ISSN 1870-4069

Implementación del algoritmo MBS para resolver instancias de bin packing

Luis Alberto Silva Escamilla, Hilda Castillo Zacatelco, Rafael de la Rosa Flores

Benemérita Universidad Autónoma de Puebla, Puebla, México

siel_alb@me.com, hildacz@gmail.com, rafael@cs.buap.mx

Resumen. El Minimum Bin Slack es una heurística propuesta por Gupta y Ho [6]. Dicho algoritmo es definido por sus autores como un algoritmo orientado al contenedor. El MBS parte de la idea de que minimizar el número de contendores a utilizar es equivalente a reducir el espacio sobrante de los mismos: es decir, a menor cantidad de contenedores, menor espacio sobrante. Considerando la idea anterior, este algoritmo utiliza el concepto de búsqueda lexicográfica para ir llenando cada contenedor buscando optimizar el espacio del mismo y una vez lleno, proceder a llenar el siguiente contenedor, sin perder nunca el objetivo de reducir el espacio sobrante en los mismos. En este trabajo, se presenta una propuesta para la implementación de este algoritmo y se realiza una comparación de los resultados obtenidos con él contra los resultados generados por el algoritmo FFD. Finalmente se dan a conocer los resultados encontrados.

Palabras clave: Bin packing problem, herurística, minimum bin slack.

Implementation of the MBS Algorithm to Solve Instances of the Bin Packing Problem

Abstract. The Minimum Bin Slack is a heuristic proposed by Gupta and Ho [6]. Such algorithm is defined by its authors as an algorithm focused on the container. MBS starts from the idea that minimizing the number of containers is equivalent to reduce the left space in the containers: thus, less quantity of containers means less left space in the containers. Considering this idea, MBS algorithm uses the concept of lexicographic search in order to fill each container, searching to optimize space in container and once it is filled, proceed to fill the next one, without forgetting the goal of reducing the space behind. In this article, a proposal is presented to implement this algorithm and the results obtained from it are compared to results obtained from FFD. Finally, the results of this comparison are presented.

Keywords: Bin packing problem, heuristic, minimum bin slack.

1. Introducción

El *bin packing problem* (BPP) [4] es un problema NP-combinatoria [6] (es decir, NP y NP-hard) de optimización combinatoria en el cual dado un conjunto de ítems de tamaño variable, se busca acomodarlos dentro de contenedores de tamaño fijo, buscando optimizar el número de contenedores a utilizar, es decir, usando el menor número de contendores para colocar el mayor número de ítems posible. Para este problema existen tres variantes tales como el *bin packing* unidimensional (1D-BPP), de 2 dimensiones (2D-BPP) [9] o de 3 dimensiones (3D-BPP) [5]; en dichas variantes [3] solo cambia el número de dimensiones tanto de los ítems como de los contenedores no obstante, en el presente trabajo nos limitaremos al bin packing de una dimensión.

Aunque existe una variedad de trabajos relacionados a la búsqueda de la mejor solución a este problema, uno de los trabajos más conocidos y del cual parten muchas otras propuestas de soluciones es el *Minimum bin slack* presentado por Gupta y Ho en 1999. En dicho trabajo, se muestra una clara mejora respecto a otros algoritmos conocidos, tales como el *first fit decreasing* [10] o el *next fit decreasing* [2] sin embargo, las instancias presentadas por los autores, no son instancias reconocidas en la literatura, por lo cual no se puede tener la certeza del desempeño de dicho algoritmo con otros conjuntos de datos creados explícitamente para este problema.

Considerando lo anterior, el trabajo desarrollado a continuación presenta una implementación del algoritmo de MBS buscando si dicho algoritmo es igual de eficiente para conjuntos de datos particularmente diseñados para el *bin packing problem*. De esta forma, el documento se divide como sigue: en la sección 2 se presenta la idea básica sobre el funcionamiento del algoritmo, los dos conceptos fundamentales que son la base de dicho algoritmo y se describe de forma breve el funcionamiento del mismo. En la sección 3 se presentan las instancias utilizadas para probar el desempeño de esta implementación, de igual forma, se menciona de forma breve el algoritmo con el que se comparó MBS y el cual busca mejorar. Finalmente en las secciones 4 y 5 se muestran los resultados y las conclusiones respectivamente.

Respecto a la implementación, es importante mencionar que aunque los autores nos muestran una sugerencia para la implementación del mismo, hay ciertas funciones que no detallan, por lo cual la implementación y por ende los resultados pueden variar, ya que dependen de la interpretación que da el lector.

