Proyecto Final: Bin packing

Fernando Carrillo A01194204

25/11/20

Descripción del problema:

En el Bin Packing problem se quiere encontrar la mejor manera de empacar N objetos de distintos tamaños de tal forma que se minimicen la cantidad de contenedores M necesarios para guardarlos. Solucionar este problema tiene varias aplicaciones en el mundo real como llenar contenedores de paqueterías de envío, acomodar objetos en bodegas, o llenar particiones de memoria en una computadora, que pueden resultar en un ahorro en costos, materiales e impacto ambiental. Existen distintas variaciones del problema, diferenciados por la cantidad de dimensiones que se toman en cuenta (1D, 2D, 3D), y la habilidad de rotar objetos. En todos los casos esto resulta en un problema combinatorio NP-Hard, ya que existen diferentes maneras de acomodar objetos en los diferentes contenedores, por lo que puede ser tardío encontrar la solución óptima cuando la cantidad de objetos a ordenar es elevada.

Trabajo relacionado:

En el trabajo de Blum y Schmid (2013), se resolvió el 2D Bin Packing problem sin rotaciones utilizando una mezcla de algoritmos evolutivos y Tabu search, el cual se puede comparar al proceso de recocido simulado. Además de elegir su método de solución, tuvieron que escoger un método de organización para los objetos dentro de los contenedores. Se mencionan dos técnicas distintas: One-phase y Two-phase approach. La diferencia entre estas dos es que One-phase inserta los objetos directamente dentro del contenedor mientras que Two-phase primero llena las filas y procede a hacer un 'stack' de filas. Se optó por un proceso de One-phase llamado Improved Lowest Gap Fill (LGFi), que busca minimizar el espacio desperdiciado decidiendo si debería colocar el siguiente objeto en la misma fila que el último objeto, o en la misma columna.

Lam, Ho y Logotafu (2017) en cambio exploran la idea de que algoritmos de búsqueda avanzados como Búsqueda local, Recocido Simulado, y Algoritmos Genéticos solamente se enfocan en mejorar el orden de los objetos a empacar, y que se pueden utilizar otros algoritmos más simples como First-Fit Decreasing, donde se enfoca más en acomodar los objetos dentro de un contenedor para minimizar el espacio desperdiciado. Argumentan que los algoritmos complejos estocásticos pueden dar buenos resultados intentando soluciones distintas al azar, pero la verdadera mejora viene en el algoritmo que ordena los objetos dentro de los contenedores. Si se encuentra un mejor algoritmo para minimizar el espacio desperdiciado por contenedor, no tendría mucha relevancia el orden de los objetos, y este obtendría consistentemente mejores resultados que aquellos que se enfocan en mejorar el orden.

Descripción de la solución:

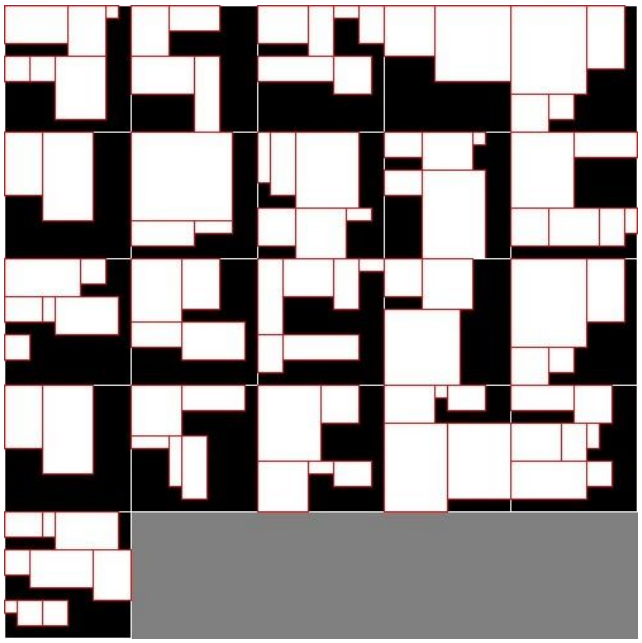
Se trabajara con el siguiente problema:

