Uma heurística GRASP para resolução do problema de carregamento e descarregamento de contêineres em navios porta-contêineres

Amaro José de Souza Neto
amaro.neto@gmail.com

Dalessandro Soares Vianna
dalessandro@pq.cnpq.br

Marcilene de Fátima Dianin Vianna
marcilenedianin@vm.uff.br

Erenildo da Silva Rios
erenildo@gmail.com

RESUMO

Neste artigo foi proposta uma heurística GRASP para resolução do problema de carregamento e descarregamento de contêineres em navios porta-contêineres. Esses navios possuem espaços celulares onde os contêineres são empilhados. Ao atracar em um terminal portuário, pode ser necessário realizar diversas operações de carregamento e descarregamento. Algumas vezes, na operação de descarregamento de um contêiner que necessita ser desembarcado, este pode estar posicionado abaixo de outros contêineres que não serão descarregados nesse momento. Os contêineres que estão acima precisarão ser removidos para descarregar os de baixo, e essa operação se chama remanejamento. Este trabalho aborda o problema carregamento e descarregamento de contêineres em navios porta-contêineres, tendo como objetivo encontrar a melhor sequência de carregamento, minimizando, dessa forma, o número de remanejos. A heurística proposta foi comparada com uma heurística gulosa e um método de busca local, e os resultados indicaram adequação da heurística GRASP ao problema abordado.

***Palavras-chave**: Otimização, Carregamento, Contêineres, Meta-heurísticas, GRASP.*

A GRASP heuristic for solving the problem of loading and unloading of containers on container ship

ABSTRACT

In this paper we propose a GRASP heuristic for solving the problem of loading and unloading of containers on container ships. These ships have cellular spaces where the containers are stacked. When a ship docks at a port terminal, it may be necessary to perform several loading and unloading operations. Sometimes, in the unloading operation of a container which needs to be unloaded, it may be located below other containers that will not be unloaded at that moment. These containers which are on top must be "relocated" to unload the required container. This paper addresses the problem of loading and unloading containers on containerships, aiming to find the best loading sequence, minimizing, thus, the number of relocations. The proposed heuristics was compared with greedy heuristics and a local search method. The results show the suitability of the GRASP heuristic to the problem addressed.

***Key words**: Optimization; Stowage; Containers; Metaheuristics; GRASP.*

1. Introdução

Os navios porta-contêineres são embarcações especializadas em transporte de carga conteinerizada. Essas embarcações dispõem de espaços celulares (baias) onde os contêineres são empilhados (Figura 1). A movimentação da carga ocorre tanto nas baias quanto no convés do navio, através de equipamentos de bordo ou de terra (GÓES, 2002). Devido à estrutura do navio e à forma que a carga deve estar disposta, o acesso aos contêineres é feito pelo topo da pilha. Pode ser necessário movimentar alguns contêineres para descarregar outros que estão numa posição inferior. Essa operação é denominada *remanejo*.

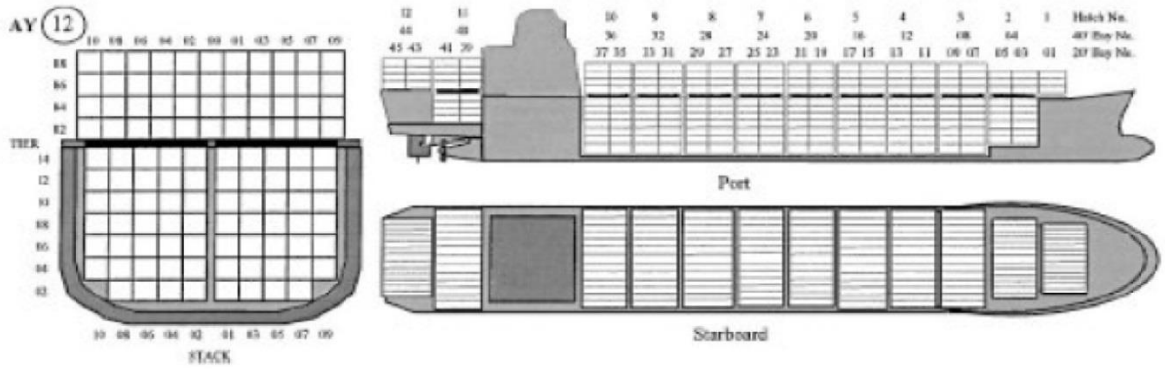

Figura 1 - Estrutura de um navio

Fonte: WILSON; ROACH, 2000.

