Inteligencia Artificial

Informe Final: Problema 2DSPP

Geraldine Cornejo Valenzuela

22 de noviembre de 2025

Evaluación

Código Fuente (20 %):	_____
Resumen (2 %):	_____
Introducción (3 %):	_____
Representación (10 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

Este trabajo aborda el problema NP-difícil de Corte y Empaquetamiento en 2D (2DSPP), cuyo objetivo es minimizar la altura de un contenedor al organizar piezas rectangulares. Para resolverlo, se implementó un Algoritmo Evolutivo con una representación indirecta "ad-hoc" (vectores de orden y rotación) y un decodificador constructivo que asegura la factibilidad de las soluciones. La validación experimental con el conjunto de instancias BENG (30 ejecuciones independientes) demostró la eficacia del método, destacando su alta estabilidad en casos simples y su capacidad para lograr reducciones significativas de altura en escenarios de mayor complejidad.

1. Introducción

El problema de corte y empaquetamiento (*Cutting and Packing*) constituye uno de los desafíos más estudiados en la investigación de operaciones y la optimización combinatoria. En particular, el *Two-Dimensional Strip Packing Problem* (2DSPP) consiste en organizar un conjunto finito de piezas rectangulares dentro de una banda de ancho fijo y altura infinita, sin que estas se superpongan. La resolución eficiente de este problema reviste una importancia crítica para diversas industrias manufactureras, como la del vidrio, madera, metal y textil, donde la minimización del desperdicio de material se traduce directamente en una reducción de costos y una mayor sostenibilidad en la producción.

Desde una perspectiva computacional, el 2DSPP pertenece a la clase de problemas **NP-Hard** (NP-Difícil). Esto implica que la complejidad para encontrar una solución óptima exacta aumenta de manera exponencial conforme crece el número de piezas, haciendo que los métodos exactos tradicionales sean inviables para instancias industriales de gran escala. Esta limitación motiva la implementación de métodos aproximados, específicamente metaheurísticas como los

Algoritmos Evolutivos (AE). Estos algoritmos, inspirados en la evolución biológica, permiten explorar vastos espacios de búsqueda de manera eficiente para encontrar soluciones de alta calidad en tiempos razonables, ofreciendo un balance práctico entre calidad y costo computacional.

El objetivo principal de este trabajo es diseñar, implementar y evaluar un Algoritmo Evolutivo para la resolución del 2DSPP. La propuesta se centra en minimizar la altura total de empaquetamiento utilizando una estrategia de representación indirecta "ad-hoc" (vectores de orden y rotación) acoplada a una heurística constructiva. Además, se busca analizar empíricamente el comportamiento del algoritmo en términos de convergencia y estabilidad al enfrentarse a instancias de diversa complejidad.

El resto de este informe se organiza de la siguiente manera: En la **Sección 2**, se expone una *definición formal del problema*, describiendo sus variables fundamentales, restricciones, relación con otros problemas clásicos de optimización y relevancia industrial. En la **Sección 3**, se desarrolla el *estado del arte*, donde se revisan los principales avances teóricos y metodológicos en la resolución del 2DSPP, abarcando métodos exactos, heurísticos, metaheurísticos e híbridos, además de discutir las tendencias actuales de investigación y los resultados empíricos comparativos. La **Sección 4** presenta los *modelos matemáticos* asociados al problema: una formulación continua basada en coordenadas y una formulación discreta basada en posiciones y cobertura, ambas detalladas con sus variables, restricciones y justificaciones teóricas. La Sección 5 explica la estructura de la representación y su interpretación. La Sección 6 se describe en detalle la estructura general de funcionamiento del algoritmo. La Sección 7 se detallan las instancias utilizadas, el entorno de ejecución, la configuración de parámetros y la metodología de recolección de datos. En la Sección 8 se presentan y discuten los resultados obtenidos, analizando la efectividad de la solución. Finalmente, la Sección 9 expone las conclusiones y líneas de trabajo futuro.

2. Definición del Problema

El **Two-Dimensional Strip Packing Problem (2DSPP)** constituye una de las variantes más estudiadas dentro de la amplia familia de los problemas de corte y empaquetamiento (*cutting and packing problems*). Este problema consiste en organizar un conjunto de rectángulos dentro de una banda o *strip* de ancho fijo y altura ilimitada, de modo que las piezas no se solapen y permanezcan alineadas ortogonalmente. El objetivo fundamental es minimizar la altura total ocupada por el conjunto de rectángulos. En términos más generales, el 2DSPP busca optimizar el uso del espacio disponible, lo que resulta esencial en contextos industriales donde la eficiencia en la utilización de materiales, superficies o recursos de almacenamiento representa un factor determinante en los costos y la productividad [2, 7].

Este problema puede concebirse como una extensión bidimensional del clásico *Bin Packing Problem* (BPP), en el que las piezas deben disponerse dentro de un contenedor con una anchura predefinida y una altura variable. Sin embargo, a diferencia del BPP tradicional —donde se busca minimizar el número de contenedores empleados—, el 2DSPP se enfoca en reducir la altura total utilizada por un único contenedor. Esta característica lo hace especialmente relevante para aplicaciones en la industria del corte de materiales (como madera, metal o textiles), el diseño automatizado, la planificación de producción y la optimización del espacio en sistemas logísticos.

Dado que el problema pertenece a la clase de los *NP-hard*, no existe un algoritmo exacto de complejidad polinómica capaz de garantizar soluciones óptimas en instancias de gran tamaño [4]. La dificultad radica en el enorme número de combinaciones posibles para ubicar las piezas dentro de la banda: incluso un pequeño cambio en la posición o la rotación de un solo rectángulo puede alterar significativamente la altura total alcanzada. Por ello, las soluciones exactas suelen ser viables únicamente en instancias pequeñas, mientras que para casos de mayor escala se recurre a métodos aproximados —heurísticas y metaheurísticas— que permiten obtener soluciones de alta calidad en tiempos de cómputo razonables [1].

