O PROBLEMA DE CARREGAMENTO E DESCARREGAMENTO DE CONTÊINERES EM TERMINAIS PORTUÁRIOS: UMA ABORDAGEM VIA SIMULATED ANNEALING

Anibal Tavares Azevedo
FEG - Faculdade de Engenharia de Guaratinguetá - UNESP
Av. Ariberto Pereira da Cunha 333. Pedregulho – Guaratinguetá –SP – CEP: 12516-410
anibal@feg.unesp.br

Galeno José de Sena
FEG - Faculdade de Engenharia de Guaratinguetá - UNESP
Av. Ariberto Pereira da Cunha 333. Pedregulho – Guaratinguetá –SP – CEP: 12516-410
gsena@feg.unesp.br

Antônio Augusto Chaves
UNIFESP, Universidade Federal de Sao Paulo
Rua Talim, 330 – São Jose dos Campos – SP, CEP: 12.231-280
antonio.chaves@unifesp.br

Cassilda Maria Ribeiro
FEG - Faculdade de Engenharia de Guaratinguetá - UNESP
Av. Ariberto Pereira da Cunha 333. Pedregulho – Guaratinguetá – SP – CEP: 12516-410
cassilda@feg.unesp.br

RESUMO

Neste artigo é apresentada uma abordagem via *Simulated Annealing* para resolver o problema de carregamento de contêineres num terminal portuário, comparando-se os resultados obtidos com os produzidos por um algoritmo do tipo *Beam Search*. Num navio porta contêiner os contêineres são colocados em pilhas verticais, localizadas em diversas seções. O acesso aos contêineres é feito somente através do topo da pilha. Muitas vezes para se descarregar um contêiner num determinado porto j, é necessário remover o contêiner cujo destino é o porto j+1, porque ele está acima do contêiner que se deseja descarregar. Esta operação é chamada de remanejamento. Um navio porta contêiner transportando carga para vários portos necessita de muitas operações de remanejamento. Esses remanejamentos possuem custo e despendem tempo. É possível evitar alguns remanejamentos através de um planejamento eficiente do carregamento de contêineres. Um objetivo crucial para o problema de carregamento de contêineres é minimizar o número de movimentos de contêineres. Neste artigo é apresentado um modelo de solução para este problema, implementando a metaheurística Simulated Annealing e comparando os resultados com um método Beam Search encontrado na literatura. O modelo apresentado utiliza uma representação bastante compacta da solução do problema assegurando que as soluções sejam factíveis.

PALAVRAS CHAVES. Carregamento de Contêineres. Simulated Annealing. Beam Search. Metaheurísticas. Otimização Combinatória.

ABSTRACT

This paper proposes a Simulated Annealing approach to solve the Container Ship

Stowage Problem, comparing the obtained results with the ones produced by a *Beam Search* kind algorithm. Containers on board a container ship are placed in vertical stacks, located in different sections. The access to the containers is done only through the top of the stack. Many times to unload a container at a given port j, it is necessary to remove the container whose destination is the port j +1, because it is above the container one wants to download. This operation is called "shifting". A ship container carrying cargo to several ports may require a large number of shifting operations. These operations have spent time and cost. It is possible to avoid some shifting operations by means of an efficient stowage planning. A key objective of stowage planning is to minimize the number of container movements. In this paper it is presented a solution model to this problem, which has been implemented in an Simulated Annealing algorithm application environment, whose results are compared with the ones obtained with a beam search kind algorithm. The presented method uses a very compact representation of the solution and ensures that all solutions are feasible.

KEYWORDS. Container Ship Stowage Problem. Simulated Annealing. Beam Search. Metaheuristics. Combinatorial Optimization. `

1. Introdução

Navios porta contêineres são navios que possuem uma estrutura que facilita a movimentação de contêineres de carga, pois em cada porto ao longo de uma viagem, contêineres são descarregados e contêineres adicionais, destinados aos portos subseqüentes são carregados.

A eficiência de um terminal portuário depende de uma adequada programação da movimentação de contêineres, especialmente durante o processo de carregamento dos navios, uma vez que a programação poderá refletir em redução de tempo e conseqüentemente redução de custo. A estiva e o plano de carregamento associado são determinados fundamentalmente por dois critérios: estabilidade do navio e número mínimo de remanejamentos requeridos nos diversos pontos de entrega (AVRIEL *et al*., (2000); WILSON e ROACH, (2000); AMBROSINO *et al*., (2006)). O último critério é baseado no fato de que muitos navios possuem uma estrutura celular, conforme pode ser observado na Figura 1, e os contêineres devem ser carregados de modo a formarem pilhas verticais, o que acarreta, em muitos casos, a necessidade de movimentar contêineres na parte superior da pilha a fim de se descarregar os contêineres que estão posicionados na parte inferior. A este tipo de movimentação dá-se o nome de remanejamento. Os remanejamentos são necessários porque os contêineres que estão numa pilha só podem ser acessados pelo topo.

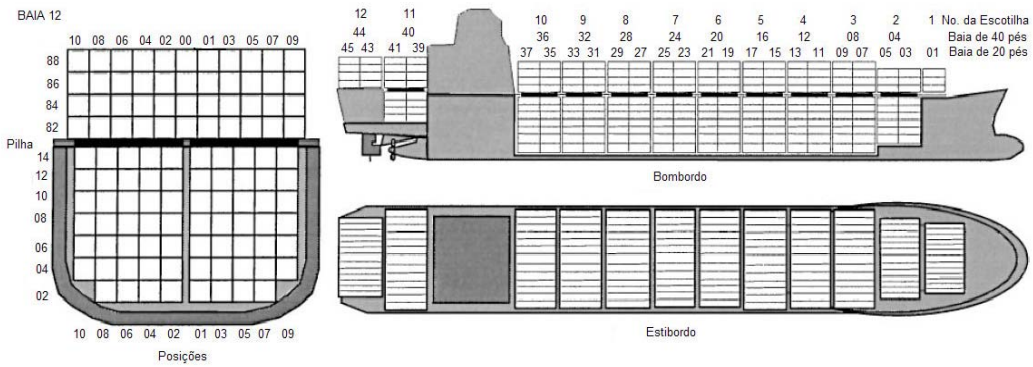

Figura 1: Estrutura celular de um navio. Fonte: WILSON e ROACH (2000).

