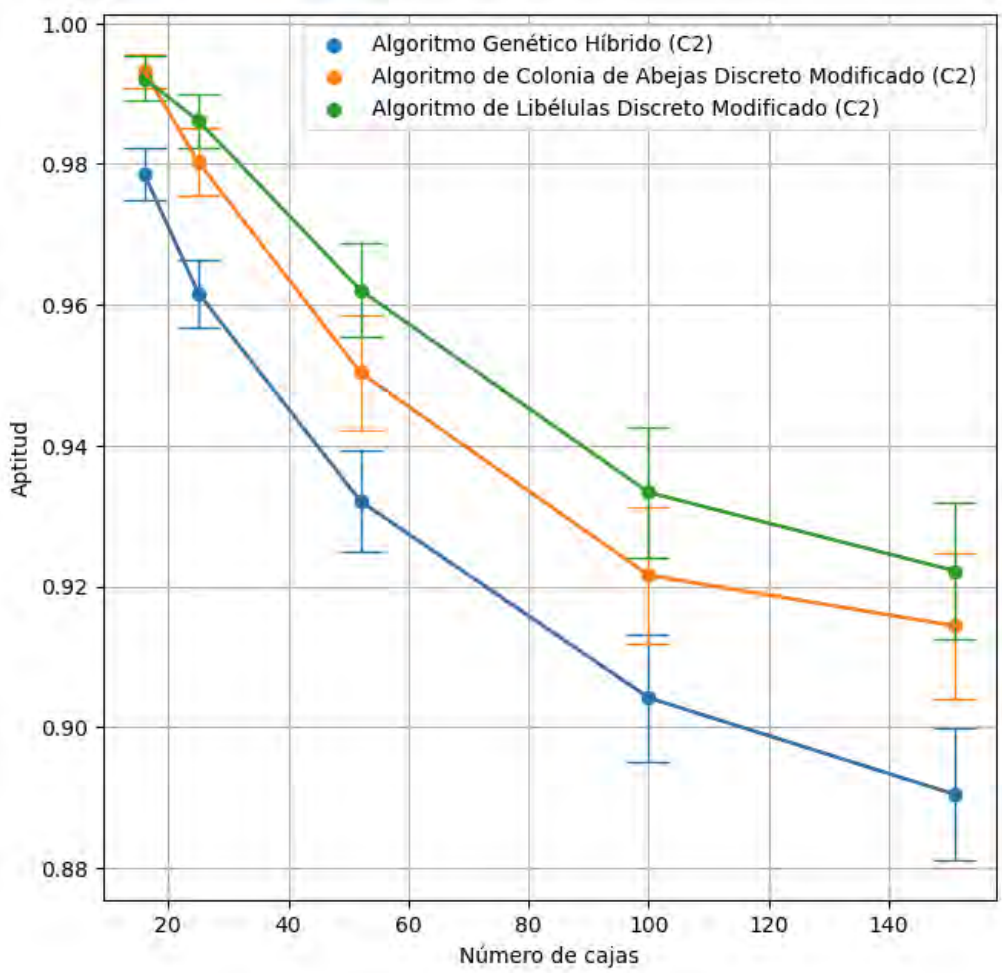


Figura 5.22: Comparativa de aptitud promedio de metaheurísticas por anexión con 2-rotación.

(b) **Tiempo promedio**

La Tabla 5.23 muestra los tiempos promedios de ejecución de las metaheurísticas con 2-rotación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.23 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.23: Tabla de tiempo promedio entre metaheurísticas con 2-rotación

	GAH (C2)	ACADM (C2)	ALDM (C2)
P1A2	0.1191 min	0.1202 min	0.1035 min
P2A2	0.3946 min	0.4613 min	0.4471 min
P3A2	2.3217 min	2.6341 min	2.6954 min
P4A2	18.3530 min	14.7341 min	14.7662 min
P5A2	59.8212 min	46.7931 min	46.3875 min
	16.2019 min	12.9486 min	**12.8799** min