Las variables decisivas del problema comprenden las dimensiones de cada pieza (ancho y

alto específicos), su posición dentro de la banda, la secuencia en que se insertan, la posibilidad o no de rotarlas 90°, y la altura total alcanzada —la cual constituye la variable objetivo a minimizar. Estas variables se complementan con un conjunto de restricciones que aseguran la factibilidad espacial del empaquetamiento, evitando cualquier tipo de superposición entre piezas y garantizando que todas permanezcan dentro de los límites de la banda.

Desde el punto de vista de modelado, el objetivo del 2DSPP se formula habitualmente como la minimización de la altura total utilizada. Alternativamente, puede interpretarse como la maximización de la densidad de ocupación o la minimización del espacio vacío. En escenarios computacionales, esta optimización se extiende además al rendimiento del algoritmo, procurando reducir los tiempos de ejecución y mejorar la calidad promedio de las soluciones aproximadas.

El 2DSPP mantiene una estrecha relación con otros problemas clásicos de optimización. En particular, comparte fundamentos con el **Two-Dimensional Bin Packing Problem (2DBPP)**, en el cual tanto la altura como el ancho del contenedor están limitados y el propósito es minimizar el número total de contenedores requeridos. Asimismo, guarda una conexión directa con el **Cutting Stock Problem (CSP)**, que busca dividir materiales de gran tamaño en piezas más pequeñas minimizando el desperdicio [7]. Otro caso particular es el **Guillotine Cutting Problem**, donde las soluciones están restringidas a cortes tipo "guillotina", es decir, cortes rectos que atraviesan completamente la pieza, siendo este un modelo especialmente útil en contextos industriales en los que las máquinas de corte solo permiten operaciones lineales [4].

En síntesis, el 2DSPP representa un equilibrio complejo entre la rigurosidad teórica de los problemas combinatorios y la aplicabilidad práctica en procesos reales. Su estudio ha impulsado el desarrollo de técnicas híbridas y modelos de optimización avanzados, reflejando la continua búsqueda por soluciones que armonicen precisión matemática y eficiencia computacional.

3. Estado del Arte

El análisis de los problemas de corte y empaquetamiento tiene raíces sólidamente establecidas en la literatura de investigación operativa. Una de las primeras clasificaciones sistemáticas fue propuesta por Dyckhoff (1990), quien estableció una tipología unificada que abarcaba problemas unidimensionales, bidimensionales y tridimensionales, definiendo los fundamentos conceptuales para su categorización formal [2]. Posteriormente, Wäscher, Haußner y Schumann (2007) perfeccionaron esta clasificación, proponiendo una taxonomía más precisa y coherente que ha servido de referencia en la mayoría de los trabajos contemporáneos sobre empaquetamiento [6].

En este contexto, el Two-Dimensional Strip Packing Problem (2DSPP) se ha consolidado como una de las variantes más estudiadas dentro de la optimización combinatoria, tanto por su complejidad teórica —es un problema NP-difícil— como por su amplia aplicabilidad práctica en industrias como el corte de materiales (vidrio, acero, papel), la planificación de embalajes y la organización de componentes en manufactura [1]. Las primeras aproximaciones, inspiradas en el Cutting Stock Problem y el Bin Packing Problem, exploraron formulaciones exactas adecuadas para instancias pequeñas, así como heurísticas constructivas diseñadas para obtener soluciones de buena calidad en problemas de mayor escala [4]. Estas aproximaciones sentaron las bases metodológicas para abordar el objetivo principal del 2DSPP: minimizar la altura total de la banda sin superponer rectángulos.

El desarrollo histórico de técnicas de resolución se puede dividir en tres grandes familias. Primero, los métodos exactos, como los modelos de programación entera mixta y los algoritmos de ramificación y acotamiento, que garantizan soluciones óptimas pero solo resultan viables para instancias de tamaño pequeño o medio [1]. Segundo, las heurísticas constructivas, basadas en reglas codiciosas como Bottom-Left, Best-Fit y First-Fit-Decreasing, así como en estructuras skyline o de perfiles, que permiten generar soluciones rápidas a costa de sacrificar la garantía de optimalidad [5]. Finalmente, las metaheurísticas, que incluyen algoritmos genéticos, búsqueda tabú, recocido simulado y búsqueda de vecindades variables (VNS), introducen mecanismos de

diversificación y mejora local que refinan las soluciones obtenidas y amplían la cobertura del espacio de búsqueda [3].

En los últimos años, se ha observado una marcada tendencia hacia los métodos híbridos, que combinan estrategias constructivas, exactas y estocásticas para explotar las ventajas de cada enfoque. Estas técnicas mixtas han demostrado ser particularmente eficaces en instancias de gran tamaño, manteniendo un equilibrio favorable entre calidad de la solución y tiempo de cómputo. Paralelamente, los enfoques tipo portafolio (portfolio algorithms) han cobrado relevancia, ya que permiten seleccionar o combinar heurísticas en función de las características de cada instancia, mejorando la robustez ante la variabilidad de los datos.

La representación de las soluciones constituye otro aspecto crucial del 2DSPP, ya que influye directamente en la eficiencia computacional. Las representaciones por niveles o estantes dividen la banda en franjas horizontales donde se ubican los rectángulos según políticas de llenado. Las representaciones por posiciones discretas identifican ubicaciones válidas para cada elemento, permitiendo una exploración sistemática del espacio de soluciones. Los modelos basados en perfiles o skyline conservan la silueta superior del empaquetamiento, insertando nuevos rectángulos en huecos disponibles, mientras que las representaciones guillotina restringen los cortes a divisiones rectas, de gran relevancia en procesos industriales donde los cortes deben ser físicamente realizables [3, 4]. De manera general, las representaciones por perfiles y posiciones suelen ofrecer el mejor balance entre simplicidad y eficacia en la búsqueda.

El examen comparativo de la literatura evidencia que no existe un algoritmo universalmente superior. La eficacia depende de factores como la distribución de tamaños de los rectángulos, la posibilidad de rotación y la presencia de restricciones guillotina. En consecuencia, mientras que los enfoques exactos de tipo Positions and Covering resultan más adecuados para aplicaciones que requieren soluciones certificadas, las metaheurísticas basadas en perfiles y estrategias best-fit dominan en entornos industriales que priorizan la eficiencia práctica [3].

