Simulated Annealing aplicado ao problema do empacotamento unidimensional

José Guilherme P. Lima[1], **Phillipe I. M. Silva** [1], **Walberto M. dos Santos** [1], **Glaubos Clímaco** [1]

[1]Departamento de informática – Universidade Federal do Maranhão (UFMA)
Caixa Postal 1966 – Vila Bacanga 65080-805 – São Luís – MA – Brasil

guilherme-lima@hotmail.com,phillipe.mendonca20@gmail.com
walberto.marques@gmail.com, francisco.glaubos@ufma.br

Abstract. *The one-dimension bin packing problem (PEU) is an NP-hard combinatorial optimization problem, with several applications in the real world. There are several methods and forms present in the literature to find solutions to this type of problem. In this paper, a Simulated Annealing metaheuristic was developed for the resolution of the PEU, and mathematical solver Gurobi was used to obtain optimal solutions. Computational experiments were conducted, and the proposed SA proved to be robust in solving instances from the literature.*

Resumo. *O problema do empacotamento unidimensional (PEU) é um problema de otimização combinatória NP-difícil, e com várias aplicações no mundo real. Existem vários métodos e formas presentes na literatura para encontrar soluções para este tipo de problema. Neste trabalho, foi desenvolvida uma metaheurística Simulated Annealig para a resolução do PEU, e a utilizada do solver matemático Gurobi para a obtenção de soluções ótimas. Experimentos computacionais foram conduzidos, e o SA proposto mostrou-se robusto ao resolver instâncias da literatura.*

1. Introdução

Dado um conjunto $N = \{1, ..., n\}$ de itens com pesos w_i, $i = 1, ..., n$, o problema do empacotamento unidimensional (PEU) consiste em encontrar o número mínimo de caixas de capacidade C necessário para embalar os itens sem violar as restrições de capacidade. Alternativamente, o problema também pode ser visto como o particionamento do conjunto de itens em um número mínimo de subconjuntos, de modo que a soma dos pesos dos itens em cada subconjunto seja menor ou igual a C.

O PEU é um problema clássico da literatura e possui um vasto número de aplicações no mundo real. Como por exemplo: Alocação de memória em sistemas paginados, escalonamento de processadores, escalonamento de comerciais para televisão e rádio, carregamento de veículos, corte e estoque de objetos unidimensionais, etc [Alvim et al. 2004].

Por ser um problema NP-Difícil [Karmarkar and Karp 1982], resolver o PEU de maneira exata torna-se inviável à medida que N cresce exponencialmente. Para contornar esse problema, surgiram as metaheurísticas, que são procedimentos que controlam diferentes heurísticas a fim de evitar ótimos locais. Diferente dos métodos exatos, as

metaheurísticas não focam em encontrar uma solução ótima para o problema, mas em encontrar soluções de boa qualidade em um tempo computacional razoável.

Neste trabalho, para a resolução do PUE, propõe-se a aplicação de um solver matemático a partir de uma formulação da literatura, e o desenvolvimento de uma metaheurística Simulated Annealing, que faz uso da heurística *First Fit Decreasing* (FFD) [Dósa 2007]. O restante deste artigo está organizado da seguinte maneira: na Seção 2, é descrita uma formulação matemática para o PEU; na Seção 3, os métodos propostos são exibidos; na Seção 4, são expostos resultados de experimentos computacionais realizados com o solver e o SA; e por fim, na Seção 5, são apresentadas conclusões e trabalhos futuros.

2. Formulação matemática

Uma maneira de descrever o problema do empacotamento, consiste em empacotar itens de tamanhos distintos em um número finito de caixas ou pacotes, de forma a minimizar o número de recipientes utilizados. No qual todos os recipientes têm a mesma capacidade, e há permissão de espaço vago independente de qual abordagem utilizada.

