TRATAMENTO DO PROBLEMA DE EMPACOTAMENTO UNIDIMENSIONAL VIA SIMULATED ANNEALING

Samuel Estevão Vendramini
Instituto Tecnológico da Aeronáutica
Praça Marechal Eduardo Gomes, 50 - Vila das Acacias, São José dos Campos - SP, 12228-900
samuelsev@ita.br

Sarah da Silva Pereira
Instituto Tecnológico da Aeronáutica
Praça Marechal Eduardo Gomes, 50 - Vila das Acacias, São José dos Campos - SP, 12228-900
sarahsp@ita.br

Vinicius Lacerda
Instituto Tecnológico da Aeronáutica
Praça Marechal Eduardo Gomes, 50 - Vila das Acacias, São José dos Campos - SP, 12228-900
vinicius.lopes@ga.ita.br

RESUMO

O problema de empacotamento unidimensional é um problema NP-difícil que objetiva minimizar o número de recipientes necessários para armazenar um conjunto fixo de objetos. Por sua natureza combinatória e sua consequente inviabilidade de solução exata para grandes instâncias, foram escolhidas a meta-heurística *Simulated Annealing* (SA) e a heurística *First Fit Decreasing* (FFD) na busca de obter boas respostas para o problema em sua forma unidimensional. O solver matemático Gurobi foi, ainda, utilizado como medida comparativa para os resultados computacionais do SA, demonstrando este ter, em geral, boa performance.

PALAVRAS CHAVE. Problema de empacotamento, Simulated Annealing, Meta-heurística

ABSTRACT

The unidimensional Bin Packing Problem is an NP-difficult problem that aims to minimize the number of containers needed to store a fixed set of objects. Due to their combinatorial nature and consequent unfeasibility of an exact solution for large instances, the meta-heuristic *Simulated Annealing* (SA) and the heuristic *First Fit Decreasing* (FFD) were chosen in order to obtain good answers for the problem in its one-dimensional form. The mathematical solver Gurobi was also used as a comparative measure for the computational results of SA, demonstrating that it has, in general, good performance.

KEYWORDS. Bin Packing Problem, Simulated Annealing, Metaheuristic

1. Introdução

O Problema do Empacotamento, ou *Bin Packing Problem* (BPP), é um clássico da Pesquisa Operacional, o qual encontra o número mínimo de recipientes necessários para guardar um conjunto de diferentes objetos. Sua aplicação mais óbvia está relacionada a estágios do *Supply Chain*, como carregamento de contêineres e caminhões [Ugur e Tursel, 2009]. Porém, a pluralidade de áreas em que o BPP aparece vai desde a criação de backups de arquivos em mídia até o mapeamento de tecnologia em design de chip semicondutor de matriz de porta programável em campo, escalonamento de comerciais para televisão e corte de objetos, caracterizando-se, assim, como um problema abrangente quanto a seu uso [Lima et al.]. Há, ainda, três classificações possíveis para o problema do empacotamento: unidimensional, bidimensional e tridimensional.

A solução ótima do Problema do Empacotamento pode ser encontrada por algoritmos como *Branch and Bound* ou Programação Dinâmica. No entanto, devido à natureza combinatória do problema, os métodos computacionais exatos, que trabalham com enumeração do espaço de solução, são inviáveis para resolução de grandes instâncias em tempos computacionais razoáveis. Assim, devido a essa limitação, torna-se necessário o uso de métodos heurísticos que, ainda que não consigam garantir o ótimo global, trazem bons resultados em curto espaço de tempo. Alguns exemplos de meta-heurísticas e heurísticas que podemos encontrar na literatura aplicados a esse problema são: *Tabu Search, Weight Annealing, Variable Neighbourhood Search e Best Fit Heuristic* [Sonuc et al., 2017]. Nesse trabalho, foram escolhidos como meta-heurística o Simulated Annealing (SA) e, para a definição da solução inicial, o algoritmo *First Fit Decreasing* (FFD) a fim de solucionar uma particularização do *Bin Packing*, o Problema de Empacotamento Unidimensional (PEU) com caixas de mesmo tamanho. Ainda, para fins comparativos, foi também utilizado o Solver Gurobi.

Este trabalho está dividido em três partes. Inicialmente, será abordada a modelagem matemática do PEU, em seguida, será detalhado o funcionamento do *Simulated Annealing* e do *First Fit Decreasing* e, por fim, serão discutidos os resultados computacionais.