Terminais Concentradores são portos que têm a finalidade de atender à concentração de cargas (conteinerizada) de toda uma região para posterior distribuição para outros portos (BERTOLANI; LEME, 2004). Sua eficiência está relacionada à ordenação e forma de lidar com os contêineres (SOBRAL et al., 2009). Um navio porta-contêineres pode atracar em terminais concentradores para realizar diversas operações de carga e descarga de contêineres. Há dois critérios fundamentais no problema de planejamento de estiva: a minimização do número de remanejos e as restrições de estabilidade.

Neste artigo, o foco é a embarcação porta-contêineres, sendo também discutida a posição onde as cargas são armazenadas no navio, de modo que em cada porto, mesmo necessitando realizar operações de carregamento e descarregamento de contêineres, o número de remanejos seja o menor possível. De acordo com Avriel et al. (2000), esse problema é NP-Completo.

O objetivo deste trabalho foi desenvolver uma heurística GRASP para o problema de carregamento e descarregamento de contêineres em um navio porta-contêineres, visando minimizar o número de remanejos.

Na literatura podem ser encontrados trabalhos que abordam problemas semelhantes ao tratado neste estudo, entre os quais os de:

- Sobral et al. (2009), que desenvolveram um algoritmo Beam Search para resolução do problema de carregamento e descarregamento de contêineres em terminais portuários.
- Campos (2008), que elaborou um estudo sobre a integração entre carregamento e roteamento de veículos.
- Morabito e Arenales (1997), que mostraram diferentes abordagens para o problema de carregamento de contêineres.
- Raidl (1999), que propôs um algoritmo genético para o problema de empacotamento de múltiplos contêineres.
- Martins et al. (2009), que estudaram o problema de estocagem de contêineres.
- Azevedo (2009), que propôs um algoritmo genético para resolver o problema de carregamento e descarregamento de contêineres em terminais portuários.

O restante do trabalho está organizado da seguinte maneira: na Seção 2 é apresentado o problema a ser resolvido. Na Seção 3 é mostrada a representação da demanda de contêineres, da baia e

da solução, além da função de avaliação. Na Seção 4 é apresentada a meta-heurística GRASP desenvolvida. Na Seção 5 são apresentados os resultados computacionais. Na Seção 6 vêm as conclusões.

2. Apresentação do problema

Os navios porta-contêineres são embarcações de grande porte e possuem uma estrutura celular que facilita a manipulação da carga armazenada em contêineres. As células são agrupadas por seções (baias) onde contêineres podem ser empilhados. Possui uma rota $R = \{p_0, p_1, p_2, ..., p_n\}$ que é conhecida antes de sua partida. Geralmente são circulares, partindo de um Porto p_0, percorrendo todos os Portos p_i, descarregando as cargas nos portos de destino e carregando novas cargas destinadas a outros portos e, no fim, retorna ao Porto p_0. Cada um dos portos p que compõem R tem uma demanda de contêineres que necessitarão ser carregados na embarcação. Além disso, o navio possui uma capacidade que é medida em TEU (*Twenty-Foot Equivalent Units*). Por exemplo, uma embarcação com capacidade para 1.000 TEUs pode armazenar 1.000 contêineres de 20 pés.

Ao atracar em um Porto p_i, o navio primeiro efetuará o descarregamento de contêineres destinados a p_i, caso haja. Nessa operação pode ser necessário descarregar temporariamente os contêineres que estão armazenados numa pilha de determinada baia do navio, para conseguir descarregar um contêiner que está localizado em uma posição inferior dessa pilha. Posteriormente, caso haja demanda, os contêineres serão carregados do terminal para o navio, e isso implica uma sequência de carregamento $S = \{c_0, c_1, c_2, ..., c_n\}$ em cada porto que o navio irá atracar, onde deve ser buscado um melhor arranjo dessas sequências, de forma a minimizar o número de remanejos para os portos seguintes.

3. Representação

Nesta seção, discute-se a representação da demanda de contêineres, da baia, da solução e, também, da função de avaliação.

