Una búsqueda tabú para el *Bin Packing Problem*

Mª del Carmen Delgado, Pablo Cortés, Alejandro Escudero, Jesús Muñuzuri

Grupo Ingeniería de Organización. Escuela Superior de Ingenieros. Universidad de Sevilla. Avd. de los Descubrimientos s/n, 41092 Sevilla. m_delgado@us.es, pca@esi.us.es, aescudero@esi.us.es, munuzuri@esi.us.es

Resumen

Se presenta el estudio de una implementación de una Búsqueda Tabú para el problema de empaquetamiento en cajas (Bin Packing Problem) de una dimensión. La valoración de las numerosas posibilidades y formas para adaptar el algoritmo genérico al problema tratado, así como la influencia del contexto en la solución final muestran los numerosos parámetros que han de tratarse en el problema.

Palabras clave: bin packing problem, empaquetamiento en cajas, búsqueda tabú

1. Introducción

El denominado problema de empaquetamiento en cajas (*Bin Packing Problem*, BPP) consiste en empaquetar n ítems indivisibles, de pesos w_i concretos, en el menor número posible de cajas de capacidad finita c. Existen numerosas variantes de este problema, según se quiera minimizar la capacidad de un número concreto de cajas, se considere el problema en dos o tres dimensiones, etc. No obstante, muchos de los problemas de optimización reales relativos al área de corte y empaquetamiento necesitan de la asignación de ítems a otras unidades mayores (cajas) sin que exista solapamiento y minimizando el número de cajas requeridas. Situación que es adecuadamente modelada por el BPP.

El BPP tiene numerosas aplicaciones en la vida real pues permite modelar, entre otras situaciones, la carga de camiones con limitaciones de peso, situaciones de almacenamiento, el diseño de páginas de periódicos, la distribución de los anuncios de televisión en las desconexiones o paradas publicitarias, la asignación de canales en redes de comunicación celulares, la asignación de trabajos de producción, o el diseño de circuitos VLSI entre otros.

2. Planteamiento del problema

A pesar de la aparente sencillez de su enunciado, el BPP es un problema combinatorio NP-duro. El objeto de estudio de este trabajo corresponde a la versión unidimensional del mismo que busca minimizar el número de cajas empleadas. Formalmente, se trata de encontrar una partición del conjunto de ítems en el mínimo número de subconjuntos, de tal manera que se cumpla que la suma de los pesos de los ítems en cada conjunto sea menor que la capacidad (Monaci y Toth, 2006).

El modelo correspondiente a la situación descrita se presenta en las ecuaciones (1) a (6) . Las variables y parámetros empleados en el modelo aparecen en la Tabla 1:

$$\min \sum_{k} y_k \quad \forall k \in B \tag{1}$$

$$\text{s.a.} \quad \sum_{l \in I} w_l x_{kl} \leq c \quad \forall k \in B \tag{2}$$

$$\sum_{k \in B} x_{kl} = 1 \qquad \forall l \in I \tag{3}$$

$$y_k \geq x_{kl} \qquad \forall l \in I \tag{4}$$

$$x_{kl} \in \{0,1\} \quad \forall l \in I, k \in B \tag{5}$$

$$y_k \in \{0,1\} \quad \forall k \in B \tag{6}$$

Tabla 1: Variables, datos y parámetros del modelo.

I : conjunto de n ítems	x_{kl} : variable binaria que indica si el ítem l se asigna a la caja k.
B: conjunto de cajas disponibles	y_k : variable binaria que indica si la caja k es utilizada.
c: capacidad tipo de cada caja k	$w_l \in \mathbb{Z} \ \ \forall l$ Condición de pesos enteros
w_l : peso de cada ítem $l \in$ I	$w_l \leq c \ \ \forall l$ Condición de cabida de los ítems en las cajas

Este problema presenta la ventaja frente a otros problemas NP-duros, de que puede resolverse en un tiempo polinomial, gracias a algoritmos de aproximación. Su estudio es interesante, no sólo por el elevado número de aplicaciones que tiene directamente por sí mismo, sino porque aparece además como parte de otros problemas mayores.