Durante las pruebas que se realizaron, se pudo observar que existen ciertos casos, los cuales los autores no mencionan que pueden afectar la ejecución del algoritmo, en particular aquellos casos en donde se repiten elementos, sin embargo, este aspecto se discute en los resultados.

2. Minimum Bin Slack

El algoritmo MBS (*Minimum Bin Slack*) es definido por sus autores como un algoritmo enfocado al contenedor [3]. Dicho algoritmo parte de la idea de que minimizar el número de contendores a utilizar es equivalente a reducir el espacio sobrante de los mismos: es decir, a menor cantidad de contenedores, menor espacio

sobrante. Considerando lo anterior, este algoritmo a diferencia del FFD o BFD [8] (los cuales se enfocan en colocar elementos en contenedores priorizando la velocidad y por ende, las soluciones ofrecidas por estos requieren más contenedores); se enfoca realmente en optimizar el espacio dentro del contenedor. El MBS utiliza el concepto de búsqueda lexicográfica [6] para ir llenando cada contenedor buscando optimizar el espacio del mismo y una vez lleno, proceder a llenar el siguiente contenedor, sin perder nunca el objetivo de reducir el espacio sobrante en los mismos.

Una vez entendiendo la idea anterior, existen 2 puntos que se consideran claves para el funcionamiento del algoritmo.

Para empezar, el concepto de backtracking. Recordemos que esta estrategia consiste en probar varias combinaciones de elementos hasta hallar una solución, sin embargo, cuando esta posible solución no cumple con una restricción previamente definida, se regresa a la combinación anterior y se exploran otras soluciones. El MBS utiliza esta estrategia de búsqueda para hallar y eliminar elementos en el contenedor.

El segundo punto es el orden. Considerando la estrategia anterior se intuye que ordenando los elementos el tiempo de búsqueda se reduce, por lo cual se asume que para el correcto funcionamiento del algoritmo los elementos están ordenados de forma descendente.

A continuación se describe una serie de pasos que detallan de manera breve el funcionamiento del algoritmo:

1. Se toma el primer elemento de la instancia y se agrega al contenedor.
2. Si la suma de los elementos contenidos hasta ese momento en el contenedor es exactamente igual al valor del contenedor, se termina la ejecución del algoritmo; en caso contrario se continúa con la ejecución.
3. Se agrega el siguiente elemento de la instancia al contenedor, en caso de que el elemento no pueda ingresarse debido a que no existe el espacio mínimo para ubicarse, se procede a buscar de entre los elementos contenidos en el contenedor, un elemento cuyo valor sea igual o mayor al valor del nuevo elemento que se pretende ingresar y se le reemplaza. Sin embargo, para esta búsqueda, se procederá por iniciar a partir del último elemento ingresado, es decir, en la cola de la lista. Se va a paso 4. Caso contrario regresamos a paso 2.
4. Mientras que la instancia no este vacía es decir, no se llegue al final de la lista o el contenedor quede exactamente lleno, se realiza el paso 2.
5. Al finalizar la ejecución del algoritmo debido a que el contenedor se llenó exactamente o se llegó al último elemento de la instancia, se procede a quitar los elementos que se agregaron al contenedor, de la instancia y se vuele a ejecutar el algoritmo ahora con los elementos sobrantes de la instancia. Todo esto hasta que ya no queden elementos en la misma.

La implementación de este algoritmo se puede realizar tanto recursiva como iterativamente [7], lo único importante a considerar es que la forma en que se acomodan los elementos en los contenedores ha de realizarse utilizando el concepto de búsqueda lexicográfica; y que esta búsqueda se repite hasta ya no existan más elementos en la instancia.

3. Instancias

Uno de los objetivos de este trabajo, es ver la eficiencia del MBS en conjuntos de datos que se crearon explícitamente para el BPP. De esta manera para la realización de las pruebas, se eligieron los siguientes conjuntos de datos: 53NIRUP, bin1data, bin2data y hard28. Se eligieron estos conjuntos de datos ya que son los que otros autores emplean en sus pruebas (disponibles en la literatura), de esta manera, se estará busca medir nuestra implementación contra benchmarks perfectamente conocidos e igualmente, se evita el sobreajuste al no utilizar conjuntos de datos propios.

