HEURÍSTICAS GRASP E ILS PARA O PLANEJAMENTO DE ESTOCAGEM DE CONTÊINERES EM NAVIOS PORTA-CONTÊINERES

Amaro José de S. Neto
Universidade Candido Mendes (UCAM-Campos)
Rua Anita Pessanha, 100, Parque São Caetano, Campos dos Goytacazes-RJ. 28040-320.
amaro.neto@gmail.com

Dalessandro S. Vianna
Universidade Federal Fluminense, Pólo Universitário Rio das Ostras – UFF/PURO
Rua Recife, s/n, Jardim Bela Vista, Rio das Ostras-RJ. 22890-000.
dalessandro@pq.cnpq.br

Marcilene de Fátima D. Vianna
Universidade Federal Fluminense, Pólo de Campos dos Goytacazes – UFF/PUCG
Rua Recife, s/n, Jardim Bela Vista, Rio das Ostras-RJ. 22890-000.
marcilenedianin@vm.uff.br

RESUMO

Em um navio porta-contêineres, algumas vezes, na operação de descarregamento, o contêiner alvo que será desembarcado pode estar posicionado abaixo de outros contêineres que não serão descarregados. Estes devem ser removidos para descarregar o contêiner alvo. Cada operação deste tipo é tradada por "remanejo". O objetivo é minimizar o número destas operações. Para resolução do problema são propostas heurísticas GRASP e ILS, cujas soluções obtidas foram comparadas entre si e com uma heurística de descida. Testes computacionais apontam melhoria em relação a heurística de descida.

PALAVRAS-CHAVE: **Otimização; GRASP; ILS**.

Área principal: Metaheurísticas**.**

ABSTRACT

On a container ship, sometimes in the unloading operation, the target container which has to be unloaded may be positioned under other containers that do not have to be unloaded. These ones need to be removed so that the target container can be unloaded. This operation is called "rearrangement". The goal is to minimize the number of such operations. To solve the problem we propose GRASP and ILS heuristics, whose solutions were compared among themselves and with a descent heuristics. Computational experiments show improvement over the descent heuristics.

KEYWORDS: Optimization; GRASP; ILS.

Main area: Metaheuristics**.**

1. Introdução

Os navios porta-contêineres são embarcações especializadas em transporte de carga conteinerizada. Estas embarcações dispõem de espaços celulares (baias), onde os contêineres são empilhados (ver Figura 1). A movimentação da carga ocorre tanto nas baias quanto no convés do navio através de equipamentos de bordo ou de terra. Devido a estrutura do navio e a 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. A essa operação dá-se o nome *remanejo*.

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).

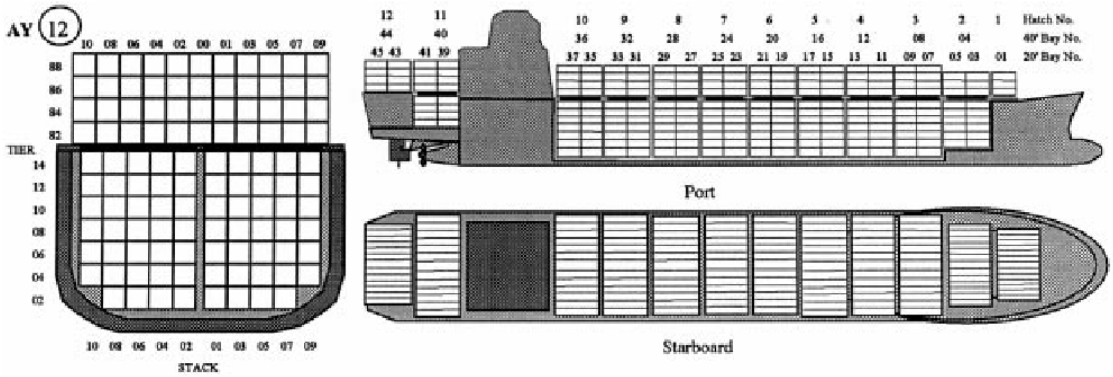

Figura 1. Estrutura de um navio. Fonte: Wilson e Roach (2000).

