Problema Bin Packing com Abordagem Heurística First Fit Decreasing

Daniel Scheidemantel Camargo
Eduardo José de Borba

[1]Programa de Pós Graduação em Computação Aplicada – PPGCA/DCC
Universidade do Estado de Santa Catarina - UDESC
Zona Industrial Norte, Joinville - SC

daniel@colmeia.udesc.br , eduardojoseborba@gmail.com

Abstract. *O presente trabalho visa mostrar uma implementação heurística do problema Bin Packing (empacotamento), cujo propósito é descobrir o menor número de bins (caixas) necessários para empacotar um dado conjunto de itens. A menos que P = NP, esse problema não possui uma solução determinística em tempo polinomial que forneça uma solução ótima. Nesse trabalho utiliza-se a solução heurística First Fit Decreasing (FFD) para encontrar uma solução aproximada em tempo polinomial. É mostrada também uma função de limite inferior para indicar o valor ótimo, e analisar o FFD pelo critério de otimalidade. Como resultados, os testes mostram que o algoritmos desenvolvido possui boa otimalidade, encontrando o valor ótimo para uma pequena instancia de itens.*

1. Introdução

Em Ciência da Computação, um problema é chamado de NP quando existe um algoritmo não-determinístico que o resolva em tempo polinomial ou é possível verificar a resposta do problema em tempo polinomial [Cook 1971]. Um problema de decisão X é chamado de NP-completo se faz parte de NP e que existe um algoritmo polinomial para reduzir outro problema NP-completo (por exemplo: caixeiro viajante, 3-CNF-SAT, etc.) para uma instância de X. Já os problemas P são aqueles para os quais se conhece uma solução determinística que o resolva em tempo polinomial.

Considerando que $P \neq NP$ as soluções para problemas NP-completos em tempo satisfatório permanecem desconhecidas [Fortnow 2009]. Entretanto, pesquisadores usam métodos heurísticos com o objetivo de encontrar as soluções mais aproximadas da solução exata/ótima em tempo polinomial. O objetivo de uma heurística é produzir uma solução aproximada da ótima em um espaço de tempo razoavelmente aceitável para resolver um determinado problema. Neste sentido, o presente trabalho visa mostrar a aplicação de um método heurístico eficiente para encontrar uma solução aproximada para o problema do Bin Packing (empacotamento), mostrando a sua análise segundo critérios de proximidade da solução ótima. O propósito deste problema é usar o menor número de *bins* (caixas) necessárias para empacotar um determinado conjunto de itens.

O problema Bin Packing pode ser encontrado em exemplos práticos, como na logística (carga de caminhões/contêineres) e na virtualização de servidores (consolidação de máquinas virtuais (VMs)) [Monil and Rahman 2016]. Especificamente no problema de consolidação de VMs, busca-se o provisionamento eficiente de recursos de servidores físicos através da alocação ou realocação de VMs (itens), objetivando usar o menor

número de servidores físicos (*bins*). Como este é um problema de otimização, também é classificado como pertencente à classe NP-Difícil. Em uma notação simplificada deste problema, cada *bin* do conjunto de *bins* possui capacidade igual a 1.0, representando 100%, e cada item possui um tamanho de (0.0, 1.0], representando um dado percentual da capacidade de um *bin*.

De modo genérico, para selecionar uma heurística que resolva um determinado problema, pode-se considerar quatro critérios importantes, conforme relacionados na Tabela 1 [Kunche and Reddy 2016]. Existe um *trade-off* entre estes critérios, pois deve-se maximizar o desempenho (otimização, completude e/ou precisão) e minimizar o tempo de execução.

Critérios	Descrição
Otimalidade:	Quando existem várias soluções para um dado problema, a heurística garante que a melhor solução será encontrada até que limite?
Completude:	Quando várias soluções existem, a heurística encontra todas elas? São necessárias todas as soluções?
Acurácia e precisão:	A heurística pode fornecer um intervalo de confiança para a solução?
Tempo de execução:	Esta é a heurística mais rápida para resolver o problema? (Algumas heurísticas convergem mais rápido que outras.)

