IMPLEMENTAÇÃO DE UM ALGORITMO HÍBRIDO PARA A SOLUÇÃO DO PROBLEMA *BIN-PACKING*

Marcone Jamilson Freitas Souza
Departamento de Computação, Universidade Federal de Ouro Preto, Campus Universitário, CEP 35400-000, Ouro Preto, MG, Brasil,
e-mail: marcone@iceb.ufop.br

Letícia Oliveira Rezende
Graduanda em Ciência da Computação, Universidade Federal de Ouro Preto,
e-mail: lehhrezende@hotmail.com

Philipe Lemos Parreira
Graduando em Ciência da Computação, Universidade Federal de Ouro Preto,
e-mail: philipelemos@hotmail.com

Victor Augusto Alves Coelho
Graduando em Ciência da Computação, Universidade Federal de Ouro Preto,
e-mail: victoracoelho@hotmail.com

Resumo:

Este trabalho apresenta o problema do *bin-packing* em relação aos navios de carga. Isto é, abordando a necessidade da distribuição das cargas com menor custo. Com isso, o principal objetivo é utilizar o menor número possível de contêineres para o transporte. Esse problema é solucionado através dos seguintes procedimentos heurísticos escolhidos: a utilização da heurística construtiva conhecida como Solução Gulosa para a geração da solução inicial, o uso da heurística de refinamento denominada Método da Descida (*Best Improvement Method - BI*) e, por último a metaheurística *Iterated Local Search (ILS)* para encontrar a ótima solução.

Palavras-chaves: *Bin-Packing*, Metaheurísticas, *Iterated Local Search.*

Abstract:

This work shows the problem of bin-packing in relation to cargo ships. That is, by addressing the need to distribute the loads with lower cost. With this, the main objective is to use the smallest number of containers for the transport. This problem is solved through the following chosen heuristic procedures: the utilization of the constructive heuristic known as Greedy Solution for the generation of the initial solution, the use of the refinement heuristic denominated Descent Method or Best Improvement Method (BI) and, finally the metaheuristics Iterated Local Search (ILS) to find the optimal solution.

Keywords: Bin-Packing, Metaheuristics, Iterated Local Search.

1. Introdução

Sabe-se que o problema do *bin-packing* (BP) ou *bin-packing problem* pode ser apresentado por diversas formas, como por exemplo, de maneira bidimensional. Porém, foi escolhido o *bin-packing* unidimensional através do empacotamento por peso (*packing by weight*) para a atividade, pois aparenta ser resolvido com maior facilidade.

Dada uma certa quantidade de objetos com diferentes pesos, o *bin-packing problem* consiste em agrupa-los de modo a utilizar o menor número possível de contêineres de certa capacidade suportada para armazená-los, uma vez que é normal, em alguns transportes de carga, os contêineres estarem com sua capacidade máxima não sendo alcançada.

Além disso, como o *bin-packing* é considerado um problema NP-difícil, então ele deve ser resolvido através de heurísticas a afim de obter uma solução aceitável computacionalmente. As heurísticas escolhidas para este trabalho foram: *Iterated Local Search* (ILS), Método da Descida (*Best Improvement Method - BI*) para realizar a busca local e a Heurística de Construção Gulosa para gerar a solução inicial.

Este trabalho foi dividido em 5 seções. Na seção 2 é descrito o problema, assim como a sua representação. Na seção posterior, é mostrado como os métodos são desenvolvidos e também a geração de cada solução e o algoritmo desenvolvido. Na seção 4 estão os resultados obtidos computacionalmente. E, finalmente, na seção 5, a conclusão do trabalho é apresentada.

2. Descrição do problema

O problema é composto por itens, contêineres e pesos. Os itens representam os objetos a serem colocados nos contêineres para serem transportados. Já os pesos representam quantos quilos possuem cada item, sendo necessário para verificar se cabem ou não nos contêineres.