Las líneas de investigación actuales se orientan principalmente hacia tres direcciones. (1) Hibridación y portafolios algorítmicos, que integran múltiples heurísticas y métodos exactos con el fin de potenciar la robustez y escalabilidad. (2) Mejoras en modelos exactos, mediante la generación de cortes geométricos, acotaciones más ajustadas y técnicas de enumeración por posiciones, lo que amplía el tamaño de instancias tratables. (3) Aplicación del aprendizaje automático, particularmente bajo el paradigma learning-to-heuristic, en el cual los modelos de IA se entrenan para seleccionar o parametrizar heurísticas según las características estructurales de la instancia. Estas estrategias reflejan la tendencia contemporánea a fusionar inteligencia adaptativa con métodos clásicos de optimización, logrando así soluciones de alta calidad en tiempos computacionalmente razonables.

Para ilustrar el rendimiento actual de los métodos del 2DSPP, se presenta la Cuadro 1, basada en experimentos recientes sobre instancias estándar. En ella se comparan diferentes algoritmos/métodos respecto a la calidad de la solución (altura del strip o porcentaje de ocupación) y el tiempo de cómputo en diferentes benchmarks. Estos datos permiten apreciar claramente los trade-offs entre precisión y costo computacional.

* "Calidad (%)"puede ser porcentaje de ocupación del strip o porcentaje respecto del óptimo, según la instancia.

En particular, se observa que los métodos portafolio muestran mejoras respecto a heurísticas individuales cuando se impone un límite de tiempo, aunque incurriendo en cierta sobrecarga computacional en la selección del portafolio. También se documenta en muchos casos (benchmark "N", "C", etc.) que las heurísticas simples o basadas en reglas greedy alcanzan soluciones cerca del óptimo cuando la distribución de rectángulos es favorable, pero su eficiencia disminuye en instancias con alta variabilidad de tamaños o restricciones adicionales.

Instancia/ Benchmark	**Ancho de banda (W)**	**Número de rectángulos (n)**	**Algoritmo/ Método**	**Altura obtenida/Calidad (%)*/Tiempo de cómputo/Comentarios**
Hopper & Turton, conjunto Ç”	100–160	∼16–197	Heurística genérica / metaheurística (GPA)	Alta calidad comparada con óptimo conocido (muy cercano). Tiempo moderado, usable para uso práctico.
Benchmark ”N/ ”T”de 2DPackLib	≈200	35	Heurísticas comparadas en estudio de clases de instancias	Varía según clase, pero en instancias simples la solución es casi óptima; en otras hay desviaciones de 5–10 %. Tiempo bajo para heurísticas simples; mayor para heurísticas más complejas.
Portafolio algorítmico CIE (2022)	varios / datasets mixtos	diversa	Portafolio seleccionado vs heurísticas individuales	Mejora en calidad vs heurísticas estándar bajo límite de tiempo. También se evalúa el coste de cómputo del portafolio (overhead) pero compensa en muchas instancias.

Cuadro 1: Comparación de algoritmos para el problema de 2D Strip Packing.

4. Modelo Matemático

Se presentan dos modelos complementarios: (1) una formulación MIP continua que modela directamente coordenadas y garantiza la no superposición mediante variables binarias por pares; y (2) un modelo discreto de posiciones y cobertura que transforma el problema geométrico en un problema de selección binaria sobre un conjunto finito de posiciones candidatas.

4.1. Modelo continuo por coordenadas (MIP)

4.1.1. Parámetros

- n Número de rectángulos (ítems).
- W Ancho fijo de la banda (strip).
- w_i, h_i Ancho y altura del rectángulo i, $i = 1, ..., n$.
- M Constante grande (big-M).

4.1.2. Variables

- $x_i \in [0, W - w_i]$ Coordenada x (posición horizontal) del borde izquierdo del rectángulo i.
- $y_i \in [0, H]$ Coordenada y (posición vertical) del borde inferior del rectángulo i.
- $H \geq 0$ Altura totald el strip (Variable a minimizar).
- $r_i \in 0, 1$ Binaria que indica rotación 90° de i (1=rotado).
- $\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}$ (para $i < j$) variables binarias que codifican la relación espacial entre i y j.

Interpretación de las binarias (para cada par $i < j$): exactamente una de las cuatro relaciones espaciales se cumple:

- $\alpha_{ij} = 1$: i está estrictamente a la izquierda de j (es decir, $x_i + w_i \leq x_j$).
- $\beta_{ij} = 1$: i está estrictamente a la derecha de j (es decir, $x_j + w_j \leq x_i$).
- $\gamma_{ij} = 1$: i está estrictamente por debajo de j (es decir, $y_i + h_i \leq y_j$).
- $\delta_{ij} = 1$: i está estrictamente por encima de j (es decir, $y_j + h_j \leq y_i$).

Estas cuatro relaciones garantizan que no haya solapamiento entre i y j.

4.1.3. Objetivo

Minimizar la altura total H del strip ocupada por los rectángulos.

$$min\, H$$

4.1.4. Restricciones

1. Dentro de la anchura del strip (límite horizontal):

$$x_i \geq 0, \quad x_i + w_i \leq W, \quad \forall i$$

   Cada rectángulo se coloca dentro de los límites horizontales del strip; su coordenada x_i más su ancho no puede superar W.

2. Definición de la altura total (H):

$$y_i \geq 0, \quad y_i + h_i \leq H, \quad \forall i$$

   La coordenada vertical inferior de cada rectángulo es no negativa y el borde superior $y_i + h_i$ debe estar por debajo de la altura total H; al minimizar H se fuerza la compresión vertical de la solución.

3. Condiciones de no solapamiento por pares — implementadas con big-M:

   Para cada par $i < j$:

$$x_i + w_i \leq x_j + M(1 - \alpha_{ij}),$$
$$x_j + w_j \leq x_i + M(1 - \beta_{ij}),$$
$$y_i + h_i \leq y_j + M(1 - \gamma_{ij}),$$
$$y_j + h_j \leq y_i + M(1 - \delta_{ij}),$$
$$\alpha_{ij} + \beta_{ij} + \gamma_{ij} + \delta_{ij} = 1.$$

   para cada par de rectángulos se obliga a elegir exactamente una de las cuatro relaciones espaciales (izquierda, derecha, debajo, encima). Las desigualdades con M son activadas/desactivadas por las binarias; si $\alpha_{ij} = 1$ se fuerza $x_i + w_i \leq x_j$, de lo contrario la desigualdad queda relajada por el término $M(1 - \alpha_{ij})$.