Dichos conjuntos tienen instancias que son particularmente difíciles de resolver ni siquiera por métodos de reducción. Finalmente, tales conjuntos son también difíciles de resolver para el FFD.

El First Fit Decreasing (FFD) funciona como sigue: cada ítem es colocado dentro del contenedor siempre que haya espacio, en caso contrario, el ítem es colocado dentro del siguiente contenedor con espacio disponible [1]. Si hay espacio en un contenedor anterior, el ítem se coloca dentro, de este modo se elimina la restricción que tiene el NF, es decir, siempre se considerarán todos los contenedores siempre que exista espacio. Se eligió el FFD como algoritmo de comparación debido a que a pesar de que existen otros algoritmos derivados de éste, tales como los son el NFD o el BFD (y sus respectivas variantes), el FFD aún muestra un mejor desempeño por si mismo respecto a tiempo y cantidad de contenedores utilizados.

4. Resultados

El programa se corrió sobre una computadora portátil con procesador Intel Core i3 a 1.40 GHz con el sistema operativo Windows 7 de 64 bits. El algoritmo se implementó en lenguaje C++ debido principalmente a que su paradigma permite enfocarse en la solución del problema. Otra razón importante es que C++ tiene un buen manejo de los recursos computacionales, esto se ve reflejado a la hora de ejecutar el programa, donde los tiempos son relativamente cortos. El algoritmo fue puesto a prueba con las siguientes instancias: 53NIRUP, bin1data (Tabla 3, Figura 3), bin2data (Tabla 4. Figura 4) y hard28.

Nuestra implementación emplea arreglos de la clase vector para representar los contenedores, esto nos permitió enfocarnos en la implementación y no en la gestión de recursos, no obstante el mayor reto durante se dio a la hora de interpretar el algoritmo y codificarlo ya que de la forma de implementar podría haber modificado la eficiencia del algoritmo. Se encontró particularmente difícil interpretar la función de búsqueda lexicográfica y aunque como veremos más adelante los resultados obtenidos fueron aceptables, una mejor interpretación podría mejorar al menos ligeramente el desempeño del programa.

Como se puede observar, el algoritmo da buenos resultados al compararlo con el FFD y el BFD y cumple con la tesis propuesta por los autores, ya que el número de contendores disminuye al optimizar el espacio en cada uno [6]. Adicionalmente, con la implementación que se realizó del algoritmo, los tiempos también fueron mejores.

Por el contrario, los resultados varían para cada conjunto de instancias. Por ejemplo, MBS presenta un desempeño similar en cuanto al número de contenedores utilizados para las instancias del conjunto 53NIRUP y solo mostrando una ligera mejoría en cuanto al tiempo como se muestra en la Tabla 1 y Figura 1.

Tabla 1. Instancias de 53NIRUP. Se muestra el tiempo que hizo cada instancia utilizando el MBS y el FFD. Tambien se muestra el número de contenedores para cada instancia.

MBS			FFD	
Instancia	Tiempo	# Contenedores	Tiempo	# Contenedores
BPP02015007_0918.BPP	0.001	10	0.001	9
BPP04000107_0697.BPP	0.005	16	0	15
BPP04005007_0612.BPP	0.003	16	0	15
BPP04005009_0361.BPP	0.006	20	0.001	20
BPP18015010_0605.BPP	0.105	105	0.004	104
BPP20000108_0025.BPP	0.086	77	0.004	76
BPP20000108_0175.BPP	0.069	84	0.004	83
BPP08000107_0624.BPP	0.019	28	0.001	27
BPP06015009_0403.BPP	0.013	31	0.001	30
BPP06005008_0755.BPP	0.009	24	0.001	23

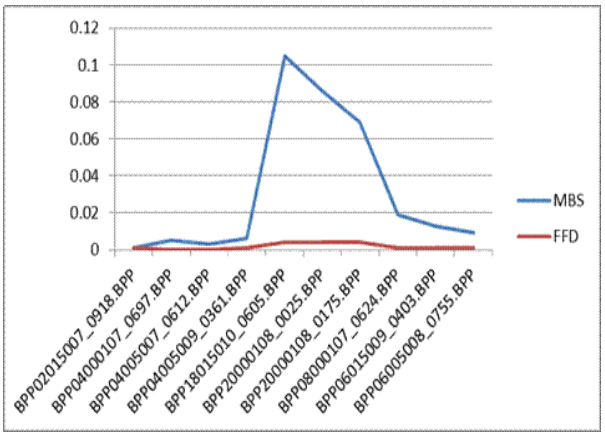

Fig. 1. Tiempos para la instancia 53nirup.