O problema de carregamento de contêineres em terminais portuários (PCCTP) consiste

em determinar como carregar um conjunto de contêineres de diferentes tipos em um navio porta-contêiner (*containership*), respeitando restrições operacionais relacionadas aos contêineres e navio de modo a minimizar o tempo de carregamento e posterior descarregamento, ou seja, o remanejamento. Em AVRIEL *et al.* (2000) se mostra que este problema é NP-Completo.

Neste artigo é apresentado um modelo alternativo ao modelo apresentado por AVRIEL *et al.* (2000) e para o qual será realizada a solução do PCCTP por meio da aplicação de um algoritmo do tipo *Simulated Annealing (SA)* e também de um algoritmo do tipo *Beam Search (BS)*. A abordagem contemplada no modelo descrito para resolver esse problema é inovadora porque reduz consideravelmente o número de variáveis do problema e também permite que seja utilizada a experiência do profissional portuário através de uma representação adequada dos seus conhecimentos. Este artigo está organizado do seguinte modo: a Seção 2 apresenta a formulação matemática do problema; a Seção 3 contém a representação matricial utilizada na resolução do problema; a Seção 4 descreve a metaheurística *SA* aplicada ao PCCTP; a Seção 5 apresenta os resultados computacionais obtidos e a Seção 6, as conclusões. O algoritmo tipo *BS* construído com base no modelo encontra-se descrito em (RIBEIRO e AZEVEDO, 2010).

2. Modelo Matemático do Problema

Navios porta contêineres são navios que possuem uma estrutura que facilita a movimentação de contêineres. De fato, eles possuem uma estrutura celular (vide Figura 1) onde são alojados os contêineres. Essas células em geral, têm 20 pés de comprimento e 8 pés de largura. Os contêineres são empilhados nas células formando pilhas verticais. O conjunto de pilhas que compreende toda a largura do navio é chamado de baia, em inglês *bays.* Então, uma baia é um agrupamento de células com capacidade para se empilhar um certo número de contêineres. Na Figura 1 tem-se o corte lateral da baia 12 de um navio. Em geral cada baia tem capacidade para alocar quarenta contêineres de vinte pés de comprimento. A baia tem então linhas horizontais numeradas r = 1, 2, ...R, (a linha 1 é a linha que está em baixo na pilha, e a linha R é a linha do topo da pilha) e colunas numeradas c= 1, 2, ..., C (coluna 1 é a primeira coluna da esquerda). A capacidade do navio porta contêiner é medida em TEU (*Twenty-foot Equivalent Units*) ou Unidade Equivalente de Vinte pés. Por exemplo, um navio com capacidade de 8000 TEUs pode carregar 8000 contêineres de vinte pés.

A formulação matemática aqui apresentada para o PCCTP respeita as restrições operacionais relacionadas aos contêineres, navio e pátio do terminal portuário e aparece de modo mais detalhado em AVRIEL *et al.* (1998). Considere um navio porta contêineres que possui uma única baia. A baia tem R linhas horizontais numeradas r = 1, 2, ...R, (a linha 1 é a linha que está em baixo, e a linha R é a linha do topo da pilha) e C colunas verticais numeradas c= 1, 2, ..., C (coluna 1 é a primeira coluna da esquerda). Apesar de a baia ter um formato tridimensional, a mesma pode ser representada, sem perda de generalidade, por um formato bidimensional, em particular uma matriz. Então, uma baia pode alocar no máximo RxC contêineres. É assumido também que todos os contêineres têm o mesmo tamanho. O navio chega no porto 1 completamente vazio e seqüencialmente visita os portos 2, 3, ..., N. Em cada porto i=1,...,N-1, o navio recebe o carregamento de contêineres com destino aos portos i+1,...,N. No último porto ele descarrega os contêineres e fica totalmente vazio. Seja $T=[T_{ij}]$ a matriz de transporte de dimensão (N-1)×(N-1), onde T_{ij} é o número de contêineres com origem em i e destino em j. Esta matriz é triangular superior porque $T_{ij}=0$ para todo $i \geq j$. Seja $x_{ijv}(r,c)$ a variável binária que assume o valor 1 se existir um contêiner no compartimento (r, c) que foi ocupado no porto i e tem como destino final o porto j e movido no porto v; caso contrário assume valor zero. Por compartimento (r,c) entende-se a linha r e a coluna c no compartimento de carga do navio. É necessário salientar que a numeração das linhas é feita de baixo para cima. Assim a linha de número 5 está acima da linha de número 4, e a numeração das colunas é feita da esquerda para a direita. Seja $y_i(r,c)$ a

variável binária que possui valor 1 se saindo do porto *i* o compartimento (r, c) for ocupado por um contêiner; caso contrário assume valor 0. A formulação em programação linear inteira do PCCTP é dada pelas Eqs. (1)-(6).