3. Algoritmo de Búsqueda Tabú

Para la resolución de BPP se han desarrollado, tradicionalmente, dos tipos de estrategias: unas basadas en métodos de aproximación a la solución, basados en heurísticas o metaheurísticas, y otras basadas en métodos exactos (Scholl et al, 1997).

En este caso, se propone para su resolución una Búsqueda Tabú. El algoritmo de Búsqueda Tabú se apoya explícitamente en la historia de la búsqueda, tanto para escapar de óptimos locales, como para explorar el espacio de soluciones (Blum y Roli, 2003).

La obtención de una solución inicial al problema, a partir de la cual se aplica el algoritmo de Búsqueda Tabú puede realizarse mediante distintas heurísticas, tales como las estrategias de peor ajuste, primer ajuste o mejor ajuste, entre otras. Dos de las heurísticas más rápidas para la obtención de una solución aproximada al BPP son FFD (*First Fit Decreasing*) y BFD (*Best Fit Decreasing*) (Alvim et al., 2004). En este caso concreto se emplea el método de aproximación FFD para obtener una primera solución admisible.

La característica más destacable de la Búsqueda Tabú es el uso de memoria adaptativa y de estrategias especiales de resolución de problemas. La Búsqueda Tabú guía el proceso de búsqueda local más allá de los posibles óptimos locales. Para ello, cada vez que se ejecuta un movimiento, se imponen restricciones o condiciones tabú sobre estos. Estas condiciones impiden la repetición de movimientos recientes y se imponen para determinados períodos de tiempo, período de la lista tabú (Emmert et al., 2003).

Esta metaheurística ha sido muy exitosa en la resolución de problemas de optimización duros, especialmente aquellos que surgen en aplicaciones del mundo real (Glover y Melián, 2003). De ahí, la elección de la misma para el estudio del problema planteado.

3.1. Particularización del algoritmo

La estructura concreta del algoritmo puede apreciarse en la Figura 1. El programa desarrollado considera numerosos grados de libertad que permiten la adaptación del mismo al ejemplo concreto que se esté tratando con el objetivo de conseguir unos mejores resultados. Estos grados de libertad, que a continuación se comentan, hacen referencia a aspectos diversos de la propia Búsqueda Tabú, así como las heurísticas en las que se apoya o la propia concreción del problema entre manos.

```
Sc=Genera_Sol_Inicial_PFD
Inicializa_Lista_Tabu

while(!limite_inferior && k<max_iter)

      if(k=periodo_sin_cambios)
            Vaciado_Nivels_Cajas     %Diversificación
            Sc=Reasignacion             %Intensa
      else
            W=Vecindario_Admisible   %Intercambio+Desplazamiento
            We=Elige_Mejor(W)              +VaciadoCaja

            if(Verifica_Aspiracion)
                  Sc=We
                  Actualiza_Lista_Tabu
                  Actualiza_Criterio_Aspiracion
            elseif(We_Mejora_Sc)
                  Comprueba_Condicion_Tabu
                  if(!tabu)
                        Sc=We
                        Actualiza_Lista_Tabu
                        Actualiza_Criterio_Aspiracion
                  end
            elseif(We_No_Mejora_Sc)
                  Acepta_con_Probabilidady
                  if(aceptado)
                        Actualiza_Lista_Tabu
                        Actualiza_Criterio_Aspiracion
                  end
            end
      end
end
```

Figura 1: Pseudocódigo del algoritmo desarrollado.

3.1.1 Caracterización de la Búsqueda Tabú

Para la implementación de un algoritmo de Búsqueda Tabú es de especial importancia la definición de la estructura de vecindario en el espacio de soluciones y de la lista tabú que se va a emplear.

