**Revisión de Heurísticas para el Problema Bin Packing de una y dos dimensiones**

Marcel Valdez

A00

Ciencias Computacionales

Tecnológico de Monterrey

Priscila Angulo López

A00813460@itesm.mx

Ciencias Computacionales

Tecnológico de Monterrey

**1. Introducción**

El problema de Bin Packing sirve para caracterizar muchos problemas de la vida real, este problema consiste en tener un conjunto ilimitado de contenedores con capacidad *W*, y un conjunto de *n* piezas para empacar, con pesos *w1, w2, w3, …, wn*; los pesos de las piezas pueden repetirse, sin embargo, se cumple la restricción . La tarea consiste en acomodar las piezas dentro de los contenedores de tal modo que se minimice el número total de contenedores utilizados [1].

Aunque este problema sea fácil de entender, su solución es NP difícil, lo cual implica que llegar a la solución óptima en todas las instancias del problema implica un esfuerzo computacional exponencial. Es por este motivo que existe amplia investigación para llegar a soluciones aceptables en tiempo aceptable, y aquí es donde entran las heurísticas y las hiperheurísticas, las primeras llegan a soluciones aproximadas y aceptables para una familia particular de instancias del problema, las segundas permiten la selección de heurísticas según las características de la instancia.

El presente reporte muestra la aplicación de dos heurísticas de selección, una para el problema de una dimensión y otra para el de dos dimensiones. El contenido está organizado alrededor de esas heurísticas, en la primera sección se aborda la heurística “First Fit Decreasing” (FFD), la cual tiene su aplicación en el problema de Bin Packing de una dimensión. Esta heurística fue implementada y probada con 3 instancias de problemas. Los objetivos que se siguieron al implementar esta heurística fueron: hacer un análisis detallado del algoritmo que implementa la heurística, probar su efectividad en las instancias propuestas y comprobar que se cumple con la cota superior para el número de contenedores (objetos) obtenidos en el peor caso.

En la segunda sección de este reporte se trata el problema de Bin Packing de dos dimensiones, para este problema se consideraron 20 instancias de problemas, las cuales fueron resueltas aplicando dos versiones de la heurística de selección de piezas de Djang y Finch (DJD): y , teniendo que la primera aplica un llenado inicial del 33% y la segunda del 25%. Dados los resultados de estas heurísticas se procedió a proponer una hiperheurística que selecciona cuál de las dos versiones es mejor para cada tipo de instancia.

En el contenido de ambas secciones se incluyen detalles de las heurísticas utilizadas, descripción de los algoritmos desarrollados y su complejidad, datos de las instancias con las que se experimentó y los resultados obtenidos.

**2. FFD para Bin Packing de una dimensión**

La heurística FFD es un algoritmo muy popular porque, aunque no siempre obtiene una solución óptima, tiene un buen comportamiento aún en el peor caso, estudios han probado que nunca usará más de contenedores, dónde es la cantidad mínima, u óptima, de contenedores necesarios para la instancia del problema. Originalmente esta cota superior estaba fijada en , pero se fue reduciendo con investigaciones posteriores [1] [2].

Los pasos que el algoritmo sigue son los siguientes:

1. Ordena descendentemente las piezas a empacar por su peso.
2. Abre un contenedor.
3. Busca la siguiente pieza más grande disponible.
   1. Si encuentra una pieza va al paso 4.
   2. Si terminó de recorrer las piezas y siguen existiendo piezas sin acomodar, regresa al paso 2.
   3. Si ya no existen más piezas pendientes de acomodar va al paso 5.
4. Intenta acomodar la pieza en el contenedor actual. De ser exitoso, la agrega al contenedor y la remueve de la lista de piezas disponibles. Regresa al paso 3.
5. Termina el algoritmo, regresa los contenedores utilizados con sus piezas contenidas.