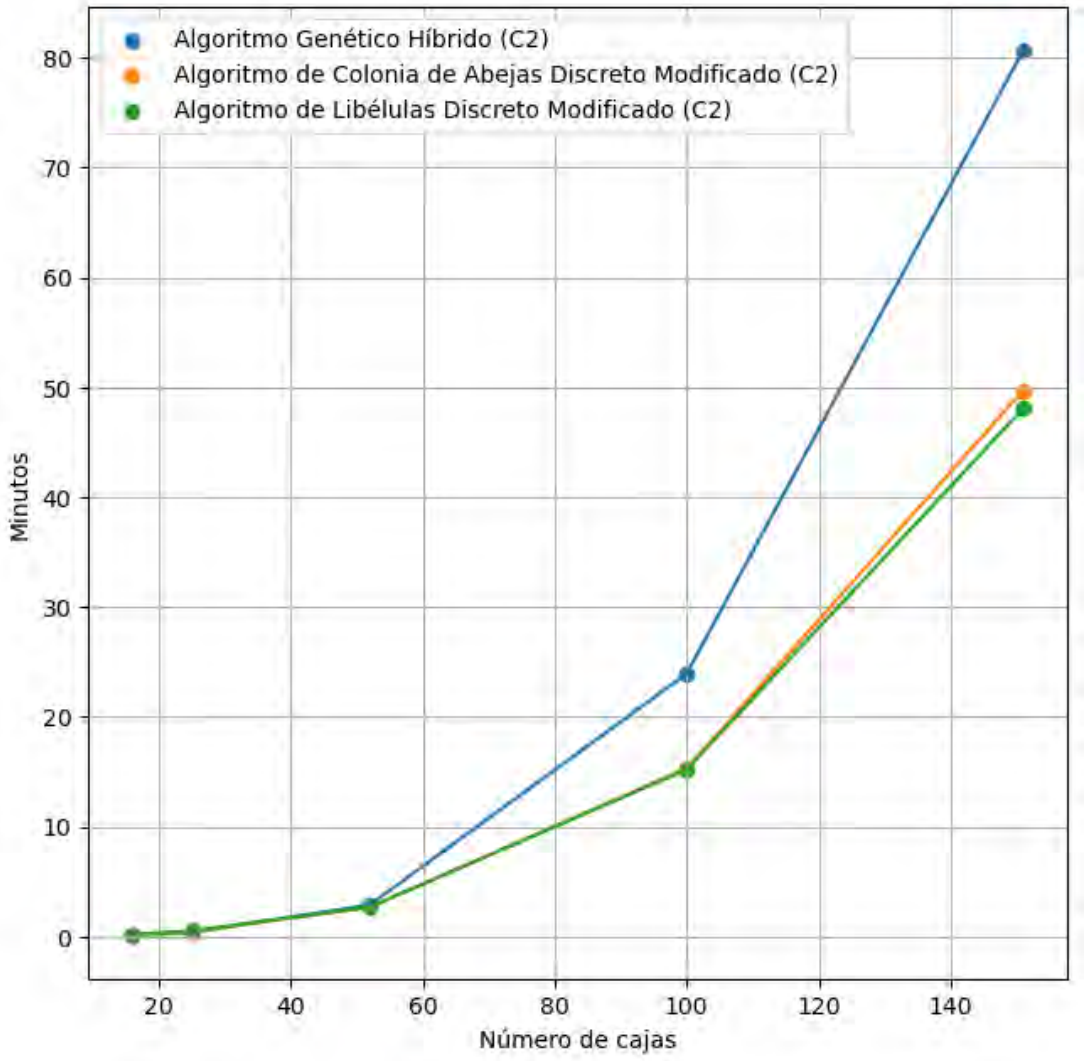


Figura 5.23: Comparativa del tiempo promedio de metaheurísticas por anexión con 2-rotación.

(c) **Análisis de resultados**

De manera similar a los resultados con 0-rotación, la Figura 5.22 y la Tabla 5.22 muestran que

el ALDM obtuvo, en promedio, el mejor rendimiento entre las 3 metaheurísticas para todos los problemas. En segunda posición, aunque no muy alejado del ALDM, está el ACADM, que cae dentro del intervalo de confianza del ALDM. El peor desempeño lo obtuvo el AGH.

La Figura 5.23 y la Tabla 5.23 muestran que el AGH fue, de nueva cuenta, el que más se tardó en resolver el problema en comparación del ACADM y el ALDM, aproximadamente $80\,\%$ más. Mientras que no hay diferencia significativa entre el ACADM y el ALDM, aproximadamente ya es solo del $1\,\%$.

5.7.3 Resultados para 6-rotación

(a) **Aptitud promedio**

En la Tabla 5.22 se muestran los promedios del porcentaje de ocupación en el contenedor con 6-rotación. Se remarca en negrita la metaheurística con el mejor rendimiento. La Figura 5.22 muestra la gráfica Aptitud vs Número de Cajas en donde cada curva representa el uso de una metaheurística.