2. Definição do Problema

A modelagem matemática do PEU pode ser escrita da seguinte maneira [Sonuc et al., 2017]:

$$Minimizar: \ B = \sum_{i=1}^{n} b_i \tag{1}$$

$$\sum_{j=1}^{n} w_j a_{ij} \leq c \cdot b_i, \ a_i \in N, \ i \in \{1, ..., n\} \tag{2}$$

$$\sum_{i=1}^{n} a_{ij} = 1, \ j \in N \tag{3}$$

$$b_i, \ a_{ij} \in \mathbb{B} \tag{4}$$

Onde w_i representa o pesa de cada objeto, c a capacidade da caixa e

$$b_i = \begin{cases} 1, \text{ se a caixa } i \text{ é usada} \\ 0, \text{ caso contrário} \end{cases}$$

$$a_{ij} = \begin{cases} 1, \text{ se o objeto } j \text{ é colocado na caixa } i \\ 0, \text{ caso contrário} \end{cases}$$

A equação (1) estabelece o objetivo do problema: a minimização da quantidade de caixas que guardarão os objetos, enquanto a equação (2) restringe o peso dos objetos dentro de uma caixa à capacidade máxima dela e a equação (3) garante que o objeto estará em apenas uma caixa. Vale ressaltar que $\mathbb{B}$ representa o espaço binário.

3. Metodologia

3.1. Simulated Annealing

É uma meta-heurística que se baseia no processo de recozimento, em que um metal é aquecido além de seu ponto de fusão e, depois, resfriado. Foi proposta por Kirkpatrick em 1983, que sugeriu que a simulação desse processo físico poderia ser utilizado para busca de soluções [Kirkpatrick et al., 1983].

O funcionamento básico do SA consiste em, estando numa solução S, fazer um movimento aleatório que resulte em outra solução S'. Essa solução S' será aceita se seu resultado for melhor que S ou será aceita de forma probabilística caso não seja melhor que S. A probabilidade de aceitação de S' é dada por $e^{-\Delta/T}$, em que Δ é a variação da função objetivo de S e S' e T a temperatura atual. Assim, esse valor de probabilidade é alto a temperaturas elevadas e baixo a temperaturas baixas. Ou seja, no início, o algoritmo percorre amplamente o espaço de soluções, já que começa com alta temperatura e, consequentemente, aceita quase todas as soluções, mesmo que ruins. À medida que a temperatura vai diminuindo, o SA é mais seletivo, tendendo a aceitar apenas soluções melhores que a anterior. [Lima et al.]

Para o PEU, a escolha de vizinhos no algoritmo do SA pode dar-se de duas formas: intercâmbio de objetos de caixas diferentes ou realocação de um objeto em outra caixa [Rao e Iyengar, 1994]. A figura a seguir representa esses movimentos em busca de novas soluções:

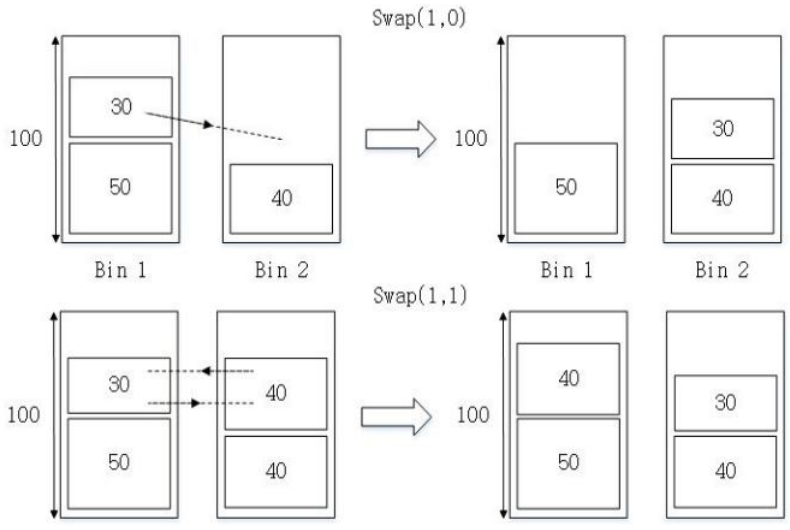

Figura 1: Movimentos possíveis para a busca de vizinhos no algoritmo Simulated Annealing

Os parâmetros iniciais do *Simulated Annealing* são: solução inicial, temperatura inicial, número de iterações máximo e taxa de redução de temperatura. Todos esses parâmetros, com exceção da primeira, são adaptados para esse problema num processo de tentativa e erro, já a solução inicial será construída através da heurística FFD.