Tabela 1. Critérios para análise de heurísticas. Adaptado de [Kunche and Reddy 2016]

Especificamente para o problema Bin Packing, será utilizada a abordagem heurística denominada First Fit Decreasing (FFD) (detalhada na Seção 3), sendo esta abordagem heurística analisada segundo os critérios da Tabela 1. A completude é um critério trivial no contexto do presente trabalho, pois refere-se à possibilidade de existir mais de uma solução ótima em relação ao arranjo do conjunto de itens distribuídos entre os *bins*. A otimalidade consiste em encontrar uma faixa de valores aproximados possíveis, considerando o valor ótimo de *bins*, *i.e.*, até o número possível de *bins* para empacotar um determinado conjunto de itens nesta heurística. Neste sentido, o presente trabalho mostra uma análise sobre a estimativa da solução exata para o problema do Bin Packing com a heurística FFD, utilizando uma função de limite inferior como base indicativo da solução ótima e a análise do critério de otimalidade como o limite superior para o FFD.

O presente trabalho está organizado da seguinte forma. É fornecido uma breve descrição do problema Bin Packing na Seção 2, com algumas variações e aplicação em um problema real. A Seção 3 explica o funcionamento e comportamento da solução heurística FFD, bem como é feita a análise de sua otimalidade. São apresentados na Seção 4, os resultados da aplicação do FFD em uma instância de vinte e dois itens, bem como os detalhes do desenvolvimento em linguagem Java.

2. O problema Bin Packing

Dado como entrada um conjunto de números positivos $S = \{s_1, s_2, s_3, ..., s_n\}$ e uma capacidade C, deve-se separar os elementos (itens) do conjunto S em subconjuntos (*bins*)

mutuamente exclusivos e coletivamente exaustivos, com somas menores ou iguais a C, de modo a minimizar a quantidade de *bins* [Souza 2011]. Este é um problema de natureza combinatória, que cresce exponencialmente em relação ao tamanho das entradas.

A função de limite inferior (lb) para o Bin Packing calcula o número mínimo de *bins* com soma igual ou inferior a capacidade C necessária para empacotar todos os itens do conjunto de entrada S. Se uma solução é encontrada, a busca pode terminar imediatamente ao encontrar o valor de lb como uma solução ótima. Em Martello and Toth 1990, é especificada uma função lb, conforme descrita na Equação 1, sendo seu resultado somado com um valor w para que seja arredondado para um inteiro.

$$l_b = \frac{Soma\ dos\ Itens\ em\ S}{Capacidade\ do\ Bin} + w = \frac{\sum_{i=0}^{n} s_i}{C} + w \tag{1}$$

Na Equação 1, não é considerado a limitação de que os itens em S são indivisíveis, e pode ocorrer de a solução ótima fornecida pela função lb ser inalcançável em alguns casos. Contudo existem outras funções de limite inferior que podem remover esta limitação, considerando os espaços remanescentes em cada *bin* para alocar os itens de menor valor.

2.1. Variações

Podem existir duas variações do problema Bin Packing: Online, aonde os itens chegam um de cada vez (em uma ordem desconhecida) e cada item deve ser colocado em um bin antes do próximo chegar; e Offline, sendo todos os itens do conjunto fornecidos antes de executar o algoritmo.

O problema Online possui um grau de dificuldade maior. Inclusive, pode-se perceber que raramente a solução ótima para esse problema poderá ser alcançada. Como exemplo, consideremos a entrada: M itens "menores" de tamanho 1/2 - e chegam primeiro, seguidos de M itens "grandes" de tamanho 1/2 + e, para qualquer $0 < e < 0.001$. Uma boa solução seria colocá-los em pares (um pequeno, um grande), utilizando-se de $M\ bins$.