Neste artigo o foco é a embarcação porta-contêiner. É 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), este problema é NP-Completo.

O objetivo deste trabalho é desenvolver uma heurística GRASP (*Greedy Randomized Adaptive Search Procedure*) e uma ILS (*Iterated Local Search*) para o problema de carregamento e descarregamento de contêineres em um navio porta-contêiner, visando minimizar o número de remanejos.

Na literatura podem ser encontrados trabalhos que abordam problemas semelhantes ao tratado neste trabalho. Dentre estes, pode-se citar: Sobral *et al.* (2009) desenvolveram um algoritmo *Beam Search* para resolução do problema de carregamento e descarregamento de contêineres em terminais portuários; Campos (2008) elaborou um estudo sobre a integração entre carregamento e roteamento de veículos; Morabito e Arenales (1997) mostraram diferentes abordagens para o problema de carregamento de contêineres; Raidl (1999) propôs um algoritmo genético para o problema de empacotamento de múltiplos contêineres; Martins *et al.* (2009) estudaram o problema de estocagem de contêineres; e Azevedo *et al.* (2009) 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 heurística construtiva,

a de busca local, além das heurísticas GRASP e ILS propostas. Na Seção 5 são apresentados os resultados computacionais obtidos e na Seção 6 as conclusões.

2. Apresentação do problema

Os navios porta-contêineres são embarcações de grande porte que possuem uma estrutura celular que facilita à 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 ao fim retorna ao Porto p_0. Cada um dos portos p_i que compõe R tem uma demanda de contêineres que necessitarão ser carregados na embarcação. Além disto, o navio possui uma capacidade que é medida em TEU (*Twenty-foot Equivalent Units*). Por exemplo, uma embarcação com capacidade de 1000 TEUs pode armazenar 1000 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. Nesta operação, pode ser necessário descarregar temporariamente os contêineres que estão armazenados numa pilha de uma determinada baia do navio, para conseguir descarregar um contêiner que está localizado em uma posição inferior desta pilha. Posteriormente, caso haja demanda, os contêineres serão carregados do terminal para o navio, isto implica numa sequência de carregamento S = {c_0, c_1, c_2, ..., c_n} para cada porto que o navio irá atracar, onde deve ser buscado um melhor arranjo destas sequências de forma a minimizar o número de remanejos para os portos seguintes.

3. Representação

Nesta seção será discutida a representação da demanda de contêineres, da baia, da solução e também a função de avaliação.

3.1 Demanda de contêineres

A representação da demanda de contêineres será através de uma matriz de transporte T_{od}, onde 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.

PORTO ORIGEM \ PORTO 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}, onde l=1,2,...,L e c = 1, 2, ..., H (onde L e H significam quantos contêineres cabem na largura e na 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. Veja no exemplo da Figura 3, o valor na posição (3,2) = 3, corresponde ao porto no qual o contêiner da posição (3,2) deverá ser descarregado, neste caso o Porto 3.

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

Figura 3. Exemplo da representação de uma baia de 9 contêineres.

Quando ocorre a atracação do navio em um porto, ocorrem operações de carregamento e descarregamento de contêineres. Devido à estas operações as informações nesta matriz de ocupação são atualizadas constantemente. A cada porto atracado, será necessário pelo menos uma destas 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, onde o primeiro item será o primeiro a ser armazenado no navio, e assim sucessivamente. A Figura 4 ilustra uma solução para um problema com 5 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 destas posições refere-se 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 4 contêineres com destino ao Porto 4, em seguida serão carregados 6 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 isto, 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 um determinado trajeto, realizando descarregamentos e carregamentos, quando necessários, de acordo com as respectivas sequências de carregamento. Um exemplo desta função é mostrado pela Tabela 1, a qual utiliza como dado de entrada a