Este algoritmo fue implementado en Java, a continuación se menciona la complejidad de cada uno de los pasos del algoritmo, excepto del último, ya que es sólo para presentación de los detalles del resultado:

|  |  |  |
| --- | --- | --- |
| **Paso** | **Descripción** | **Complejidad** |
| 1 | Ordenamiento de *n* piezas |  |
| 2 | Ciclo que recorre cada pieza |  |
| 2.1 | Ciclo que recorre cada contenedor existente. | En el peor de los se requerirá un contenedor por pieza. |
| 2.1.1 | Si la pieza cabe, agregarla. | Operaciones de tiempo constante: comparar y agregar. |
| 2.1.2 | De lo contrario, revisar si ya se acabaron las posibilidades de contenedores y no fue posible agregar la pieza a ninguno, de ser así, agregar nuevo contenedor. | Operaciones de tiempo constante: comparar y agregar. |

La complejidad total que se obtiene es entonces: y en términos generales Esta complejidad polinomial es mucho mejor que la se obtendría si se intentara probar todas las posibles combinaciones de piezas para su acomodo.

**2.1 Experimentos**

Primeramente se trabajó con 3 instancias de problemas, adicionalmente, a la instancia 1 se le hizo una modificación: se eliminó la pieza de peso 46, de este modo se generó una segunda versión de la instancia, esto se hizo con el propósito de probar un comportamiento inconsistente que se ha documentado que puede ocurrir con el algoritmo, este comportamiento es discutido más adelante. El origen de estas primeras instancias es de otros trabajos de investigación, no fueron generadas por nosotros.

Después de revisar el comportamiento obtenido con el cambio en la instancia 1, se buscó la forma de replicar ese comportamiento con una instancia de problema diferente, se logró identificar otra instancia que se comporta de la misma manera, la cual fue nombrada Instancia 4, y se obtiene el comportamiento inconsistente al remover la pieza de peso 38.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instancia 1** | | | | | |
| Capacidad Contenedor: 524 | | | | | |
| Cantidad de Piezas: 33 | | | | | |
| 37 | 10 | 106 | 85 | 252 | 252 |
| 252 | 10 | 37 | 10 | 46 | 127 |
| 127 | 252 | 252 | 10 | 12 | 12 |
| 442 | 9 | 252 | 106 | 10 | 106 |
| 252 | 127 | 106 | 84 | 127 | 9 |
| 12 | 10 | 127 |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instancia 2** | | | | | |
| Capacidad Contenedor: 1,000 | | | | | |
| Cantidad de Piezas: 60 | | | | | |
| 351 | 357 | 269 | 252 | 261 | 350 |
| 303 | 473 | 347 | 262 | 287 | 252 |
| 474 | 258 | 366 | 410 | 271 | 275 |
| 370 | 298 | 273 | 366 | 419 | 444 |
| 372 | 299 | 445 | 439 | 259 | 272 |
| 315 | 251 | 430 | 320 | 450 | 273 |
| 472 | 395 | 275 | 288 | 292 | 269 |
| 298 | 495 | 274 | 252 | 355 | 307 |
| 350 | 366 | 283 | 466 | 414 | 361 |
| 363 | 255 | 272 | 254 | 263 | 268 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instancia 3** | | | | | |
| Capacidad Contenedor: 10,000 | | | | | |
| Cantidad de Piezas: 57 | | | | | |
| 4812 | 4122 | 3326 | 2067 | 1492 | 468 |
| 4812 | 4122 | 3326 | 2067 | 1492 | 246 |
| 4812 | 3959 | 3168 | 1912 | 1308 | 246 |
| 4783 | 3787 | 3168 | 1897 | 1308 | 117 |
| 4778 | 3534 | 3168 | 1762 | 1274 | 117 |
| 4769 | 3534 | 3168 | 1762 | 1274 | 63 |
| 4769 | 3534 | 2649 | 1762 | 724 | 63 |
| 4738 | 3412 | 2317 | 1594 | 511 | 55 |
| 4199 | 3412 | 2317 | 1574 | 511 | 26 |
| 4199 | 3412 | 2156 |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instancia 4** | | | | | |
| Capacidad Contenedor: 1300 | | | | | |
| Cantidad de Piezas: 43 | | | | | |
| 665 | 561 | 12 | 12 | 12 | 150 |
| 500 | 500 | 280 | 10 | 10 | 212 |
| 500 | 500 | 280 | 10 | 10 | 9 |
| 500 | 500 | 280 | 10 | 10 | 37 |
| 500 | 280 | 280 | 222 | 9 | 38 |
| 243 | 211 | 200 | 200 | 37 | 158 |
| 189 | 162 | 150 | 150 | 150 | 154 |
| 197 |  |  |  |  |  |

Los experimentos consistieron en ingresar de forma individual cada una estas instancias al algoritmo implementado y registrar los resultados obtenidos, los datos que se obtuvieron fueron: cantidad de contenedores utilizados, piezas en cada uno de los contenedores y espacio no utilizado del contenedor.