$$Min \quad f(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{v=i+1}^{j-1} \sum_{r=1}^{R} \sum_{c=1}^{C} x_{ijv}(r,c) \qquad i = 1,\Lambda\ ,N-1,\ j = i+1,\Lambda\ ,N; \qquad (1)$$

$$s.a: \quad \sum_{v=i+1}^{j} \sum_{r=1}^{R} \sum_{c=1}^{C} x_{ijv}(r,c) - \sum_{k=1}^{i-1} \sum_{r=1}^{R} \sum_{c=1}^{C} x_{kji}(r,c) = T_{ij} \qquad i = 1,\Lambda\ ,N-1,\ j = i+1,\Lambda\ ,N; \qquad (2)$$

$$\sum_{k=1}^{j} \sum_{j=i+1}^{N} \sum_{v=i+1}^{j} x_{kjv}(r,c) = y_i(r,c) \qquad i = 1,\Lambda\ ,N-1, r = 1,\Lambda\ ,R; \quad c = 1,\Lambda\ ,C; \qquad (3)$$

$$y_i(r,c) - y_i(r+1,c) \geq 0 \qquad i = 1,\Lambda\ ,N-1, r = 1,\Lambda\ ,R-1; \quad c = 1,\Lambda\ ,C; \qquad (4)$$

$$\sum_{i=1}^{j-1} \sum_{p=j}^{N} x_{ipj}(r,c) + \sum_{i=1}^{j-1} \sum_{p=j+1}^{N} \sum_{v=j+1}^{p} x_{ipv}(r+1,c) \leq 1 \qquad i = 2,\Lambda\ ,N, r = 1,\Lambda\ ,R-1; \quad c = 1,\Lambda\ ,C; \qquad (5)$$

$$x_{ijv}(r,c) = 0 \text{ ou } 1;\ \ y_i(r,c) = 0 \text{ ou } 1 \qquad (6)$$

A Eq. (1) fornece o custo total de movimentação dos contêineres em todos os portos, assumindo que a movimentação de um contêiner possui um custo unitário e igual para todos os portos. A restrição (2) é a restrição de conservação de fluxo de contêineres para cada porto *i*, onde T_{ij} é o elemento da matriz de transporte que representa o número de contêineres que embarcam no porto *i* com destino ao porto *j*. A restrição (3) garante que cada segmento de rota tem pelo menos um único contêiner. A restrição (4) é necessária para garantir que existem contêineres embaixo do contêiner que ocupa o compartimento (r, c). A restrição (5) é responsável por definir a movimentação dos contêineres: se um contêiner que ocupa a posição (r, c) é descarregado no porto *j*, então, ou não existem contêineres acima dele, ou o índice v do contêiner que ocupa o compartimento (r+1, c) não é maior que *j*.

O tamanho que o problema assume com a formulação dada pelas Eqs. (1)-(6) é proibitivo para problemas reais e só pode ser resolvido, de maneira ótima, para problemas pequenos.

3. Representação matricial do PCCTP

A seguir será apresentada a representação matricial desenvolvida para a resolução do PCCTP. Esta representação tem a vantagem de ser bastante compacta e de assegurar que todas as soluções geradas, tanto pelo método *BS* como pelo *SA* sejam factíveis.

Na Figura 1 viu-se que os navios possuem uma estrutura celular de modo que os locais onde os contêineres serão alocados são pré-determinados fazendo com que os contêineres sejam empilhados verticalmente. Este empilhamento sugere uma representação matricial dos contêineres no navio. Deste modo pode-se definir uma matriz de ocupação B que fornece a quantidade de espaços disponíveis e a localização dos contêineres no navio em cada porto. Para tanto, cada elemento da matriz B_{rc} representa o estado de uma célula (r,c), isto é se $B_{rc}=0$ significa que a célula (r,c) está vazia e se $B_{rc}= j$ significa que a célula (r,c) contém um contêiner cujo destino é o porto *j*. As linhas são numeradas de baixo para cima, ou seja, a linha 1 representa a parte inferior da pilha, e as colunas, da esquerda para a direita. Assim, no exemplo da Figura 2, o elemento $B_{1,1}$ é igual a 5 significando que neste local existe um contêiner que será descarregado no porto 5. De modo análogo, o elemento $B_{3,3}=0$ significa que esta célula está vazia.

2	2	0	0
4	3	0	5
5	2	3	4
5	5	4	2

B

Figura 2: Matriz de Ocupação, considerando um navio com capacidade de 16 contêineres e transporte para 5 portos.

Em todos os portos a matriz B de ocupação é modificada devido à entrada e saída de novos contêineres no porto. Isto porque, quando o navio chega num porto *j*, os contêineres cujo destino é o porto *j*, serão descarregados e depois serão carregados os contêineres destinados aos portos *j+1, j+2,...,N*. Portanto, podem ser identificadas duas classes de operações com contêineres que modificam a matriz B de ocupação do navio no porto *j*: *carregamento* e *descarregamento*. É possível elaborar diferentes estratégias para se carregar e descarregar o navio. Por exemplo, o carregamento de um navio pode ser feito com a formação de pilhas verticais ou ainda formação de camadas horizontais. O mesmo pode ser realizado para o descarregamento. O produto da transformação de uma estratégia, para realizar carregamento ou descarregamento, em um algoritmo, resulta em uma *regra_k* no modelo, correspondendo a um par de regras, contendo uma regra de carregamento e uma de descarregamento para um determinado porto *j*. Uma vantagem desta estratégia é que ela permite incorporar o conhecimento do planejador, sob a forma de regras, na resolução do problema.

Neste artigo foram criadas 12 regras_k, a partir da combinação de quatro para o carregamento (Rc1, Rc2, Rc3 e Rc4) com três para o descarregamento (Rd1, Rd2, Rd3). A Tabela 1, a seguir, apresenta as 12 regras_k empregadas nas soluções computacionais discutidas neste artigo. Observa-se na Tabela 1 que a regra Rc2 foi obtida utilizando a regra Rc1 para o carregamento dos contêineres e a regra Rd2 para o descarregamento. Além do mais, esta forma de combinar regras permite uma representação compacta da solução, pois para se determinar o número de movimentos basta ter a informação de qual regra_k será aplicada em cada porto *j* e a conseqüente atualização da matriz de ocupação B. Inicialmente a matriz B está com todos os elementos iguais a zero e ela começa a ser preenchida no porto 1.