Tabla 5.24: Tabla de aptitud porcentual promedio entre metaheurísticas con 6-rotación

	GAH (C2)	ACADM (C2)	ALDM (C2)
P1A2	$93.2344 \pm 0.6042\,\%$	$95.8305 \pm 0.7241\,\%$	$96.8592 \pm 0.6903\,\%$
P2A2	$90.1782 \pm 0.7774\,\%$	$94.2362 \pm 0.8501\,\%$	$94.9112 \pm 0.7550\,\%$
P3A2	$88.0440 \pm 0.9598\,\%$	$90.7151 \pm 0.9816\,\%$	$92.2823 \pm 0.9997\,\%$
P4A2	$84.9060 \pm 0.9632\,\%$	$89.7153 \pm 0.9626\,\%$	$90.7231 \pm 0.9799\,\%$
P5A2	$83.7111 \pm 1.0544\,\%$	$88.6505 \pm 1.0866\,\%$	$89.1002 \pm 1.0080\,\%$
	$88.0147 \pm 0.8718\,\%$	$91.8295 \pm 0.9210\,\%$	$\mathbf{92.7752 \pm 0.8866\,\%}$

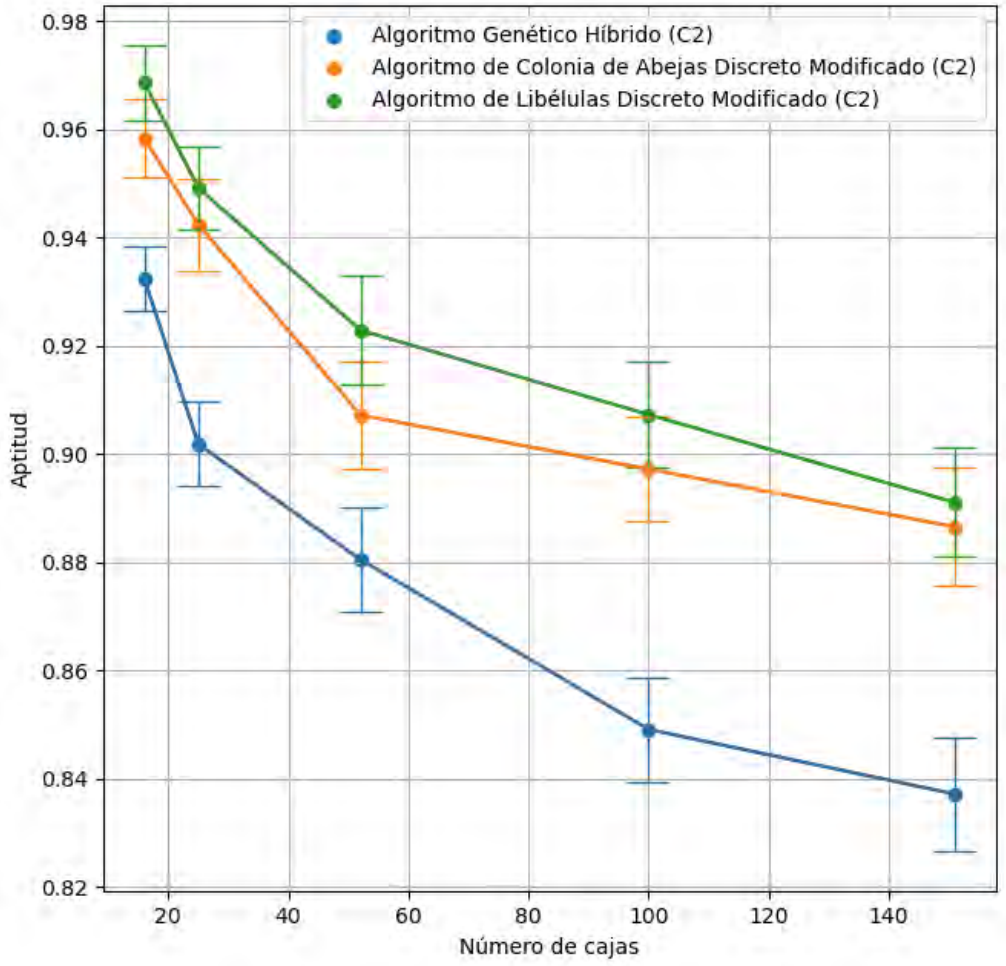


Figura 5.24: Comparativa de aptitud promedio de metaheurísticas por anexión con 6-rotación.

(b) **Tiempo promedio**

La Tabla 5.25 muestra los tiempos promedios de ejecución de las metaheurísticas con 6-rotación. Se remarca en negrita la metaheurística con menor tiempo. La Figura 5.25 muestra la gráfica de los tiempos promedios en minutos por el número de cajas a empacar.

