RESOLUÇÃO DO PROBLEMA DE CARREGAMENTO E DESCARREGAMENTO DE CONTÊINERES EM TERMINAIS PORTUÁRIOS VIA BEAM SEARCH

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

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

RESUMO

Neste artigo é apresentado um algoritmo do tipo beam search para resolver o problema de carregamento e descarregamento de contêineres num terminal portuário. 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. Aqui é apresentado um algoritmo do tipo beam search para resolver esse problema. O algoritmo apresentado utiliza uma representação bastante compacta da solução do problema assegurando que elas sejam factíveis.

PALAVRAS CHAVES. Carregamento de Contêiner. Beam Search. Método heurístico Otimização Combinatória.

ABSTRACT

This paper proposes a Beam Search method to solve the Container Ship Stowage Problem. 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 we want 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 and it can be avoided by efficient stowage planning. A key objective of stowage planning is to minimize the number of container movements. Here is presented a beam search method to solve this problem. The method presented uses a very compact representation of the solution and ensures that all solutions are feasible.

KEYWORDS. Container Ship Stowage Problem. Beam Search. Heuristic Method 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 remanejamento requerido 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.

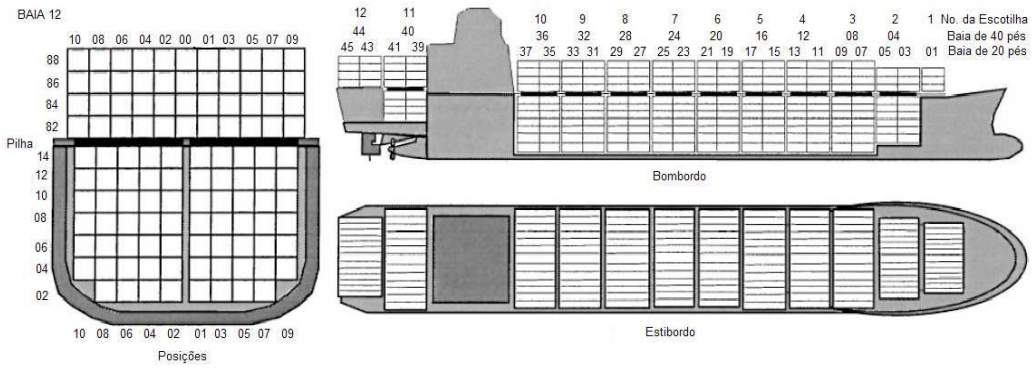

Figura 1: Estrutura celular de um navio. Fonte: WILSON e ROACH (2000).

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. O artigo de AVRIEL et al.,(2000) mostra que este problema é NP-Completo.

Neste artigo será apresentado um algoritmo do tipo Beam Search para a solução do PCCTP. A abordagem aqui apresentada para resolver esse problema é inovadora porque ela 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; seção 4 apresenta o algoritmo de Beam Search; a seção 5 apresenta os resultados computacionais obtidos e a seção 6 as conclusões.

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 at 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 tem o mesmo tamanho. O navio chega no porto 1 completamente vazio e seqüencialmente ele 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).

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

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

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

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

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

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

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 pelo método do Beam Search 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. 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. Lembrando que a linha 5 representa o topo da pilha de carregamento e a linha 1 representa a parte inferior da pilha.

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 com destinos aos portos *j+1, j+2,...,N*. Portanto podem ser identificados 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 é denominado de regra de carregamento e descarregamento, respectivamente. Assim é possível incorporar o conhecimento do planejador, sob a forma de regras, na resolução do problema.