Por otra parte, para las instancias del conjunto de hard28 incluso llego a mostrar un peor desempeño respecto a la cantidad de contenedores utilizados como se observa en la tabla y figura 2, aunque como se puede apreciar, solo de manera sutil.

Esta variación en el desempeño puede tener dos causas principales. Por una parte, cada uno de los conjuntos presenta ciertas características como por ejemplo, el número de elementos repetidos en cada instancia. Igualmente, se podría considerar relevante el

número de dígitos de cada elemento de la instancia, ya que originalmente algunas de las instancias utilizadas por los autores para sus pruebas, se limitaba a utilizar conjuntos con datos de 1 a 100. Por otra parte, algunos de los conjuntos utilizados para nuestra prueba, utilizan una distribución de 1 a 1000

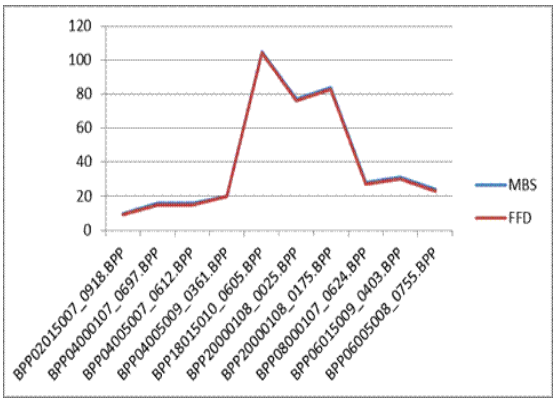

Fig. 2. Gráfica comparativa de contenedores utilizados para instancia 53 nirup.

Tabla 2. Instancias de 28hard. Se muestra el tiempo que hizo cada instancia utilizando el MBS y el FFD. También se muestra el número de contenedores para cada instancia.

MBS			FFD	
Instancia	Tiempo	# Contenedores	Tiempo	# Contenedores
BPP 13.BPP	0.065	68	0.003	67
BPP 40.BPP	0.048	60	0.002	59
BPP 60.BPP	0.065	64	0.002	63
BPP 175.BPP	0.068	84	0.004	83
BPP 359.BPP	0.073	76	0.004	76
BPP 781.BPP	0.076	72	0.004	71
BPP 814.BPP	0.08	82	0.003	81
BPP 419.BPP	0.084	81	0.004	80
BPP 645.BPP	0.055	59	0.002	58
BPP 531.BPP	0.081	84	0.004	83

Para la segunda causa se debe considerar a la siguiente premisa señalada por los autores: “el algoritmo propuesto es óptimo si la suma del requerimiento de los ítems, es menor o igual al doble de la capacidad del contenedor” [6]. Considerando que los conjuntos de instancias utilizadas para las pruebas, fueron diseñados explícitamente para “romper” cualquier propuesta de solución, no sería extraño que alguno de esos conjuntos tuviera alguna característica que haga que la premisa anteriormente mencionada no se cumpla.

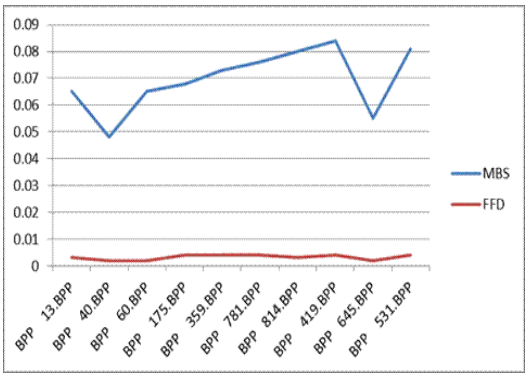

Fig. 3. Tiempos para la instancia *hard28.*

Tabla 3. Instancias de bin1data. Se muestra el tiempo que hizo cada instancia utilizando el MBS y el FFD. También se muestra el número de contenedores para cada instancia.