Segundo [Hall et al. 1988], dado um conjunto de caixas $S_1, S_2...$ com o mesmo tamanho C e uma lista de n itens com tamanhos $a_1, \ldots, a_n$ para empacotar, uma formulação matemática para o PEU pode ser descrita da seguinte maneira:

$$Minimizar \qquad B = \sum_{i=1}^{n} y_i \tag{1}$$

$$s.a. : B \geq 1 \tag{2}$$

$$\sum_{j=1}^{n} a_j x_{ij} \leq V y_i, \qquad \forall i \in \{1, \ldots, n\} \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j \in \{1, \ldots, n\} \tag{4}$$

$$y_i \in \{0, 1\}, \qquad \forall i \in \{1, \ldots, n\} \tag{5}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in \{1, \ldots, n\} \, \forall j \in \{1, \ldots, n\} \tag{6}$$

No qual as variáveis $y_i = 1$ se a caixa i é utilizada (0 caso contrário) e, $x_{ij} = 1$ se o item j é colocado dentro da caixa i. A função objetivo (1) visa minimizar a quantidade de caixas utilizadas. A restrição (2) garante ao modelo que pelo menos uma caixa será utilizada. As restrições (3) asseguram que a capacidade de uma caixa i não será extrapolada. As restrições (4) garantem que cada item poderá ser armazenado em apenas uma caixa. Por fim, as restrições (5) e (6) definem o domínio das variáveis.

3. Abordagem proposta

Como abordagens para o PEU, foram utilizados o solver Gurobi para a resolução do modelo matemático e a metaheurística SA. O Gurobi Optimizer é uma ferramenta computacional consolidada pela indústria devido ao seu desempenho em programação linear, quadrática e inteira mista.

Por utilizar o paradigma computacional de análise, projeto e programação orientado a objetos, traz consigo a facilidade de comunicação com plataformas de desenvolvimento de softwares e linguagens como C++, Java, Matlab, entre outros [Optimization 2018].

3.1. Simulated Annealing

Simulated Annealing (SA) é uma meta-heurística para otimização de problemas que se fundamenta numa analogia com a termodinâmica. Annealing (recozimento) é o processo térmico utilizado para fundir um metal, no qual este é aquecido e em seguida resfriado lentamente de modo que o material final fique homogêneo. Primeiramente proposto por [Kirkpatrick et al. 1983] para encontrar o mínimo global em uma função de custo que pode ter vários mínimos locais.

O funcionamento básico da SA consiste em: i) estando numa solução S, fazer um movimento aleatório, resultando em uma solução S'; e ii) se o movimento resultar em melhoria $(c(S') < c(S))$, adotar S'. Se não for uma melhoria, adote S' probabilisticamente dependendo da diferença de custo e um parâmetro de temperatura, T. S' é adotado com probabilidade $e^{\frac{-(c(S')-c(S))}{T}}$, e se T for decrementado em uma velocidade lenta suficiente, então o SA encontrará um ótimo global com probabilidade próxima a 1. O pseudo-código do SA utilizado é apresentado no pseudo-código do Algoritmo 1.

Por padrão, os parâmetros iniciais são: uma solução inicial aleatória (S), temperatura inicial (T_0), número de iterações ($SAmax$) máximo ou um limite de tempo de processamento, e uma taxa de redução da temperatura (α). O Algoritmo 1 começa com um valor alto para o parâmetro de temperatura e uma solução inicial. Em seguida, enquanto a temperatura for maior que zero, para cada temperatura T gradativamente menor, tenta-se aprimorar S buscando uma solução melhor em sua vizinhança $S' \in N(S)$. Essa busca é realizada enquanto um número de buscas máximo $SAmax$ não for atingido.

3.2. *First Fit Decreasing* (FFD) e busca na vizinhança

Para a construção da solução inicial S_0, foi utilizado o algoritmo *First Fit Decreasing* (FFD) [Baker 1985]. O FFD é um algoritmo guloso aproximativo que requer um tempo $O(nlogn)$. O algoritmo processa os itens em uma ordem arbitrária. Para cada item, ele tenta colocar o item na primeira caixa que possa acomodá-lo. Se nenhuma caixa é encontrada, o algoritmo abre uma nova caixa e põe o item dentro da nova caixa. Esse algoritmo atinge um fator de aproximação igual a 2; ou seja, o número de caixas utilizada por esse algoritmo não é mais que duas vezes o número ótimo de caixas a serem utilizadas. Um pseudo-código do FFD utilizado é apresentado no pseudo-código do Algoritmo 2.