3.1. Demanda de contêineres

A representação da demanda de contêineres será através de uma matriz de transporte T_{od}, em que $o = 1, 2, ..., N$ e $d = 1, 2, ..., N$ (N é o número de portos). Considere o sendo o porto de origem e d, o porto de destino. O número na posição T_{od} corresponde à quantidade de contêineres que deverão ser carregados no Porto o para posteriormente serem descarregados no Porto d, ou seja, as demandas correspondentes. Na Figura 2 é mostrado um exemplo de uma matriz de transporte. Por exemplo, quando o navio atracar no Porto 2 será necessário carregar $T_{21} = 5$ contêineres que têm como destino o Porto 1 e $T_{24} = 4$ contêineres com destino ao Porto 4.

ORIGEM \ DESTINO	1	2	3	4	5
1	0	2	4	2	4
2	5	0	0	4	0
3	0	2	0	2	6
4	6	0	0	0	0
5	0	6	0	4	0

Figura 2 - Exemplo de uma matriz de transporte de um navio

3.2. Baia

Uma baia será representada computacionalmente por uma matriz de ocupação O_{lc}, em que $l = 1, 2, ..., L$ e $c = 1, 2, ..., H$ (em que L e H significam quantos contêineres cabem na largura e altura do navio, respectivamente). Nela serão registradas suas posições (l,c) livres e ocupadas. Em cada célula O_{lc} ocupada, o valor inteiro corresponde ao destino em que a carga deverá ser entregue. $O_{lc} = 0$ significa que a posição (l,c) da matriz está livre. A simplicidade dessa representação facilita a compreensão do armazenamento, assim como a manipulação dos itens. Veja no exemplo da Figura 3 que o valor na posição (1,1) = 0 significa que essa posição está livre e pronta para receber alguma carga. Enquanto a posição (3,2) = 3 corresponde ao porto no qual o contêiner da posição (3,2) deverá ser descarregado, no caso o Porto 3.

	1	2	3
1	0	5	0
2	0	3	3
3	2	3	2

Figura 3 - Exemplo da representação da baia de um navio com capacidade para nove contêineres

Quando ocorre a atracação do navio em um porto acontecem operações de carregamento e descarregamento de contêineres. Devido a essas operações, as informações nessa matriz de ocupação

são atualizadas constantemente. A cada porto atracado será necessário pelo menos uma dessas operações; caso contrário, a atracação não ocorreria.

3.3. Solução

Neste trabalho, uma solução é representada por uma lista, com o tamanho determinado pelo número de portos. Cada posição *i* da lista armazena um vetor que possui as demandas que devem ser embarcadas no Porto *i*. Cada vetor é uma sequência de carregamento, em que o primeiro item será o primeiro a ser armazenado no navio, e assim sucessivamente. A Figura 4 ilustra uma solução para um problema com cinco portos. A demanda considerada será a mesma da Figura 2. Suponha a rota de visitação do navio: 1-3-5-2-4-1. Cada posição do vetor corresponde a um contêiner, e o número contido em cada uma dessas posições se refere ao porto que o contêiner deve ser descarregado. Por exemplo, quando a embarcação atracar no Porto 5, de acordo com a respectiva sequência, primeiramente serão carregados no navio quatro contêineres com destino ao Porto 4 e, em seguida, os seis contêineres destinados ao Porto 2.

PORTO DE DESTINO

PORTO ORIGEM												
1	4	4	2	2	5	5	5	5	3	3	3	3
2	1	1	1	1	1	4	4	4	4			
3	4	4	2	2	5	5	5	5	5	5		
4	1	1	1	1	1	1						
5	4	4	4	4	2	2	2	2	2	2		

