Inteligencia Artificial

Estado del Arte: Problema 2D-Strip Packing

Ignacio Loayza Campos - 201273604-8

17 de enero de 2020

Evaluación

Resumen (5 %):	_____
Introducción (5 %):	_____
Definición del Problema (10 %):	_____
Estado del Arte (35 %):	_____
Modelo Matemático (20 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100 %):	_____

Resumen

En este trabajo se ha realizado un estudio del problema de empaque 2-dimensional en cinta, en donde un determinado número de artículos deben ser empacados en un contenedor con ancho fijo y altura infinita, el objetivo de dicho problema es encontrar un emplazamiento de los artículos de forma tal que se minimice la altura total del empacado.
Este trabajo es un estudio de algunos de los avances relevantes para el problema en cuestión, además de presentar resultados experimentales de una solución implementada utilizando una aproximación meta-heurística conocida como Hill Climbing.

1. Introducción

En varias industrias de manufactura, como por ejemplo la de vidrio, cuero, madera y papel, se observa reiteradamente el problema de alocar formas a cortar sobre una superficie de material, de forma tal que se minimice el material desperdiciado, reduciendo costos de material que en muchos casos no es despreciable. Este problema recibe el nombre en la literatura como "*2D-Strip packing problem*"(desde ahora referido como **2DSPP**). El 2DSPP tiene numerosas variantes dependiendo de las restricciones que se necesite aplicar (cortes guillotinables, dimensionalidad de los objetos, dimensiones del contenedor, etc..) que han sido ampliamente estudiadas en la literatura.
El objetivo de trabajo se centra en presentar un estudio de algunas de las propuestas más relevantes para el 2DSPP describiendo la aproximación utilizada por los autores, estudiando su rendimiento con respecto al benchmark ampliamente utilizado para este problema, propuesto

por Hopper y Turton en [8]

Este documento se encuentra organizado de la siguiente forma: La sección 2 discute la definición del problema de empaque, especificando las variantes existentes, complejidades y restricciones aplicadas. La sección 3 detalla el estado del arte comprendiendo diversos autores y sus aproximaciones. La sección 4 contiene el modelo matemático presentado para el problema de 2DSPP. Finalmente, la sección 5 presenta las conclusiones extraídas del estudio de las aproximaciones de diversos autores asi como las limitantes de estas y una propuesta de implementación para este problema.

2. Definición del Problema

Los problemas de empaque son problemas de optimización donde se busca obtener una buena configuración espacial de artículos dentro de un determinado contenedor, por lo general, el objetivo es minimizar el tamaño del espacio desperdiciado dentro del contenedor. Las dificultades asociadas a la búsqueda de soluciones para los problemas de empaque radican en varias aristas, por ejemplo, la explosión combinatorial del espacio de búsqueda, con el incremento de la cantidad de objetos a emplazar, producto de la alta cantidad de posiciones y combinaciones posibles entre los artículos dentro de la cinta.

En esta investigación se tratará el problema de empacar artículos 2-dimensionales dentro de un contenedor (referido como "*cinta*") también 2-dimensional, sin que ocurran superposiciones entre los artículos dentro de contenedor.

Como menciona Wei et al. [14], los problemas de corte y empaque pueden ser caracterizados por distintas cualidades, algunas de estas son:

1. **Por dimensionalidad:** Los artículos a ser empacados suelen ser 2-dimensionales o 3-dimensionales, sin embargo, el problema puede generalizarse a dimensiones más altas. Las aproximaciones 2-dimensionales como la tratada en esta investigación, son aquellas en las que el objeto donde deben ser empacados los artículos es considerado un pliego de material, con largo ya sea finito o infinito y los artículos son figuras en dos dimensiones. Por otro lado, la versión 3-dimensional considera un contenedor con un cierto volumen asociado, al igual que los artículos a ser empacados dentro de él.

2. **Por forma:** La mayoría de la literatura se enfoca en el caso de artículos rectangulares, sin embargo, las figuras pueden ser más complejas. En la industria textil, por ejemplo, es común el requerimiento de cortes de tela con formas específicas no rectangulares.

3. **Por *rotabilidad*:** Se refiere a si el problema admite la rotación de los artículos al momento de ser emplazados en el contenedor.

4. **Por *guillotinabilidad*:** Se refiere a si se requiere que la solución final encontrada permita hacer cortes guillotinables, es decir, los cortes sobre el contenedor, para separar los artículos, deben ser paralelos a los ejes del contenedor y deben atravesarlo de un limite al opuesto.

5. **Por objetivo:** El problema puede ser generalizado de tal forma que el objetivo sea maximizar el beneficio/ganancia total de los artículos incluidos en el contenedor. Notar que, como se menciona en [14], en este caso el valor de cada artículo no necesariamente se debe corresponder con su área, [13] por ejemplo, utiliza una función de ajuste apropiada en conjunto con una meta-heurística basada e algoritmos genéticos.

De acuerdo a esta caracterización el problema que se estudia en esta investigación es el problema de empaque 2-dimensional de artículos rectangulares, permitiendo rotaciones de 90º, sin

la restricción de que la solución final sea guillotinable y buscando minimizar la altura final de la disposición de los artículos.

Definiendo el problema de manera más formal, el problema de empaque es aquel en el que se tiene un conjunto inicial de n artículos rectangulares , cada uno definido con un ancho y alto específicos. Se observan dos variantes para el caso 2-dimensional, dependiendo de las características del contenedor:

1. **Problema de empaque 2-dimensional en cinta (2DSPP):** El contenedor (referenciado comúnmente como "*cinta*") tiene un ancho fijo W y altura infinita. El objetivo es emplazar los artículos en la cinta sin que se superpongan minimizando al altura total.
2. **Problema de empaque 2-dimensional en recipiente (2DBPP):** Se tiene una cantidad infinita de contenedores con ancho W y altura finita H. El objetivo es emplazar los artículos en la menor cantidad de recipientes.

El tipo de contenedor referenciado en éste estudio corresponde al de una cinta, es decir, 2DSPP.