> *Se tiene una cantidad infinita de contenedores con tamaño 10 x 10, y se tienen los siguientes N=100 objetos (ancho, largo): (5, 3), (3, 4), (1, 1), (2, 2), (2, 2), (4, 5), (3, 4), (4, 2), (5, 3), (2, 6), (4, 3), (2, 4), (2, 1), (2, 3), (6, 2), (3, 3), (4, 4), (6, 6), (6, 7), (3, 5), (3, 3), (2, 2), (3, 5), (4, 7), (8, 7), (5, 2), (3, 1), (1, 4), (2, 5), (5, 6), (3, 3), (4, 4), (2, 1), (3, 2), (4, 3), (1, 1), (3, 2), (5, 7), (5, 6), (5, 2), (3, 3), (4, 3), (2, 3), (1, 2), (6, 3), (2, 2), (3, 2), (1, 2), (5, 3), (2, 2), (5, 3), (3, 4), (1, 1), (2, 2), (2, 2), (4, 5), (3, 4), (4, 2), (5, 3), (2, 6), (4, 3), (2, 4), (2, 1), (2, 3), (6, 2), (3, 3), (4, 4), (6, 6), (6, 7), (3, 5), (3, 3), (2, 2), (3, 5), (4, 7), (4, 4), (5, 2), (3, 1), (1, 4), (2, 5), (5, 6), (3, 3), (4, 4), (2, 1), (3, 2), (4, 3), (1, 1), (3, 2), (5, 7), (5, 6), (5, 2), (3, 3), (4, 3), (2, 3), (1, 2), (6, 3), (2, 2), (3, 2), (1, 2), (5, 3), (2, 2)*

En esta ocasión se intentará solucionar el problema con dos dimensiones, y sin la opción de poder hacer rotaciones. Los individuos en este problema son representados como arreglos de objetos, donde cada objeto tiene un tamaño especificado en dos dimensiones (ancho x largo). El orden de los objetos influye en la manera en la que estos serán acomodados dentro de los contenedores. Los objetos serán posicionados dentro de los contenedores empezando por la esquina superior izquierda, y siguiendo las siguientes reglas:

1. Colocar el objeto en la fila actual a la derecha del objeto anterior si hay espacio
2. Si no hay espacio en la fila, comenzar una nueva fila debajo de la previa
3. Si no hay espacio para una nueva fila, crear un nuevo contenedor

Colocar los objetos en el orden original siguiendo un orden de top-left requiere de M=21 contenedores:



Cada recuadro negro representa un contenedor, y cada recuadro blanco representa un objeto

Para solucionar este problema se utilizará el framework DEAP para crear algoritmos genéticos. Regularmente para utilizar los algoritmos genéticos se recomienda representar las soluciones o individuos con un arreglo de bits, ya que esto facilita los cruces y mutaciones. Sin embargo, para este problema los individuos son representados como arreglos de objetos de dos dimensiones, donde cada individuo debe tener los mismos objetos originales pero un orden distinto.

Descripción de cruce y mutación:

Debido a la representación de los individuos, no se pueden utilizar los métodos de cruce y mutaciones incluidos en la librería de DEAP para representaciones binarias cómo Cx.OnePoint, Cx.TwoPoint, BitFlip, etc. Por esta razón se creó una función personalizada de cruce y otra de mutación.

La función de cruce fue un poco complicada de diseñar ya que no se podían simplemente combinar las dos soluciones, pues esto podría ocasionar que se pierda la integridad de una solución. Combinar dos soluciones simplemente seleccionando un punto de cruce creaba otra solución con objetos duplicados y otros extraviados, por lo que se tuvo que tomar en cuenta el orden de ambas soluciones y llevar un registro de los objetos que ya se utilizaron en la nueva solución producto del cruce. Teniendo individuo1, individuo2, y un arreglo con los objetos disponibles (no utilizados), el resultado fue la siguiente función:

1. Seleccionar un pivote
2. Copiar los primeros elementos de ind1 hasta el pivote en la nueva solución
3. Quitar los objetos utilizados en la nueva solución del registro de objetos no utilizados
4. A partir de pivote:
    a. Intentar copiar los elementos de ind2, y actualizar el registro de objetos disponibles
    b. En caso de que ya se haya utilizado un objeto de ind2, utilizar el de ind1
    c. En caso de que ya se hayan utilizado ambas opciones de ind1 e ind2, seleccionar un objeto al azar del arreglo de objetos no utilizados

De esta manera se llega a algo similar al cruce de un punto, tomando en consideración que los objetos no se deben duplicar ni perder.