Especificamente para o PEU, algo a ser notado é que a função objetivo do modelo matemático, como definida anteriormente, pode apresentar nenhuma variação para uma quantidade considerável de soluções vizinhas. Isso não é adequado e torna a meta-heurística muito pouco eficiente. Assim, a alternativa que foi empregada nesse trabalho foi a alteração da função objetivo para

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

```
1: s* ← s (Melhor solução obtida até então)
2: IterT ← 0 (Número de iterações na temperatura T)
3: T ← T_0 (temperatura corrente)
4: Enquanto (T > 0) faça
5:   Enquanto (Iter < SAmax) faça
6:     IterT ← IterT + 1
7:     Gere um vizinho qualquer s' ∈ N(s)
8:     △ = f(s') − f(s)
9:     Se (△ < 0) então
10:      s ← s'
11:      Se (f(s') < f(s*) então
12:        s ← s'
13:      Senão
14:        tome x ∈ [0, 1]
15:        Se (x < e^(−△/T)) então
16:          s ← s'
17:        Fim-se
18:      Fim-se
19:    Fim-se
20:  Fim-enquanto
21:  T ← α × T
22:  IterT ← 0
23: Fim-enquanto
24: s ← s*
25: Retorne s //Saída do algoritmo
```

Figura 2: Pseudo-código para o Simulated Annealing

a soma do quadrado do peso que cada caixa carrega [Henderson et al., 2003], segundo a equação:

$$f = \sum_{i=1}^{n} \left(\sum_{k_i} w_j \right)^2 \tag{5}$$

Na função objetivo, k_i é o conjunto de itens na caixa i. Assim, é possível obter uma medida de eficiência no empacotamento dos itens, ainda que seja utilizado o mesmo número de caixas. Vale ressaltar que, para essa função objetivo, temos um problema de maximização.

3.2. First Fit Decreasing

Um dos parâmetros do *Simulated Annealing* é a solução inicial. Para sua construção, foi utilizada o FFD, o qual consiste em um algoritmo guloso que ordena os itens em ordem decrescente e tenta colocá-los na primeira caixa que possa acomodá-lo. Caso nenhuma caixa o suporte, uma nova é criada e o item é colocado ali. O FFD pode ser implementado em $O(n \log n)$, mostrando-se muito eficiente em termos computacionais. O algoritmo básico da heurística é apresentado na figura 3. [Sonuc et al., 2017]

4. Resultados Computacionais

Todos os algoritmos foram implementados em Python 3.8.5 e executados em uma máquina Intel(R) Core(TM) i3-3337U 1.80GHz com ambiente Windows.

As instâncias utilizadas foram selecionadas a partir das produzidas por E. Falkenauer e disponíveis em um banco de dados *online* [Delorme et al., 2018]. No total, foram testadas 16 instâncias, com 8 pertencendo a uma classe U (*Uniform*) e outras 8 a uma classe T (*Triplets*). As instâncias U consistem de um conjunto de objetos com pesos uniformemente distribuídos entre 20 e 100 que devem ser alocados em mochilas com capacidade 150. As intâncias T, de objetos com pesos

Algorithm 2: FFD

```
1: Classificar os objetos em ordem decrescente por peso
2: Para todos Objeto i = 1...n faça
3:   Para todos Pacote j = 1...n faça
4:     Se objeto i couber no pacote j então
5:       colocar objeto i em pacote j
6:       Pare e continue com o próximo objeto
7:     Fim-se
8:   Fim-para
9:   Se Objeto i não couber em qualquer pacote aberto então
10:    criar novo pacote e colocar objeto i
11:  Fim-se
12: Fim-para
```

Figura 3: Pseudo-código para o First Fit Decreasing

entre 250 e 500 que devem ser alocados em mochilas com capacidade 1000. Estas são naturalmente mais difíceis, pois exigem que uma mochila bem preenchida tenha exatamente um objeto grande (mais de um terço da capacidade) e dois menores (menos de um terço), tendendo a gerar soluções sub-ótimas. Cada instância possui uma quantidade de itens, que pode variar entre 60 e 1000. As soluções ótimas (LB) para cada uma das instâncias escolhidas constam na literatura [Falkenauer, 1996].