Tabla 5.25: Tabla de tiempo promedio entre metaheurísticas con 6-rotación

	GAH (C2)	ACADM (C2)	ALDM (C2)
P1A2	0.1372 min	0.1850 min	0.2075 min
P2A2	0.3847 min	0.4592 min	0.5174 min
P3A2	2.9021 min	2.6771 min	2.7072 min
P4A2	23.9390 min	15.3031 min	15.2160 min
P5A2	80.6338 min	49.7059 min	48.0527 min
	21.5994 min	13.6660 min	**13.3402 min**

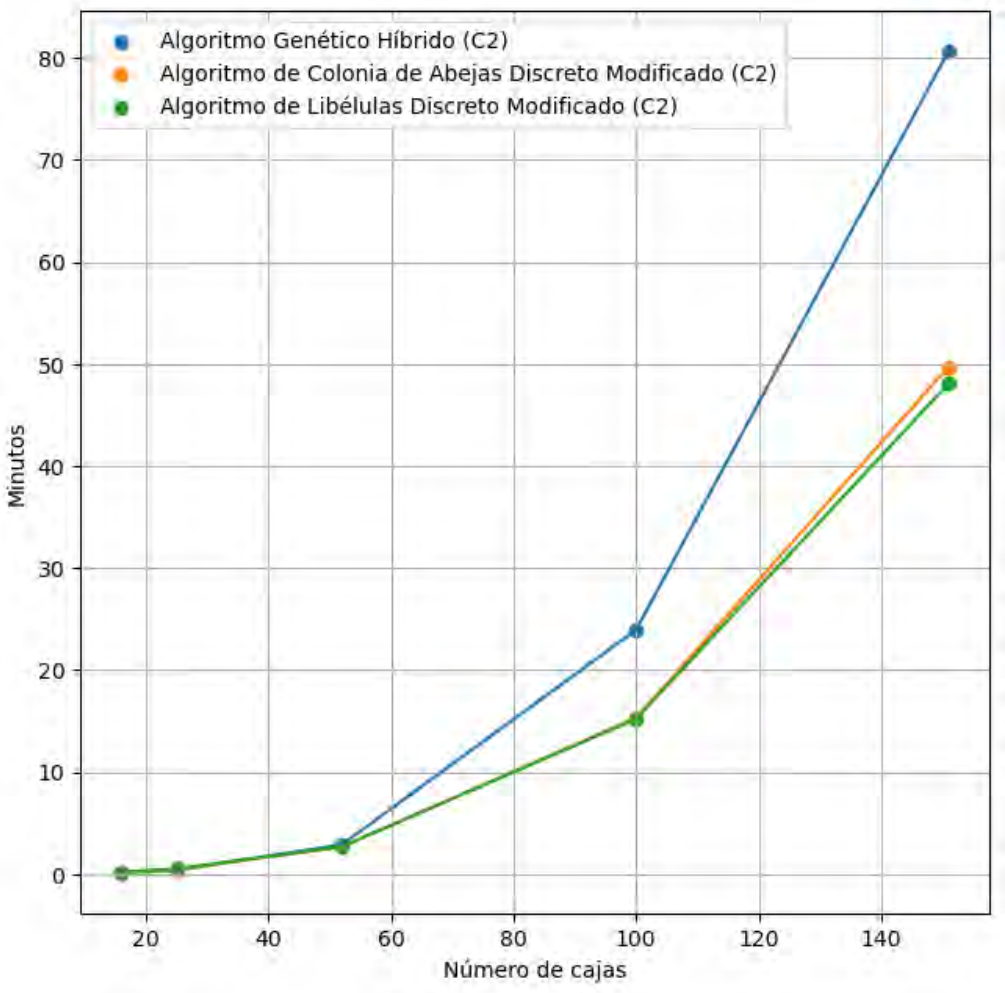


Figura 5.25: Comparativa del tiempo promedio de metaheurísticas por anexión con 6-rotación.

(c) **Análisis de resultados**

De manera similar a los resultados con 0-rotación y de 2-rotación, Figura 5.24 y Tabla 5.24, muestran que el ALDM obtuvo de nueva manera el mejor rendimiento promedio entre las 3 metaheurísticas para todos los problemas. El peor desempeño lo obtuvo el AGH, siendo más notoria la distancia entre de las curvas respecto a las metahurísticas bio-inspiradas, además, los púntos del AGH no cayeron dentro del intervalo de confianza de las otras metaheurísticas.

La Figura 5.25 y la Tabla 5.25 muestran que el AGH fue el que más se tardó en resolver el problema en comparación del ACADM y el ALDM, aproximadamente $63\,\%$ más, aunque esta vez es menor que con los resultados de 0-rotación y 2-rotación. Entre el ACADM y el ALDM hay aproximadamente $1.02\,\%$ de diferencia.

CAPÍTULO 6
CONCLUSIONES

6.1 Conclusiones generales

6.1.1 Conclusiones de los algoritmos genéticos

Aunque el enfoque de anexión de soluciones mostró una mayor eficiencia comparado con el enfoque de sustitución de individuos en la población, los algoritmos genéticos híbridos no lograron superar a las metaheurísticas bioinspiradas en términos de rendimiento global.