MBS			FFD	
Instancia	Tiempo	# Contenedores	Tiempo	# Contenedores
N1C1W1_A.BPP	0.007	25	0.004	25
N1C1W1_B.BPP	0.009	31	0.004	31
N1C1W1_C.BPP	0.006	20	0.001	21
N1C1W1_D.BPP	0.006	28	0.003	28
N1C1W1_M.BPP	0.006	30	0.003	30
N1C1W1_N.BPP	0.007	25	0.0004	25
N1C1W2_O.BPP	0.006	29	0.004	29
N3C1W2_F.BPP	0.128	123	0.001	123
N4C2W4_R.BPP	0.576	300	4.041	300
N4C3W1_I.BPP	0.222	168	0.2	168

Tabla 4. Instancias de bin2data. Se muestra el tiempo que hizo cada instancia utilizando el MBS y el FFD. Tambien se muestra el número de contenedores para cada instancia.

MBS			FFD	
Instancia	Tiempo	# Contenedores	Tiempo	# Contenedores
N1W1B1R0.BPP	0.009	20	0.004	18
N1W1B1R1.BPP	0.009	19	0.004	18
N1W1B1R2.BPP	0.009	21	0.003	19
N1W1B2R0.BPP	0.009	18	0.003	17
N1W1B2R3.BPP	0.009	17	0.003	16
N1W2B1R9.BPP	0.008	11	0.001	11

N4W4B3R9.BPP	0.123	56	0.17	56
N4W4B3R8.BPP	0.132	57	0.155	57
N4W4B3R7.BPP	0.011	57	0.13	57
N2W1B1R6.BPP	0.033	35	0.022	34

Finalmente, existen conjuntos de instancias que manejan números no enteros sin embargo, estos conjuntos no se contemplaron para la implementación. No obstante, su implementación es posible, siempre y cuando se considere la naturaleza de estos conjuntos de números en las estructuras de datos a utilizar.

5. Conclusiones

El algoritmo MBS mostró su mejor desempeño para el conjunto de instancias 53NIRUP donde se redujo ligeramente el tiempo de solución y el número de contenedores utilizados. Sin embargo, tales resultados se aprecian en un número reducido de instancias que podría aumentar al efectuar más pruebas con más instancias del mismo conjunto. Por otra parte, para el resto de los conjuntos de datos donde el desempeño en el mejor de los casos se mostró igual al compararlo con el FFD, se buscara mejorar el desempeño implementando el MBS' [7] y observando si esta implementación puede mejorar respecto al MBS y al FFD.

Como se puede observar, el algoritmo da buenos resultados al compararlo con el FFD y cumple con la tesis propuesta por los Gupta y Ho [6] sobre el MBS, ya que el número de contendores disminuye al optimizar el espacio en cada uno. Adicionalmente con la implementación que se realizó del algoritmo, los tiempos también fueron mejores.

Referencias

1. Coffman Jr., E. G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: a survey. Approximation algorithms for NP-hard problems, pp. 46–93 (1996)
2. Garey, M.R., Johnson, D.S.: Approximation Algorithms for Bin Packing Problems: A Survey. In: Ausiello G., Lucertini M. (eds) Analysis and Design of Algorithms in Combinatorial Optimization. International Centre for Mechanical Sciences (Courses and Lectures), Vol 266, Springer (1981)
3. Bansal, N., Correa, J. R., Kenyon, C., Sviridenko, M.: Bin Packing in Multiple Dimensions: Inapproximability Results and Approximation Schemes. Mathematics of Operations Research, Vol. 31, No. 1, pp. 31–49 (2006)
4. Falkenauer, E., Delchambre, A.: A Genetic Algorithm for Bin Packing and Line Balancing. In: Proceedings IEEE International Conference on Robotics and Automation (1992)
5. Bortfeldt, A.: A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research 172, pp. 814–837 (2006)
6. Gupta, J. N.D., Ho, J.C.: A new heuristic algorithm for one-dimensional bin-packing problem. Production Planning & Control 10, pp. 598–603 (1999)

7. Flesznar, K., Hindi, K. S.: New heuristics for one dimensional bin-packing. Computers & Operations Research 29, pp. 821–839 (2002)
8. Maiza, M., Radjef, M.S.: Heuristics for Solving the Bin-Packing Problem with Conflicts. Applied Mathematical Sciences, Vol. 5(35), pp. 1739–1752 (2011)
9. Lodi, A., Martello, S., Vigo, D.: Heuristic algorithms for the three-dimensional bin packing problem. European Journal of Operational Research 141, pp. 410–420 (2002)
10. Dósa, G.: The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is $FFD(I) \leq 11/9OPT(I) + 6/9$. In: Chen, B., Paterson, M., Zhang, G. (eds), Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, Lecture Notes in Computer Science, Vol. 4614, pp. 1–11 (2007)