Inicialmente, utilizou-se o solver Gurobi 9.1.0, com tempo restrito a 300 segundos para obter uma referência de performance para o problema. Foi dada como solução inicial o resultado obtido pela heurística (com Z_0). Dentro dessa limitação, o solver chega a uma solução com Z_f. Como parâmetro de medida, calculou-se o gap da solução final pela equação

$$\text{Gap}(\%) = \frac{Z_f - \text{LB}}{\text{LB}} \cdot 100 \tag{6}$$

Os resultados computacionais foram reunidos na Tabela 1.

Tabela 1: FFD+Gurobi com restrição de tempo.

Classe	Id.	Qtd.	LB	Z_0 (FFD)	Z_f	Gap (%)
U	1	120	48	49	48	0
	2	120	50	**50**	50	0
	3	250	99	100	**100**	1,00
	4	250	101	102	**102**	0,99
	5	500	198	201	200	1,11
	6	500	206	209	208	0,96
	7	1000	399	403	**403**	**1,00**
	8	1000	397	402	**402**	**1,26**
T	1	60	20	23	20	0
	2	60	20	24	20	0
	3	120	40	45	41	2,50
	4	120	40	46	41	2,50
	5	249	83	94	85	2,41
	6	249	83	95	85	2,41
	7	501	167	190	170	**1,80**
	8	501	167	191	169	**1,20**

Em seguida, utilizou-se a meta-heurística SA, também a partir da solução inicial obtida pela heurística. Para cada instância, o algoritmo foi executado 3 vezes. A partir dos resultados registrados, determinou-se a melhor solução obtida (Z_b), bem como o gap e o tempo de execução médios (média aritmética entre as três execuções), por se tratar de um algoritmo não determinístico.

Para a aplicação deste procedimento, definiu-se os parâmetros iniciais do SA a partir de ensaios anteriores para problemas de empacotamento [Dowsland, 1993]. Ainda, os parâmetros foram ajustados para melhorar a performance especificamente nas instâncias utilizadas, com base em algumas execuções anteriores. Dessa forma, foram estabelecidos: temperatura inicial de 100, temperatura final de 0.1, número máximo de iterações igual a 100 e taxa de resfriamento igual a 0.95. Analogamente ao caso anterior, os resultados constam na Tabela 2.

Tabela 2: FFD+SA.

Classe	Id.	Qtd.	LB	Z_b	Gap (%)	Tempo (s)
U	1	120	48	**48**	0	24,3
	2	120	50	**50**	0	23,5
	3	250	99	**99**	0,67	97,6
	4	250	101	**101**	0,33	101,2
	5	500	198	199	0,50	396,7
	6	500	206	**206**	0,16	394,8
	7	1000	399	400	**0,50**	1565,6
	8	1000	397	399	**0,67**	1622,7
T	1	60	20	21	5,00	5,8
	2	60	20	21	5,00	6,0
	3	120	40	41	2,50	19,2
	4	120	40	41	2,50	19,3
	5	249	83	84	1,20	82,2
	6	249	83	84	1,20	78,7
	7	501	167	168	**0,60**	314,6
	8	501	167	168	**0,60**	305,7

A partir das tabelas apresentadas, alguns pontos merecem destaque. Pela Tabela 1, pode-se notar que a heurística FFD já apresenta resultados muito bons para as instâncias da classe U. Em alguns casos, para essa classe, o solver não conseguiu encontrar uma solução melhor do que a da heurística dentro do limite de tempo. Para a classe T, naturalmente, o gap das soluções finais foi maior. A solução ótima foi alcançada apenas para instâncias menores. Pela Tabela 2, pode-se notar que a meta-heurística alcançou soluções com gap relativamente baixo. Em metade dos casos da classe U, a melhor solução alcançada foi a solução ótima. Para os casos da classe T, a solução alcançada foi sempre pior do que a solução ótima por exatamente uma unidade. Em geral, o tempo de execução foi baixo, com exceção para as duas maiores instâncias da classe U.

Comparando as duas tabelas, nota-se que o gap médio para o SA foi menor em geral. Em alguns casos, a solução foi significativamente melhor, com o gap reduzido em mais da metade. Para as instâncias Id. 6 de cada classe, as soluções foram monitoradas durante a execução dos algoritmos (conforme a melhor solução encontrada era atualizada) e os resultados podem ser observados nas figuras 4 e 5. Nessas instâncias, o SA mostrou-se competitivo com o Gurobi. Pode-se apontar dentre os fatores para tal desempenho a baixa complexidade computacional da meta-heurística. Vê-se que a lógica de se obter uma boa solução inicial por uma heurística gulosa e então refiná-la pela meta-heurística produz ótimos resultados. Portanto, dentro do contexto de tempo e recursos limitados, o FFD+SA mostra-se como uma excelente alternativa para resolução do problema apresentado.