Por otro lado, aunque ya se mencionaron algunas restricciones para el problema de 2DSPP, es conveniente referirse a ellas nuevamente. En este estudio se considerarán las siguientes restricciones básicas, algunas de las cuales son descritas por Riff et al en [12]:

1. Todos los artículos deben estar contenidos dentro del contenedor, es decir, no puede darse el caso de que un objeto tenga parte de su área fuera de los límites del contenedor.
2. Los artículos no deben solaparse en la disposición final.
3. El ancho de un artículo no puede ser mayor al del contenedor.
4. El alto de un artículo no puede ser mayor a la del contenedor.

3. Estado del Arte

El primer intento de abordar el problema de empaque y corte 2-dimensional nace de mano de los autores Gilmore y Gomory en [7], quienes lo tratan como una aproximación más general al problema de corte y empaque 1-dimensional. En dicho trabajo, proponen un método basado en generación de columnas.

Dowsland [6] (1993) prueba una aproximación al problema de 2DSP mediante SA, explorando tanto soluciones feasibles como soluciones donde los artículos se superponen. El autor utiliza como función objetivo el área total superpuesta de los artículos. El vecindario de la solución, por otro lado, lo define como todas aquellas soluciones donde se apliquen movimientos horizontales o verticales a la solución actual.

Jakobs [9] (1996) propone el uso de GA para el 2DSPP utilizando una representación del patrón de empaque en la que se hacen permutaciones de acuerdo al orden en el cual los items son empacados, mientras que la posición en la que es empacado está dictada por la heurística BL.

Más recientemente, Hopper y Turton (2001) [8] elaboran una investigación fundamental para el 2DPP ("*2D packing problem*") en donde compararon el rendimiento de algoritmos híbridos que utilizan meta-heurísticas: algoritmos genéticos (GA), evolución ingenua (NE) y simulated annealing (SA, "*recocido simulado*") entre sí para problemas de empaque de diferentes tamaños

en conjunto con heurísticas de empaque. Para todos los métodos estudiados se escogió una aproximación híbrida de dos etapas, donde la meta-heurística es usada para encontrar la secuencia de entrada de los artículos a empacar y una heurística es encargada de encontrar la posición en la que empacar el artículo. Los autores observan en su trabajo el alto costo computacional y temporal que requiere resolver el problema para las instancias de clases 5, 6 y 7 mostradas en 1. Las heurísticas de empaque usadas por [8] son aquellas que pertenecen a la clase de heurísticas "Bottom-left"(abajo-izquierda). Baker et al ([2], 1980) propuso esta clase de heurísticas para el problema de empaque en dos dimensiones, estás heurísticas se caracterizan por preservar la estabilidad en la esquina inferior izquierda del emplazamiento. Se distinguen dos variaciones principales:

1. **BL (Bottom-Left)**: Jakobs (1996)[9] especifica esta rutina, utilizándola en conjunto con un algoritmo genético para el emplazamiento de polígonos en una superficie. Comienza emplazando el artículo en la esquina superior derecha del contenedor, luego, este es desplazado tanto como se pueda hacia el fondo y luego tanto como se pueda hacia la izquierda de la cinta. El proceso de movimientos verticales y horizontales se repite hasta que el artículo está en una posición estable, el problema de esta aproximación es que dicha rutina de movimientos es propensa a generar áreas vacías en el emplazamiento cuando los artículos de mayor tamaño bloquean el movimiento de los siguientes a emplazar. Baker et al [2] muestra que tiene complejidad $O(N^2)$ donde N es el número de artículos a empacar. En la figura 1 se muestra un ejemplo del posicionamiento que realiza BL cuando se trata de empacar el artículo 6.

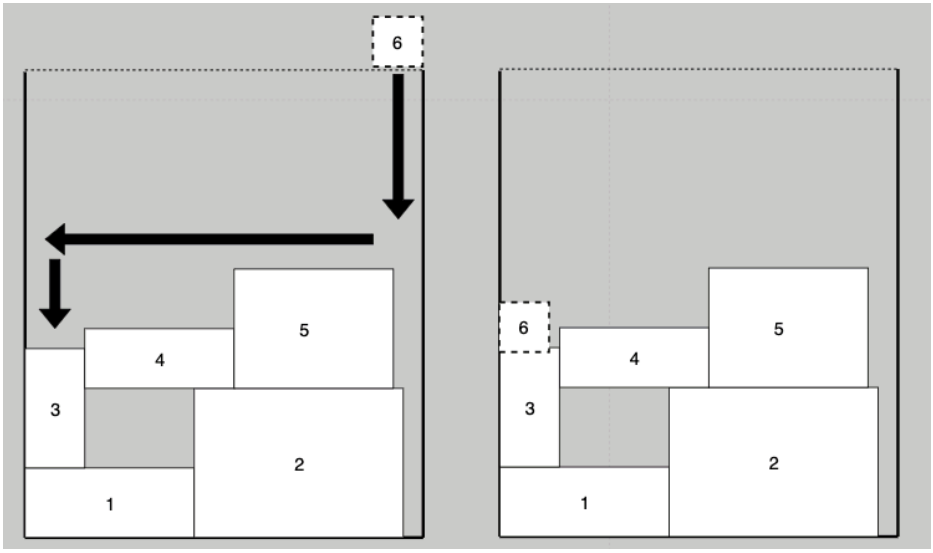

Figura 1: Ejemplo del funcionamiento de empaque del algoritmo BL (Jakobs, 1996 [9])

2. **BLF (Bottom-Left-Fill)**: Nace como una solución al problema de BL de generar grandes espacios vacíos en el emplazamiento. La estrategia consiste en emplazar un rectángulo en la posición más baja disponible y luego justificarla hacia la izquierda. Esta heurística resulta en espacios más densos que BL, como indica Chazelle (1983) [4], sin embargo, posee una complejidad de $O(N^3)$. En la figura 2 se puede ver un ejemplo del empaque realizado por la heurística.