Figura 4 - Representação de uma solução

3.4. Função de avaliação

O objetivo principal do problema proposto é minimizar o número de remanejos. Para isso, a função que irá avaliar o problema terá como entrada a lista de sequências de carregamento (solução). Ao término será retornado o número total de remanejos efetuados, através da simulação do cumprimento de determinado trajeto, realizando descarregamentos e carregamentos, quando necessários, de acordo com as respectivas sequências de carregamento. Um exemplo dessa função é mostrado na Tabela 1, a qual utiliza como dado de entrada a solução descrita na Figura 4 e possui duas baias de tamanho 3x3 (*L* x *H*, em que *L* e *H* significam quantos contêineres cabem na largura e altura do navio, respectivamente). Por exemplo, na iteração 1, de acordo com a ordem de visitação, o navio atracará no Porto 1 e estará vazio. Como não há contêineres a serem descarregados, é feito o carregamento de dois contêineres com destino para o Porto 4, dois com destino para o Porto 2, quatro para o Porto 5 e quatro para o Porto 3. Na iteração 2, o navio está parcialmente ocupado e atracado no Porto 3. É feito o descarregamento de quatro contêineres que tinham como destino o Porto 3, e em seguida realiza-se o carregamento. Na iteração 3, o navio está totalmente ocupado e atracado no Porto 5. Será feito o descarregamento de contêineres com destino ao Porto 5, mas como pôde ser observado, na baia 2 existem contêineres que precisam ser descarregados e estão no fundo da baia e com contêineres sobre eles. Será necessário descarregar esses contêineres numa pilha auxiliar para, então, fazer a descarga dos contêineres que têm como destino o Porto 5. Computa-se o número de remanejos *rem* = 2, e então se inicia o processo de carregamento, e o navio fica totalmente ocupado. Esse processo é repetido ao longo de todo o trajeto de visitação.

Tabela 1 - Execução da função de avaliação

Iteração	Ordem visitação	Situação Navio	Descarregar contêineres	Carregar contêineres	Total remanejos
1	1	0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0	0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0	0 0 0 | 0 0 0 5 5 3 | 3 3 3 4 4 2 | 2 5 5	0
2	3	0 0 0 | 0 0 0 5 5 3 | 3 3 3 4 4 2 | 2 5 5	0 0 0 | 0 0 0 5 5 0 | 0 0 0 4 4 2 | 2 5 5	5 5 5 | 5 5 5 5 5 2 | 4 4 2 4 4 2 | 2 5 5	0
3	5	5 5 5 | 5 5 5 5 5 2 | 4 4 2 4 4 2 | 2 5 5	0 0 0 | 0 0 0 0 0 2 | 4 0 0 4 4 2 | 2 4 2	2 2 2 | 2 2 2 4 4 2 | 4 4 4 4 4 2 | 2 4 2	2
4	2	2 2 2 | 2 2 2 4 4 2 | 4 4 4 4 4 2 | 2 4 2	0 0 0 | 0 0 0 4 4 0 | 0 4 0 4 4 0 | 4 4 4	1 4 4 | 4 4 0 4 4 1 | 1 4 1 4 4 1 | 4 4 4	4
5	4	1 4 4 | 4 4 0 4 4 1 | 1 4 1 4 4 1 | 4 4 4	0 0 0 | 0 0 0 0 0 1 | 0 0 0 1 0 1 | 1 1 0	0 0 0 | 0 0 0 1 1 1 | 1 1 0 1 1 1 | 1 1 1	7
6	1 . . .	0 0 0 | 0 0 0 1 1 1 | 1 1 0 1 1 1 | 1 1 1	0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0	0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0	7

4. Heurística GRASP proposta

A meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*) é um processo iterativo de múltiplos inícios propostos por Feo e Resende (1995). Consiste, basicamente, em duas fases: construção e melhoria. A fase de construção cria uma solução viável iterativamente, elemento por elemento. Em seguida, na fase de melhoria é aplicada uma busca local, na qual se pode refinar a solução inicial através de uma busca em sua vizinhança. Ao fim dessa fase é obtida uma solução ótima local, devido ao fato de esta se destacar em relação à solução inicial e suas soluções vizinhas. A melhor solução encontrada ao longo de todas as iterações GRASP é retornada como resultado.

O sucesso do uso da meta-heurística GRASP em problemas de otimização é descrito em diversos trabalhos da literatura, entre os quais se podem citar os de Nogueira et al. (2006), Júnior et al. (2009) e Mota e Ochi (2009).

Nas próximas subseções serão apresentadas as etapas da heurística proposta. As subseções 4.1 e 4.2 apresentam as fases de construção de uma solução e de busca local, respectivamente.

4.1. Fase de construção

Esta fase tem como objetivo prover uma solução factível e de boa qualidade para a fase de busca local. A solução será construída iterativamente, elemento por elemento. Para cada porto são selecionados todos os contêineres que devem ser carregados. Em seguida, são ordenados numa lista, de forma que os contêineres destinados ao porto mais distante no trajeto de visitação ficarão como primeiros na sequência (critério guloso).