Regra_k usada no porto j	**Regra de carregamento**	**Regra de descarregamento**	**Regra_k usada no porto j**	**Regra de carregamento**	**Regra de descarregamento**
1	Rc1	Rd1	7	Rc3	Rd1
2	Rc1	Rd2	8	Rc3	Rd2
3	Rc1	Rd3	9	Rc3	Rd3
4	Rc2	Rd1	10	Rc4	Rd1
5	Rc2	Rd2	11	Rc4	Rd2
6	Rc2	Rd3	12	Rc4	Rd3

Tabela 1: Regras_*k*, a serem utilizadas em cada porto *j*.

Para melhor ilustrar o uso dessas regras, será utilizada uma matriz de transporte T, que fornece a quantidade de contêineres que devem ser embarcados no porto *i* com destino ao porto *j*, tal como ilustrado na Figura 3. Nesta Figura, T_{12}= 3, significa que no porto 1, serão embarcados 3 contêineres com destino ao porto 2 e T_{24}=2, significa que no porto 2, serão embarcados 2 contêineres com destino ao porto 4. Neste exemplo considerou-se que o navio tem capacidade para transportar 16 contêineres, num roteiro de 5 portos. Ao chegar em um porto *p*, o navio deve ser descarregado de acordo com uma regra de descarregamento e depois carregado de acordo com uma regra de carregamento. As regras de descarregamento (**Rd)** e de carregamento (**Rc)** adotadas para se montar a matriz B de ocupação são apresentadas a seguir. Observe-se que a regra de descarregamento para o porto N (último porto) é irrelevante.

D2 D3 D4 D5

O1	3	5	1	3
O2	0	3	2	1
O3	0	0	2	2
O4	0	0	0	3

Figura 3: Matriz de transporte T.

Regras de Descarregamento

Regra Rd1: Nesta regra quando o navio chega a um porto *p*, são descarregados todos os contêineres cujo destino é *p* e todos os contêineres que estão acima dos contêineres do porto p cujos destinos são os portos *p+1*,..., *N*. Suponha, por exemplo, que ao se chegar no porto 2 a matriz de ocupação B seja a da Figura 4(a) mostrada a seguir. De acordo com esta regra a matriz B, após o descarregamento dos contêineres, ficaria como mostrada na Figura 4(b). A Figura4(c) mostra os contêineres que estavam no navio com destino aos demais portos e foram remanejados. Esses contêineres serão re-embarcados juntamente com os contêineres que serão embarcados nesse porto.

0	0	0	0
5	5	2	3
3	2	3	5
2	3	3	4

Figura 4(a): matriz B antes da aplicação da regra **Rd1.**

0	0	0	0
0	0	0	3
0	0	3	5
0	3	3	4

Figura 4(b): matriz B depois da aplicação da regra **Rd1.**

5	5	3

Figura 4(c): Vetor contendo os contêineres do pátio de remanejamento

Regra Rd2: Nesta regra quando o navio chega ao porto *p*, remove-se completamente todas as pilhas que contenham um contêiner cujo destino é o porto atual. A seguir, os contêineres que forem removidos neste porto, mas cujos destinos são os portos *p+1*, *p*+2,...N, serão re-embarcados de acordo com a regra de embarque adotada para o porto. Suponha por exemplo que ao se chegar no porto 2 a matriz de ocupação B seja a da Figura 5(a). De acordo com esta regra a matriz de ocupação ficará quase vazia (a exceção de uma coluna todas as demais serão removidas) e quase todos os contêineres que devem seguir viagem irão para o pátio de remanejamento. Como no caso da regra anterior, esses contêineres serão recarregados juntamente com os contêineres que devem ser embarcados no porto 2. A Figura 5(c) mostra os contêineres do pátio de remanejamento.

0	0	0	0
5	5	2	3
3	2	3	5
2	3	3	4

Figura 5(a): matriz B antes da aplicação da regra **Rd2.**

0	0	0	0
0	0	0	3
0	0	0	5
0	0	0	4

Figura 5(b): matriz B depois da aplicação da regra **Rd2.**

5	5	3	3	3	3

Figura 5(c): Vetor contendo os contêineres do pátio de remanejamento

Regra Rd3: Nesta regra quando o navio chega ao porto *p*, remove-se completamente todas as pilhas de contêineres do navio, independentemente de conterem ou não um contêiner cujo destino é o porto atual, objetivando-se com isto uma reorganização geral dos contêineres no navio. A seguir, os contêineres que forem removidos neste porto, mas cujos destinos são os portos *p+1*, *p*+2,...N, são re-embarcados de acordo com a regra de embarque adotada para o porto. Suponha por exemplo que ao se chegar no porto 2 a matriz de ocupação B seja a da Figura 6(a). De acordo com esta regra a matriz de ocupação ficará vazia e todos os contêineres que devem seguir viagem irão para o pátio de remanejamento. Como nas regras anteriores, esses contêineres deverão ser recarregados juntamente com os contêineres que devem ser embarcados no porto 2. A Figura 6(b) mostra os contêineres do pátio de remanejamento.