Para el BPP tratado, se considera como vecindad de una solución a aquellas que pueden obtenerse a partir de dicha solución mediante la realización de movimientos de desplazamiento de ítems de unas cajas a otras y de movimientos de intercambio de ítems entre cajas. La contribución de estos dos movimientos a la creación del vecindario de una solución es controlable. No obstante, con el objetivo de facilitar la aparición de vecinos mejores (más cercanos al óptimo) en el entorno de una solución, se emplean también heurísticas para vaciado de alguna de las cajas.

La aceptación de una nueva solución como solución actual implica la actualización de la lista tabú. La lista tabú definida en este caso es estática y con un período tabú igual a la raíz cuadrada del número de ítems (Glover y Melián, 2003), si bien podrían haberse empleado otras posibilidades, como la contemplada en (Scholl et al, 1997). En el trabajo desarrollado, las soluciones son guardadas íntegramente, sin necesidad de ser caracterizadas mediante atributos.

Por último, para evitar caer en óptimos locales, existen dos procesos de diversificación que, con cierta probabilidad, o forzosamente, permiten la aceptación de soluciones peores que la actual. El período de diversificación forzosa puede adaptarse a la situación concreta considerada.

3.1.2 Caracterización de las heurísticas de apoyo

Para mejorar el rendimiento del algoritmo desarrollado, en la creación de la vecindad pueden emplearse distintas heurísticas, además de las estrategias de intercambio y desplazamiento de ítems ya nombradas. Estas heurísticas tienen como objetivo identificar, según distintos criterios aquellas cajas que pueden vaciarse en la solución actual. Estas cajas, que cumplen criterios tales como tener un número de ítems mínimo o máximo, o bien, tener mínimo peso, se vacían repartiendo sus ítems entre las demás. De esta forma, la solución vecina así obtenida tiene un número de cajas que se aproxima más al óptimo del problema.

El método empleado para la diversificación forzosa pretende obtener una solución alejada del óptimo, a partir de la cual reiniciar el proceso de búsqueda. Para ello, se toma la solución actual y se vacían las cajas, como mínimo, un cierto nivel determinado. La estrategia de asignación de los ítems para la creación de la nueva solución puede apoyarse, a su vez, en estrategias de mejor o peor ajuste. La calidad de la solución obtenida también es controlable según la forma de asignación de los ítems a las cajas y el nivel de vaciado que se elija.

Así pues, los aspectos previamente descritos, muestran la adaptabilidad del código desarrollado a la concreción del problema BPP que se esté manejando. De este modo, y tras la adecuada caracterización y comprensión del método y del problema entre manos, se facilita la consecución de resultados adecuados en tiempos razonables.

4. Resultados

La codificación del algoritmo ha sido realizada en MATLAB y para las pruebas se emplearon los ficheros disponibles en la librería OR-Library (Beasley, 2007).

Si bien esta implementación no alcanza el óptimo de los distintos representantes del BPP

probados, sí que se obtienen buenas aproximaciones al mismo.

La Figura 2 muestra, a modo de ejemplo, la aproximación al óptimo del problema para dos de las distintas heurísticas de generación del vecindario que pueden existir, según las combinaciones que de ellas se hagan. Por un lado se representa el resultado empleando vaciado de la caja con menor número de ítems y asignación de mejor ajuste, y por otro, vaciado de la de mayor número de ítems y asignación de peor ajuste. Puede observarse cómo las soluciones obtenidas se encuentran en torno a los mismos valores en ambos casos.