Neste artigo foram criadas 8 regras, sendo quatro para o carregamento (Rc1, Rc2, Rc3 e Rc4) e duas para o descarregamento (Rd1, Rd2). Essas regras foram combinadas de modo que a

combinação de uma regra de carregamento com uma de descarregamento forneça uma regra k, para o porto *j*. A Tabela 1, mostrada a seguir fornece as 8 regras (regras k) a serem utilizadas nos portos *j*, oriundas da combinação das regras de carregamento e descarregamento. Observe na Tabela 1 que a regra 2 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	5	Rc3	Rd1
2	Rc1	Rd2	6	Rc3	Rd2
3	Rc2	Rd1	7	Rc4	Rd1
4	Rc2	Rd2	8	Rc4	Rd2

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 dado na Figura 3. Então, no exemplo da Figura 3, T_{12}= 2, significa que no porto 1, serão embarcados 2 contêineres com destino ao porto 2 e T_{24}=3, significa que no porto 2, serão embarcados 3 contêineres com destino ao porto 4. Observe que a primeira coluna de T tem todos os elementos iguais a zero porque o porto 1 é sempre o porto de onde o navio parte, logo não existe o destino D1. Neste exemplo considerou-se que o navio tem capacidade para transportar 16 contêineres, num roteiro de 5 portos. Suponha agora que em cada porto *i* o navio deve ser carregado de acordo com uma regra de carregamento e ao chegar no porto *j* ele deve ser descarregado de acordo com uma regra de descarregamento. As regras de descarregamento (**Rd)** e carregamento (**Rc)** adotadas para se montar a matriz B de ocupação são apresentadas a seguir.

	D1	**D2**	**D3**	**D4**	**D5**
O1	0	2	5	0	0
O2	0	0	3	2	1
O3	0	0	0	2	2
O4	0	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 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 de descarregamento a matriz B antes de se carregar os contêineres que devem ser embarcados no porto 2 ficaria como mostrada na Figura 4(b). A Figura4(c) mostra os contêineres que estavam no navio com destino aos demais portos e tiveram que ser remanejados. Esses contêineres serão re-embarcados juntamente com os contêineres que serão embarcados nesse porto.

0	0	0	0
5	5	2	2
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	0
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

foram removidos neste porto, mas cujos destinos são os portos *p+1*, *p*+2,...N, serão re-embarcados de acordo com a regra de embarque que foi adotada no 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 ficaria vazia e todos os contêineres que devem seguir viagem iriam para o pátio de remanejamento para que eles sejam recarregados juntamente com os contêineres que devem ser embargados no porto 2. A Figura 5(b) mostra os contêineres do pátio de remanejamento.

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

Figura 5(a): Matriz B antes de **Rd2.**

5	5	5	4	3	3	3	3

Figura5 (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 cujo destino é mais distante. A Figura 6, a seguir, fornece um exemplo da aplicação da regra **Rc1**, considerando que o navio está no porto 1. Observe na Figura 6 que os elementos B_{11}, B_{12}, B_{13}, B_{14} e B_{21} são iguais a 3, e os elementos B_{22}, B_{23} são iguais a 2 porque no porto 1 são embarcados respectivamente 5 contêineres com destino ao porto 3 e dois contêineres com destino ao porto 2.

0	0	0	0
0	0	0	0
3	2	2	0
3	3	3	3

} **B**

Figura 6: 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, começando 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 pegando-se a função teto, resultante da soma do total de contêineres que estavam no navio vindos dos portos anteriores; menos a quantidade de contêineres que serão desembarcados em *p*, mais a quantidade de contêineres que serão embarcados em *p*, dividido pelo número de colunas da matriz B. O número dessa linha pode ser calculado sobre a 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 6. Ao se descarregar os contêineres cujo destino é o porto 2, a matriz de ocupação passa a ser a da Figura 7(a), mostrada a seguir. Aplicando-se agora a regra de carregamento **Rc2**, a matriz de ocupação **B** será preenchida até a linha 3, tal como mostrado na Figura 7(b). O objetivo dessa regra ao se calcular θ_p, é se obter de pilhas de contêineres mais homogêneas e assim tentar diminuir o número de remanejamentos (BISCHOFf e RATCLIFF (1995)).

0	0	0	0
0	0	0	0
3	0	0	0
3	3	3	3

Figura 7(a): Matriz de ocupação **B** no porto 2 antes a aplicação da regra Rc2.

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

Figura 7(b): Matriz de ocupação **B** no porto 2 após a aplicação da regra Rc2.

Regra Rc3: Esta regra é o espelho da regra **Rc1**, isto é, 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 cujo destino é mais distante.