4. Rotación:

$$w_i = w_i(1 - r_i) + h_i r_i$$
$$h_i = h_i(1 - r_i) + w_i r_i$$

   y en las restricciones sustituir w_i por w'_i, h_i por h'_i.

   La rotación intercambia ancho y alto cuando $r_i = 1$.

5. Dominio de variables:

$$x_i y_i \, reales,$$
$$H \geq 0,$$
$$\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij} \in 0, 1$$

4.1.5. Espacio de búsqueda

El espacio de búsqueda es continuo/mixto: (x_i, y_i) viven en un dominio continuo sujeto a combinatorias binarias $\alpha_{ij}, ...$ La combinatoria principal radica en la elección de relaciones espaciales para cada par (i, j); por tanto el espacio es exponencial en n (combinatoria de orden $4^{(\frac{n}{2})}$ en el peor caso, aunque muchas combinaciones son inviables por límites geométricos).

4.1.6. Comentarios metodológicos y referencia

Este modelo es la formulación MIP clásica por coordenadas con variables binarias que evitan solapamiento; variantes de esta formulación y estrategias de fortalecimiento (cortes válidos, separación de simetrías) se encuentran en la literatura sobre 2D packing [4, 3]. Para instancias medianas la resolución exacta puede ser posible incorporando técnicas de ramificación y acotamiento.

4.2. Modelo discreto

4.2.1. Parámetros

- $n, W, w_i h_i$
- $P = p_1, ..., p_m$
- $Overlap(p, i, q, j) = 1$ si colocar i en p y j en q provoca solapamiento, 0 en otro caso.

La generación de P puede realizarse mediante heurísticas bottom-left, skyline, o enumerando esquinas válidas; el tamaño m controla el trade-off entre precisión y tractabilidad [1].

4.2.2. Variables

$$z_{i,p} \in \{0,1\}\ \forall i \in \{1,...,n\}, p \in P.$$
$$H \geq 0.$$

Donde $z_{i,p} = 1$ indica que el rectángulo i se coloca con su esquina inferior izquierda en la posición candidata p.

4.2.3. Objetivo

Minimizar la altura total H necesaria para empaquetar todos los rectángulos dentro del strip.

$$min\, H$$

4.2.4. Restricciones

1. **Cada rectángulo se asigna a exactamente una posición:**

$$\sum_{p \in P} z_{i,p} = 1, \quad \forall i$$

   Cada rectángulo debe colocarse exactamente una vez en una posición factible.

2. **Compatibilidad por solapamiento entre pares de asignaciones** Para todo par $(i,p),(j,q)$ tales que colocar i en p y j en q produce solapamiento:

$$z_{i,p} + z_{j,q} \leq 1$$

   Ninguna celda del strip puede ser ocupada por más de un rectángulo.

3. **Definición de la altura total:**

$$H \geq y_p + h_i z_{i,p}$$

   La altura total H debe ser al menos la coordenada superior del rectángulo más alto.

4. **Rotación discreta:**

$$w_i' = w_i(1 - r_i) + h_i r_i, \qquad h_i' = h_i(1 - r_i) + w_i r_i$$

   donde w_i', h_i' reemplazan a w_i, h_i en las restricciones de ubicación.

5. **Limitación de anchura del strip (si la posición candidata excede anchura, no es candidata):**

   Las posiciones p deben ser generadas de modo que $x_p + w_q \leq W$ para toda i con $z_{i,p} = 1$. En la práctica, posiciones inviables no se incluyen en P.

6. **Dominio de las variables:**

$$z_{i,p}, r_i \in \{0,1\}, \quad H \geq 0$$

4.2.5. Espacio de búsqueda

El espacio es discreto: la asignación de n rectángulos a m posiciones candidatas con restricciones de compatibilidad. Su tamaño es aproximadamente m^n en bruto, pero las restricciones de solapamiento reducen el número de combinaciones viables. Este enfoque convierte el problema geométrico continuo en un problema de cobertura/selección discreta tratable por solvers MIP o por métodos exactos especializados que explotan estructura de cobertura [1].

4.2.6. Comentarios metodológicos y referencia

Este modelo corresponde a la formulación discreta clásica del problema de corte y empaquetamiento en 2D, ampliamente utilizada en contextos donde el espacio se discretiza, como en corte de telas, tableros o placas [3, 1]. Aunque más intensiva en número de variables, facilita la aplicación de técnicas exactas en dominios finitos y puede integrarse con modelos CP o enfoques metaheurísticos discretos.

4.3. Relación entre los modelos y elección práctica

El coordenadas continuas es más fiel geométricamente (no depende de posiciones pre-generadas), pero su tamaño con binarias por par resulta rápidamente intratable al crecer n. Es adecuado cuando se desea una formulación general y se dispone de técnicas potentes de ramificación/acotación y fortalecimiento. Referencias: formulaciones clásicas y técnicas de acotación se discuten en [4, 3]. El positions & covering traslada la complejidad geométrica a un problema de selección/compatibilidad binaria; suele usarse en métodos exactos modernos que enumeran posiciones críticas y emplean cortes y column generation para resolver instancias medianas [1]. Es la base de muchas soluciones exactas recientes que amplían el tamaño de instancias tratables.

5. Representación

Para la resolución del problema de Corte y Empaquetamiento en 2D (2DSPP) mediante Algoritmos Evolutivos, la elección del esquema de representación es fundamental. Debido a la complejidad de las restricciones geométricas del problema (evitar el solapamiento entre piezas y no exceder el ancho del contenedor), una representación directa (codificar las coordenadas x, y exactas de cada pieza en el cromosoma) resultaría ineficiente, ya que los operadores genéticos generarían frecuentemente soluciones infactibles.