solução descrita na Figura 4 e que possui duas baias de tamanho 3x3 ($L \times H$, onde L e H significam quantos contêineres cabem na largura e na altura do navio respectivamente). Por exemplo, na iteração 1, de acordo com a ordem de visitação, o navio atracará no Porto 1 e encontra-se vazio. Como não há contêineres a serem descarregados, é feito o carregamento de 2 contêineres com destino para o Porto 4, 2 com destino para o Porto 2, 4 com destino para o Porto 5 e 4 com destino para o Porto 3. Na iteração 2, o navio está parcialmente ocupado e atracado no Porto 3. É feito o descarregamento de 4 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 pode 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 descarregar os contêineres que têm como destino o Porto 5. Computa-se o número de remanejos = 2, e então inicia-se o processo de carregamento e o navio fica totalmente ocupado. Este 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 5 5 3 4 4 2	0 0 0 3 3 3 2 5 5	0
2	3	0 0 0 5 5 3 4 4 2	0 0 0 3 3 3 2 5 5	0 0 0 5 5 0 4 4 2	0 0 0 0 0 0 2 5 5	5 5 5 5 5 2 4 4 2	5 5 5 4 4 2 2 5 5	0
3	5	5 5 5 5 5 2 4 4 2	5 5 5 4 4 2 2 5 5	0 0 0 0 0 2 4 4 2	0 0 0 4 0 0 2 4 2	2 2 2 4 4 2 4 4 2	2 2 2 4 4 4 2 4 2	2
4	2	2 2 2 4 4 2 4 4 2	2 2 2 4 4 4 2 4 2	0 0 0 4 4 0 4 4 0	0 0 0 0 4 0 4 4 4	1 4 4 4 4 1 4 4 1	4 4 0 1 4 1 4 4 4	4
5	4	1 4 4 4 4 1 4 4 1	4 4 0 1 4 1 4 4 4	0 0 0 0 0 1 1 0 1	0 0 0 0 0 0 1 1 0	0 0 0 1 1 1 1 1 1	0 0 0 1 1 0 1 1 1	7
6	1	0 0 0 1 1 1 1 1 1	0 0 0 1 1 0 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. Metodologia

Nesta seção serão discutidas as heurísticas: construtiva, de busca local, GRASP e ILS desenvolvidas.

4.1 Heurística construtiva

A solução nesta fase é 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 à 9 garante que será montada uma sequência de carregamento para cada porto. O laço que vai da linha 3 à 8 garante a montagem da sequência na ordem inversa a 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.

Na Figura 6 é ilustrada à construção de uma solução. Considere que o navio visitará 3 portos, sua ordem de visitação é 1-3-2-1. Observe 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 este 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 da sequência. Em seguida, seleciona-se a demanda do porto que será visitado antes do Porto 2, neste caso, o Porto 3, que têm sua demanda selecionada e colocada na sequência também. A montagem das sequências para os outros portos segue a mesma lógica.

Figura 5. Algoritmo de construção de uma solução

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

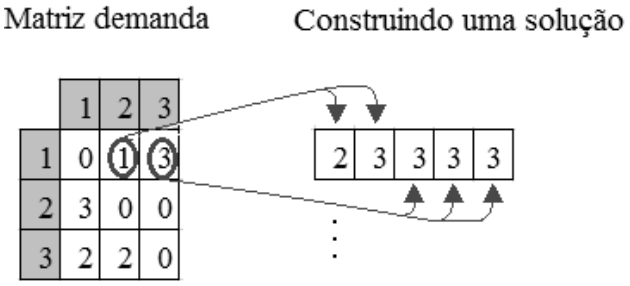

Figura 6. Montagem de uma solução

4.2 Heurística de busca local

O objetivo desta fase é possibilitar uma 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 a solução inicial que pode ser obtida por um movimento, onde o algoritmo de busca local poderá procurar por uma solução melhor. Neste trabalho, um movimento consiste em trocar nas sequências de carregamento, um contêiner de uma determinada posição por um 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 com que se busque na sequência a próxima posição para efetuar a troca. Na linha 5 e 6, é realizada a troca e a avaliação, respectivamente. O bloco delimitado pela linha 7 e 11 serve para verificar se a troca gerou uma solução melhor; caso contrário a troca é desfeita e a solução volta a seu estado anterior.