Se observó que los AGH requieren un tiempo significativamente mayor para obtener soluciones en cada iteración. Esto implica que, aunque los AGH exploran más el espacio de soluciones, no necesariamente logran los mejores resultados, lo cual se refleja en los tiempos y la aptitud obtenida en comparación con otras metaheurísticas como el Algoritmo de Luciérnagas Discreto Modificado (ALDM).

El AGH puede adobar problemas de combinatoria de manera eficiente, pero frente a las metaheurísticas bioinspiradas se observa que se podría explorar alternativas con enfoques evolutivos para la mejora su rendimiento.

6.1.2 Conclusiones de los algoritmos bioinspirados

Los algoritmos bioinspirados demuestran que superan consistentemente a los algoritmos genéticos híbridos, en términos de rendimiento para el problema de carga en un solo contenedor tridimensional. El Algoritmo de Luciérnagas Discreto Modificado (ALDM) y el Algoritmo de Colonia de Abejas Discreto Modificado (ACADM) obtuvieron los mejores resultados en todas las instancias,

destacando especialmente el ALDM como la metaheurística con mayor aptitud promedio y menor tiempo de ejecución.

Los algoritmos bioinspiradas exploran de manera más eficiente el espacio de soluciones, reduciendo los tiempos computacionales en comparación de los AGH. Además, el uso de técnicas avanzadas de mutación (como el uso del grupo C2) optimizó aún más el rendimiento de las metaheurísticas bioinspiradas, maximizando el porcentaje de ocupación del contenedor.

Estos resultados resaltan la adaptabilidad de los algoritmos bioinspirados frente a otros enfoques evolutivos. Aunque los algoritmos genéticos logran una mayor explotación del espacio de búsqueda, su efectividad fue superada por los modelos bioinspirados, los cuales ofrecen una mejor relación entre calidad de solución y eficiencia temporal.

6.1.3 Conclusiones de los operadores de mutación

La importancia en la selección de los operadores de mutación para los algoritmos bioinspirados es tal que influye significativamente en la calidad y eficiencia de las soluciones obtenidas. El grupo C2 mostró los mejores resultados en el uso de las metaheurísticas evaluadas en este trabajo.

6.2 Objetivos logrados

Objetivos	Logros
Utilización de técnicas evolutivas en el problema de carga en un solo contenedor, con y sin rotación.	Se desarrolló un sistema para la evaluación de metaheurísticas que adoptan una estrategia evolutiva, la cual es el enfoque de anexar nuevas soluciones al conjunto de soluciones.
Generar instancias que ayuden a medir el rendimiento de las metaheurísticas	Se generaron las instancias de prueba y se presentó el medio de obtenerlas, así como la interpretación de la lectura de los archivos que describen estas instancias.
Utilizar una heurística de empacado en conjunto con algoritmos metaheurísticos con un enfoque evolutivo	Se utilizó la heurística DBL sobre un conjunto de metaheurísticas para desarrollar el enfoque evolutivo de anexión de soluciones.
Modelo de las técnicas evolutivas para acomodarse al problema de carga en un solo contenedor.	Se diseñó un sistema general que trabaja sobre un espacio de soluciones discreto con permutaciones de secuencias como soluciones.
Aplicar el modelo en una instancia de pruebas generadas.	Se modelaron dos metaheurísticas bioinspiradas: *algoritmo de luciérnagas discreto modificado* y el *algoritmo de colonia de abejas discreto modificado*, al darle un enfoque evolutivo para resolver problemas de un solo contenedor fuertemente heterogéneo.
Obtener resultados de 100 pruebas en cada técnica evolutiva.	Se creó un conjunto de 500 instancias de 5 problemas fuertemente heterogéneos. Cada problema contiene 100 instancias. Se utilizó el modelo y se demostró con los resultados experimentales que la utilización de un enfoque evolutivo sobre el enfoque común, da mejores resultados.
Interpretar resultados y obtener la metaheurística con los mejores resultados	Se concluyó que la metaheurística con mejor desempeño es el algoritmo de luciérnagas discreto modificado.

6.3 Aportaciones

A. Modificación de los algoritmos bio-inspirados: Algoritmo de Libélulas Discreto Modificado y Algoritmo de Colonia de Abejas Discreto Modificado, teniendo un enfoque evolutivo en la obtención de nuevas soluciones.

B. Creación de instancias generadas para la carga de un solo contenedor fuertemente heterogéneo, la mayoría de las cajas a empacar tienen diferente tamaño de entre sí.