A representação inicial do problema é composta por itens colocados em contêineres de acordo com seus pesos para que depois sejam reacomodados.

A solução final é um vetor de itens com o número do contêiner onde estarão e o número de contêineres que serão utilizados.

3. Modelagem

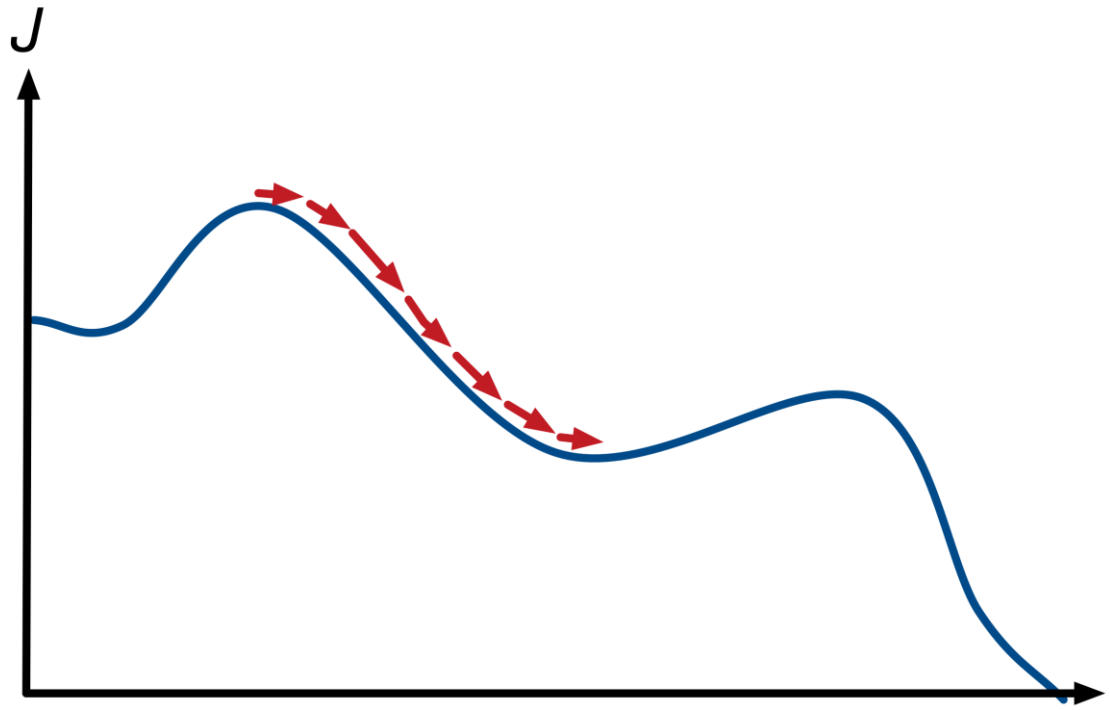
Nos próximas seções, serão mostrados os métodos utilizados para chegar a solução do problema do *Bin-Packing*.

3.1. Estrutura de vizinhança

Seja S o conjunto de soluções do problema. Um vizinho de $s \in S$ é criado causando perturbações ao efetuar algumas trocas entre dois itens (movimento em que se troca o contêiner de um item pelo contêiner do outro) e realizando a realocação de um item (movendo-o de um contêiner para o outro) através da busca local, analisando todas realocações possíveis. O melhor movimento é realizado.

3.2. Busca Local – *Descent Method*

O *Descent Method* (Método da Descida) é um método que realiza a busca local a partir de uma solução inicial qualquer analisando todos os seus possíveis vizinhos e realizando apenas o movimento que representa uma melhora (redução) no valor atual da função objetivo, alcançando um ótimo local. A figura abaixo exemplifica o seu funcionamento.