Figura 7. Algoritmo de busca local

```
Algoritmo buscalocal(solucao)
Entrada
   solucao - matriz onde são registradas as sequências de carregamento de cada porto
Início
1.  remanejos ← ∞
2.  para cada porto i faça
3.     para cada elemento j de solucao[i] faça
4.        para cada elemento k, onde k > j de solucao[i] faça
5.           trocar elemento solucao[i][j] por solucao[i][k]
6.           remanejoObtido ← avalia(solucao)
7.           se remanejoObtido < remanejo então
8.              remanejos←remanejoObtido
9.           senão
10.             trocar elemento solucao[i][j] por solucao[i][k]
11.          fim-se
12.       fim-para
13.    fim-para
14. fim-para
Fim- buscalocal
```

4.3 Heurística GRASP proposta

A Metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) é um processo iterativo de múltiplos inícios, proposto 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 pode-se refinar a solução inicial através de uma busca em sua vizinhança. A melhor solução encontrada ao longo de todas as iterações GRASP é retornada como resultado.

Em um algoritmo GRASP, em geral, a fase de construção é gulosa-aleatorizada. Para isso, em cada porto, todos os elementos pertencentes a sequência de carregamento serão considerados candidatos a compor uma nova solução, caso ainda não tenham sido escolhidos e não inviabilizem à solução com sua participação. Então é constituída à lista de candidatos LC. Em seguida é montada a lista restrita de candidatos LRC, escolhendo-se os α melhores elementos de LC (os contêineres cujo destino sejam os mais distantes). Em LRC serão realizados sorteios, cujo objetivo é escolher um elemento para compor a nova solução. Após a escolha de um elemento de LRC, a lista de candidatos é reconstruída, e é montada uma nova LRC.

Na Figura 8 é mostrado o pseudo-código padrão da metaheurística GRASP. Como entrada deve ser passado o número de iterações do algoritmo (critério de parada). Na linha 1 é atribuído a *melhor_valor* um valor infinito. O laço que vai da linha 2 à 9 garante que a execução do algoritmo *n_iter* vezes. Na linha 3 a heurística de construção parcialmente gulosa determina uma solução *s*. Na linha 4 é realizada uma busca local em *s*, obtendo uma solução *s'* que é ótima local. Da linha 5 à 8 é mantida a melhor solução encontrada em *s_melhor*. Desta forma, a cada iteração do GRASP a melhor solução encontrada é mantida, e serve como parâmetro de comparação para iterações posteriores.

Figura 8. Metaheurística GRASP

```
Algoritmo GRASP(n_iter)
Entrada
   n_iter – numero de iterações do GRASP
Saída
   s_melhor – melhor solução da vizinhança de s
inicio
1.   melhor_valor ← ∞
2.   para i de 0 até n_iter faça
3.      s ← construcaoGulosaAleatorizada()
4.      s' ← buscaLocal(s)
5.      se (f(s') < melhor_valor) entao
6.         s_melhor ← s'
7.         melhor_valor ← f(s')
8.      fim-se
9.   fim-para
10.  retorne s_melhor
fim-GRASP
```

4.4 Heurística ILS proposta

A Metaheurística ILS (*Iterated Local Search*) é um método de busca local que procura em um subespaço do espaço de busca de soluções, definido por soluções que são ótimas locais de determinado procedimento de otimização (LOURENÇO *et al.*, 2002). Existem 5 pontos importantes para o funcionamento do algoritmo: a geração de uma solução de partida; o procedimento de perturbação; o procedimento de busca local; o critério de parada; e o critério de aceitação. Primeiramente é realizada uma busca local na solução de partida *s* (gerada após método de construção guloso), posteriormente *s* será perturbada e realizada uma busca local gerando uma nova solução *s''* e caso seja mais apta que *s* passará a ser a nova solução de partida e o processo de perturbação e busca local se repete até que um critério de parada seja satisfeito. A solução mais apta encontrada ao longo de todas as iterações ILS é retornada como resultado.