C. Sistema de obtención de óptimos resultados para el problema de carga utilizando metaheurística que trabajen sobre un espacio de permutaciones de secuencias.

D. Demostración que el cambio de enfoque a uno que anexe soluciones al conjunto de soluciones que se está operando en cada iteración obtiene mejores resultados que los convencionales.

E. Demostración que, para el problema de carga de contenedores fuertemente heterogéneo, el uso del grupo de combinación C2 de [Kıran et al., 2013] en donde utilizó el algoritmo de abejas para el problema del viajero, obtiene mejores resultados también en el problema de carga de contenedores.

6.4 Trabajos futuros

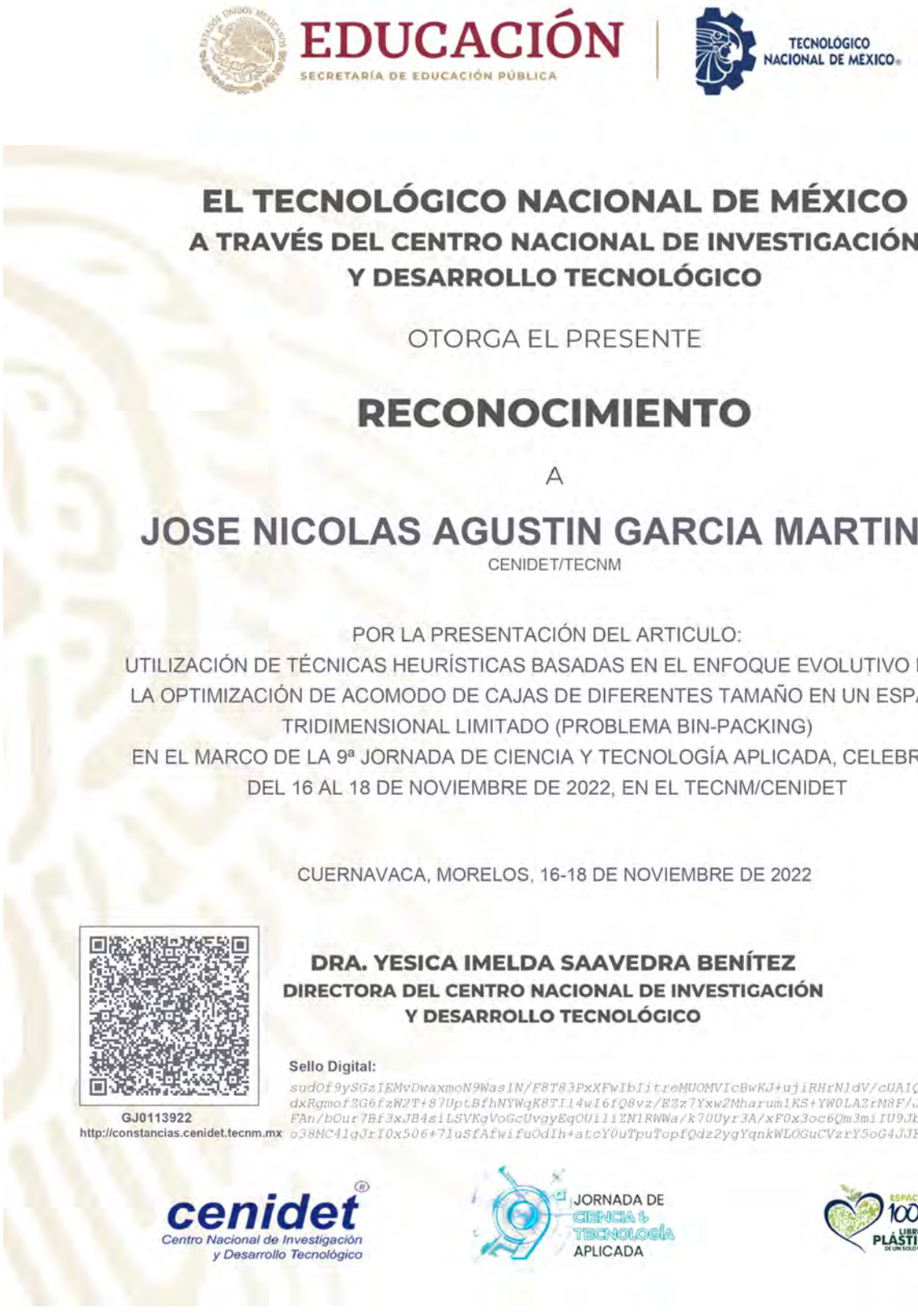
En la actualidad, en el problema da carga en un contenedor se está utilizando aprendizaje profundo Chien et al. [2024] [Jin et al., 2024] [Verma et al., 2020], el trabajo mediante metaheurísticas puede seguir siendo mejorado con la utilización de heurísticas con menor complejidad temporal que la DBLF. Además, este trabajo se extiende del problema de carga en un solo contenedor ya que este es un problema aplicable al conjunto de soluciones generalizado. De igual manera se puede utilizar las instancias generadas fuertemente heterogéneo con la finalidad de comparar resultados con otros enfoques.