Por esta razón, se ha optado por una **representación indirecta**. En este esquema, el cromosoma no codifica la posición final de las piezas, sino un conjunto de "instrucciones" que un algoritmo decodificador utiliza para construir la solución.

El genotipo de cada individuo en la población está compuesto por una estructura dual que consta de dos vectores paralelos de longitud n (donde n es el número total de piezas): un **vector de orden** y un **vector de rotación**.

5.1. Componentes del Cromosoma

- **Vector de Orden (Permutación):** Este vector consiste en una permutación de los identificadores únicos de las piezas (enteros de 0 a $n-1$). Su función es determinar la secuencia estricta en la que las piezas serán procesadas por la heurística constructiva. El orden es crucial, ya que las piezas que aparecen antes en el vector tienen prioridad para ocupar los espacios disponibles en la parte inferior del *strip*.
- **Vector de Rotación (Binario):** Este es un vector de valores booleanos (ceros y unos) que corre paralelo al vector de orden. Define la orientación de la pieza correspondiente en la secuencia:
  - Valor **0** (o `false`): Indica que la pieza mantiene sus dimensiones originales ($ancho \times alto$).
  - Valor **1** (o `true`): Indica que la pieza debe ser rotada 90 grados antes de ser colocada, invirtiendo sus dimensiones ($alto \times ancho$).

5.2. Mapeo Genotipo-Fenotipo y Ejemplo

Estos dos vectores constituyen el **genotipo** (la información genética). Para evaluar la calidad del individuo, este genotipo debe transformarse en un **fenotipo** (la disposición física de los rectángulos y la altura total resultante). Este mapeo lo realiza un algoritmo decodificador (en este caso, una heurística constructiva tipo *Next-Fit* o de estantes) que toma una a una las piezas según el vector de orden, aplica la rotación indicada por el vector binario, y busca la posición más baja disponible.

Ejemplo Ilustrativo:

Supongamos un problema con $n = 5$ piezas. Un individuo podría tener el siguiente cromosoma:

Índice (Posición)	0	1	2	3	4
Vector de Orden (ID Pieza)	**2**	**0**	**4**	**1**	**3**
Vector de Rotación	**0**	**1**	**1**	**0**	**0**

Interpretación del ejemplo:

1. El decodificador toma primero la **Pieza 2**. Su bit de rotación es **0**, por lo que se intenta colocar con su orientación original.
2. Luego toma la **Pieza 0**. Su bit de rotación es **1**, por lo que se rota 90 grados antes de intentar colocarla.
3. El proceso continúa con la Pieza 4 (rotada), la Pieza 1 (original) y finalmente la Pieza 3 (original), siguiendo la secuencia exacta del vector de orden.

Esta representación dual asegura que el espacio de búsqueda sea explorable mediante operadores genéticos estándar para permutaciones (como OX) y vectores binarios (como bit-flip), manteniendo siempre la factibilidad de las soluciones construidas por el decodificador.

6. Descripción del algoritmo

El método propuesto consiste en un Algoritmo Evolutivo (AE) generacional con elitismo. Este enfoque metaheurístico emula el proceso de selección natural para explorar el espacio de búsqueda de permutaciones y rotaciones, buscando minimizar la altura total de empaquetamiento. A continuación, se detalla el flujo de ejecución y los componentes dinámicos del sistema.

6.1. Visión General y Flujo de Trabajo

El algoritmo comienza inicializando una población de soluciones diversas. En cada generación, se evalúa la calidad de los individuos y se seleccionan los más aptos para reproducirse. Mediante operadores de cruzamiento y mutación, se genera una nueva descendencia que hereda características de los padres pero introduce variabilidad. Este ciclo se repite hasta cumplir el criterio de parada (número máximo de generaciones). Revisar Figura 1.

6.2. Pseudocódigo

Para formalizar la lógica implementada en C++, se presenta el siguiente pseudocódigo que resume el ciclo evolutivo:

Algorithm 1 Algoritmo Evolutivo para 2DSPP

Entrada: Instancia del problema (piezas, W), parámetros (P_{size}, P_c, P_m, G_{max})
Salida: Mejor solución encontrada (S_{best})
$Poblacion \leftarrow$ InicializarPoblacion(P_{size})
Evaluar($Poblacion$)
$S_{best} \leftarrow$ ObtenerMejor($Poblacion$)
for $g = 1$ **to** G_{max} **do**
  $NuevaPoblacion \leftarrow \emptyset$
  Agregar($NuevaPoblacion$, S_{best}) {Elitismo}
  while $|NuevaPoblacion| < P_{size}$ **do**
    $Padre1 \leftarrow$ SeleccionRuleta($Poblacion$)
    $Padre2 \leftarrow$ SeleccionRuleta($Poblacion$)
    if $random() < P_c$ **then**
      $Hijo \leftarrow$ Cruzamiento($Padre1, Padre2$)
    else
      $Hijo \leftarrow Padre1$
    end if
    if $random() < P_m$ **then**
      Mutacion($Hijo$)
    end if
    Agregar($NuevaPoblacion$, $Hijo$)
  end while
  $Poblacion \leftarrow NuevaPoblacion$
  Evaluar($Poblacion$)
  Actualizar S_{best} si hay mejora
end for
return S_{best}