A perturbação considerada consiste em efetuar *r* trocas em cada sequência de carregamento; a escolha dos contêineres trocados é aleatória.

A Figura 9 mostra o pseudo-código da metaheurística ILS, o qual tem como ponto de partida uma solução s_0 gerada a partir do método construtivo guloso, na linha 1. Na linha 2 é aplicada uma busca local em s_0, obtendo a solução *s*. O laço que vai da linha 3 à 9 garante a repetição do processo de perturbação e busca local *n_iter* vezes. Na linha 4 é obtida a solução *s'* a partir de uma perturbação em determinados elementos de *s*. Na linha 5 é feita uma busca local em *s'* e é obtido *s''*. Na estrutura que vai da linha 6 até 8 é verificado se a nova solução obtida *s''* é melhor que a melhor solução armazenada *s*. Caso afirmativo *s''* passa a ser a melhor armazenada. Na linha 10 é retornada a melhor solução encontrada *s*.

5. Resultados

Os testes gerados consideram navios com capacidade de 600 a no máximo 2400 unidades de 20 pés, de navios de pequeno a médio porte, respectivamente. A demanda de cada porto não excede a capacidade do navio. Na geração da rota é considerado que o navio parte de um porto origem, visita todos os portos e volta para este porto, no final.

Figura 9. Metaheurística ILS

Algoritmo ILS(s_0, *n_iter*)
Entrada
s_0 – solução inicial
n_iter – numero de iterações do ILS
Saída
s – melhor solução encontrada
Inicio
1. s_0 ← constroiSolucao()
2. *s* ← buscalocal(s_0)
3. **para** *i* de 0 **ate** *n_iter* **faca**
4. *s'* ← perturbação(*s*)
5. s'' ← buscalocal (s')
6. **se** ***f***(*s''*) melhor ***f***(*s*) **entao**
7. *s* ← *s''*
8. **fim**-se
9. **fim**-para
10. **retorne** *s*

Fim-Algoritmo

Por questões de balanceamento do navio, o atendimento a uma 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 × largura da baia. Neste trabalho foi considerado que o número de camadas é a metade da altura. Por exemplo, um navio com 2 baias que suportam 7 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 de 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. Além disto, é considerado que a demanda gerada não supera a capacidade do navio e ocupa 100% de sua capacidade. As demandas foram geradas aleatoriamente.

Para os testes computacionais, foram gerados problemas testes com navios de diferentes capacidades, classificados em grupos de pequenos, médios e grandes, de 600 a 1040, de 1092 a 1456, e 1560 a 2400 TEUs (Twenty-foot Equivalent Units) respectivamente. A visitação considerada foi para 10, 20 e 30 portos, totalizando 54 instâncias. Estas têm as seguintes informações: número de portos; número de baias do navio (Q); a dimensão (L×H) das baias; a ordem de visitação do navio; e a demanda que o navio deve atender em cada porto. Nas tabelas os valores correspondentes a coluna TEU´s são obtidos através da multiplicação entre L × H × Q. O tempo computacional obtido (T) é expresso em segundos.

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.

Para avaliar os resultados computacionais obtidos, através da resolução das instâncias geradas, pelos métodos GRASP e ILS propostos, foi observada a porcentagem de melhoria do número de remanejos em relação aos alcançados com método construtivo puramente guloso.

A lista restrita de candidatos utilizada no método construtivo aleatorizado tem tamanho 3. O número de iterações do GRASP e ILS foi 20. O número de movimentos r considerado no

procedimento de perturbação foi 2. Estes parâmetros foram obtidos a partir de testes de adequação das heurísticas propostas com diferentes parâmetros de entrada.