El algoritmo implementado no tiene prerrequisitos de entrenamiento, siempre aplica el mismo método para resolver cada instancia. En la siguiente sección se comparan los resultados obtenidos para cada instancia con los resultados óptimos conocidos para cada una de ellas, también se menciona el comportamiento inconsistente obtenido en las instancias 3 y 4.

**2.2 Resultados**

En general los resultados obtenidos por FFD para las 3 instancias iniciales son muy buenos, el 66% de los resultados fueron óptimos, en 2 de las 3 instancias revisadas se obtuvo un resultado óptimo. Esto concuerda con los resultados de otros estudios realizados sobre esta heurística. La cuarta instancia no se contempla en esta tabla ya que fue generada de tal modo que el resultado fuera óptimo al resolverse con esta heurística en particular. La siguiente tabla muestra los resultados obtenidos para cada instancia y permite compararlos con el resultado óptimo conocido de las mismas:

|  |  |  |
| --- | --- | --- |
| **Instancia** | **Cantidad de contenedores FDD** | **Cantidad de contenedores óptima** |
| 1 | 7 | 7 |
| 2 | 23 | 20 |
| 3 | 15 | 15 |

Para las instancias 1 y 3, es claramente observable que la cota superior de la heurística se cumple, para la instancia 2 se puede aplicar la fórmula para corroborar que se cumpla:

El comportamiento inconsistente observado en la Instancia 1 y replicado en la Instancia 4, es que con las piezas originales se logra un empacado óptimo, con cero desperdicio de espacio, pero al *remover* uno de los elementos en particular, sucede que la cantidad de contenedores aumenta, dejando una gran cantidad proporcional de espacio no utilizado.

Este comportamiento es mencionado por [1], quien comenta que esto es una aparente discontinuidad en el desempeño de la heurística, la instancia mencionada en ese artículo es la Instancia 1 considerada en este reporte. Las únicas características que en ese artículo se identifican como factores para que este comportamiento ocurra son:

* FDD debe ser capaz de dar una solución sin desperdicios de espacio para la instancia.
* El elemento a eliminar no debe ser igual en peso que la suma de 2 o 3 elementos menores.

El detalle de los resultados arrojados por el algoritmo, para cada instancia, son mostrados en las siguientes tablas:

|  |  |  |
| --- | --- | --- |
| **Resultados Instancia 1** | | |
| Cantidad de Contenedores utilizados: 7 | | |
| **Contenedor** | **Piezas** | **Espacio no utilizado** |
| 1 | 442, 46, 12, 12, 12 | 0 |
| 2 | 252, 252, 10, 10 | 0 |
| 3 | 252, 252, 10, 10 | 0 |
| 4 | 252, 252, 10, 10 | 0 |
| 5 | 252, 127, 127, 9, 9 | 0 |
| 6 | 127, 127, 127, 106, 37 | 0 |
| 7 | 106, 106, 106, 85, 84, 37 | 0 |

|  |  |  |
| --- | --- | --- |
| **Resultados Instancia 1 (Removiendo pieza 46)** | | |
| Cantidad de Contenedores utilizados: 8 | | |
| **Contenedor** | **Piezas** | **Espacio no utilizado** |
| 1 | 442, 37, 37 | 8 |
| 2 | 252, 252, 12 | 8 |
| 3 | 252, 252, 12 | 8 |
| 4 | 252, 252, 12 | 8 |
| 5 | 252, 127, 127, 10 | 8 |
| 6 | 127, 127, 127, 106, 10, 10, 10 | 7 |
| 7 | 106, 106, 106, 85, 84, 10, 10, 9 | 8 |
| 8 | 9 | 515 |