3.3. *Iterated Local Search* (ILS)

Após gerar uma solução inicial através do método guloso, o Iterated Local Search (Lourenço, 1995) aplica uma busca local sobre essa solução buscando melhorá-la e, em seguida, realiza perturbações, de modo que sejam fortes o bastante para explorar outras soluções aplicando a busca local sobre elas e, ao mesmo tempo, fracas o suficiente para a exploração aleatória, para evitar encontrar a mesma solução. Através dessas perturbações, o algoritmo procura sair de um ótimo local para achar a melhor solução. A seguir, na figura abaixo, é apresentado o pseudocódigo.

```
Procedimento ILS(vezesSemMelhora ;NivelsemMelhora; n; p)
    Fo ← descida(n; p)
    Iter ← 0
    MelhorIter ← 0
    Enquanto(iter-MelhorIter < NivelsemMelhora) faça {loop aonde tem a perturbação}
        iter ← iter + 1 vezes ← 0
        enquanto(vezes < vezesSemMelhora) faça
            Fo' ← perturbação(n; p; NivelsemMelhora)
            Fo' ← descida(n; p)
            Se(Fo' < Fo)
                Fo ← Fo'
                MelhorIter = iter
            fim se
            vezes ← vezes + 1
        fim se enquanto
    Fim se enquanto
Retorna Fo
```

4. Função objetivo

Para avaliar as soluções geradas, utiliza-se uma função objetivo *f* que é aplicada sobre uma solução. Como não foram aceitas soluções ruins, ou seja, soluções onde existe algum contêiner que teve sua capacidade máxima excedida, o resultado de *f* é o somatório de todos os contêineres que estão ocupados.

Caso aceitasse soluções ruins, o resultado de *f* seria o somatório de todos os contêineres usados e mais uma penalidade que equivale ao número total de contêineres multiplicado pelo excesso.

5. Resultados computacionais

O algoritmo proposto foi implementado em C++. Todos os experimentos foram realizados em um computador com processador Intel Core i5-5200U de 2.20 GHz, 8 GB de RAM e Windows 10.

A tabela a seguir mostra os resultados encontrados em quatro testes. Nela encontram-se: a quantidade de itens (N° de itens), o melhor resultado já registrado em artigos (Melhor Fo), o melhor resultado obtido nessa atividade (Fo encontrado), o tempo de execução para um teste em cada caso (Tempo(s)) e a porcentagem de piora em relação ao melhor resultado já registrado (Gap).

N° de itens	Melhor Fo	Fo encontrado	Tempo(s)	Gap
60	20	20	2.42	0%
120	40	40	8.91	0%
249	83	85	34.84	2,41%
501	167	183	1214.54	9.58%

6. Conclusões

Neste trabalho foi utilizando a metaheurística ILS para a resolução do Problema do *Bin-Packing*, A ideia foi construir uma solução gulosa inicialmente e que usasse a metaheurística ILS e a técnica do Método da Descida para a busca local.

Com base nos resultados obtidos, pode-se concluir que, apesar da nossa implementação do ILS obter bons resultados, não foram obtido resultados páreos aos que foram obtidos na literatura. Além disso, percebe-se que o tempo de execução do programa aumenta exponencialmente para um número maior de itens.

Referências

SOUZA, Fernanda S.H.; BRAGA, Thiago H.; COELHO, Viviane S. **Aplicação do Simulated Annealing ao Bin Packing Problem**

Bin Packing.
Disponível <https://www2.wiwi.uni-jena.de/Entscheidung/binpp/index.htm> em: Acesso em: 7 fevereiro. 2018.

OR-Library.
Disponível <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.htm> em: Acesso em: 2 fevereiro. 2018.

SOUZA, Marcone J. F., **Notas de aula da disciplina Inteligência Computacional para Otimização**

Imagem referente ao Gradient Descent
Disponível <http://www.deepideas.net/deep-learning-from-scratch-iv-gradient-descent-and-backpropagation/> em: Acesso em: 2 fevereiro. 2018.