Regra Rc4: Esta regra também é o espelho da regra **Rc2.** Ela faz o preenchimento da matriz de ocupação B em um porto *p* preenchendo cada coluna até a linha θ_p, começando 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 calculado na regra **Rc1.**

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 são factíveis. Neste trabalho a heurística utilizada é o Beam Search, mas futuramente serão utilizadas outras heurísticas para que se possa fazer uma comparação entre elas.
- 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 all., (2000).

4. O Algoritmo do tipo Beam Search

O algoritmo do Beam Search é um método do tipo Enumeração Implícita para resolver problemas de Otimização Combinatória (SABUNCUOGLU E BAVIZ (1999); DELLA CROCE E T'KINDT (2000); FOX (1983); VALENTE E ALVES, (2005)). Pode-se dizer que ele é uma adaptação do método de Branch and Bound onde somente os nós mais promissores de cada nível da árvore de decisões (atribuições) são guardados na memória para serem visitados, enquanto que os demais nós são descartados permanentemente.

Como uma grande parte dos nós da árvore de atribuições é descartada, sem ser analisado, o tempo de execução do Beam Search é polinomial com relação ao tamanho do problema. Em resumo pode-se dizer que o Beam Search é uma técnica de busca em árvore de decisão que em cada nível da árvore é analisado um número fixo de nós e, por conseguinte um número fixo de soluções. O número de nós analisados em cada nível é chamado de **largura da busca** e é denotado por **β.** Para se construir a árvore de decisões é necessário estabelecer as seguintes definições:

(D.1) A árvore é construída por nível e em cada **nível *i*** é feita à atribuição de uma regra ***k*** ao ***i*-ésimo** porto.

(D.2) Os nós que estão no **nível 1** da árvore são chamados de nós semente porque cada um deles vai gerar uma sub-árvore de decisões.

(D.3) A cada nível ***i,*** ao se realizar a atribuição no ***i*-ésimo** porto de uma regra ***k***, são contabilizados os movimentos de carregamento e descarregamento da regra ***k*** adotada no porto ***i*** mais os movimentos de carregamento e descarregamento das regras utilizadas nas (***i*-1**) atribuições anteriores.

Tendo em vista que a árvore possui **N-1** níveis **(D.1)**, onde **N** é o número total de portos, uma solução completa só será obtida ao se definir as atribuições até o nível **N-1** da árvore. No porto **N** (último porto) não é feita a atribuição de nenhuma regra porque neste porto só existe descarregamento. Observe agora que a quantidade de nós da árvore tem uma relação direta com o número de regras que serão utilizadas e com o número de portos. Na realidade o tamanho do problema a ser resolvido, com esta forma de representação, é exponencial com o número de portos. Assim o número total de nós (***Qn***) da árvore de decisões será dado por:

$$Qn = \frac{m(m^{(N-1)} - 1)}{(m-1)}$$, onde ***m***= número de regras e **N**= número total de portos.

O número total de soluções factíveis **(*Fs*)** será dado por: $Fs = m^{(N-1)}$. Assim para o exemplo o utilizado neste artigo, que tem 8 regras e 5 portos, a árvore de decisões teria 32760 nós. Está árvore iria gerar 4096 soluções sendo todas elas factíveis. Mas, vale lembrar que, o Beam Search não vai analisar todas as soluções possíveis, ele utiliza uma largura de busca **β** = ***m***. O método gera então uma árvore de decisões cujo número total de nós é dado por $Qn = m(N-1)$ e o número de total de soluções factíveis analisadas é dado por $Fs = \beta$.

Antes de apresentar o algoritmo proposto, será utilizado um exemplo para mostrar como seria a árvore de decisões (atribuições) se fossem enumeradas todas as soluções possíveis. Neste exemplo serão consideradas somente duas regras para carregamento (**Rc1** e **Rc2**) e uma única regra para o descarregamento, a **Rd1**, cujas combinações resultam em R1 e R2. O número de portos considerado é **N = 5**. Considere também que a matriz de transporte T é a da Figura 3 e que cada navio tem uma capacidade de 16 contêineres.