Como relatan los autores, la calidad del emplazamiento ("layout") contruido utilizando las heurísticas BL o BLF depende de la secuencia de artículos presentada a la heurística, puesto

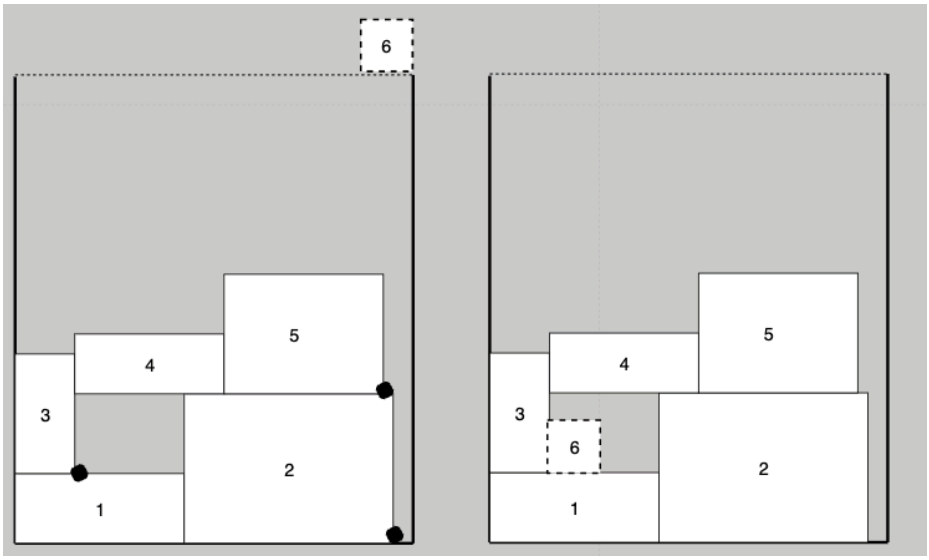

Figura 2: Ejemplo del funcionamiento de empaque del algoritmo BLF

que el número de combinaciones es demasiado grande como para ser explorado de forma exhaustiva, se prefieren estrategias de búsqueda metaheurística para resolver este problema y evaluar el orden encontrado con las metaheurísticas con el valor de la función de evaluación luego de empacar los items de la secuencia encontrada.
Hopper y Turton [8] experimentan con aproximaciones hibridas utilizando tres meta-heurísticas: GA, NE, SA, un algoritmo de búsqueda local: Hill Climbing (HC) y dos heurísticas de empaque: BL y BLF. Los autores midieron la calidad de sus soluciones con una función de ajuste que realiza una suma ponderada con 70 % para la altura del empacado y 30 % para la densidad del mismo.
Los resultados encontrados indican que la heurística BL es más eficiente en términos de tiempo de ejecución que BLF, debido a que la capacidad de esta última de llenar agujeros en el emplazamiento aumentan su costo. Con respecto a la calidad de solución los algoritmos meta-heurísticos superan a las heurísticas de empaque (BL y BLF) y a HC, siendo SA la meta-heurística con mejores resultados. Los autores hacen notar también que GA y NE son mejores que SA en cuanto a tiempo de ejecución.

Además del completo estudio realizado por Hopper y Turton [8], uno de sus más gran aportes en el citado trabajo fue la creación de instancias para formular un benchmark de algoritmos en la resolución de 2DSPP. Los autores evalúan el rendimiento de los algoritmos con siete tamaños diferentes para el problema de empaque, variando desde 17 a 197 artículos. La dimensión de los rectángulos fue generada de forma aleatoria y los problemas fueron construidos de forma que la solución óptima es conocida. En la Tabla 1 se muestran las clases y las alturas óptimas propuestas por Hopper y Turton.

Chen y Huang (2007) [5] proponen una aproximación diferente a los trabajos hasta el momento, su algoritmo se basa en el principio Less-Flexibility-First (LFF), una heurística determinista propuesta por Wu et al (2002) [16], el principio radica en emplazar los artículos en las esquinas (del contenedor y del emplazamiento), luego en los límites y finalmente en los espacios vacíos dentro del emplazamiento en dicho orden de importancia/prioridad. Los autores muestran dos algoritmos, de los cuales A_1 es el que muestra tener mejores resultados, este método usa una estrategia de búsqueda utilizando el beneficio (basándose en la densidad del emplazamiento) de colocar una artículo en una determinada posición colindante con esquinas del emplazamiento

Categoría del problema	Número de artículos	Altura óptima	Dimensiones de la cinta
C1	16 o 17	20	20x20
C2	25	15	40x15
C3	28 o 29	30	60x30
C4	49	60	60x60
C5	72 o 73	90	60x90
C6	97	120	80x120
C7	196 o 197	240	160x240

Cuadro 1: Clases de instancias generadas por Hopper y Turton [8].

actual, esto como señalan, hace que A_1 tenga una complejidad $O(n^8)$, sin embargo, los autores mencionan que dicha cota es una estimación bastante gruesa debido a que la mayoría de las esquinas que observa el algoritmo son infeasibles, disminuyendo la cantidad de esquinas candidatas. Chen y Huang comparan sus resultados con dos de las mejores meta-heurísticas implementadas por Hopper y Turton (2001) [8]: GA+BLF y SA+BLF, además con los algoritmos de otros autores como HR de Zhang et al. (2006) [17], LFFT de Wu et al (2005) [15] y SPGAL de Bortfeldt (2006) [3], usando 21 instancias de prueba en el rango 16 a 197 definidas en el trabajo de Hopper y Turton [8].
Los resultados experimentales muestran que A_1 genera soluciones óptimas para 8 de las 21 instancias probadas, para las 13 restantes la diferencia entre la altura alcanzada por el algoritmo y la óptima difieren en una unidad de largo. Midiendo la calidad de las soluciones como la brecha porcentual a la altura óptima, Chen y Huang (2007) [5] encuentran que su algoritmo A_1 se desempeña mejor que las meta-heurísticas de Hopper y Turton [8] y HR [17] en términos de la densidad del empaquetado. Comparando con LFFT [15] y SPGAL [3], la brecha media de A_1 es mejor que LFFT, pero mayor que SPGAl. Una de las grandes ventajas de [5] es que el algoritmo propuesto: A_1 es determinista.