6.5 Actividades Académicas

Presentación del póster ”Heurísticos con enfoque evolutivo y bioinspirados para el problema de carga en un solo contendedor (BPP3D)” en la escuela de inteligencia computacional y robótica 2022. Universidad Tecnológica Emiliano Zapata (UTEZ), 16 y 20 de agosto del 2022, Fig, 6.1.

Figura 6.1: Póster.

Se presentó, y se publicó, de manera presencial en el auditorio del TecNM/CENIDET, el artículo Ütilización de técnicas heurísticas basadas en enfoque evolutivo para la optimización de acomodo de cajas de diferente tamaño en un espacio tridimensional limitado (problema del binpacking)" en el marco de la jornada de ciencia y tecnología aplicada, que se celebró del 16 al 18 de novimebre de 2022 en el TecNM/CENIDET, Fig. 6.2.


EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

EL TECNOLÓGICO NACIONAL DE MÉXICO
A TRAVÉS DEL CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO

OTORGA EL PRESENTE

RECONOCIMIENTO

A

JOSE NICOLAS AGUSTIN GARCIA MARTINEZ
CENIDET/TECNM

POR LA PRESENTACIÓN DEL ARTICULO:
UTILIZACIÓN DE TÉCNICAS HEURÍSTICAS BASADAS EN EL ENFOQUE EVOLUTIVO PARA LA OPTIMIZACIÓN DE ACOMODO DE CAJAS DE DIFERENTES TAMAÑO EN UN ESPACIO TRIDIMENSIONAL LIMITADO (PROBLEMA BIN-PACKING)
EN EL MARCO DE LA 9ª JORNADA DE CIENCIA Y TECNOLOGÍA APLICADA, CELEBRADA DEL 16 AL 18 DE NOVIEMBRE DE 2022, EN EL TECNM/CENIDET

CUERNAVACA, MORELOS, 16-18 DE NOVIEMBRE DE 2022

DRA. YESICA IMELDA SAAVEDRA BENÍTEZ
DIRECTORA DEL CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO

Sello Digital:

GJ0113922
http://constancias.cenidet.tecnm.mx

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

JORNADA DE CIENCIA & TECNOLOGÍA APLICADA

ESPACIO 100% LIBRE DE PLÁSTICO DE UN SOLO USO

Figura 6.2: Artículo.

BIBLIOGRAFÍA

Alvarado Lara, I. L. (2012). Desarrollo de una herramienta de apoyo al análisis experimental del desempe' no de algoritmos metaheurísticos. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico.

Bayraktar, T., Ersöz, F., and Kubat, C. (2021). Effects of memory and genetic operators on artificial bee colony algorithm for three-dimensional bin packing problem. Technical report, EasyChair.

Bischoff, E. E. and Marriott, M. D. (1990). A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, 44(2):267–276.

Bischoff, E. E. and Ratcliff, M. (1995). Issues in the development of approaches to container loading. *Omega*, 23(4):377–390.

Chien, C.-F., Lan, Y.-B., Sethanan, K., and Peng, C.-C. (2024). Digital system for dynamic container loading with neural network-based memory exploiting hybrid genetic algorithm for carbon reduction. *Computers & Industrial Engineering*, 191:110149.

Cruz-Reyes, L., Quiroz C., M., Alvim, A. C. F., Fraire Huacuja, H. J., Gómes S., C., and Torres-Jiménez, J. (2012). Heuísticas de agrupación híbridas eficientes para el problema de empacado de objetos en contenedores. *Computación y Sistemas*, 16:349 – 360.

Delorme, M. and Wagenaar, J. (2024). Exact decomposition approaches for a single container loading problem with stacking constraints and medium-sized weakly heterogeneous items. *Omega*, 125:103039.

Dereli, T. and Daş, G. (2011). A hybrid 'bee(s) algorithm' for solving container loading problems. *Applied Soft Computing Journal*, 11:2854–2862.

Duan, L., Hu, H., Qian, Y., Gong, Y., Zhang, X., Xu, Y., and Wei, J. (2018). A multi-task selected learning approach for solving 3d flexible bin packing problem. *arXiv preprint arXiv:1804.06896*.

Escamilla, L. A. S., Zacatelco, H. C., and de la Rosa Flores, R. (2017). Implementación del algoritmo MBS para resolver instancias de bin packing. *Res. Comput. Sci.*, 134:45–53.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA.