As Tabelas 6, 7 e 8 mostram os resultados médios obtidos após 5 execuções de cada algoritmo. Conforme pode ser observado, os algoritmos GRASP e ILS se mostraram bem superior ao método construtivo guloso em todos os casos testados. O algoritmo ILS foi superior ao construtivo seguido de uma busca local em todos os casos. O algoritmo GRASP obteve melhores resultados em relação ao construtivo seguido de uma busca local, exceto em: Tabela 6, teste 7; Tabela 7, testes 13, 14 e 15; e Tabela 8, testes 2 e 12. Os melhores resultados encontrados pelos métodos abordados estão destacados em negrito. Em um total de 54 instâncias, o GRASP obteve 31 melhores resultados, enquanto o ILS obteve 21, sendo que houveram 2 empates.

Analisando a Tabela 6, pode-se perceber que o algoritmo GRASP para instâncias pequenas obteve os melhores resultados em todos os casos. Para instâncias médias o GRASP obteve os melhores resultados em 67% dos casos, e o algoritmo ILS em 33%. Em instâncias grandes o GRASP obteve somente 33% dos melhores resultados, e o ILS obteve 67%. Comparando as médias gerais de melhoria, o GRASP obteve 1% a mais que o ILS.

Analisando a Tabela 7, pode-se perceber que o algoritmo GRASP para instâncias pequenas obteve os melhores resultados em 67% dos casos, o algoritmo ILS em 16,5% e houve igualdade em 16,5%. Para instâncias médias o GRASP obteve os melhores resultados em 33% dos casos, o algoritmo ILS em 50% e houve igualdade em 17%. Em instâncias grandes o GRASP obteve somente 33% dos melhores resultados, e o ILS obteve 67%. Comparando as médias gerais de melhoria, o GRASP obteve 1% a mais que o ILS.

Analisando a Tabela 8, pode-se perceber que o algoritmo GRASP para instâncias pequenas obteve os melhores resultados em 67% dos casos, o algoritmo ILS em 33%. Para instâncias médias o GRASP obteve os melhores resultados em 67% dos casos, o algoritmo ILS em 33%. Em instâncias grandes o GRASP obteve 50% dos melhores resultados, e o ILS obteve 50%. Em relação as médias gerais de melhoria, o GRASP e o ILS foram iguais.

Tabela 2. Resultados obtidos para instâncias com 10 portos

Nº	BAIA			TEUs	CONSTRUÇÃO		BUSCA LOCAL			GRASP			ILS		
	Q	L	H		CUSTO	T	CUSTO	%	T	CUSTO	%	T	CUSTO	%	T
1	12	5	10	600	558	0	470	16%	1,6	**413**	26%	36,86	421	25%	42,30
2	12	5	13	780	1713	0	1444	16%	8,87	**816**	52%	96,29	842	51%	110,72
3	16	5	10	800	809	0	652	19%	3,12	**502**	38%	67,36	564	30%	84,63
4	12	7	10	840	979	0	674	31%	3,73	**588**	40%	87,71	618	37%	108,81
5	12	5	15	900	777	0	638	18%	3,57	**562**	28%	83,81	584	25%	104,88
6	16	5	13	1040	1352	0	1165	14%	6,17	**1005**	26%	146,13	1029	24%	171,51
7	12	7	13	1092	835	0	544	35%	3,28	559	33%	74,91	**503**	40%	83,57
8	16	7	10	1120	1169	0	805	31%	6,68	**632**	46%	146,39	657	44%	174,67
9	12	10	10	1200	945	0	721	24%	6,89	**679**	28%	162,50	701	26%	175,35
10	16	5	15	1200	1130	0	888	21%	7,82	837	26%	176,36	**803**	29%	199,76
11	12	7	15	1260	1024	0	708	31%	7,04	**650**	37%	158,54	694	32%	186,08
12	16	7	13	1456	1303	0	985	24%	9,95	**903**	31%	229,87	881	32%	261,74
13	12	10	13	1560	1569	0	1299	17%	13,23	1211	23%	284,17	**1199**	24%	344,97
14	16	10	10	1600	993	0	695	30%	10,62	671	32%	248,88	**661**	33%	292,21
15	16	7	15	1680	1302	0	924	29%	13,5	**793**	39%	295,54	835	36%	331,31
16	12	10	15	1800	1489	0	992	33%	15,51	**887**	40%	363,24	927	38%	390,77
17	16	10	13	2080	1665	0	1456	13%	25,48	1376	17%	586,43	**1363**	18%	653,58
18	16	10	15	2400	3899	0	3042	22%	40,96	2976	24%	903,91	**2893**	26%	1091,77
MÉDIA							24%			33%			32%		