|  |  |  |
| --- | --- | --- |
| **Resultados Instancia 2** | | |
| Cantidad de Contenedores utilizados: 23 | | |
| **Contenedor** | **Piezas** | **Espacio no utilizado** |
| 1 | 495, 474 | 31 |
| 2 | 473, 472 | 55 |
| 3 | 466, 450 | 84 |
| 4 | 445, 444 | 111 |
| 5 | 439, 430 | 131 |
| 6 | 419, 414 | 167 |
| 7 | 410, 395 | 195 |
| 8 | 372, 370, 258 | 0 |
| 9 | 366, 366, 268 | 0 |
| 10 | 366, 363, 271 | 0 |
| 11 | 361, 357, 275 | 7 |
| 12 | 355, 351, 292 | 2 |
| 13 | 350, 350, 299 | 1 |
| 14 | 347, 320, 315 | 18 |
| 15 | 307, 303, 298 | 92 |
| 16 | 298, 288, 287 | 127 |
| 17 | 283, 275, 274 | 168 |
| 18 | 273, 273, 272 | 182 |
| 19 | 272, 269, 269 | 190 |
| 20 | 263, 262, 261 | 214 |
| 21 | 259, 255, 254 | 232 |
| 22 | 252, 252, 252 | 244 |
| 23 | 251 | 749 |

|  |  |  |
| --- | --- | --- |
| **Resultados Instancia 3** | | |
| Cantidad de Contenedores utilizados: 15 | | |
| **Contenedor** | **Piezas** | **Espacio no utilizado** |
| 1 | 4812, 4812, 246, 117 | 13 |
| 2 | 4812, 4783, 246, 117, 26 | 16 |
| 3 | 4778, 4769, 63, 63, 55 | 272 |
| 4 | 4769, 4738, 468 | 25 |
| 5 | 4199, 4199, 1594 | 8 |
| 6 | 4122, 4122, 1574 | 182 |
| 7 | 3959, 3787, 2156 | 98 |
| 8 | 3534, 3534, 2649 | 283 |
| 9 | 3534, 3412, 2317, 724 | 13 |
| 10 | 3412, 3412, 3168 | 8 |
| 11 | 3326, 3326, 3168 | 180 |
| 12 | 3168, 3168, 2317, 1308 | 39 |
| 13 | 2067, 2067, 1912, 1897, 1762 | 295 |
| 14 | 1762, 1762, 1492, 1492, 1308, 1274, 511 | 399 |
| 15 | 1274, 511 | 8215 |

|  |  |  |
| --- | --- | --- |
| **Resultados Instancia 4** | | |
| Cantidad de Contenedores utilizados: 7 | | |
| **Contenedor** | **Piezas** | **Espacio no utilizado** |
| 1 | 665, 561, 38, 12, 12, 12 | 0 |
| 2 | 500, 500, 280, 10, 10 | 0 |
| 3 | 500, 500, 280, 10, 10 | 0 |
| 4 | 500, 500, 280, 10, 10 | 0 |
| 5 | 500, 280, 280, 222, 9, 9 | 0 |
| 6 | 243, 212, 211, 200, 200, 197, 37 | 0 |
| 7 | 189, 162, 158, 154, 150, 150, 150, 150, 37 | 0 |

|  |  |  |
| --- | --- | --- |
| **Resultados Instancia 4 (Removiendo pieza 38)** | | |
| Cantidad de Contenedores utilizados: 8 | | |
| **Contenedor** | **Piezas** | **Espacio no utilizado** |
| 1 | 665, 561, 37, 37 | 0 |
| 2 | 500, 500, 280, 12 | 8 |
| 3 | 500, 500, 280, 12 | 8 |
| 4 | 500, 500, 280, 12 | 8 |
| 5 | 500, 280, 280, 222, 10 | 8 |
| 6 | 243, 212, 211, 200, 200, 197, 10, 10, 10 | 7 |
| 7 | 243, 212, 211, 200, 200, 197, 10, 10, 10 | 8 |
| 8 | 9 | 1291 |