Porém, o algoritmo online não sabe o valor dos itens que estão por vir, e nem quantos itens ainda faltam. Para o exemplo mostrado anteriormente, o algoritmo separaria em *bins* de 2 itens os M primeiros itens que chegaram, porém os M últimos ficaria com apenas um item. Portanto o algoritmo teria $M/2 + M$ *bins*. Para chegar a solução ótima, o algoritmo deveria colocar os M primeiros itens em um *bin* diferente. Porém, ao realizar isso, o algoritmo teria feito uma solução que para os primeiros M itens resultou em M *bins*, que não faria muito sentido.

2.2. Exemplo de Aplicação: Alocação de recursos computacionais

Em ambientes computacionais que utilizam a virtualização, é necessário que o provisionamento de recursos acorra de modo automatizado, sendo possível modelar a alocação de recursos e a consolidação de servidores como o problema do Bin Packing [Beloglazov and Buyya 2012, Dow 2016]. Nesse sentido, os servidores físicos são os *bins*, enquanto cada maquina virtual (VM) é um item a ser empacotado. No modelo apresentado no presente trabalho, assume-se que todos os servidores físicos são homogêneos, *i.e.*, com a

mesma capacidade C unitária. A demanda de recursos é normalizada como uma fração desta capacidade unitária, *e.g.*, se uma VM requer 20% da memória física de um servidor, então este valor corresponde a um item de tamanho $0,2$, e assim por diante.

Ainda existem outros modos de correlacionar o problema Bin Packing com a alocação de recursos para VMs [Song et al. 2014], como por exemplo: considerar os bins heterogêneos, *i.e.*, bins com diferentes valores de C entre si; ou ainda modelar os *bins* com um valor multidimensional, considerando a quantidade de processadores, memória, rede e disco, etc, como dimensões dos itens a serem empacotados pelos servidores e equipamentos de rede.

3. Solução Heurística

Uma das abordagens heurísticas clássicas de aproximação para o problema Bin Packing, é o *First-Fit Decreasing* (FFD) [Schreiber and Korf 2013]. O FFD consiste em ordenar o conjunto de itens S inicialmente em ordem decrescente, analisando-se um item por vez e cada elemento é colocado no primeiro compartimento em que se encaixa. O Algoritmo 1 mostra uma implementação simplificada do FFD. Analisando sua complexidade de tempo, no melhor caso é de $O(n^2)$, considerando que n é o conjunto de itens do consjunto de entrada S. Para o caso médio ao melhor caso, considerando que os n itens sejam alocados em b *bins*, a complexidade é de $O(n * b)$.

Algoritmo 1: First Fit Decreasing

```
Dados: Itens; bins
Entrada: Dois vetores: itens e bins.
Saída: Itens alocados nos Bins
/*ordene o vetor em decrescente*/
decreasingOrder(Itens);
para cada Item s no vetor de Itens hacer
    para cada Bin b no vetor de Bins hacer
        se s.addItem(p) == SUCESSO então
            break;
        fim
    fin
fin
```

Na análise de otimalidade do FFD, considerando que a função de limite inferior lb forneça o valor ótimo de m *bins*, seu grau de proximidade desta solução ótima pode ser dada por $(4m + 1)/3$. Em outras palavras, o valor aproximado da abordagem FFD pode variar em até 34% acima de seu valor ótimo de *bins*. Por exemplo, se um valor ótimo de *bins* para uma determinada instância de itens com valores igualmente distribuídos for 5, então o FFD pode empacotar estes itens entre 5 e 7 *bins*.

4. Resultados

Como um estudo de caso, foi implementado o problema do Bin Packing na linguagem Java-8. O algoritmo FFD foi implementado como a solução heurística para resolver uma determinada instância de itens, que são introduzidas na implementação como um arquivo.

Bins	Preenchido	Itens em cada bin
Bin_1	0,995	[0,53, 0,45, 0,015]
Bin_2	1,000	[0,42, 0,33, 0,25]
Bin_3	1,000	[0,32, 0,3, 0,28, 0,1]
Bin_4	0,979	[0,23, 0,2, 0,199, 0,18, 0,17]
Bin_5	0,895	[0,16, 0,15, 0,13, 0,125, 0,12, 0,11, 0,1]

Tabela 2. Tabela com resultados da entrada S.