Na Figura 5 é mostrado o algoritmo de construção de uma solução. O laço que vai da linha 1 até 9 garante que será montada uma sequência de

carregamento para cada porto. O laço que vai da linha 3 até 8 garante a montagem da sequência na ordem inversa à de visitação, ou seja, os contêineres destinados aos portos mais distantes no trajeto de visitação serão colocados como primeiros na lista. O laço que vai da linha 4 até 7 garante que todas as demandas serão colocadas na sequência de carregamento. Na linha 5 é montada a sequência de carregamento de cada porto para então, na linha 10, retornar à solução completa.

Algoritmo constrói solução (*ordem visita*, *matriz-demanda*)
Entrada
Ordem visita – ordem na qual o navio fará os carregamentos e descarregamentos *matriz demanda*- matriz, onde são registradas as demandas de carregamento de cada porto
Saída
Solução – matriz onde são registradas as sequências de carregamento de cada porto
Início
1. Para cada porto *i* faça
2. $s \leftarrow 0$
3. Para cada elemento *j* de *ordem visita* do último para o primeiro faça
4. para cada elemento *k* de *matriz-demanda* [*i*][*j*] faça
5. *Solução* [*i*][*s*] $\leftarrow j$
6. $s \leftarrow s + 1$
7. Fim-para
8. Fim-para
9. Fim-para
10. Retorne *solução*

Fim-constrói solução

Figura 5 - Algoritmo de construção de uma solução

Na Figura 6 é ilustrada a construção de uma solução. Considere que o navio visitará três portos; sua ordem de visitação é 1-3-2-1. Observa-se, nessa figura, que na montagem da ordem de carregamento do Porto 1, primeiro, são selecionadas as cargas que têm como destino o porto que será visitado por último. De acordo com a ordem de visitação, o último é o Porto 1, mas, como a sequência em construção pertence ao Porto 1, não teria sentido esse porto ter demanda de carregamento para ele mesmo. Então, o Porto 2 tem sua demanda selecionada e colocada como os primeiros contêineres que serão carregados na sequência. Em seguida, seleciona-se a demanda do porto que será visitado antes do Porto 2; nesse caso, o Porto 3, que tem sua demanda selecionada e colocada na sequência também. A montagem das sequências para os outros portos segue a mesma lógica.

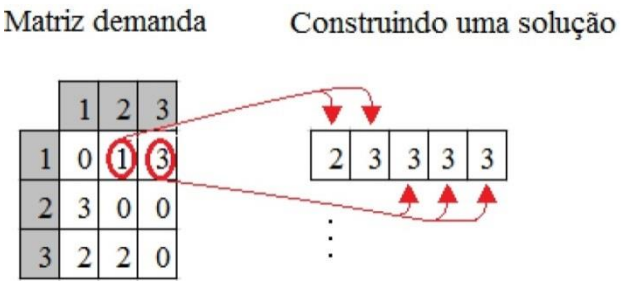

Figura 6 - Montagem de uma solução

Em um algoritmo GRASP é necessário que a fase de construção seja gulosa-aleatorizada. Para isso, em cada porto todos os elementos pertencentes à sequência de carregamento serão considerados candidatos a compor uma nova solução, caso ainda não tenham sido escolhidos e não inviabilizam a solução com sua participação. Então, é constituída a lista de candidatos LC. Em seguida é montada a lista restrita de candidatos LRC, escolhendo-se os α melhores elementos de LC. Em LRC serão realizados sorteios, cujo objetivo é escolher um elemento para compor a nova solução.

4.2. Fase de busca local

O objetivo desta fase é possibilitar melhoria para a solução construída através da busca em uma vizinhança por uma solução de melhor qualidade.

A vizinhança é o conjunto de soluções próximas à solução inicial que pode ser obtida por um movimento, em que o algoritmo de busca local poderá procurar por uma solução melhor. Neste trabalho, o movimento consiste em trocar, nas sequências de carregamento, um contêiner de determinada posição por outro na mesma sequência em posição diferente. Com a troca realizada, a

respectiva sequência é alterada e reavaliada para, assim, obter seu número de remanejos.