0	0	0	0
5	5	2	3
3	2	3	5
2	3	3	4

Figura 6(a): Matriz B antes de **Rd2.**

5	5	5	4	3	3	3	3	3

Figura 6(b): Vetor contendo os contêineres do pátio de remanejamento

Regras de Carregamento

Regra Rc1: Esta regra preenche a matriz de ocupação B (no porto *p*) por linha, da esquerda para a direita, colocando na parte inferior da pilha as cargas destinadas a portos mais distantes. A Figura 7, a seguir, fornece um exemplo da aplicação da regra **Rc1**, com base na matriz de transporte da Figura 3, considerando que o navio está no porto 1. Observe na Figura 7 que os elementos B_{11}, B_{12} e B_{13} são iguais a 5, B_{14} é igual a 4, B_{21}, B_{22}, B_{23}, B_{24} e B_{31} são iguais a 3, e B_{32}, B_{33} e B_{34} são iguais a 2 porque no porto 1 são embarcados, respectivamente, 3 contêineres destinados ao porto 5, 1 contêiner cujo destino é o porto 4, 5 contêineres com destino ao porto 3, e 3 contêineres cujo destino é o porto 2.

0	0	0	0
3	2	2	2
3	3	3	3
5	5	5	4

B

Figura 7: Matriz de Ocupação no porto 1, após a aplicação da **regra Rc1**

Regra Rc2: Nesta regra o preenchimento da matriz de ocupação B em um porto *p* é feito por coluna até uma linha θ_p, iniciando-se pela coluna da esquerda e colocando-se em primeiro lugar os contêineres cujos destinos são os portos mais distantes. A linha θ_p é calculada tomando-se a função teto, a qual resulta da soma do total de contêineres que estavam no navio vindos dos portos anteriores, menos a quantidade de contêineres que serão desembarcados em *p*, e mais a quantidade de contêineres que serão embarcados em *p*, dividindo-se este resultado pelo número de colunas da matriz B. O índice dessa linha pode ser calculado a partir da matriz de transporte T, através da equação (7).

$$\theta_p = \left\lceil \frac{\sum_{i=1}^{p} \sum_{j=p+1}^{N} T_{ij}}{C} \right\rceil \qquad (7)$$

onde: *p* é o porto atual do navio, T_{ij} é o número total de contêineres a serem embarcados no porto *i* com destino ao porto *j* e C é o número total de colunas da matriz B de ocupação do navio.

Supondo que a matriz de ocupação **B** quando o navio chega ao porto 2 seja a da Figura 7, ao se descarregar os contêineres cujo destino é o porto 2, ela terá os elementos mostrados na Figura 8(a) a seguir. Aplicando-se agora a regra de carregamento **Rc2**, a matriz de ocupação **B** será preenchida até a linha 4, tal como mostrado na Figura 7(b). O objetivo dessa regra, ao se calcular θ_p, é se obter pilhas de contêineres mais homogêneas, com isto procurando-se diminuir o número de remanejamentos (BISCHOFF e RATCLIFF (1995)).

0	0	0	0
3	0	0	0
3	3	3	3
5	5	5	4

Figura 8(a): Matriz de ocupação **B** no porto 2 antes a aplicação da regra Rc2.

5	4	3	0
3	4	3	3
3	3	3	3
5	5	5	4

Figura 8(b): Matriz de ocupação **B** no porto 2 após a aplicação da regra Rc2.

Regra Rc3: Esta regra é similar a regra **Rc1**, mas difere no sentido do preenchimento de B tal que no porto *p* a matriz de ocupação B será preenchida por linha, da direita para a esquerda, colocando na parte inferior da pilha as cargas destinadas aos portos mais distantes.

Regra Rc4: Esta regra é similar a regra **Rc2**, mas difere no sentido do preenchimento de B tal que ela efetua o preenchimento da matriz de ocupação B em um porto *p* preenchendo cada coluna até a linha θ_p, iniciando-se pela coluna da direita e colocando-se em primeiro lugar os contêineres cujos destinos são os portos mais distantes. A linha θ_p é calculada pela equação (7), de modo idêntico ao cálculo efetuado na regra **Rc2.**

A utilização de regras de carregamento e descarregamento possui algumas vantagens tais como:

- A facilidade de incorporar conhecimento prévio do planejador sob a forma de regras.
- Todas as matrizes de ocupação produzidas pelas regras são factíveis. Isto facilita e garante que todas as soluções obtidas pelo método heurístico sejam factíveis. Neste trabalho são comparadas entre si soluções obtidas com as metaheurísticas *SA* e *BS*.
- A codificação da solução que determina como será realizado o carregamento e o descarregamento de um navio para N portos é um vetor de tamanho N-1. Esta representação é muito mais compacta se comparada com outras abordagens da literatura, como por exemplo, a utilizada em AVRIEL *et al*., (2000).

4. Abordagem via *Simulated Annealing*

A metaheurística *Simulated annealing (SA)* é um algoritmo de busca local capaz de escapar de ótimos locais, permitindo movimentos para soluções com valor da função objetivo piores que o valor da função objetivo da solução corrente. A facilidade de implementação, as propriedades de convergência e o uso de movimentos de "subida" (hill-climbing) com o objetivo de escapar de ótimos locais contribuíram para que se tornasse uma técnica muito popular em Otimização Combinatória (HENDERSON *et al.*, 2003). As origens do algoritmo se fundamentam na Mecânica Estatística, a idéia da metaheurística tendo sido inspirada no processo de recozimento de metais e vidro, o qual assume uma configuração de baixa energia quando inicialmente aquecido e então resfriado lentamente (BLUM et al., 2008). O algoritmo *SA* foi inicialmente apresentado como um algoritmo de busca para problemas de Otimização Combinatória em (KIRKPATRICK *et al*., 1983).

Apresenta-se a seguir o algoritmo básico referente à metaheurística *SA*, adaptado de (MCMULLEN e FRAZIER, 2000).