A Figura 9 a seguir, mostra a árvore de decisões para o exemplo dado, com todas as decisões possíveis. Cada nó da árvore representa a aplicação de uma regra. Assim os nós com o número 1 representam a aplicação da regra 1, (**Rc1** e **Rd1**) e os nós com o número 2 representam a aplicação da regra 2 (**Rc2** e **Rd1**). Observe que a árvore foi construída por nível. O nível 1 tem dois nós. São os chamados nós sementes. A construção da árvore inicia-se no nó semente e a partir daí segue-se para os demais níveis, criando um nó para cada regra e considerando todas as combinações possíveis de regras em cada nível. Como neste exemplo existem duas regras, então existem 2 nós sementes. Cada nó semente gera então uma sub-árvore. Para cada sub-árvore e em cada nível ***i*** são decididas as atribuições de cada uma das regras a um porto ***i*** de modo que: no nível 1 têm-se as atribuições das regras 1 e 2 ao porto p1; no nível 2 têm-se as atribuições das regras 1 e 2 ao porto p2, considerando-se as atribuições que foram feitas no nível 1; no nível 3 são feitas as atribuições das regras 1 e 2 ao porto 3 levando-se em conta as atribuições feitas nos níveis anteriores, e assim sucessivamente.

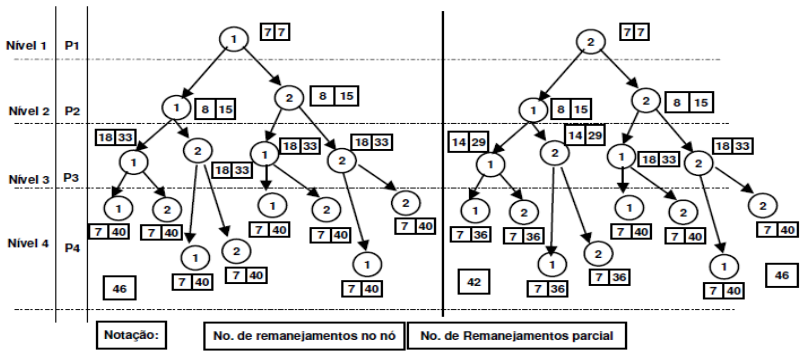

Figura 9: Árvore com todas soluções factíveis considerando os nós semente da regra R_1 (sub-árvore da esquerda) e da regra R2 (sub-árvore da direita) no Nível 1.

Observe na Figura 9 que ao lado de cada nó são apresentados dois valores que indicam (da esquerda para a direita): o número de remanejamentos feitos no nó, decorrente da aplicação

da regra em questão e o número de remanejamentos parcial, isto é, o total de remanejamentos feitos até aquele ponto. O cálculo desses remanejamentos é feito através da matriz de ocupação, ou seja, em cada porto ***i*** é obtida a matriz de ocupação B, levando-se em conta a regra que está sendo empregada naquele porto. A solução completa do problema só é obtida no último nível quando for adicionado ao último remanejamento parcial, o número de contêineres que serão desembarcados no último porto. No nosso exemplo, no porto 5 serão desembarcados 6 contêineres. Então, a melhor solução obtida é, por exemplo, a solução obtida na sub-árvore da esquerda, com a aplicação das regras 2-1-2-1, cujo número total de remanejamentos é 36 + 6 = 42. Observe também que neste problema obteve-se mais 3 outras soluções com o mesmo número de remanejamentos (2-1-1-1; 2-1-1-2 e 2-1-2-2).

Árvore de Soluções Gerada pelo Beam Search

O algoritmo proposto não cria todos os nós da árvore como foi feito no exemplo mostrado e por esta razão não é feita uma enumeração completa de todas as soluções factíveis. Isso evita o crescimento exponencial da árvore. Para se criar os nós, o algoritmo segue algumas regras que são mostradas a seguir:

Inicio
nível = 0
Enquanto (**nível** < número de portos **N-1**) faça:
1º Passo:
Faça **nível = nível + 1**
P1.1. Criar os nós deste novo nível, um nó para cada regra.
P1.2. Para cada nó criado no **nível *i***, faz-se:

- Armazenar a identificação da regra e do porto;
- Armazenar a matriz B de carregamento antes da aplicação da regra. No nível 1, B começa com zero em todas as posições.
- Calcular o número de descarregamento e carregamento relativos à regra aplicada ao nó.
- Atualizar a matriz B de carregamento após a aplicação da regra.
- Calcular o custo parcial da solução, isto é, o número de movimentos de descarregamento e carregamento relativos à atribuição realizada do nó semente até o nó criado;