Tabela 3. Resultados obtidos para instâncias com 20 portos

Nº	BAIA			TEUs	CONSTRUÇÃO		BUSCA LOCAL			GRASP			ILS		
	Q	L	H		CUSTO	T	CUSTO	%	T	CUSTO	%	T	CUSTO	%	T
1	12	5	10	600	1389	0	983	29%	6,98	**808**	42%	152,80	908	35%	174,78
2	12	5	13	780	1984	0	1444	27%	12,54	1406	29%	285,82	**1379**	30%	317,41
3	16	5	10	800	1098	0	802	27%	11,26	**737**	33%	248,04	768	30%	278,61
4	12	7	10	840	1863	0	1346	28%	14,65	**1214**	35%	332,34	1241	33%	368,95
5	12	5	15	900	1974	0	1618	18%	17,32	**1525**	23%	393,20	1573	20%	442,89
6	16	5	13	1040	2171	0	1600	26%	22,81	1546	29%	531,65	1546	29%	594,71
7	12	7	13	1092	2490	0	1903	24%	24,71	1777	29%	553,47	**1771**	29%	625,95
8	16	7	10	1120	2250	0	1668	26%	23,07	**1519**	32%	539,66	1534	32%	603,43
9	12	10	10	1200	3044	0	2263	26%	28,98	**1822**	40%	646,12	2075	32%	739,58
10	16	5	15	1200	3187	0	2457	23%	29,31	2323	27%	698,29	2323	27%	761,76
11	12	7	15	1260	3628	0	2860	21%	38,78	2636	27%	813,23	**2570**	29%	937,71
12	16	7	13	1456	3471	0	2758	21%	45,59	2655	24%	1078,85	**2652**	24%	1209,77
13	12	10	13	1560	3293	0	2335	29%	42,95	2399	27%	1016,13	**2275**	31%	1105,53
14	16	10	10	1600	2333	0	1596	32%	39,67	1605	31%	918,54	**1580**	32%	993,85
15	16	7	15	1680	2487	0	1904	23%	47,46	1911	23%	1061,49	**1861**	25%	1144,37
16	12	10	15	1800	3140	0	2380	24%	59,56	**2249**	28%	1327,76	2317	26%	1500,95
17	16	10	13	2080	5940	0	4551	23%	98,82	**4256**	28%	2308,43	4258	28%	2597,19
18	16	10	15	2400	5740	0	4364	24%	128,78	4171	27%	2913,04	**4164**	27%	3261,53
MÉDIA							25%			30%			29%		