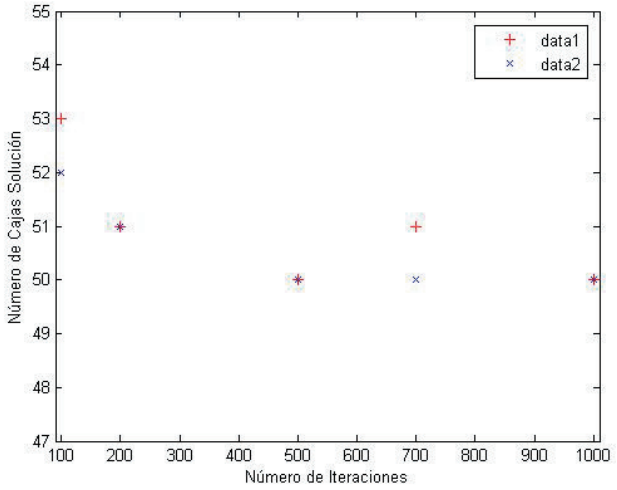

Figura 2: Aproximaciones al óptimo de distintos métodos: Mayor número de ítems y peor ajuste (azul); Menor número de ítems y mejor ajuste (rojo). Problema de empaquetamiento de 120 ítems en cajas de capacidad 150.

La Tabla 2 recoge, para los mismos casos anteriores, el número de cajas de la solución obtenida para distinto número de iteraciones. Observándola, se puede apreciar cómo en el segundo caso, la aproximación a la mejor solución obtenida es más rápida que en el primero.

Tabla 2: Soluciones obtenidas frente al número de iteraciones permitidas para el problema de empaquetamiento en cajas de 120 ítems en cajas de capacidad 150.

Número de Iteraciones	Solución obtenida con heurística de menor nº ítems y mejor ajuste	Solución obtenida con heurística de mayor nº ítems y peor ajuste
50	55	52
100	53	52
200	51	51
350	51	50
500	50	50
700	51	50
1000	50	50

5. Conclusiones

A partir de los experimentos realizados se extraen una serie de conclusiones referidas a la implementación y resolución del problema.

Mediante la experimentación, se comprobó que la generación de la solución inicial mediante el algoritmo FFD hacía que se alcanzase un valor muy cercano al óptimo en las primeras fases del problema. Esto dificultaba la apreciación del funcionamiento del algoritmo de búsqueda tabú, cuyo rendimiento se hace especialmente patente tras las fases de diversificación forzosa.

En general, los resultados obtenidos y el rendimiento de la implementación realizada empeoran conforme el tamaño del problema (número de ítems) crece, necesitándose un mayor número de iteraciones para alcanzar resultados aceptables.

Es destacable también la influencia que la concreción del problema tratado tiene sobre los resultados obtenidos. A través del nivel de vaciado que se establece en los procesos de diversificación, se ve la dependencia de la calidad de la solución empeorada que se quiere conseguir con el tamaño de los ítems considerados.

Por último, reseñar que la mayor dificultad a la hora de estudiar la caracterización y resolución del problema nace de la propia definición del mismo. Es decir, de su función objetivo, que permite que numerosas soluciones distintas resulten en un mismo valor de la misma, por lo que no resulta inmediata la identificación del camino hacia el óptimo.

Referencias

Alvim, A.; Ribeiro, C.; Glover, F.; Aloise, D. J. (2004). A Hybrid Improvement Heuristic for the One-Dimensional Bin Packing Problem. Journal of Heuristics, No. 10, pp. 205-229.

Beasley, J.E.; (2007) http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html

Blum, C.; Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys, Vol. 35, No. 3, pp. 268-308.

Emmert, J.M.; Lodha, S.; Bhatia, D.K. (2003). On Using Tabu Search for Design Automation of VLSI Systems. Journal of Heuristics, No. 9, pp. 75-90.

Glover, F.; Melián, M. (2003). Búsqueda Tabú. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, No. 19, pp. 29-48.

Monaci, M.; Toth, P. (2006). A Set-Covering-Based Heuristic Approach for Bin-Packing Problems. INFORMS Journal on Computing, Vol. 18, No. 1, pp. 71-85.

Scholl, A.; Klein, R.; Jürgens, C. (1997). BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Computers and Operations Research, Vol. 24, No. 7, pp. 627-645.