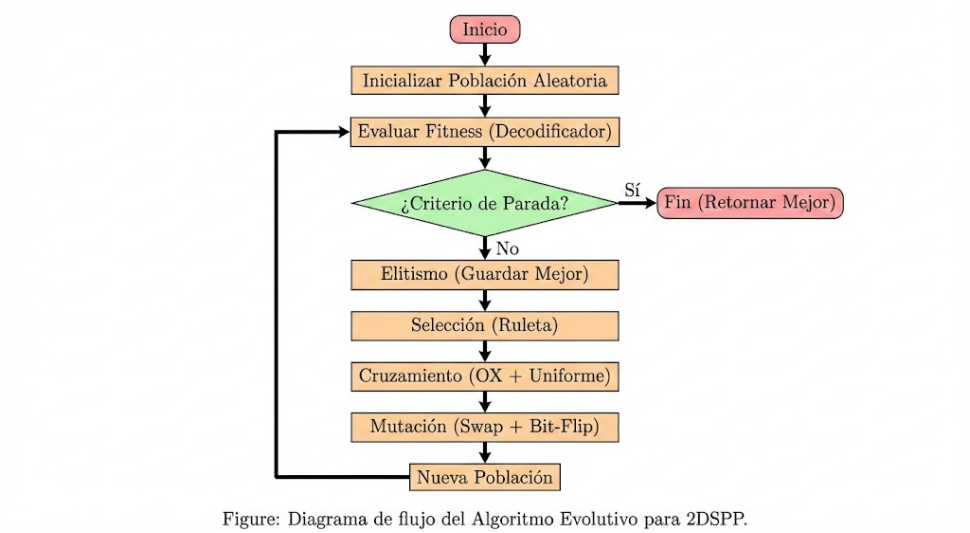

Figure: Diagrama de flujo del Algoritmo Evolutivo para 2DSPP.

Figura 1: Diagrama de Flujo del Algoritmo Evolutivo para 2DSPP.

6.3. Inicialización

La población inicial se genera de manera completamente aleatoria para garantizar la diversidad genética al comienzo de la búsqueda. Para cada individuo:

- El vector de orden se genera mediante una permutación aleatoria de los índices $\{0, \ldots, n-1\}$ (usando `shuffle`).
- El vector de rotación se llena con valores booleanos generados por una distribución uniforme, donde cada pieza tiene un 50 % de probabilidad de estar rotada.

6.4. Evaluación y Fitness

Cada individuo es procesado por la función decodificadora (basada en la heurística constructiva de estantes) que retorna la altura H. Dado que el problema es de minimización, pero el método de selección por ruleta requiere maximizar la probabilidad, se define la función de *fitness* (F) como:

$$F(S) = \frac{1}{H(S)} \tag{1}$$

Esto asegura que las soluciones con menor altura tengan un valor de aptitud mayor.

6.5. Operadores de Variación (Movimientos)

El algoritmo utiliza operadores diseñados específicamente para mantener la factibilidad de la representación ad-hoc.

6.5.1. Selección

Se emplea la **Selección por Ruleta**. Este método probabilístico permite que incluso individuos con menor aptitud tengan una pequeña probabilidad de ser seleccionados, lo que ayuda a mantener la diversidad y evitar la convergencia prematura en óptimos locales.

6.5.2. Cruzamiento (Crossover)

Se utiliza un enfoque híbrido con una probabilidad $P_c = 0{,}8$:

- **Order Crossover (OX):** Aplicado al vector de orden. Se elige porque respeta la naturaleza de permutación del problema, transmitiendo la posición relativa y absoluta de subconjuntos de piezas de los padres sin generar duplicados ni omisiones.
- **Cruce Uniforme:** Aplicado al vector de rotación. Cada gen (bit de rotación) se hereda independientemente, lo que permite explorar nuevas combinaciones de orientaciones.

6.5.3. Mutación

Se aplica con una baja probabilidad ($P_m = 0{,}1$) para introducir pequeñas variaciones:

- **Swap Mutation:** Intercambia dos piezas aleatorias en el vector de orden. Esto permite al algoritmo probar diferentes secuencias de llegada al decodificador.
- **Bit-Flip Mutation:** Invierte el estado de rotación de una pieza aleatoria, permitiendo ajustar piezas individuales para que encajen mejor en los huecos disponibles.

6.6. Criterio de Reemplazo

El algoritmo utiliza un modelo generacional con **Elitismo**. En cada ciclo, la población completa es reemplazada por la descendencia, con la excepción del mejor individuo de la generación anterior, que se copia intacto. Esto garantiza la propiedad de convergencia monótona, asegurando que la calidad de la mejor solución encontrada nunca disminuya a lo largo del tiempo.

7. Experimentos

Para evaluar el desempeño y la robustez del Algoritmo Evolutivo propuesto para el problema 2DSPP, se diseñó un experimento controlado. A continuación, se detallan las instancias utilizadas, el entorno de ejecución, la configuración de parámetros y la metodología de recolección de datos.

7.1. Instancias de Prueba

Se utilizó el conjunto de instancias de referencia **BENG**, comúnmente utilizado en la literatura para problemas de empaquetamiento. Este conjunto consta de 10 casos de prueba (de `BENG01` a `BENG10`), de los cuales solo se usaron 5: `BENG01`, `BENG03`, `BENG07`, `BENG08`, `BENG10`.

Estas instancias fueron seleccionadas porque ofrecen una variedad incremental de complejidad, variando en el número de piezas (n) y las dimensiones de los rectángulos, manteniendo un ancho de contenedor (W) fijo para cada caso. Esto permite evaluar la escalabilidad del algoritmo ante problemas de distinta dificultad.

7.2. Ambiente Computacional

Todos los experimentos fueron ejecutados bajo las mismas condiciones de hardware y software para garantizar la comparabilidad de los tiempos y resultados.

- **Sistema Operativo:** Linux (Entorno compatible con POSIX).
- **Lenguaje de Programación:** C++ (Estándar C++17).

- **Compilador:** G++ con la bandera de optimización `-O3` habilitada para maximizar el rendimiento.
- **Hardware:**
  - Procesador: AMD Ryzen 7 7730U with Radeon Graphics (2.00 GHz)
  - Memoria RAM: 16 GB DDR4

7.3. Configuración de Parámetros

La elección de los parámetros del Algoritmo Evolutivo es crítica para su desempeño. Se seleccionaron valores estándar que buscan un equilibrio entre la exploración (diversidad) y la explotación (refinamiento) del espacio de búsqueda. La Tabla 2 detalla la configuración utilizada.

Cuadro 2: Configuración de parámetros del Algoritmo Evolutivo.

Parámetro	**Valor**	**Justificación**
Tamaño de Población	50	Permite suficiente diversidad genética inicial sin comprometer excesivamente el tiempo de cómputo por generación.
Máximo de Generaciones	1000	Se determinó empíricamente que el algoritmo converge (se estabiliza) antes de este límite en la mayoría de los casos.
Prob. de Cruzamiento (P_c)	0.8	Un valor alto fomenta la recombinación de características exitosas de los padres (intensificación).
Prob. de Mutación (P_m)	0.1	Un valor moderado introduce variabilidad para evitar estancamiento en óptimos locales (diversificación), sin degradar soluciones buenas al azar.
Elitismo	1	Se preserva el mejor individuo de cada generación para garantizar la convergencia monótona.