Ghaisani, F. and Suyanto, S. (2019). Discrete firefly algorithm for an examination timetabling. In *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 1–4. IEEE.

Gonçalves, J. F. and Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Jati, G. K., Manurung, R., et al. (2013). Discrete firefly algorithm for traveling salesman problem: A new movement scheme. In *Swarm Intelligence and Bio-Inspired Computation*, pages 295–312. Elsevier.

Jiao, G., Huang, M., Song, Y., Li, H., and Wang, X. (2024). Container loading problem based on robotic loader system: An optimization approach. *Expert Systems with Applications*, 236:121222.

Jin, J., Cui, T., Bai, R., and Qu, R. (2024). Container port truck dispatching optimization using real2sim based deep reinforcement learning. *European Journal of Operational Research*, 315(1):161–175.

Kanna, S. R., Udaiyakumar, K., Kumar, S. D., and Lingaraj, N. (2018). 3d heterogeneous bin packing framework for multi-constrained problems using hybrid genetic approach. In *IOP Conference Series: Materials Science and Engineering*, volume 402, page 012203. IOP Publishing.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization.

Karabulut, K. and İnceoğlu, M. M. (2004). A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method. In *International Conference on Advances in Information Systems*, pages 441–450. Springer.

Kim, H. C., Lee, Y. J., Han, S. H., and Oh, J. (2021). Hybrid genetic algorithm for packing segments of decommissioned nuclear reactor components.

Kıran, M. S., İşcan, H., and Gündüz, M. (2013). The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem. *Neural computing and applications*, 23(1):9–21.

Kröger, B., Schwenderling, P., and Vornberger, O. (1990). Parallel genetic packing of rectangles. In *International Conference on Parallel Problem Solving from Nature*, pages 160–164. Springer.

Lodi, A., Martello, S., and Vigo, D. (2002). Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research*, 141(2):410–420.

Martello, S., Pisinger, D., and Vigo, D. (2000). The three-dimensional bin packing problem. *Operations research*, 48(2):256–267.

Martello, S. and Toth, P. (1990). *Knapsack problems: Algorithms and computer implementations*. John Wiley & Sons, Inc.

Mokshin, V., Maryashina, D., Stadnik, N., Zolotukhin, A., and Sharnin, L. (2020). Modified genetic algorithm as a new approach for solving the problem of 3d packaging. pages 654–661.

Nieto, D. (2007). Hibridación de algoritmos metaheurísticos para problemas de bin packing. Master's thesis, Instituto tecnológico de ciudad madero. División de estudios de posgrado e investigación.

Osaba, E., Yang, X.-S., Diaz, F., Onieva, E., Masegosa, A. D., and Perallos, A. (2017). A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy. *Soft Computing*, 21(18):5295–5308.

Perez-Ortega, J., Castillo-Zacatelco, H., Vilarino-Ayala, D., Mexicano-Santoyo, A., Zavala-Diaz, J. C., Martinez-Rebollar, A., and Estrada-Esquivel, H. (2016). Una nueva estrategia heurística para el problema de bin packing. *Ingenieria, Investigacion y Tecnologia*, 17(2):155–168.

Sayadi, M., Ramezanian, R., and Ghaffarinasab, N. (2010). A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. *International Journal of Industrial Engineering Computations*, 1.

Sundar, S., Singh, A., and Rossi, A. (2010). An artificial bee colony algorithm for the 0–1 multidimensional knapsack problem. In *International Conference on Contemporary Computing*, pages 141–151. Springer.

Tilahun, S. L. and Ngnotchouye, J. M. T. (2017). Firefly algorithm for discrete optimization problems: A survey. *KSCE Journal of civil Engineering*, 21(2):535–545.

Verma, R., Singhal, A., Khadilkar, H., Basumatary, A., Nayak, S., Singh, H. V., Kumar, S., and Sinha, R. (2020). A generalized reinforcement learning algorithm for online 3d bin-packing. *arXiv preprint arXiv:2007.00463*.

Wang, H. and Chen, Y. (2010a). A hybrid genetic algorithm for 3d bin packing problems. In *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, pages 703–707. IEEE.

Wang, H. and Chen, Y. (2010b). A hybrid genetic algorithm for 3d bin packing problems. In *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, pages 703–707.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*, pages 169–178. Springer.

Zhao, C., Jiang, L., and Teo, K. L. (2020). A hybrid chaos firefly algorithm for three-dimensional irregular packing problem. *Journal of Industrial & Management Optimization*, 16(1):409.

Zhong, Y., Lin, J., Wang, L., and Zhang, H. (2017). Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem. *Information Sciences*, 421:70–84.