Se (**nível** < número de portos **N-1**)
Então
2º Passo.
Para cada nó criado no ***1º. Passo*** **(P1.1)** e considerando-se os custos calculados até então, ache uma **solução gulosa** para o problema, através de uma **busca em profundidade** na árvore, partindo deste nó.
3º. Passo
Conserve na árvore, neste nível, os nós que geraram as **β** melhores soluções gulosas. Descarte os demais nós.
Fim Enquanto

Inicialmente o algoritmo encontra-se no **nível zero** de solução, pois as atribuições começarão a serem feitas agora. A seguir faz-se **nível = 1** e começam a serem criados os nós deste nível **(P1.1)**. Os nós do **nível** 1 são os nós sementes e é criado um nó semente para cada regra. Em **(P1.2**), para cada nó criado em **(P1.1)** é atribuída uma regra. A seguir calculam-se o número de movimentos de descarregamento e carregamento do nó, considerando a matriz de carregamento antes da aplicação da regra. A seguir, é feita a atualização da matriz de carregamento. A busca em profundidade realizada no ***2º Passo*** é feita utilizando um algoritmo do tipo guloso e ela visa escolher, no ***3º. Passo***, quais os nós, do ***nível i***, devem permanecer na árvore de soluções factíveis, de modo a respeitar a largura de busca **β**. Por exemplo, se **β=2**, em cada nível vão permanecer somente dois nós. A busca em profundidade é feita do seguinte modo:

Busca em Profundidade do *Passo2*- Algoritmo Guloso.

Para cada nó ***j*** criado no ***1º. passo*** **(P1.1)** fazer uma arborescência em profundidade, a partir deste nó, descendo até o último nível, isto é:

- Seja um nó ***j*** criado no **nível *i*** com custo de solução parcial ***Sj***, calculada até o **nível *i***.
- Escolha para fazer parte desta solução o nó ***q*** do **nível *i+1*** que é factível e que gere a menor solução parcial **S*q***, onde **S*q*= *Sj* +** número de movimentos de descarregamento +número de movimentos de carregamento relativa à atribuição efetuada no nó ***q*.**
- Repetir este processo a partir do nó ***q***, descendo até o último nível. Somente após descer no último nó do ramo é que se terá o custo total de uma solução gulosa.

Como já foi dito anteriormente as soluções gulosas servem para escolher dentre os nós, do **nível *i*** que foram criados no ***1º. passo,*** aqueles que vão permanecer na árvore. Essa escolha é feita da seguinte maneira: Ordenar os nós criados no **nível *i***, em ordem crescente, de acordo com os resultados dos custos das soluções gulosas que cada um deles gerou. Entre os nós do **nível *i*** escolher para permanecer na árvore os **β** nós que geraram as soluções gulosas de menor custo, Cortar os demais nós do **nível *i***, isto é: Se largura de busca é **β=2**, só vão continuar a fazer parte da árvore os dois nós que geraram as duas soluções gulosas de menor custo. O valor da menor solução gulosa calculada no ***2º. Passo***, também é usado como limitante superior (corte) para se gerar ou não os outros nós da árvore, no ***1º. passo***. Este limitante superior deve ser atualizado à medida que forem encontradas soluções gulosas menores. Com isto têm-se como critério de corte o custo da menor solução gulosa, isto é, se o custo parcial da solução no nó que acabou de ser criado, for maior que o custo da melhor solução gulosa, este nó deve ser excluído da solução. Depois de terminada a **Busca em Profundidade** deve-se verificar se o **nível** da árvore de decisões é igual **(N-1)**. Se este for o caso, o algoritmo termina, pois já se chegou ao final da árvore. Senão, o algoritmo volta ao ***1º. passo*** para gerar os nós do próximo nível.

O método de solução do PCCTP apresentado neste artigo combina uma representação compacta da solução do problema com o esquema de regras descrito na Seção 3. Deste modo às soluções obtidas são sempre factíveis e é possível o tratamento de problemas de maior porte em tempo computacional razoável, visto que neste método de solução a dimensão do problema cresce exponencialmente com do número de regras e não com a quantidade de contêineres.