Uma vez criada uma solução inicial, o algoritmo tenta aprimorá-la ao longo das iterações utilizando uma das duas técnicas descritas em [Rao and Iyengar 1994], que consideramos como 1 e 2:

1. Realocação de um único item selecionado de um pacote para outro pacote selecionado aleatoriamente.
2. Seleção aleatória de dois itens alocados em dois pacotes diferentes e trocando suas posições.

À temperaturas mais baixas a técnica 2 é utilizada, enquanto a 1 é utilizada em temperaturas mais altas.

Algorithm 1: SA $(f(.), N(.), \alpha, SAmax, T_0, s)$

$s^* \leftarrow s$ (Melhor solução obtida até então)
$IterT \leftarrow 0$ (Número de iterações na temperatura T)
$T \leftarrow T_0$ (temperatura corrente)
Enquanto $(T > 0)$ **faça**
  Enquanto $(Iter < SAmax)$ **faça**
    $IterT \leftarrow IterT + 1$
    Gere um vizinho qualquer $s' \in N(s)$
    $\triangle = f(s') - f(s)$
    Se $(\triangle < 0)$ **então**
      $s \leftarrow s'$
      Se $(f(s') < f(s^*)$ **então**
        $s \leftarrow s'$
      Senão
        tome $x \in [0, 1]$
        Se $(x < e^{(-\delta/T)})$ **então**
          $s \leftarrow s'$
        Fim-se
      Fim-se
    Fim-se
  Fim-enquanto
  $T \leftarrow \alpha \times T$
  $IterT \leftarrow 0$
Fim-enquanto
$s \leftarrow s^*$
Retorne s *//Saída do algoritmo*

Algorithm 2: FFD

Classificar os objetos em ordem decrescente por peso
Para todos Objeto $i = 1...n$ **faça**
  Para todos Pacote $j = 1...n$ **faça**
    Se objeto i couber no pacote j **então**
      colocar objeto i em pacote j
      Pare e **continue** com o próximo objeto
    Fim-se
  Fim-para
  Se Objeto i não couber em qualquer pacote aberto **então**
    criar novo pacote e colocar objeto i
  Fim-se
Fim-para

4. Resultados

Esta seção apresenta os resultados das duas propostas desse trabalho: aplicação do solver Gurobi (versão 8.0) e uma metaheurística SA. Para a implementação foi utilizada a

linguagem Python 3.7.3, e os experimentos computacionais foram executados em uma máquina Core(™) i3-4010U 1.70GHz com o sistema operacional Windows.

Para validar as propostas, foram utilizadas instâncias clássicas da literatura, que podem ser encontradas em `http://or.dei.unibo.it/library/bpplib`. A Tabela 1 apresenta os resultados com o solver Gurobi, a partir da formulação matemática apresentada na Seção 2. A primeira coluna indica a instância testada, a segunda coluna o número de itens, a terceira coluna a capacidade das caixas, a quarta coluna exibe a solução ótima obtida, e a quinta coluna indica o tempo despendido em segundos. Como o solver pode consumir bastante memória da máquina no processo de branch & bound, foi determinado um tempo de 3600 segundos de processamento. O símbolo "-" indica que o solver não foi capaz de atingir uma solução dentro do tempo limite.