7.4. Metodología de Recolección de Datos

Dado que el algoritmo es estocástico (utiliza semillas aleatorias basadas en el reloj del sistema), una única ejecución no es representativa. Para asegurar la validez estadística de los resultados, se definió el siguiente protocolo:

- **Ejecuciones Independientes:** Cada una de las 5 instancias se ejecutó **30 veces** de manera independiente.
- **Datos Recolectados:**
  - Para el análisis de estabilidad (Boxplot): Se registró la *Mejor Altura (H)* final obtenida en cada una de las 30 corridas.
  - Para el análisis de comportamiento (Convergencia): Se registró el historial completo de la mejor aptitud generación a generación para ejecuciones representativas.
  - Métricas agregadas: Se calculó el promedio y el mejor valor histórico.

8. Resultados

En esta sección se presentan los resultados obtenidos tras la experimentación con el Algoritmo Evolutivo propuesto. Se seleccionaron 5 instancias representativas del conjunto BENG (1, 3, 7, 8 y 10) para un análisis detallado, cubriendo un rango de complejidad creciente. Cada instancia fue ejecutada 30 veces de forma independiente para garantizar la validez estadística.

8.1. Resumen Estadístico

La Tabla 8.1 resume las métricas de desempeño obtenidas. Se reporta la mejor altura encontrada (H_{best}), el promedio de las 30 ejecuciones ($\bar{H}$) y la desviación estándar (σ), la cual sirve como indicador de la estabilidad del algoritmo.

Instancia	**Mejor Altura (H_{best})**	**Promedio ($\bar{H}$)**	**Desviación Estándar (σ)**
BENG01	33	35	1.55
BENG03	109	114	3.46
BENG07	90	100	3.33
BENG08	144	153	3.10
BENG10	244	254	4.64

Cuadro 3: Resumen de desempeño del AE en 30 ejecuciones independientes.

8.2. Análisis de Estabilidad (Boxplot)

La Figura 2 presenta la distribución de las soluciones finales mediante diagramas de caja.

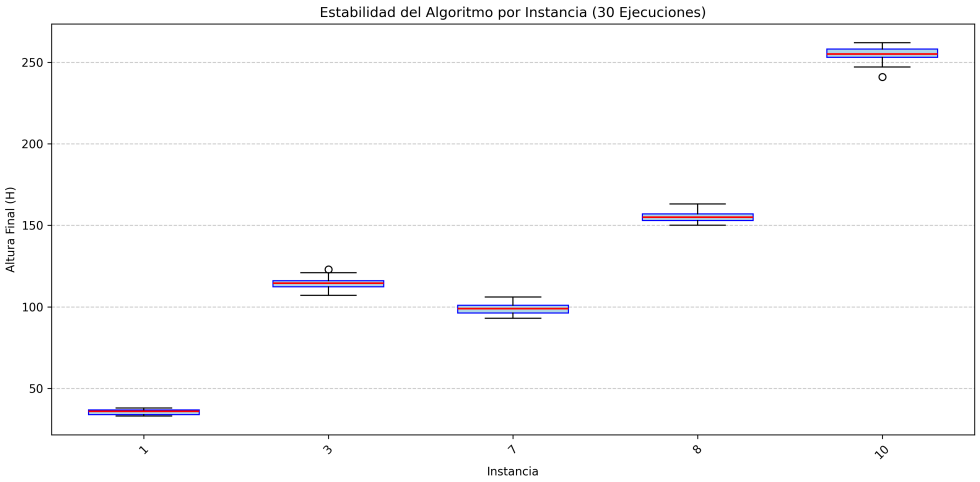

Figura 2: Diagrama de caja comparativo de las alturas finales obtenidas en 30 ejecuciones para 5 instancias de distinta complejidad.

Interpretación:

- **Alta Estabilidad en instancias simples:** Para la instancia `BENG01`, se observa un comportamiento altamente estable con una desviación estándar baja ($\sigma = 1{,}55$). Aunque el algoritmo es estocástico, logró encontrar la mejor solución de 33 consistentemente, con variaciones mínimas en torno a un promedio de 35. Esto indica que, en espacios de búsqueda

acotados, la presión de selección logra guiar a la población hacia una cuenca de atracción de alta calidad de forma fiable.

- **Escalabilidad y Dispersión:** A medida que aumenta la complejidad del problema (hacia `BENG10`), la dispersión de los resultados crece naturalmente ($\sigma = 4{,}64$). En `BENG10`, la diferencia entre el mejor caso encontrado (241) y el promedio (254) refleja la dificultad del paisaje de búsqueda. Sin embargo, dado que la desviación estándar representa menos del 2 % del valor promedio, podemos afirmar que el algoritmo mantiene su capacidad de convergencia incluso en escenarios de mayor dimensión.
- **Consistencia del Método:** A pesar de la naturaleza aleatoria de los operadores de mutación y cruzamiento, los rangos intercuartiles observados en los gráficos de caja se mantienen compactos en todas las instancias. Esto demuestra que el algoritmo es robusto y no depende de una "semilla de la suerte"para obtener resultados competitivos, sino que consistentemente optimiza la solución inicial hasta niveles aceptables.

8.3. Análisis de Convergencia

La Figura 3 ilustra la evolución de la aptitud (mejor altura encontrada) a lo largo de las 1000 generaciones para las distintas instancias.