Wei et al. (2011) [14] proponen una heurística greedy que representa el empacado de una forma distinta a la de los demás autores, Wei et al. lo representan mediante el *"horizonte"* (*skyline*), la cual definen como el contorno de un set de barras verticales consecutivas, además, usan una función de evaluación donde se incorpora la minimización de la pérdida local de espacio, sumada a través de todo el empacado (layout), sobre esto implementan Tabú search utilizando como parámetros el número de secuencias generadas en cada iteración (tamaño del vecindario) y el número de iteraciones donde se prohíbe el movimiento de *swap* (*permanencia tabú*), luego de un período de experimentación encontraron que tamaños de vecindario entre 5 y 15, junto con una permanencia tabú de entre $2n$ y $4n$ (donde n es el número de artículos de la instancia) entregaba los mejores resultados. Autores también se basaron en las instancias de [8] para la experimentación mostrando resultados competitivos en cuanto a calidad con otras aproximaciones recientes, pero recalcando que su aproximación tiene una eficiencia mayor en cuanto a tiempo de ejecución.

Thomas y Chaudhari (2013) [13] utilizan un algoritmo genético junto con una función de ajuste apropiada, dentro de sus experimentos, autores mencionan que minimizar la altura como uno de los objetivos en la función de evaluación en conjunto con el área no utilizada dentro del empaque obtenido ayuda a mejorar los resultados. Para el algoritmo genético, usan una representación tal que cada cromosoma representa una solución al problema.
Los autores diseñan una función de evaluación que incorpora información sobre la cantidad de espacio desperdiciado dejado en el empacado así como también la altura total, las ventajas de esta aproximación es que permite obtener soluciones más compactas, como desventaja se puede

mencionar el elevado costo computacional producto de la implementación del algoritmo genético, el cual requiere realizar pasos de codificación y decodificación varias veces durante la búsqueda. Los autores se basan en el benchmark de Hopper y Turton [8] usando 5 instancias de las clases $C1$, $C2$, $C3$, $C6$ y $C7$ obteniendo resultados comparables a los de implementaciones de algoritmos genéticos contemporáneas para el 2DSPP.

Babaoglu (2017) [1] propone el uso de la meta-heurística de la mosca de la fruta (fruit-fly optimization algorithm, FFOA), el cual fue propuesto por Pan (2012) [11], para resolver el 2DSPP. Su aproximación, al igual que otros autores mencionados anteriormente, contempla usar FFOA para encontrar la secuencia óptima de artículos en la cola de empaque para luego utilizar una aproximación BLF. FOA simula los comportamientos de osfresia y visión que muestran las moscas de la fruta, estas son capaces de detectar alimento y guiarse hasta el mismo encontrando la orientación usando su capacidad de osfresia, luego, usando su visión para acercarse dependiendo del comportamiento de otras moscas.

La representación para esta aproximación es tal que cada mosca en FFOA representa una secuencia de artículos. El autor prueba su aproximación con el benchmark de Hopper y Turton [8], encontrando un desempeño levemente peor que BLF-DH (Decreasing Height) y BLF-DW (Decreasing Width) para instancias de gran tamaño ($C4$, $C5$, $C6$ y $C7$), sin embargo, obtiene mejores resultados para los problemas de la clase $C1$ en comparación a otras meta-heurísticas híbridas.

La tendencia general de la mayoría de las aproximaciones más recientes y efectivas para el 2DSPP, según el estudio realizado, es a utilizar una aproximación híbrida en la que se ocupa una meta-heurística para encontrar el orden de los artículos en la linea de empaque, y luego una heurística como BLF o BL para empacar los artículos, evaluando la calidad del orden según una función de evaluación que incorpore información sobre el alto total obtenido del empaque y la densidad del mismo. Las meta-heurísticas más comunes y eficaces parecen ser simulated annealing y algoritmos genéticos.
Como menciona Riff et al [12] el 2DSPP es un problema con una gran cantidad de literatura y aproximaciones diferentes, por lo que es complejo realizar un estudio exhaustivo de las soluciones propuestas, sin embargo, en este trabajo se trató de comprender aquellos estudios más relevantes para el problema.

4. Modelo Matemático

Riff et al. [12] muestra el siguiente modelo en el suvey realizado:

Sea un conjunto N de n artículos (rectángulos) emplazados en un área (contenedor o cinta) sin superposición, tal que el alto H de empacado es minimizado considerando un ancho fijo W. Cada artículo i tiene dimensiones h_i (altura) y w_i (ancho) $\forall i = 1, ..., n$. La posición de un artículo en el contenedor rectangular es identificada usando coordenadas en el plano cartesiano tal que la esquina inferior izquierda de la cinta está en el punto $(0, 0)$ y se extiende hacia el cuadrante positivo I.
Entonces, el modelo , que buscará minimizar la altura (H), queda como sigue:

Minimizar H

Sujeto a:

$$x_i + w_i \leq W, \quad \forall i \in N \tag{1}$$

$$y_i + h_i \leq H, \quad \forall i \in N \tag{2}$$

$$\begin{aligned} &x_i + w_i \leq x_j \quad \text{or} \\ &\quad x_j + w_j \leq x_i \quad \text{or} \\ &\quad y_i + h_i \leq y_j \quad \text{or} \\ &\quad y_j + h_j \leq y_i, \\ &(i, j) \in N, \quad i \neq j \end{aligned} \tag{3}$$

$$x_i + y_i \geq 0, \quad \forall i \in N. \tag{4}$$

Donde:

$$\sum_{i \in N} h_i = H$$

La restricción (1) controla que un artículo no puede ser empacado en una posición tal que parte de su área se encuentre fuera del contenedor debido al ancho del mismo. La restricción (2) controla que ningún artículo sea empacado de forma tal que su alto supere el del contenedor, en el caso de 2DSPP, puesto que se asume una cinta de altura H infinita esta restricción se ve relajada. La restricción (3) controla que los artículos no puedan ser empacados de forma tal que se superpongan sus áreas. Finalmente, la restricción (4) controla que los objetos solo puedan ser empacados en el cuadrante I (positivo) del plano cartesiano, desde donde se define la cinta.
Notar que el modelo mostrado por Riff et al. [12] distingue la posición de cada artículo en la cinta a partir de la posición de su esquina inferior izquierda.