Tabela 1. Resultados utilizando o solver Gurobi.

instância	n	C	Sol.	T. (s)
binpack_1	10	10	5	0,5
binpack_2	50	150	19	13,5
binpack_3	120	150	48	196,2
binpack_4	1000	150	-	3600,0
binpack_5	1000	150	-	3600,0
binpack_6	1000	150	-	3600,0
binpack_7	250	150	-	3600,0
binpack_8	250	150	-	3600,0
binpack_9	250	150	-	3600,0
binpack_10	250	150	-	3600,0

A partir da Tabela 1, percebe-se que o solver precisou de pouco tempo para resolver instâncias de até 120 itens. Contudo, o mesmo não consegue atingir nenhuma solução nas instâncias maiores, em que o número de itens é maior que 120.

A Tabela 2 apresenta os resultados da metaheurística SA, na qual a coluna "Sol. Média" apresenta a solução média obtida a partir de dez execuções, com sementes aleatórias variando de 1 a 10. Para a aplicação desse método, foram definidos os seguintes parâmetros, de acordo com a aplicação do SA em outros problemas de empacotamento [Dowsland 1993]: temperatura inicial igual 1000, temperatura final igual 0.1, taxa de redução da temperatura igual 0.95, e número de iterações igual 1000.

A partir da Tabela 2, pode ser observado que o SA atinge o ótimo apenas para a menor instância. Porém, resolve todas as instâncias testadas, em um tempo máximo de 487 segundos. Portanto, para problemas reais, a resolução por meio do SA mostra-se mais promissora que pelo solver matemático.

5. Conclusão

Neste trabalho, foi apresentado os resultados do problema do empacotamento unidimensional usando o solver Gurobi e em seguida, a meta-heurística Simulated Annealing. Em termos de soluções, foi observado que o SA proporciona soluções de boa qualidade para todas as instâncias testadas, ao contrário do solver, que resolve menos da metade das instâncias.

Tabela 2. Resultados com o simulated annealing.

Instância	n	C	Melhor Sol.	Sol. Média	T. Médio (s)
binpack_1	10	10	5	5	59,8
binpack_2	50	150	20	20	152,4
binpack_3	120	150	49	49	214,9
binpack_4	1000	150	403	399	487,0
binpack_5	1000	150	411	406	466,0
binpack_6	1000	150	416	411	474,0
binpack_7	250	150	102	100	147,0
binpack_8	250	150	103	102	136,0
binpack_9	250	150	99	97	128,0
binpack_10	250	150	107	105	139,0

Ao se trabalhar com o Simulated Annealing, foi observado que: esse método é de fácil implementação computacional, possuindo uma baixa dependência de parâmetros; demanda algum mecanismo para o controle de soluções repetidas; o mecanismo de aceitação de soluções de piora com uma probabilidade decrescente, pode ser um mecanismo poderosíssimo, e deve ser agregado em outras meta-heurísticas; e que uma boa busca local é fundamental para a eficiência do método.

Como investigações futuras, pretende-se desenvolver uma técnica de pré-processamento, a fim de reduzir o tamanho do problema de entrada, e auxiliar na resolução do solver. Em relação ao SA, pretende-se desenvolver outras heurísticas de construção e busca locais mais elaboradas.

Referências

Alvim, A. C., Ribeiro, C. C., Glover, F., and Aloise, D. J. (2004). A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, 10(2):205–229.

Baker, B. S. (1985). A new proof for the first-fit decreasing bin-packing algorithm. *Journal of Algorithms*, 6:49–70.

Dósa, G. (2007). The tight bound of first fit decreasing bin-packing algorithm is ffd (i) 11/9opt (i)+ 6/9. In *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, pages 1–11. Springer.

Dowsland, K. A. (1993). Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research*, 68:389–399.

Hall, N. G., Ghosh, S., Kankey, R. D., Narasimhan, S., and Rhee, W. T. (1988). Bin packing problems in one dimension: heuristic solutions and confidence intervals. *Computers & operations research*, 15:171–177.

Karmarkar, N. and Karp, R. M. (1982). An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 312–320. IEEE.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Optimization, G. (2018). Inc.,“gurobi optimizer reference manual,” 2015.

Rao, R. and Iyengar, S. (1994). Bin-packing by simulated annealing. *Computers & Mathematics with Applications*, 27(5):71–82.