Com a finalidade de conhecer o comportamento do FFD, foi considerado um conjunto de vinte e dois itens, sendo os valores de $0,01 < s_i < 1,00$ e a capacidade unitária dos *bins* como $C = 1,00$. Segue o conjunto de entrada:

S = {0,1; 0,15; 0,2; 0,12; 0,25; 0,23; 0,11; 0,32; 0,28; 0,18; 0,3; 0,45; 0,13; 0,16; 0,10; 0,17; 0,33; 0,125; 0,53; 0,42; 0,199; 0,015}

Para esse problema, o número mínimo de *bins* que induzem a solução ótima pode ser calculado utilizando a Equação 1, encontrando o valor de cinco *bins*. Como resultados obtidos, a Tabela 2 mostra a distribuição dos itens, sendo apresentada a quantidade de $bins$, o espaço que foi utilizado de cada bin e quais os itens alocados em cada bin.

Percebe-se que, para essa instância, a solução encontrada pelo algoritmo heurístico FFD desenvolvido foi de cinco *bins* e, portanto, a configuração da Tabela 2 é uma solução ótima. Foi feito um segundo teste com uma instância maior, com 2000 itens de mesmas características do teste apresentado anteriormente. A quantidade de *bins* que a função FFD encontrou foi de 451 *bins*, sendo que o valor ótimo para esta instância era de 445 *bins*, executado em um tempo de 73ms. Segundo a análise de otimalidade dada por $(4M + 1)/3$, poderia-se tem uma variação entre $445 < lb < 594$ bins, mas neste caso, a distância foi de 6 *bins*.

5. Conclusões

Nesse trabalho foi analisado o algoritmo heurístico chamado First Fit Decresing (FFD) para resolver o problema do Bin Packing. Esse problema tem muitas aplicações reais, como em logística e alocação de recursos para Máquinas Virtuais, por exemplo. O algoritmo heurístico possibilita encontrar uma boa solução para o problema, porém não necessariamente a ótima, e em tempo aceitável. No caso do FFD, foi implementado com complexidade de $O(n^2)$ no pior caso, sendo n a quantidade de pacotes de entrada.

A abordagem FFD foi analisada segundos os critérios de otimalidade e seu limite superior de bins poed ser dado por $(4 * M + 1)/3$, sendo M o valor ótimo de bins. Em teste com instancia maior (2000 itens), foi encontrado um valor de bin próximo do ótimo, apenas seis unidades a mais.

Referências

Beloglazov, A. and Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency Computat.: Pract. Exper.*, 24(13):1397–1420.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158. ACM.

Dow, E. M. (2016). Decomposed multi-objective bin-packing for virtual machine consolidation. *PeerJ Comput. Sci.*, 2:e47.

Fortnow, L. (2009). The status of the p versus NP problem. *Communications of the ACM*, 52(9):78.

Kunche, P. and Reddy, K. (2016). *Metaheuristic Applications to Speech Enhancement*. SpringerBriefs in Electrical and Computer Engineering. Springer International Publishing.

Martello, S. and Toth, P. (1990). Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, 28(1):59–70.

Monil, M. A. H. and Rahman, R. M. (2016). VM consolidation approach based on heuristics, fuzzy logic, and migration control. *Journal of Cloud Computing*, 5(1):8.

Schreiber, E. L. and Korf, R. E. (2013). Improved bin completion for optimal bin packing and number partitioning. In *IJCAI*, IJCAI '13, pages 651–658. AAAI Press.

Song, W., Xiao, Z., Chen, Q., and Luo, H. (2014). Adaptive resource provisioning for the cloud using online bin packing. *IEEE Trans. Comput.*, 63(11):2647–2660.

Souza, A. (2011). Lecture for Humboldt University Berlin: Combinatorial Algorithms - Algorithms and Complexity.