Zhang et al. [17] modela el problema de forma tal que se busca encontrar n sets de dos tuplas cada uno, las cuales codifican la esquina inferior izquierda y la superior derecha de cada artículo. Luego, el problema queda formulado formalmente por el siguiente modelo:
Dada una cinta rectangular con ancho W fijo y alto H infinito y un conjunto de n artículos rectangulares de ancho w_i y largo l_i, $1 \leq i \leq n$, tomando el origen del plano cartesiano como el punto en el que se encuentra la esquina inferior izquierda de la cinta, (x_L, h) denota el punto de coordenadas de la esquina superior izquierda de la cinta, mientras que (x_R, y_R) denota la esquina inferior derecha de la misma. Se busca encontrar un conjunto de n pares de tuplas de la forma:

$$P = \{\langle (x_{li}, y_{li}), (x_{ri}, y_{ri}) \rangle \quad | \quad 1 \leq i \leq n, \quad x_{li} < x_{ri}, y_{li} > y_{ri}\}$$

Donde (x_{li}, y_{li}) denotan las coordenadas de la esquina superior izquierda del artículo i y (x_{ri}, y_{ri}) denota la esquina inferior derecha del artículo. Luego, para todo $i \leq i \leq n$ se tienen las siguientes restricciones:

$$x_{ri} - x_{li} = l_i \wedge y_{li} - y_{ri} = w_i \quad or \quad x_{ri} - x_{li} = w_i \wedge y_{li} - y_{ri} = l_i \tag{5}$$

$$\begin{aligned} &x_{ri} \leq x_{lj} \quad or \quad x_{li} \geq x_{rj} \quad or \\ &y_{ri} \geq y_{lj} \quad or \quad y_{li} \leq y_{rj} \qquad \forall i \leq i \leq n, j \neq i \end{aligned} \tag{6}$$

$$x_L \leq x_{li} \leq x_R, x_L \leq x_{riR} \quad and \quad y_R \leq y_{li} \leq h, y_R \leq y_{ri} \leq h \tag{7}$$

La función objetivo es:

Minimizar h

Donde:

$$\sum_{i \in N} l_i = h$$

La restricción (5) especifica que las dimensiones del artículo sean consistentes. La restricción (6) impide que haya sobreposición entre los artículos al ser empacados y la restricción (7) impide que se empaquen artículos de forma que parte de su área esté fuera de los límites de la cinta. Notar que para el caso de 2DSPP, donde se asume que la cinta tiene un largo infinito, la restricción (7) se relaja.

5. Representación

Se definirá como una representación válida toda lista ordenada de tamaño N, donde N es la cantidad de objetos a empacar de la instancia, en la que se encuentra el identificador único de cada artículo, por lo general especificado en las instancias mediante una sucesión de tamaño N de números naturales. El orden en el que aparecerán los identificadores de los artículos en una solución corresponderá al orden en el que se insertan los mismos en la cinta mediante la heurística escogida. Finalmente, puesto que en este desarrollo se considerará la posibilidad de rotación de un artículo previo ingreso a la cinta, se especificará, en una lista de tamaño N, si el artículo de una determinada posición fue rotado o no mediante codificación binara, donde 1 indicará si el artículo fue rotado y 0 si no.

Se puede representar la solución, entonces, como una lista de tuplas, en las que cada tupla contiene el identificador único del artículo y el indicador de rotación.

Finalmente, notar que todas las heurísticas de emplazamiento mencionadas en este trabajo son deterministas, es decir, dado un artículo (rotado o no), la posición que ocupa este en el emplazamiento de la cinta es siempre el mismo si se mantiene constante el emplazamiento de la cinta. Es posible conocer siempre la posición final de los artículos al aplicar la heurística sobre una solución conociendo solo el orden de ingreso, la rotación de los artículos y la heurística utilizada. Lo anterior implica que se puede calcular el espacio de búsqueda del algoritmo de búsqueda conociendo solo la cantidad de artículos a ingresar en la cinta.

Esta representación de las soluciones y teniendo en cuenta que solo se aplicará rotación solo antes del ingreso de un objeto a la cinta (no durante su movimiento dentro de esta hacia su posición final) conlleva a un espacio de búsqueda de $2^n \cdot n!$. 2^n es la cantidad de rotaciones que es posible aplicar los objetos de la solución y $n!$ es la cantidad de permutaciones totales posibles para los elementos de la lista.

Un ejemplo de representación sería la siguiente:

$$[(1,0),(2,1),(3,1),(4,0),(5,1)]$$

6. Descripción del Algoritmo

El algoritmo de búsqueda que se implementa en este trabajo es conocido como *Hill Climbing Mejor Mejora + Restart*, mostrado en **Algorithm 1**:

Algorithm 1 Hill Climbing Mejor Mejora con Restart

```
1:  procedure HC-MM+RESTART
2:      t ← 0
3:      inicializar s_best
4:      while t != MAX do
5:          local ← False
6:          s_c ← seleccionar un punto aleatorio del espacio de soluciones
7:          while not local do
8:              s_n ← seleccionar el punto de mejor calidad en N(s_c)
9:              if f(s_n) es mejor que f(s_c) then
10:                 s_c ← s_n
11:             else
12:                 local ← True
13:             t ← t + 1
14:             if f(s_c) es mejor que f(s_best) then
15:                 s_best ← s_c
```

Hill Climbing es una meta-heurística de búsqueda incompleta de soluciones que trabaja de la siguiente manera: Primero, se debe comenzar en un punto del espacio de búsqueda (solución) cualquiera calculandose su respectivo valor de la función de evaluación, luego, para la variante *Mejor Mejora*, mostrada en **Algorithm 1**, se evalúa la función de evaluación para todas las soluciones en el vecindario de la solución actual, seleccionando aquella que presenta mejor función de evaluación dentro de las alternativas del vecindario generado. Para la implementación realizada en este trabajo, la función de evaluación es la altura del emplazado final de una lista de artículos en la cinta, utilizando como heurística de emplazamiento el algoritmo BL, por lo tanto, una solución es mejor que otra si luego de aplicar la heurística de emplazamiento sobre todos los artículos de la lista obtiene una altura menor.

Para efectos de este trabajo, se definió como vecindario de una solución actual $\mathcal{N}(s_c)$ a toda aquella solución que se encuentra a un movimiento de distancia de la solución actual s_c. Al mismo tiempo, se definió como "*movimiento*.[a] un swap en el orden de dos artículos cualquiera de la lista de artículos de una solución, por lo tanto, el vecindario de una solución es toda aquella solución que está a un swap de distancia de la solución actual.