Algoritmo: *Simulated Annealing*
inicio
// s_0: uma solução inicial; T_0: temperatura inicial; α: taxa de resfriamento;
// SA_{max}: o número máximo de iterações para atingir o equilíbrio térmico;
s* ← s_0; { s* : melhor solução}
s ← s_0; { s : solução corrente}
T ← T_0; {T: temperatura corrente}
IterT ← 0; {IterT: número de iterações na temperatura T}
enquanto (T > 0.001) **faça**
inicio
enquanto (IterT < SA_{max}) **faça**
inicio
IterT ← IterT + 1;
Gere um vizinho qualquer s' ϵ N^k (s);

```
        Δ = f (s') - f (s);
        se (Δ ≤ 0)  então
           inicio
              s ← s';
              se (f (s') <f (s*)) então s* ← s';
           fim
        senão
           inicio
              seja x ϵ [0, 1];
              se (x < e^(-Δ / T)) então s ← s';
           fim;
     fim;  // enquanto
     T ← α × T;
     IterT ← 0;
  fim; // enquanto
  retorne(s*);
fim. // SA
```

O algoritmo *SA* aplicado ao PCCTP parte de uma solução inicial, gerada aleatoriamente, a qual corresponde a um vetor de dimensão N-1, onde N é o número de portos. Cada elemento do vetor conterá o índice *k* da *regra_k* escolhida aleatoriamente dentre as 12 possíveis regras_k enumeradas na Tabela 1. Note-se que a regra associada ao N-ésimo porto é irrelevante, dado que neste porto todos os contêineres são descarregados. Para cada solução determina-se o valor da função objetivo aplicando-se o algoritmo de avaliação de soluções apresentado em (AZEVEDO *et al.*, 2010).

Os passos seguintes seguem o esquema do algoritmo tradicional do *SA*. Em uma dada temperatura T, o algoritmo seleciona aleatoriamente um vizinho numa vizinhança da solução corrente. No presente trabalho adotou-se como estrutura de vizinhança a seleção aleatória de um dos portos, para o qual, também de forma aleatória, escolhe-se uma nova regra_k, distinta da regra_k corrente para o referido porto. Computa-se então o valor da função objetivo para o "vizinho" selecionado e a diferença entre os valores da função objetivo para o "vizinho" e para a solução corrente. Caso se verifique uma melhora em relação à solução corrente – caracterizada por um menor valor da função objetivo, o movimento será aceito, ou seja, o "vizinho" se tornará a nova solução corrente. Caso contrário, o movimento poderá ser aceito com uma "probabilidade de aceitação" ($x < e^{-\Delta/T}$) que decresce na medida em que as temperaturas se tornam mais baixas.

Os parâmetros de controle do *SA* são a razão de resfriamento α, o número de iterações para cada temperatura (SA_{max}) e a temperatura inicial T_0. A temperatura inicial, neste trabalho, foi fixada em T_0=100000. Para os demais parâmetros, adotou-se o seguinte esquema de variação para os seus valores: α=0,98 e SA_{max} = 1000 (para T > 1000); α = 0,95 e SA_{max} = 500 (para 100 < T <= 1000); α = 0,92 e SA_{max} = 200 (T <= 100). Desta forma, há um resfriamento mais lento e um número maior de iterações para as temperaturas altas. E, em temperaturas baixas resfria-se mais rapidamente e gera-se um número menor de vizinhos.

Utilizou-se também na implementação do *SA* um esquema de reaquecimento para elevar o valor de T (uma única vez a cada execução do algoritmo) para um novo valor equivalente a 20% do valor da temperatura inicial, T_0, quando T se torna inferior a 0,01.

5. Resultados obtidos

Para testar o modelo usando as metaheurísticas *SA* e *BS*, foram geradas automática e aleatoriamente 45 instâncias. Para cada instância é gerada uma matriz de transporte T, tal que a

capacidade do navio não será excedida em nenhum porto, isto é, o valor de θ_p dado pela equação (7) deve ser menor ou igual a R para todo porto ***p***. Isto porque a matriz de transporte é factível se:

$$\sum_{i=1}^{p}\sum_{j=p+1}^{N} T_{ij} \le R \times C \text{ para todo p} = 1,\text{K} \ , \text{N} \qquad (8)$$

onde: R é o número de linhas e C é o número de colunas da matriz de ocupação B.

De acordo com AVRIEL *et al.* (1998) foram gerados três tipos de matriz de transporte: 1-Mista, 2-Longa distância e 3-Curta distância. Uma matriz do tipo 3 se refere ao transporte de contêineres que vão percorrer poucos portos antes de serem desembarcados. Já uma matriz do tipo 2 se refere a contêineres que vão percorrer muitos portos antes de serem desembarcados. Uma matriz do tipo mista combina os dois tipos anteriores. As instâncias foram classificadas de acordo com a quantidade de portos a serem percorridos, o tipo da matriz de transporte e a capacidade do navio. Neste trabalho foram considerados navios com as seguintes dimensões (RxC): 6x50, 6x100 e 6x150, resultando, respectivamente, nas seguintes capacidades máximas: 300, 600 e 900 contêineres. As instâncias empregadas em (RIBEIRO e AZEVEDO, 2010) e utilizadas nos testes encontram-se disponíveis em: https://sites.google.com/site/projetonavio/home.