Para el método de mutación se creó una función que itera sobre los elementos de la solución, y dependiendo de una probabilidad especificada, hace un switch entre dos objetos vecinos y pasa al siguiente elemento del arreglo.

Descripción de la función de evaluación:

Inicialmente la aptitud de una solución se midió con la cantidad de contenedores que se utilizaron para acomodar todos los objetos. Posteriormente se realizaron pruebas con otra heurística, que se basaba en el espacio desperdiciado total dentro de los contenedores. Se tenía la hipótesis de que al minimizar el espacio desperdiciado, se utilizarían menos contenedores.
Se realizaron las siguientes pruebas:

Heuristica	Prom. contenedores	Prom. espacio desperdiciado	Mejor evaluación
Núm. contenedores	17	470	17 contenedores 470 espacio desperdiciado
Espacio desperdiciado	17.4	457	17 contenedores 440 espacio desperdiciado
Ambos Núm. contenedores y espacio desperdiciado	17	420	17 contenedores 390 espacio desperdiciado

Con estos resultados se llegó a la conclusión de que:

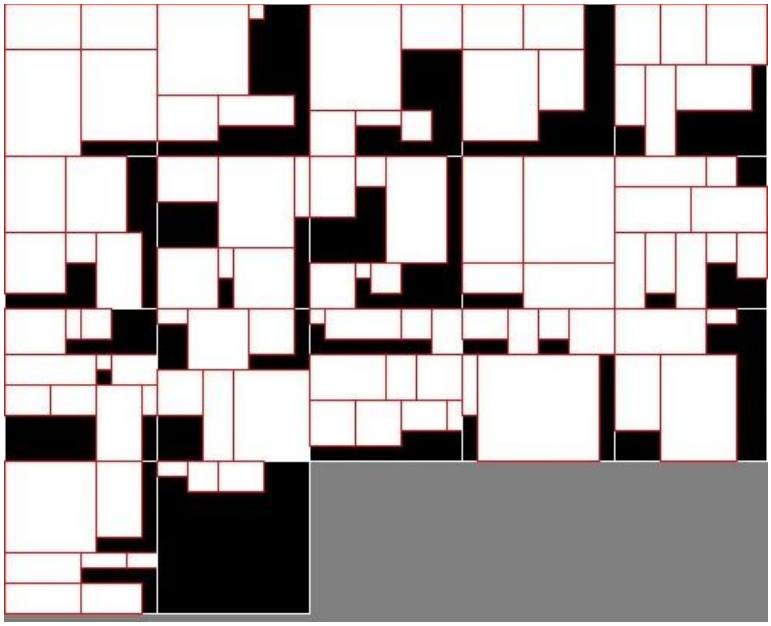
- Utilizar solamente el 'Espacio Desperdiciado' resulta en menor espacio desperdiciado en promedio, pero no siempre resulta en la menor cantidad de contenedores.
- Utilizar solamente 'Contenedores' genera soluciones más constantes, pero por lo mismo está limitada a encontrar mejores soluciones.
- Utilizar ambas heurísticas con optimización multiobjetivo resulta en las mejores evaluaciones.

Resultados obtenidos:

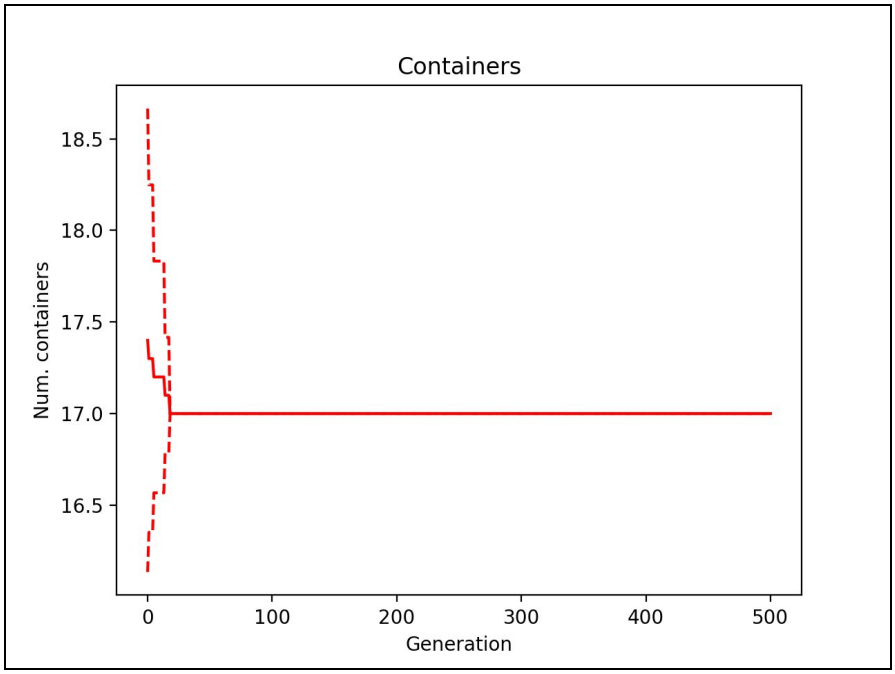
Después de correr el algoritmo con los siguientes parámetros:

Método de selección: NSGA2
Num. generaciones: 500
Num. iteraciones: 10
Mu: 50
Lambda: 100
Prob. cruce: 0.7
Prob. mutación: 0.3

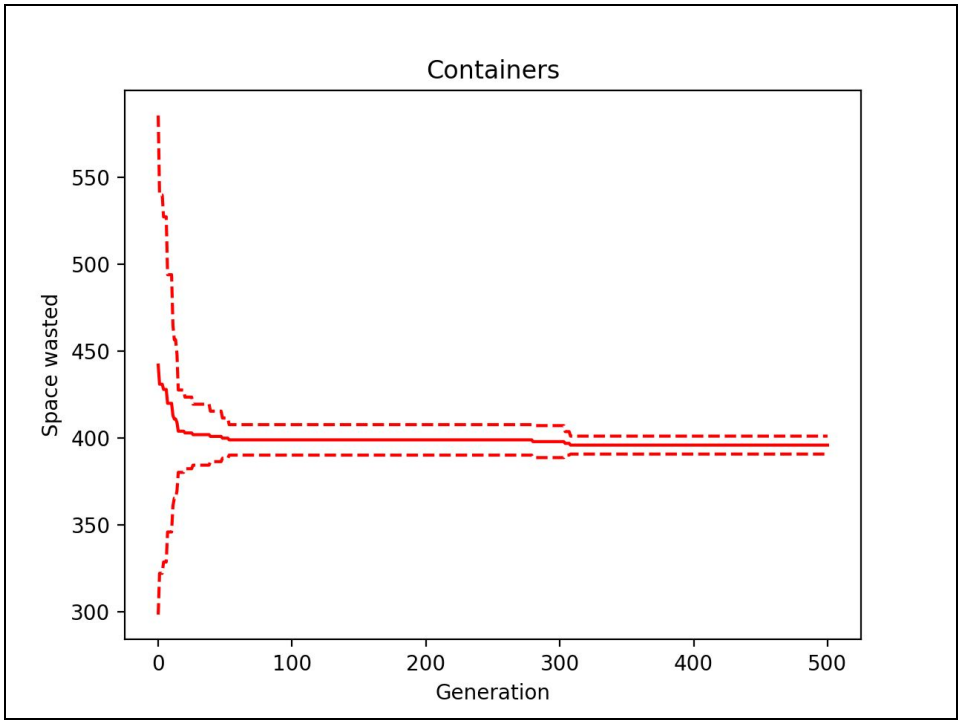
Mejor solución encontrada:



Núm. contenedores M pasó de 21 -> 17



Núm. contenedores converge en M=17 alrededor de la generación 40



El espacio desperdiciado toma más generaciones para converger en alrededor de 400 unidades.