Tabela 4. Resultados obtidos para instâncias com 30 portos

Nº	BAIA			TEUs	CONSTRUÇÃO		BUSCA LOCAL			GRASP			ILS		
	Q	L	H		CUSTO	T	CUSTO	%	T	CUSTO	%	T	CUSTO	%	T
1	12	5	10	600	2295	0	1676	27%	17,19	**1634**	29%	342,88	1645	28%	379,03
2	12	5	13	780	2497	0	1737	30%	29,28	1769	29%	584,89	**1735**	31%	639,61
3	16	5	10	800	2438	0	1652	32%	27,95	1630	33%	559,36	**1621**	34%	617,01
4	12	7	10	840	2475	0	1660	33%	27,70	**1538**	38%	574,06	1603	35%	622,62
5	12	5	15	900	3312	0	2583	22%	42,72	**2494**	25%	873,18	2511	24%	1004,98
6	16	5	13	1040	3392	0	2573	24%	50,48	**2476**	27%	1044,17	2510	26%	1154,46
7	12	7	13	1092	4052	0	2885	29%	57,72	**2799**	31%	1173,23	2808	31%	1277,67
8	16	7	10	1120	3242	0	2310	29%	52,45	**2136**	34%	1051,97	2174	33%	1182,86
9	12	10	10	1200	3782	0	2773	27%	69,48	2596	31%	1406,48	**2593**	31%	1573,47
10	16	5	15	1200	4348	0	3271	25%	76,36	**3211**	26%	1514,92	3226	26%	1673,97
11	12	7	15	1260	4405	0	3492	21%	80,22	**3342**	24%	1619,43	3380	23%	1822,04
12	16	7	13	1456	4511	0	3128	31%	93,03	3227	28%	1911,68	**3090**	32%	2072,06
13	12	10	13	1560	4645	0	3574	23%	112,63	**3497**	25%	2256,56	3529	24%	2534,79
14	16	10	10	1600	5058	0	3759	26%	115,05	3666	28%	2344,77	**3630**	28%	2604,35
15	16	7	15	1680	4945	0	3877	22%	129,27	**3583**	28%	2626,79	3694	25%	2912,52
16	12	10	15	1800	6908	0	5199	25%	176,53	5069	27%	3528,66	**5067**	27%	3869,36
17	16	10	13	2080	7599	0	5936	22%	233,41	**5669**	25%	4727,55	5688	25%	5285,66
18	16	10	15	2400	7889	0	6136	22%	288,85	5902	25%	5779,78	**5801**	26%	6438,10
MÉDIA							26%			28%			28%		

6. Conclusões

Foram propostas heurísticas baseadas nas metaheurísticas GRASP e ILS para resolução do problema de carregamento e descarregamento de contêineres em navios, no qual são minimizados o número de remanejos. A partir dos experimentos computacionais realizados nas 54 instâncias geradas é possível notar adequação das metodologias, bem como, notável melhora da solução apresentada pelo método construtivo puramente guloso. Além disto, com poucas iterações, as heurísticas GRASP e ILS se mostraram eficientes sobre o método construtivo puramente guloso seguido de uma busca local. De acordo com os resultados obtidos, comparando o somatório das médias de melhorias o ILS se mostrou 1% inferior em média ao GRASP. Analisando o desempenho de acordo com o tamanho da instância, o GRASP obteve a maior quantidade de melhores resultados para instâncias pequenas e médias enquanto o ILS foi melhor para as maiores instâncias. Vale ressaltar que a heurística ILS mostrou-se mais adequada a problemas de maior porte.

Referências

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

AZEVEDO, A. T.; SOBRAL, C. M.; DEUS, N. M. R. (2009) "Resolução do problema de carregamento e descarregamento de contêineres em terminais portuários via Algoritmo genético", XVI SIMPEP (Simpósio de Engenharia de Produção).

BERTOLANI, A. D.; LEME, F. L. (2004) "Carregamento de contêineres em navios". Revista Mackenzie On-line de Engenharia.

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

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

LOURENÇO, H. R.; MARTIN, O.; STUETZLE, T. (2002) "Iterated local search", Handbook of Metaheuristics, p. 321–353, Norwell, MA. Kluwer Academic Publishers.

MARTINS, P. T.; LOBO, V. J. A. S.; VAIRINHOS, V. (2009) "Container Stowage Problem Solution for Short Sea Shipping", 14º congresso da APDIO (Associação Portuguesa de Investigação Operacional).

MORABITO, R.; ARENALES, M. (1997) "Abordagens para o problema do carregamento de contêineres". Pesquisa Operacional, 17 (1), 29-56.

RAIDL, G. R. (1999) "A weight-coded genetic algorithm for the multiple container packing problem", 14th ACM Symposium on Applied Computing.

SOBRAL, C. M.; AZEVEDO, A. T.; LIMA, F. M. B. (2009) "Resolução do problema de carregamento e descarregamento de contêineres em terminais portuários via Beam Search", XVI SIMPEP (Simpósio de Engenharia de Produção).

WILSON, I. D., ROACH, P. A. (2000) "Container Stowage Planning: A methodology for generating computerised solutions". Journal of The Operational Research Society.