A Figura 7 mostra o algoritmo de busca local. O laço que vai da linha 2 até 14 garante que as sequências de cada porto serão contempladas com a troca. O laço que vai da linha 3 até 13 irá percorrer toda a sequência. O laço que vai da linha 4 até 10 fará que se busque na sequência a próxima posição para efetuar a troca. Nas linhas 5 e 6 são realizadas a troca e a avaliação, respectivamente. O bloco delimitado pelas linhas 7 e 11 servem para verificar se a troca gerou uma solução melhor; caso contrário, desfaz-se a troca, e a solução volta ao seu estado anterior.

```
Algoritmo busca local (solução)
Entrada
    Solução -  matriz em que são registradas as sequências de carregamento de cada porto
Início
1.   Remanejo ← ∞
2.   Para cada porto i faça
3.      Para cada elemento j de solução [i] faça
4.         Para cada elemento k, em que k > j de solução [i] faça
5.            Trocar elemento solução [i][j] por solução [i][k]
6.            Remanejo Obtido ← avalia (solução)
7.            Se remanejo Obtido < remanejo então
8.               Remanejos ← remanejo Obtido
9.            Senão
10.              Trocar elemento solução [i][j] por solução [i][k]
11.           Fim-se
12.        Fim-para
13.     Fim-para
14. Fim-para Fim- busca local
```

Figura 7 - Algoritmo de busca local

5. Resultados

Por questões de balanceamento do navio, o atendimento a determinada demanda de carregamento será de acordo com o menor valor entre a disponibilidade da baia selecionada e o bloco máximo de contêineres que pode ser atendido por uma baia. A baia selecionada é a que detém o menor número de contêineres armazenados. Para calcular o bloco máximo, é necessário que seja determinado o número de camadas da baia que pode ser preenchido sem comprometer o balanceamento da embarcação. O bloco é obtido a partir do número de camadas x largura da baia. Neste trabalho, considerou-se que o número de camadas é a metade da altura. Por exemplo, um navio com duas baias que suportam sete contêineres na largura *L* e 18 na altura *H* (*L* e *H* são dados de entrada do algoritmo) teria o número de camadas igual a 9, e o bloco máximo seria 63.

Sobre a carga que será carregada e descarregada não é considerado seu peso e são supostas cargas homogêneas, ou seja, com o mesmo tamanho.

Foram geradas 18 instâncias, considerando-se navios com capacidade de 600 a 2.400 TEUs (*Twenty-Foot Equivalent Units*), com visitação para 10, 15 e 20 portos, totalizando 54. Essas instâncias têm as seguintes informações: número de portos, número de baias do navio, a dimensão (*L* x *H*) das baias, a ordem de visitação do navio e a demanda que o navio deve atender em cada porto. Foi considerado que as demandas dos portos não excedem a capacidade do navio.

Para avaliar o desempenho dos resultados do GRASP proposto, foi usado como parâmetro de comparação o número de remanejos obtidos com o método construtivo puramente guloso (construtivo) e o construtivo puramente guloso seguido da busca local. A lista restrita de candidatos utilizada tem tamanho 3, e o número de iterações do GRASP foi de 20 iterações. Esses parâmetros foram determinados empiricamente.

Os algoritmos foram desenvolvidos na linguagem de programação C e executado em um computador Core 2 Duo 2.0, com 2 GB de memória RAM no sistema operacional Windows XP.

As Tabelas 2, 3 e 4 apresentam os resultados dos algoritmos após uma execução de cada algoritmo. Conforme pode ser observado, o algoritmo GRASP mostrou-se superior ao método puramente guloso em todos os casos testados. Em relação ao construtivo seguido da busca local, o GRASP se mostra inferior somente em um caso, o teste de número 6, da Tabela 4. A média percentual

de melhoria do GRASP em relação à construção encontrada foi de 30%. Houve casos em que a diferença percentual de melhora entre o GRASP e o construtivo seguido da busca local foi superior a 15%, como pode ser visto no teste de número 10, da Tabela 2.