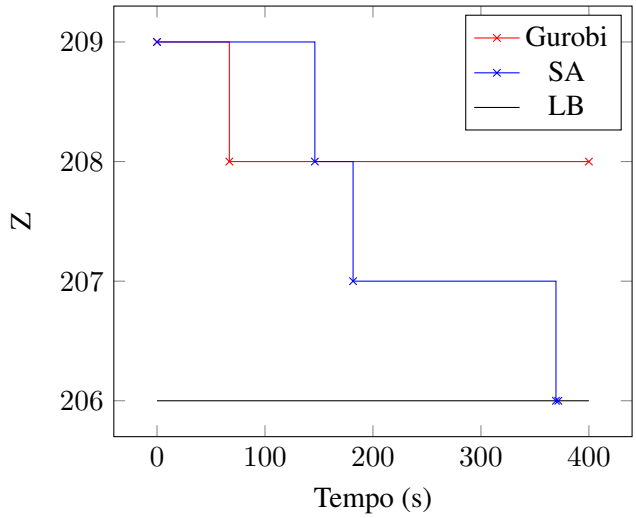

Figura 4: Classe U, Id. 6

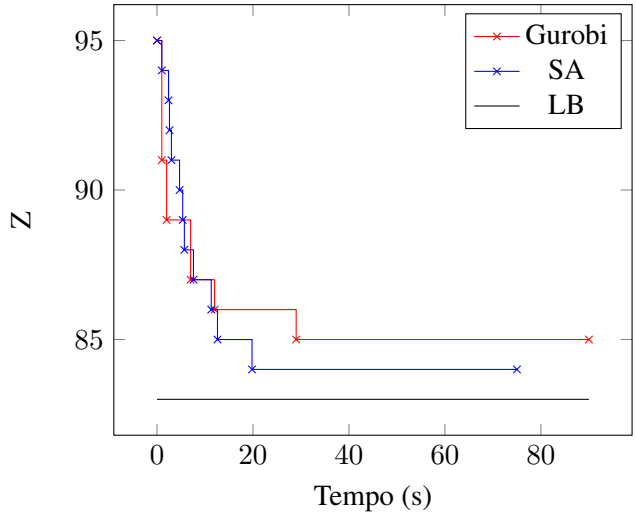

Figura 5: Classe T, Id. 6

5. Conclusão

Nesse trabalho, utilizou-se o Solver Gurobi como referência para a resolução do Problema do Empacotamento Unidimensional via Simulated Annealing. A definição dos parâmetros da meta-heurística foram baseados na literatura vigente e, para sua execução otimizada, foi necessário a alteração da função objetivo, tendo por fim distinguir diferentes soluções que ocupam o mesmo número de caixas e, consequentemente, distinguir suas probabilidades, tornando, assim, o espaço de soluções mais adequado para o algoritmo.

Foi possível averiguar uma melhor performance do algoritmo em relação ao solver tanto para instâncias de classe U quanto de classe T, quando este era limitado a um tempo de execução de 300 segundos. O método heurístico, além de baixa complexidade de implementação, apresentou baixo tempo de execução e boa performance, demonstrando-se competitivo quanto ao solver e promissor para problemas mais complexos, como *Bin Packing* na forma bidimensional e tridimensional.

Referências

Delorme, M., Martello, S., e Iori, M. (2018). Bpplib: A library for bin packing and cutting stock problems. `http://or.dei.unibo.it/library/bpplib`. Acessado: 2020-11-20.

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

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30.

Henderson, D., Jacobson, S., e Johnson, A. (2003). The theory and practice of simulated annealing. In Springer, B., editor, *Glover F., Kochenberger G.A. (eds) Handbook of Metaheuristics. International Series in Operations Research Management Science*.

Kirkpatrick, S., Gelatt, C. D., e Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 32.

Lima, J., Silva, P., Santos, W., e Clímaco, G. Simulatd annealing aplicado ao problema do empacotamento unidimensional. Technical report, Universidade Gederal do Maranhão.

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

Sonuc, E., Sen, B., e Bayir, S. (2017). Solving bin packing problem using simulated annealing. In $65^{th} ISERD\ International\ Conference, Mecca,\ Arábia\ Saudita.$

Ugur e Tursel, D. (2009). Applications of bin packing models through the supply chain. *International Journal of Business and Management*, 1.