Retos encontrados:

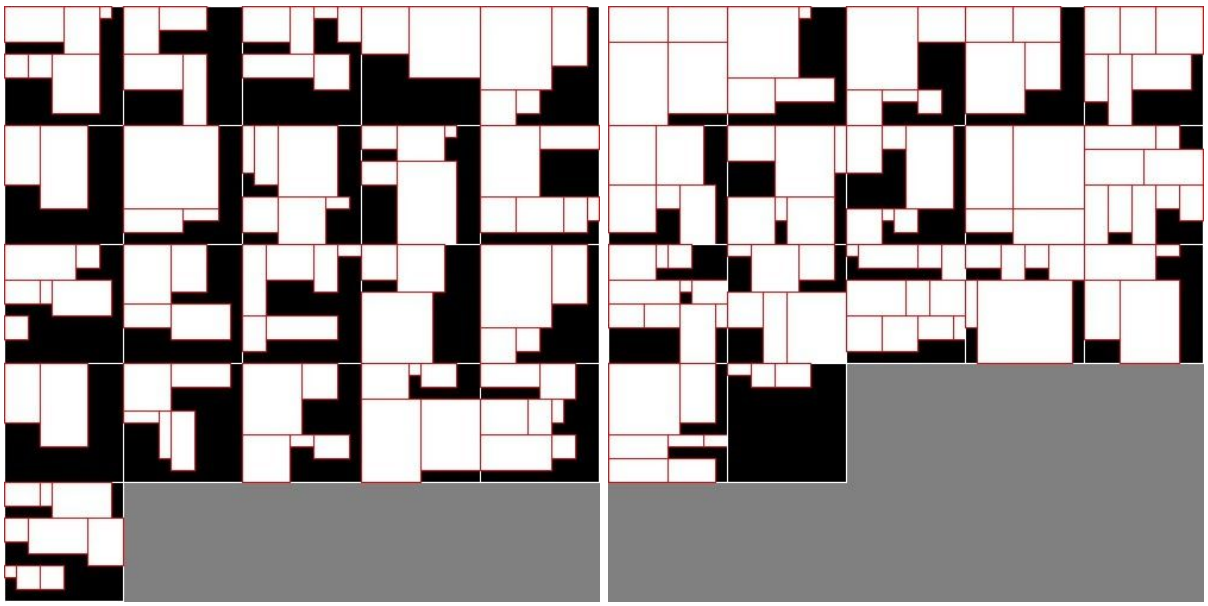
Durante el desarrollo de la solución se encontraron varios retos. El primer reto fue decidir y desarrollar el método para acomodar los objetos de dos dimensiones en los contenedores. Para solucionar este reto se tomó en consideración el trabajo relacionado y los métodos utilizados en sus soluciones. Se decidió implementar un método sencillo de Two-phase pero suficientemente bueno para encontrar mejoras.
Otro reto fue decidir las heurísticas de la función de evaluación y desarrollarla, ya que se tuvieron que calcular meticulosamente los espacios vacíos, así como los pasos para llenar los contenedores.
El último reto fue programar el método de cruce, ya que los métodos de cruce tradicionales no toman en cuenta las restricciones para mantener la integridad de las soluciones.

Conclusiones:

Al inicio pareció que no se logró desarrollar una solución buena, ya que al ver las soluciones se encontraron espacios vacíos y áreas de mejora. Pero al compararlo con una situación real se aprecia más el resultado.
Suponiendo que se tiene el problema anterior (donde se tienen 100 objetos a ordenar), y que cada contenedor utilizado tiene un costo adicional de 2,000 pesos, acomodar los objetos en su orden original costaría (21 contenedores x 2,000 pesos) 42,000 pesos. Pero si se utiliza el algoritmo (que

tarda aproximadamente 6 minutos en correr) se reduce la cantidad de contenedores de 21 a 17. Esto disminuye el costo a 34,000 pesos, con un ahorro aproximado del 19%.
Esto es siguiendo el resultado del algoritmo estrictamente, pero dado que el resultado no es perfecto, una opción es utilizar el algoritmo como una guía para después hacer ajustes manuales para llenar algunos espacios vacíos grandes.

Original *Mejora*



Existen áreas de mejora a investigar en futuros trabajos. Principalmente se podría mejorar el algoritmo que ordena los objetos dentro de los contenedores, al igual que permitir la opción de rotación de los objetos para incrementar el espacio de búsqueda.

Referencias:

C. Blum, V. Schmid, 2013. *Solving the 2D bin packing problem by means of a hybrid evolutionary algorithm*. Procedia Computer Science 18 (2013) 899 – 908

G.T. Lam, V.A. Ho, D, Logofatu, 2017. *Considerations on 2D-Bin Packing Problem: Is the Order of Placement a Relevant Factor?* 7th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2017), pp. 280-287