Si la mejor solución del vecindario es mejor que la solución actual en que se encuentra el algoritmo, se actualiza la mejor solución encontrada hasta el momento(S_c) y se utiliza esta nueva solución como punto de calculo para el nuevo vecindario.

En caso de que ninguna solución en el vecindario de la solución actual sea mejor que esta última, se está en presencia de un óptimo local y el algoritmo se detiene.

Puesto que el óptimo local encontrado por el algoritmo es fuertemente dependiente del punto de inicio del mismo, la variante implementada, denotada por *Restart*, realiza varias inicializaciones aleatorias del algoritmo en, ojalá, distintos puntos del espacio de búsqueda, retornando finalmente el mejor óptimo local encontrado entre todas las inicializaciones.

Hill Climbing con Restart es un algoritmo que intensifica en la búsqueda con el mecanismo de selección de s_n en el vecindario de una solución, la variante de mejor mejora intensifica más que otras variantes como *primera mejora*. También se encuentra regulada la intensificación mediante la cantidad de iteraciones máximas que se le permite realizar al algoritmo en búsqueda de un óptimo local. Por otro lado, la diversificación está controlada por la cantidad de reinicios (*restarts*) que realiza el algoritmo, para efectos de este trabajo, se ajusto la cantidad de reinicios a 5 y permitiéndose una máxima de iteraciones de intensificación a 30, es decir,de no ser capaz la meta-heurística de encontrar un óptimo local en 30 movimientos dentro del espacio de búsqueda, se da por terminada la ejecución y se retorna el valor actual en el que se encontraba, notar que esto no es lo mismo que generar un vecindario de hasta 30 elementos, pues este siempre es evaluado completamente por ser mejor mejora. Es interesante notar que esta limitante en la cantidad de pasos de intensificación tiene desventajas cuando se está en un espacio de búsqueda de gran tamaño, como el resultante para un alto número de artículos.

7. Experimentos

Se realizaron varios experimentos con el objetivo de medir tanto la eficacia como la eficiencia del algoritmo implementado, es decir, las dos métricas principales a estudiar son el tiempo de ejecución total del algoritmo (y cómo escala este dependiendo de la cantidad de artículos en la instancia) y la diferencia entre el óptimo global (altura mínima posible para una cinta con un determinado ancho y los artículos a emplazar) y la altura alcanzada por el algoritmo.
Se utilizo las instancias de Hopper y Turton [8] puesto que los autores entregan la altura óptima para cada instancia, lo que permitirá medir qué tanto se aleja la solución encontrada por el algoritmo con la solución óptima.
La meta-heurísitca de búsqueda utilizada fue Hill Climbing Mejor Mejora con Restart, utilizando 5 reinicios como medida de diversificación por defecto, excepto para las instancias de las clases $C6$ y $C7$ para las cuales se usaron solo 3 reinicios. Con respecto a la intensificación del algoritmo, esta se limitó a hasta 30 iteraciones de intensificación para todas las instancias. Las razones por las cuales se limitó la capacidad de diversificación del algoritmo para las clases de mayor tamaño es por los altos tiempos computacionales que implicaba encontrar soluciones para estas clases.

También se estudió el rendimiento del algoritmo en algunas de las instancias de Lodi et al. [10], para las cuales se buscó estudiar el tiempo de ejecución y la cantidad de área inutilizada, de forma similar que para las instancias de Hopper y Turton se utilizó para todas las pruebas una cantidad de reinicios de 5 y un máximo de iteraciones de intensificación de 30.

Como heurística de emplazamiento se utilizó BL debido a la facilidad de implementación y bajo coste computacional que tiene.

Finalmente, mencionar que para todos los experimentos se utilizó una semilla generada a partir del reloj del computador.

Nombre Instancia	# Artículos	# Restarts	Ancho Cinta	Altura Mínima Encontrada	Altura Óptima	Tiempo CPU (s)	Dif. A.Opt y A. Encontrada
HT c1-p1	16	5	20	22	20	0.3	2
HT c1-p2	17	5	20	22	20	0.39	2
HT c2-p1	25	5	40	16	15	2.16	1
HT c3-p1	28	5	60	34	30	5.59	4
HT c3-p2	29	5	60	33	30	5.96	3
HT c4-p2	49	5	60	67	60	54.48	7
HT c5-p1	73	5	60	99	90	276.02	9
HT c6-p1	97	3	80	137	120	645.05	17
HT c7-p1	196	3	160	278	240	18845.76	38

Cuadro 2: Resultados experimentales para las instancias de Hopper y Turton [8].

8. Resultados

En la tabla 2 se muestran los resultados obtenidos para todas las instancias probadas de Hopper y Turton [8] (adjuntas en este trabajo). Como se puede observar en la figura 3, el tiempo de ejecución del algoritmo crece exponencialmente con la cantidad de artículos a emplazar, esto se condice con lo mencionado anteriormente respecto a la explosión combinatoria del espacio de búsqueda para alta cantidad de artículos.

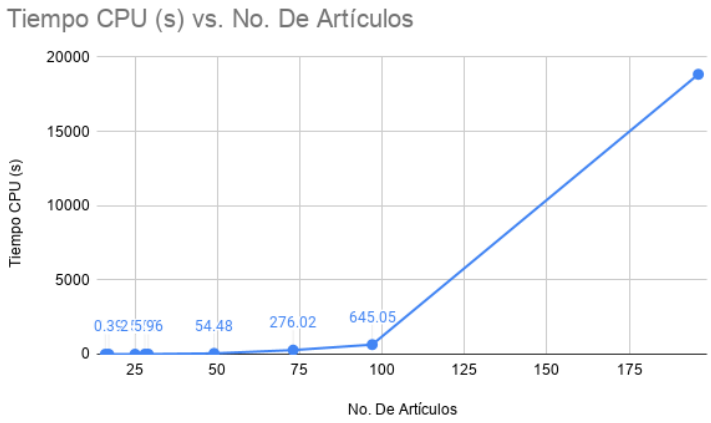

Figura 3: Tiempos de ejecución del algoritmo para distintas instancias, según cantidad de artículos.