Tabela 2 - Resultados de instâncias com 10 portos

Nº	Baia			TEUs	Construtivo		Busca local			GRASP		
	Q	L	H		Custo	Tempo	Custo	%	Tempo	Custo	%	Tempo
1	12	5	10	600	558	0	470	16%	1,6	413	26%	32,62
2	12	7	10	840	979	0	674	31%	3,73	592	40%	78,86
3	12	10	10	1200	945	0	721	24%	6,89	687	27%	144,39
4	12	5	13	780	1713	0	1290	25%	8,87	1179	31%	173,9
5	12	7	13	1092	835	0	544	35%	3,28	520	38%	65,75
6	12	10	13	1560	1569	0	1299	17%	13,23	1223	22%	247,21
7	12	5	15	900	777	0	638	18%	3,57	553	29%	74,29
8	12	7	15	1260	1024	0	708	31%	7,04	644	37%	142,4
9	12	10	15	1800	1489	0	992	33%	15,51	911	39%	319,64
10	16	5	10	800	809	0	652	19%	3,12	513	37%	61,85
11	16	7	10	1120	1169	0	805	31%	6,68	637	46%	130,9
12	16	10	10	1600	993	0	695	30%	10,62	663	33%	222,79
13	16	5	13	1040	1352	0	1165	14%	6,17	1021	24%	130,81
14	16	7	13	1456	1303	0	985	24%	9,95	865	34%	204,42
15	16	10	13	2080	1665	0	1456	13%	25,48	1367	18%	518,96
16	16	5	15	1200	1130	0	888	21%	7,82	823	27%	156,48
17	16	7	15	1680	1302	0	924	29%	13,5	841	35%	263,96
18	16	10	15	2400	3899	0	3042	22%	40,96	2958	24%	798,5

Tabela 3 - Resultados de instâncias com 15 portos

Nº	Baia			TEUs	Construtivo		Busca local			Grasp		
	Q	L	H		Custo	Tempo	Custo	%	Tempo	Custo	%	Tempo
1	12	5	10	600	558	0	470	16%	1,6	413	26%	32,62
2	12	7	10	840	979	0	674	31%	3,73	592	40%	78,86
3	12	10	10	1200	945	0	721	24%	6,89	687	27%	144,39
4	12	5	13	780	1713	0	1290	25%	8,87	1179	31%	173,9
5	12	7	13	1092	835	0	544	35%	3,28	520	38%	65,75
6	12	10	13	1560	1569	0	1299	17%	13,23	1223	22%	247,21
7	12	5	15	900	777	0	638	18%	3,57	553	29%	74,29
8	12	7	15	1260	1024	0	708	31%	7,04	644	37%	142,4
9	12	10	15	1800	1489	0	992	33%	15,51	911	39%	319,64
10	16	5	10	800	809	0	652	19%	3,12	513	37%	61,85
11	16	7	10	1120	1169	0	805	31%	6,68	637	46%	130,9
12	16	10	10	1600	993	0	695	30%	10,62	663	33%	222,79
13	16	5	13	1040	1352	0	1165	14%	6,17	1021	24%	130,81
14	16	7	13	1456	1303	0	985	24%	9,95	865	34%	204,42
15	16	10	13	2080	1665	0	1456	13%	25,48	1367	18%	518,96
16	16	5	15	1200	1130	0	888	21%	7,82	823	27%	156,48
17	16	7	15	1680	1302	0	924	29%	13,5	841	35%	263,96
18	16	10	15	2400	3899	0	3042	22%	40,96	2958	24%	798,5

Tabela 4 - Resultados de instâncias com 20 portos

Nº	Baia			TEUs	Construção		Busca local			Grasp		
	Q	L	H		Custo	Tempo	Custo	%	Tempo	Custo	%	Tempo
1	12	5	10	600	1389	0	983	29%	6,98	7,069	45%	136,9
2	12	7	10	840	1863	0	1346	28%	14,65	1207	35%	295,43
3	12	10	10	1200	3044	0	2263	26%	28,98	1817	40%	569,28
4	12	5	13	780	1984	0	1444	27%	12,54	1403	29%	253,84
5	12	7	13	1092	2490	0	1903	24%	24,71	1787	28%	490,18
6	12	10	13	1560	3293	0	2335	29%	42,95	2407	27%	900,84
7	12	5	15	900	1974	0	1618	18%	17,32	1579	20%	356,87
8	12	7	15	1260	3628	0	2860	21%	38,78	2647	27%	729,06
9	12	10	15	1800	3140	0	2380	24%	59,56	2228	29%	1179,61
10	16	5	10	800	1098	0	802	27%	11,26	766	30%	223,9
11	16	7	10	1120	2250	0	1668	26%	23,07	1557	31%	483,92
12	16	10	10	1600	2333	0	1596	32%	39,67	1583	32%	818,79
13	16	5	13	1040	2171	0	1600	26%	22,81	1565	28%	436,32
14	16	7	13	1456	3471	0	2758	21%	45,59	2637	24%	926,37
15	16	10	13	2080	5940	0	4551	23%	98,82	4289	28%	2038,29
16	16	5	15	1200	3187	0	2457	23%	29,31	2402	25%	634,2
17	16	7	15	1680	2487	0	1904	23%	47,46	1870	25%	959,34
18	16	10	15	2400	5740	0	4364	24%	128,78	4244	26%	2654,37