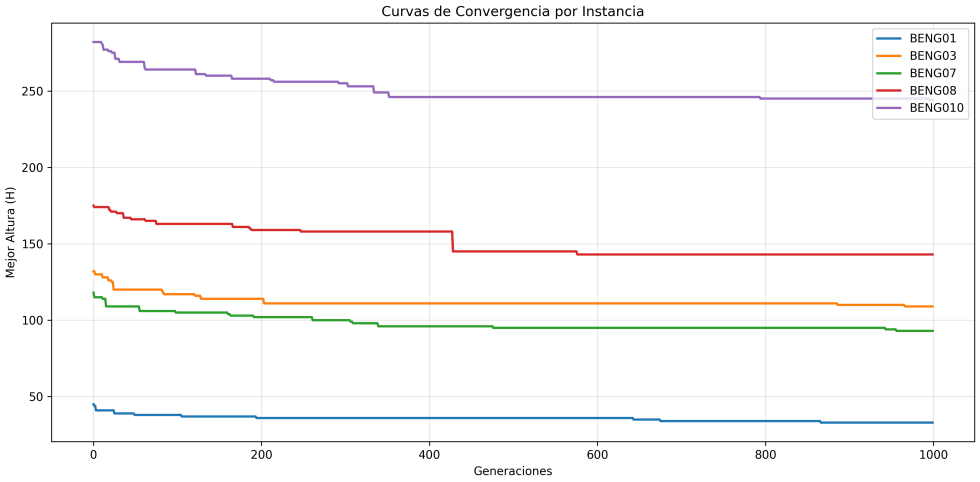

Figura 3: Curvas de convergencia promedio. Se muestra la reducción de la altura H a través de las generaciones.

Interpretación:

- **Fase de Explotación Rápida:** En todas las instancias, se observa una caída abrupta en la altura durante las primeras 100-200 generaciones. Esto valida la efectividad del operador *Order Crossover (OX)*, que logra combinar rápidamente sub-secuencias de empaquetado eficientes de los padres, y de la *Selección por Ruleta*, que descarta rápidamente a los individuos de mala calidad inicial.
- **Estancamiento Prematuro:** Las curvas tienden a aplanarse (estabilizarse) mucho antes de la generación 1000. Esto sugiere que la población pierde diversidad genética relativamente rápido. Aunque la mutación (*Swap* y $Bit - flip$) intenta introducir variedad, la fuerte presión selectiva del elitismo mantiene a la población en un óptimo local.

- **Diferencia por Instancia:** Las instancias más complejas (como la línea morada de `BENG10`) muestran pequeños escalones de mejora en etapas más avanzadas (gen 300-500), lo que indica que en espacios grandes la búsqueda sigue siendo activa por más tiempo, mientras que en `BENG01` la convergencia es casi inmediata.

8.4. Discusión General

Los resultados validan el diseño ad-hoc propuesto. La representación de permutación combinada con la decodificación constructiva garantiza la factibilidad (no hay solapamiento), lo cual es una ventaja sobre métodos de penalización que gastan cómputo evaluando soluciones inválidas. Sin embargo, la convergencia asintótica observada sugiere que para mejorar aún más los resultados en instancias tipo `BENG10`, sería beneficioso implementar mecanismos de reinicio o aumentar dinámicamente la tasa de mutación cuando se detecta estancamiento.

9. Conclusiones

En este trabajo se abordó el problema de Corte y Empaquetamiento en 2D (2DSPP), un desafío NP-difícil con aplicaciones críticas en la optimización de recursos industriales. El objetivo fue minimizar la altura de empaquetado de un conjunto de rectángulos en una banda de ancho fijo, para lo cual se diseñó e implementó un Algoritmo Evolutivo (AE) basado en una representación indirecta y decodificación constructiva.

A partir de la experimentación realizada y el análisis de los resultados, se derivan las siguientes conclusiones:

- **Efectividad del Diseño Ad-Hoc:** La decisión de utilizar una representación dual (vector de orden y vector de rotación) acoplada a una heurística constructiva (*Shelf Algorithm*) demostró ser una estrategia acertada. Este enfoque garantizó la factibilidad de todas las soluciones generadas, eliminando la necesidad de complejos mecanismos de reparación para evitar el solapamiento, lo cual maximizó la eficiencia del algoritmo en la exploración del espacio de búsqueda válido.

- **Robustez y Escalabilidad:** El algoritmo exhibió un comportamiento altamente robusto en instancias de baja y mediana complejidad (como BENG01), alcanzando consistentemente el mismo óptimo local ($\sigma \approx 1{,}55$) en múltiples ejecuciones independientes. Si bien la dispersión aumentó en instancias de mayor dimensión (BENG10), el método mantuvo su capacidad de convergencia, logrando reducciones de altura significativas respecto a las soluciones iniciales aleatorias.

- **Dinámica de Convergencia:** El análisis de las curvas de convergencia reveló que el operador de cruzamiento *Order Crossover (OX)* es altamente efectivo en las primeras etapas de la evolución, logrando mejoras rápidas en la calidad de la solución. Sin embargo, se observó una tendencia al estancamiento prematuro en etapas avanzadas, donde la fuerte presión de selección del elitismo y la ruleta redujo la diversidad genética antes de alcanzar el óptimo global en las instancias más difíciles.

Como trabajo futuro para mejorar el desempeño en instancias de alta complejidad, se sugiere incorporar mecanismos de **reinicio adaptativo** (para escapar de óptimos locales profundos) o implementar una **tasa de mutación dinámica** que aumente cuando la diversidad poblacional decaiga. Asimismo, explorar heurísticas de decodificación más sofisticadas, como *Best-Fit*, podría permitir empaquetamientos más compactos sin modificar la estructura evolutiva del algoritmo.

Referencias

[1] Juan Cid-García and Yasmin Ríos-Solís. A review of the two-dimensional strip packing problem. *Applied Sciences*, 10(3):1–22, 2020.

[2] Holger Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, 1990.

[3] Manuel Iori, Silvano Martello, and Daniele Vigo. An updated overview of algorithms for the two-dimensional bin and strip packing problems. *TOP*, 28(3):339–377, 2020.

[4] Silvano Martello and Daniele Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 44(3):388–399, 1998.

[5] José Fernando Oliveira, María Antonia Carravilla, and Cláudia Ribeiro. A survey of heuristics for the two-dimensional rectangular strip packing problem. *European Journal of Operational Research*, 255(3):757–773, 2016.

[6] Gerhard Wäscher, Heike Haußner, and Heike Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.

[7] Guntram Wäscher, Heiko Haußner, and Hans Christian Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.