Nombre Instancia	# Artículos	# Restarts	Ancho Cinta	Altura Mínima Encontrada	Espacio Inutilizado	Tiempo CPU (s)
BENG01	20	5	25	31	34	1.16
BENG02	40	5	25	63	83	16.52
BENG03	60	5	25	92	186	72.93
BENG04	80	5	25	117	236	246.88
BENG05	100	5	25	145	287	520.36
BENG08	120	5	40	111	365	1608.72

Cuadro 3: Resultados experimentales para las instancias BENG de Lodi et al. [10].

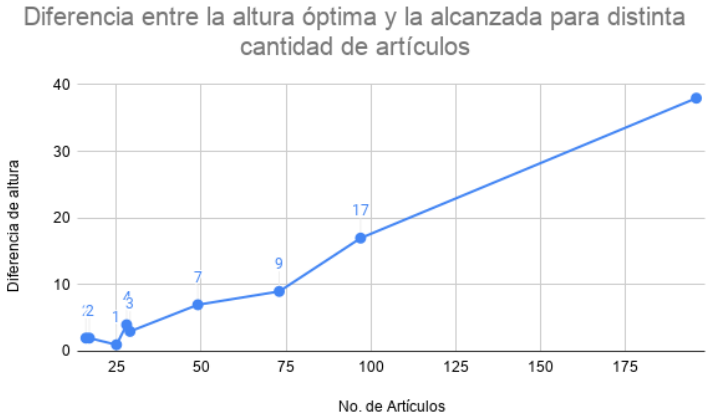

Figura 4: Diferencias entre la distancia encontrada y la óptima para las instancias probadas.

Se observa de la figura 4 que la diferencia entre la altura óptima y la encontrada por el algoritmo aumenta también con el incremento de la cantidad de artículos en la instancia, sin embargo, no de forma exponencial como el tiempo de cómputo, sino más bien lineal, esto implica que el algoritmo se hace linealmente menos eficaz a medida que aumenta el número de artículos en la instancia.
La cantidad de tiempo necesaria para poder ejecutar el algoritmo en una instancia de gran tamaño($C6$ o $C7$) es bastante grande, lo que hace que la implementación actual sea poco eficaz para este tipo de instancias, sin embargo, para instancias de menor tamaño no solo presenta tiempos de cómputo bajos sino que también poco error en comparación a la altura óptima de la instancia. Una forma en la que se puede copar con el comportamiento exponencial del tiempo de ejecución es dividir la cola de artículos en dos o más colas disjuntas y aplicar Hill-Climbing de forma paralela sobre estas colas, uniendo las soluciones finales de cada proceso, ya sea tratando cada solución parcial como si fuese un artículo en la cola con dimensiones iguales a las dimenciones del emplazamiento específico encontrado, o apilando las soluciones parciales en la cinta final.

Finalmente, en la tabla 3 se muestran los resultados obtenidos para las instancias de Lodi et al. [10], para todas las instancias mostradas se utilizó una cantidad de reinicios de 5 y un límite de movimientos de intensificación de 30.

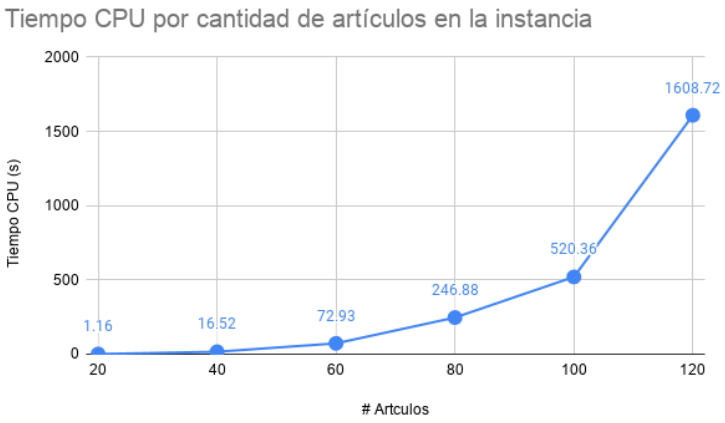

Figura 5: Tiempos de ejecución del algoritmo para distintas instancias de Lodi et al. [10], según cantidad de artículos.

Se puede observar de igual forma que para los experimentos con las instancias de Hopper y Turton que el tiempo de ejecución crece exponencialmente con la cantidad de artículos a empacar. Por otro lado, el espacio inutilizado por cantidad de artículos, mostrado en la figura 6, crece de forma lineal, de forma similar a la diferencia entre el área óptima y la alcanzada para las instancias de Hopper y Turton. Es interesante notar que el tiempo de ejecución es considerablemente menor, para instancias de similar tamaño, con respecto a las de Hopper y Turton, la razón de esto es debido a que el ancho de la cinta es menor para aquellas instancias de mayor tamaño, lo que resulta en menor cantidad de posiciones distintas a probar de cada artículo por la heurística de emplazamiento. Como se mencionó anteriormente, el algoritmo es eficiente para instancias de pequeño tamaño, por lo que cintas de ancho reducido lo favorecen.

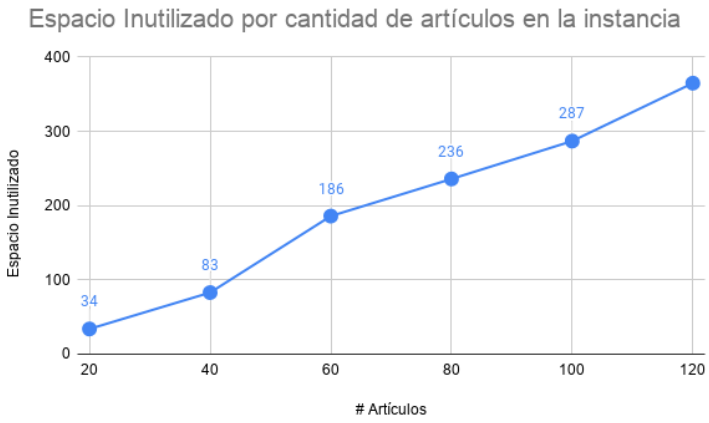

Figura 6: Área inutilizada del algoritmo para distintas instancias de Lodi et al. [10], según cantidad de artículos.