A Tabela 2 mostra os resultados obtidos com o *BS* e com o *SA*, utilizando **β=12** (número de regras) para as 45 instâncias. A tabela inclui as colunas relacionadas a seguir: **I**: o número da instância; **RxC**: a capacidade do navio em termos do número de linhas e colunas; **M**: o tipo da matriz de transporte; **N**: o número de portos; **NMin**: o número mínimo de movimentos a serem realizados com os contêineres; **F.O.**: os valores da função objetivo em termos do número total de movimentos realizados com os contêineres até a chegada no último porto; e **T**: tempo computacional em segundos. Note-se que os valores de **F.O.** e **T** são apresentados tanto para o *BS* como para o *SA***.** Note-se ainda que os resultados apresentados para o *SA* correspondem à melhor solução obtida com 20 execuções para cada instância. Em 60% das instâncias testadas (27 instâncias) a solução média em 20 execuções do SA e a melhor solução encontrada pelo SA foram iguais. Para as demais instâncias, a diferença entre a solução média e a melhor solução foi inferior a 1% em 17 instâncias e igual a 2,51% em uma instância. Estes dados confirmam a robustez do SA aplicado ao PCCTP. O número mínimo de movimentos é obtido multiplicando-se por dois o valor do somatório de $\boldsymbol{T_{ij}}$ calculado de acordo com a equação (8). Observa-se que esse valor é o limitante inferior para o número total de movimentos a serem realizados ao longo do percurso do navio. Os resultados obtidos do *BS* foram obtidos com um programa desenvolvido em Matlab 7.0 e extraído de (RIBEIRO e AZEVEDO, 2010), e os do *SA*, com um programa desenvolvido em C (DevC++, versão 4.9.9.2, http://www.bloodshed.net/), executado num computador Intel Core 2Duo 2.20 GHz (E4500), 2GB RAM, Windows XP Versão 2002 (SP 3).

					BS		SA	
I	**RxC**	**M**	**N**	**NMin**	**F.O.**	**T(s)**	**F.O.**	**T(s)**
1	6x50	1	10	1322	1366	39.27	1356	32.09
2	6x50	2	10	750	1050	40.51	**1002**	25.64
3	6x50	3	10	2282	2282	38.61	2282	32.66
4	6x50	1	15	1580	1730	170.90	**1700**	55.50
5	6x50	2	15	778	980	162.47	**974**	42.20
6	6x50	3	15	3024	3024	143.94	3024	51.56
7	6x50	1	20	1990	2204	471.22	**2118**	82.13
8	6x50	2	20	784	962	413.18	**926**	55.70
9	6x50	3	20	4880	4880	553.66	4880	95.32
10	6x50	1	25	1664	1898	1001.31	**1782**	90.08
11	6x50	2	25	944	1206	877.60	**1156**	73.56
12	6x50	3	25	5492	5492	909.81	5492	124.58
13	6x50	1	30	2262	2696	1386.19	**2410**	134.57
14	6x50	2	30	1030	1526	1475.03	**1348**	97.45
15	6x50	3	30	6380	6380	1624.49	6380	162.16
16	6x100	1	10	2878	3010	70.83	**2906**	59.15

					BS		SA	
I	**RxC**	**M**	**N**	**Nmin**	**F.O.**	**T(s)**	**F.O.**	**T(s)**
25	6x100	1	25	4390	4974	1433.76	**4586**	202.12
26	6x100	2	25	2410	3312	1356.90	**3010**	160.67
27	6x100	3	25	12236	12236	1251.10	12236	256.86
28	6x100	1	30	4654	5246	2104.61	**5038**	256.16
29	6x100	2	30	2204	3364	1968.83	**2834**	174.74
30	6x100	3	30	14432	14434	1724.57	14432	334.45
31	6x150	1	10	3222	3496	71.93	**3472**	72.31
32	6x150	2	10	2238	2374	65.55	**2268**	58.63
33	6x150	3	10	5882	5882	67.80	5882	80.59
34	6x150	1	15	4562	4878	272.75	**4754**	125.85
35	6x150	2	15	2212	3130	256.10	**3100**	92.13
36	6x150	3	15	7672	7672	216.13	7672	119.18
37	6x150	1	20	5470	6316	665.46	**5646**	190.64
38	6x150	2	20	3000	4192	605.03	**3900**	147.22
39	6x150	3	20	13190	13190	617.90	13190	244.88
40	6x150	1	25	5516	6082	1281.03	**5888**	254.42

17	6x100	2	10	1436	1698	65.02	1698	47.36
18	6x100	3	10	5354	5354	58.80	5354	65.20
19	6x100	1	15	3532	3874	263.92	**3680**	99.61
20	6x100	2	15	1656	1870	264.53	**1852**	71.93
21	6x100	3	15	6296	6304	299.35	**6300**	107.50
22	6x100	1	20	4138	4622	655.95	**4278**	148.39
23	6x100	2	20	1942	2640	620.55	**2474**	106.82
24	6x100	3	20	8714	8716	583.60	**8714**	169.66

41	6x150	2	25	2868	3294	990.97	**3044**	188.95
42	6x150	3	25	16864	16864	1283.48	16864	348.92
43	6x150	1	30	7130	7538	2482.31	**7394**	328.33
44	6x150	2	30	2104	2336	1743.29	**2276**	165.70
45	6x150	3	30	21342	21342	2292.20	21342	491.20

Tabela 2: Resultados obtidos com o *Simulated Annealing* e com o *Beam Search*

A primeira observação importante acerca dos resultados da Tabela 2 é que o tempo computacional gasto pelo *SA* é menor que o do *BS* tendo em vista a implementação do *SA* em linguagem C. É importante observar que para a formulação dada pelas equações (1)-(6) as instâncias com 30 portos, 6 linhas e 150 colunas são problemas com 24.327.000 variáveis inteiras. Estes foram resolvidos pelo *BS* e pelo *SA* em cerca de 38 minutos e 8 minutos, respectivamente. Para todas as instâncias as soluções obtidas pelo *SA* ou possuem menor número de movimentos ou têm o mesmo número de movimentos que as soluções produzidas pelo *BS*. Em 30 das 45 instâncias, o *SA* obtém soluções melhores que as produzidas pelo *BS* e nas demais consegue produzir soluções com igual número de movimentos. É importante observar que para problemas cuja matriz de transporte é do tipo curta distancia (tipo 3), as soluções encontradas são quase todas ótimas, visto que os valores das F.O são iguais ao número de movimentos mínimos (NMin). Isto ocorre mesmo para problemas grandes, como é o caso da instância de número 45 que tem 30 portos e capacidade para 900 contêineres. Pode-se então concluir que o conjunto de regras adotadas é excelente quando se trata desse tipo de matriz. No caso dos problemas cuja matriz de transporte é do tipo mista (tipo 1) as soluções encontradas estão bem próximas do limite mínimo. Ou seja, pode-se concluir que o conjunto de regras adotadas é bastante eficiente também para este tipo de matriz. No caso das instâncias com matriz de transporte do tipo longa distância (tipo 2) a diferença em relação ao valor NMin (menor número de movimentos) é bem maior. Estes resultados indicam a necessidade de se incorporar ao sistema um número maior de regras que levem em consideração a disposição dos contêineres que permanecerão um longo período de tempo dentro do navio. Agradecimentos à FAPESP pelo apoio financeiro através dos processos 2009/15107-0, 2010-16517-4, 2010/16622-2 e 2011/01667-3.