6. Conclusões

Foi implementada uma heurística GRASP para resolução do problema de carregamento e descarregamento de contêineres em navios, em que é minimizado o número de remanejos. Foram testadas 54 instâncias, a partir das quais é possível notar considerável melhora da solução apresentada pelo método construtivo. Além disso, com poucas iterações do GRASP e em tempos adequados ao problema, o algoritmo se mostrou eficiente sobre o método construtivo puramente guloso, seguido de uma busca local.

7. Agradecimentos

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), pelo financiamento deste trabalho.

8. Referências

AVRIEL, M.; PENN, M.; SHPIRER, N. Container ship stowage problem: complexity and connection to the coloring of circle graphs. **Discrete Applied Mathematics**, v. 103, p. 271-279, 2000.

AZEVEDO, A. T.; SOBRAL, C. M.; DEUS, N. M. R. Resolução do problema de carregamento e descarregamento de contêineres em terminais portuários via Algoritmo genético. In: SIMPEP, 16., 2009. **Resumos...** [S.l. : s.n.t.], 2009.

BERTOLANI, A. D.; LEME, F. L. **Carregamento de contêineres em navios**. [S.l.]: Universidade Presbiteriana Mackenzie, 2004.

CAMPOS, D. S. **Integração dos problemas de carregamento e roteamento de veículos com janela de tempo e frota heterogênea**. 2008. 119 f. Tese (Doutorado em Engenharia de Produção) - Universidade de São Paulo, São Paulo, 2008.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, p. 109-133, 1995.

GÓES, H. A. **Planejamento portuário**. Rio de Janeiro: Escola de Engenharia/Universidade Federal do Rio de Janeiro, 2002.

JÚNIOR, L. O. J.; ARROYO, J. E. C.; SOUZA, V. A. A. Heurísticas GRASP e ILS para o problema no-wait flowshop scheduling multiobjetivo. In: SBPO, 42., 2009. **Resumos…** [S.l. : s.n.t.], 2009.

MARTINS, P. T.; LOBO, V. J. A. S.; VAIRINHOS, V. Container stowage problem solution for short sea shipping. In: CONGRESSO DA APDIO, 14., 2009. **Proceedings...** [S.l. : s.n.t.], 2009.

MORABITO, R.; ARENALES, M. Abordagens para o problema do carregamento de contêineres. **Pesquisa Operacional**, v. 17, n. 1, p. 29-56, 1997.

MOTA, L. C. S.; OCHI, L. S. Meta-heurísticas com memória adaptativa para o problema de recobrimento de rotas. In: CONGRESSO BRASILEIRO DE REDES NEURAIS, 9., 2009. **Resumos...** [S.l. : s.n.t.], 2009.

NOGUEIRA, R. T.; JR, G. G. P.; PÓVOA, C. L. R. Uma heurística GRASP para o problema do pequeno investidor. In: SIMPEP, 13., 2006. **Resumos…** [S.l. : s.n.t.], 2006.

RAIDL, G. R. A weight-coded genetic algorithm for the multiple container packing problem. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 14th, 1999, San Antonio. **Proceedings…** San Antonio, july 1999. v. 1, p. 596-603.

SOBRAL, C. M.; AZEVEDO, A. T.; LIMA, F. M. B. Resolução do problema de carregamento e descarregamento de contêineres em terminais portuários via Beam Search. In: SIMPEP, 16., 2009. **Resumos…** [S.l. : s.n.t.], 2009.

WILSON, I. D.; ROACH, P. A. Container stowage planning: A methodology for generating computerised solutions. **Journal of The Operational Research Society**, v. 51, n. 11, p. 1248-1255, 2000.

Artigo selecionado entre os 10 melhores do VII Encontro Mineiro de Engenharia de Produção - EMEPRO 2011.