9. Conclusiones

2DSPP es un problema ampliamente enfrentado en la industria y por lo tanto resolverlo de forma rápida y efectiva es muy relevante para optimizar los procesos asociados, por ejemplo, agilizar los procesos de corte de material permite pasar los artículos por las siguientes etapas de procesamiento de forma más temprana y ser capaz de producir mayor cantidad de artículos en menor cantidad de tiempo. En este trabajo se resolvió el problema de 2D Strip Packing permitiendo rotabilidad, sin el requerimiento de guillotinabilidad, para objetos rectangulares y teniendo como función objetivo minimizar la altura total del emplazamiento, para esto se utilizó la meta-heurística Hill-Climbing con Mejor Mejora y Restart y la heurísitica de emplazamiento BL.
La mayoría de los autores coinciden en sus estudios en que las aproximaciones con mayor potencial son aquellas que mezclan una meta-heurística en la búsqueda del orden de los artículos en la cola, con una heurística para el emplazamiento, realizándose un proceso de dos pasos. Las meta-heurísticas que mejor desempeño muestran en cuanto a calidad del empaque son algoritmos genéticos, evolución ingenua y simulated annealing, además de ser los más estudiados, por otro lado, en cuanto a tiempo de ejecución, como observan Hopper y Turton [8], algoritmos genéticos y evolución ingenua son más rápidos que simulated annealing, teniendo resultados muy cercanos entre si en cuanto a calidad.

Es importante notar que algunos autores hacen notar que para tamaños de instancias pequeñas, una buena opción es simplemente usar una técnica de búsqueda completa o simplemente preferir aquella con mejores resultados en tiempo de ejecución (como algoritmos genéticos) ya que la diferencia entre el óptimo global y lo encontrado con la meta-heurística es pequeña.

De los experimentos se puede concluir que Hill-Climbing con Mejor Mejora y Restart es una buena aproximación al 2DSPP para instancias de limitado número de artículos ya que permite obtener resultados similares al óptimo global de las instancias probadas, sin embargo, el tiempo de ejecución exponencialmente creciente junto con el espacio de búsqueda lo hacen poco eficiente para contextos reales donde se requiere rapidez de la solución. Una solución a este problema podría ser disminuir el número de reinicios, como se hizo para las instancias de mayor tamaño en

los experimentos, paralelizar la ejecución del algoritmo lo cual es posible calculando la función de costo de distintas soluciones en un vecindario de forma paralela, otra solución es disminuir la cantidad de iteraciones de intensificación y finalmente, utilizar variantes distintas de *Mejor Mejora* conocidas como *Alguna Mejora* y *Primera Mejora*, que no evalúan todo el vecindario de una solución, sino solamente parte de este.

De forma adicional, sería relevante dejar planteada una idea de solución que surgió durante este estudio, la cual consiste en aplicar técnicas de dividir y conquistar sobre la cola de artículos y resolver el problema utilizando, para cada partición, Hill-Climbing ya sea con mejor mejora o alguna mejora para generar sub-soluciones compactas, cada una sobre una pequeña seccion de la cola de artículos, luego, utilizar estos nuevos constructos como si fueran los artículos originales en la cola de empaque y volver a aplicar una meta-heurística sobre al cola pero ahora de la forma tradicional, buscando el orden óptimo de los artículos y empacandolos con alguna heurística como BL o BLF. Una aproximación similar a generación de columnas y que es posible paralelizar.

10. Bibliografía

Referencias

[1] İsmail Babaoğlu. Solving 2d strip packing problem using fruit fly optimization algorithm. *Procedia computer science*, 111:52–57, 2017.

[2] Brenda S Baker, Edward G Coffman, Jr, and Ronald L Rivest. Orthogonal packings in two dimensions. *SIAM Journal on computing*, 9(4):846–855, 1980.

[3] Andreas Bortfeldt. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, 172(3):814–837, 2006.

[4] Bernard Chazelle. The bottomn-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, (8):697–707, 1983.

[5] Mao Chen and Wenqi Huang. A two-level search algorithm for 2d rectangular packing problem. *Computers & Industrial Engineering*, 53(1):123–136, 2007.

[6] Kathryn A Dowsland. Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research*, 68(3):389–399, 1993.

[7] PC Gilmore and Ralph E Gomory. Multistage cutting stock problems of two and more dimensions. *Operations research*, 13(1):94–120, 1965.

[8] EBCH Hopper and Brian CH Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128(1):34–57, 2001.

[9] Stefan Jakobs. On genetic algorithms for the packing of polygons. *European journal of operational research*, 88(1):165–181, 1996.

[10] Andrea Lodi, Silvano Martello, and Daniele Vigo. Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem. In *Meta-Heuristics*, pages 125–139. Springer, 1999.

[11] Wen-Tsao Pan. A new fruit fly optimization algorithm: taking the financial distress model as an example. *Knowledge-Based Systems*, 26:69–74, 2012.

[12] María Cristina Riff, Xavier Bonnaire, and Bertrand Neveu. A revision of recent approaches for two-dimensional strip-packing problems. *Engineering Applications of Artificial Intelligence*, 22(4-5):823–827, 2009.

[13] Jaya Thomas and Narendra S Chaudhari. Hybrid approach for 2d strip packing problem using genetic algorithm. In *International Work-Conference on Artificial Neural Networks*, pages 566–574. Springer, 2013.

[14] Lijun Wei, Andrew Lim, and Wenbin Zhu. A skyline-based heuristic for the 2d rectangular strip packing problem. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 286–295. Springer, 2011.

[15] Yu-Liang Wu and Chi-Kong Chan. On improved least flexibility first heuristics superior for packing and stock cutting problems. In *International Symposium on Stochastic Algorithms*, pages 70–81. Springer, 2005.

[16] Yu-Liang Wu, Wenqi Huang, Siu-chung Lau, CK Wong, and Gilbert H Young. An effective quasi-human based heuristic for solving the rectangle packing problem. *European Journal of Operational Research*, 141(2):341–358, 2002.

[17] Defu Zhang, Yan Kang, and Ansheng Deng. A new heuristic recursive algorithm for the strip rectangular packing problem. *Computers & Operations Research*, 33(8):2209–2217, 2006.