6. Conclusões e Trabalhos Futuros

Este artigo tem duas inovações, a saber: a primeira delas é a apresentação de uma nova regra de descarregamento (**rd2)** e suas conseqüentes combinações (regras), incrementando as possibilidades de representação de soluções para o problema de carregamento de contêineres em terminais portuários (PCCTP). Estas novas regras possibilitaram realizar novos estudos a partir de um modelo anteriormente apresentado em (RIBEIRO e AZEVEDO, 2010). Vale observar que a representação por regras permite reduzir consideravelmente o número de variáveis do problema, reduzindo assim a quantidade de informações necessárias para se representar uma solução, mesmo com a introdução de novas regras. Na formulação dada pelas Eqs. (1)-(6) as informações devem ser armazenadas num vetor de tamanho $(R\times C)\times(N+N^3)$. Na representação aqui utilizada o tamanho do vetor é N-1. Tendo em vista que R×C expressa o número de contêineres que podem ser armazenados em um navio e que N representa o número de portos, a representação da solução aqui utilizada permite uma enorme redução do tempo computacional para instâncias com grande número de portos. Para tanto, a representação emprega regras que definem como será carregado e descarregado o navio, regras estas que garantem que as soluções obtidas sejam sempre factíveis.

A segunda inovação é a implementação do modelo em um ambiente de aplicação de um algoritmo do tipo *SA*, cujas soluções apresentaram, em todas as intâncias, valores melhores ou iguais aos resultados fornecidos pelas soluções produzidas pelo *BS* para o novo conjunto de regras criado e detalhado neste trabalho. Como trabalhos futuros estuda-se a implementação de

novos métodos heurísticos e a introdução de novas regras com o intuito de reduzir ainda mais os valores das soluções (movimentos dos contêineres).

Referências

AMBROSINO, D., SCIOMACHEN A.,TANFANI, E. A decomposition heuristics for the container ship stowage problem, *J. Heuristics*, v.12, p. 211–233, 2006.

AVRIEL, M.; PENN, M.; SHPIRER, N.; WITTENBOON, S. (1998), Stowage planning for container ships to reduce the number of shifts, *Annals of Operations Research*, v. 76, p. 55-71.

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. ; RIBEIRO, C. M. ; DEUS, N. M. R. (2010) Resolução do Problema de Carregamento e Descarregamento de Contêineres em Terminais Portuários via Algoritmo Genético. Revista INGEPRO, v. 2, p. 38-51, 2010.

BISCHOFF, E.E. & RATCLIFF, M.S.W. (1995). Issues in the development of approaches to container loading", *Omega*, **4**, 377-390.

BLUM, C.; ROLI, A. (2008) Hybrid metaheuristics: an introduction. In: BLUM, C.; **AGUILERA, M. J. B.; ROLI, A.; SAMPELS, M. (Ed.).** Hybrid metaheuristics: an emerging approach to optimization. Berlin/Heidelberg: Springer, 2008. p. 1-30.

AGUILERA, M. J. B.; ROLI, A.; SAMPELS, M. (Ed.). Hybrid metaheuristics: an emerging approach to optimization. Berlin/Heidelberg: Springer, 2008. p. 1-30.

DELLA CROCE, F.; T'KINDT, V.(2002), A Recovering Beam Search Algorithm for the One-Machine Dynamic Total Completion Time Scheduling Problem, *Journal of the Operational Research Society*, vol 54, pp. 1275-1280.

FOX, M.S., Constraint-Directed Search: A case Study of Job-Shop Scheduling, *PhD. thesis*, Carnegie-Mellon University, USA, 1983.

HENDERSON, D.; JACOBSON, S. H.; JOHNSON, A. W. (2003) The theory and practice of Simulated Annealing. In: Glover, F.; Kochenberger, G. A. Handbook of Metaheuristics. New York: Kluwer Academic Publishers, 2003. pp. 287-320.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. (1983) Optimization by simulated annealing. Science, 220(4598):671–680, 1983.

MCMULLEN, P. R.; FRAZIER, G.V. (2000). A Simulated Annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line. *llE Transactions*, no. 32, 679-686.

RIBEIRO, C. M.; AZEVEDO, A. T. (2010) Resolução do Problema de Carregamento e Descarregamento de Contêineres em Terminais Portuários via Beam Search. In: XLII Simpósio Brasileiro de Pesquisa Operacional, 2010, Bento Gonçalves. Anais do XLII Simpósio Brasileiro de Pesquisa Operacional. Rio de Janeiro: SOBRAPO, 2010. v. 1. p. 1-12.

SABUNCUOGLU, I., BAVIZ, M., (1999), Job Shop Scheduling with Beam Search, *European Journal of Operational Research,* vol. 118, pp. 390-412.

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

VALENTE, J. M. S, ALVES, R. A. F. S., (2005), Filtered and Recovering Beam Search Algorithm for the Early/Tardy Scheduling Problem with No Idle Time, *Computers & Industrial Engineering*, vol. 48, pp. 363-375.