5. Resultados obtidos

Para testar o algoritmo, foram geradas automática e aleatoriamente 45 instâncias. Essas instâncias são classificadas de acordo com o número de portos e o tipo da matriz de transporte. 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:

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

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 geradas três tipos de matriz de transporte: 1-Mista, 2-Longa distância e 3-Curta distância. A matriz do tipo 3 transporta contêineres que vão percorrer poucos portos antes de serem desembarcados, a matriz do tipo 2 transporta contêineres que vão percorrer muitos portos antes de serem desembarcados. A matriz do tipo mista mistura os dois tipos anteriores. As instâncias foram classificadas de acordo com a quantidade de portos a serem percorridos, os tipos de matriz de transporte e capacidade do navio. Aqui foram supostos navios com as seguintes dimensões RxC: 6x50, 6x100, 6x150, resultando respectivamente nas seguintes capacidades máximas: 300, 600 e 900 contêineres.

A Tabela 2 mostra os resultados obtidos com o Beam Search, utilizando **β=8** (número de regras) para as 45 instâncias. Na primeira coluna está o número da instância (I), na segunda coluna estão as capacidades dos navios em termos do número de linhas e colunas (RxC), na terceira coluna o tipo da matriz de transporte (M), na quarta coluna o número de portos N, na quinta coluna o número mínimo de movimentos (NMim) à serem feitos com os contêineres, na sexta coluna os valores da função objetivo (F.O) em termos do número total de movimentos realizados com os contêineres até a chegada no porto de destino, na sétima coluna está o Tempo computacional (T) em segundos, na oitava coluna estão os desvios percentuais (DP) das melhores soluções obtidas em relação ao número mínimo de movimentos**.** O número mínimo de movimentos é obtido multiplicando-se por dois o valor do somatório de T_{ij} calculado de acordo com a equação (8). Observe que esse valor é limitante inferior para o número total de movimentos a serem realizados ao longo do percurso do navio. Os resultados foram obtidos com um programa desenvolvido em Matlab 7.0, Processador Intel Core Duo 1.66 GHz, memória RAM de 2 GB e Sistema Operacional Windows Vista com Service Pack 2.

I	RxC	M	N	NMin	F.O	T (s)	DP
1	6x50	1	10	1322	1366	18.73	3.33
2	6x50	2	10	750	1050	17.76	40.00
3	6x50	3	10	2282	2282	15.12	0.00
4	6x50	1	15	1580	1730	69.32	9.49
5	6x50	2	15	778	980	63.56	25.96
6	6x50	3	15	3024	3024	54.23	0.00
7	6x50	1	20	1990	2204	175.43	10.75
8	6x50	2	20	784	962	147.32	22.70
9	6x50	3	20	4880	4880	155.17	0.00
10	6x50	1	25	1664	1882	321.85	13.10
11	6x50	2	25	944	1198	290.36	26.91
12	6x50	3	25	5492	5492	297.74	0.00
13	6x50	1	30	2262	2700	576.99	19.36
14	6x50	2	30	1030	1554	551.36	50.87
15	6x50	3	30	6380	6380	495.64	0.00
16	6x100	1	10	2878	3010	24.99	4.59
17	6x100	2	10	1436	1698	25.88	18.25
18	6x100	3	10	5354	5354	21.76	0.00
19	6x100	1	15	3532	3874	99.28	9.68
20	6x100	2	15	1656	2498	99.65	50.85
21	6x100	3	15	6296	6304	86.47	0.13
22	6x100	1	20	4138	4806	253.31	16.14
23	6x100	2	20	1942	3028	247.24	55.92
24	6x100	3	20	8714	8716	214.50	0.02
25	6x100	1	25	4390	4932	455.39	12.35
26	6x100	2	25	2410	3418	502.25	41.83
27	6x100	3	25	12236	12236	431.21	0.00
28	6x100	1	30	4654	5246	827.05	12.72
29	6x100	2	30	2204	3286	802.41	49.09
30	6x100	3	30	14432	14434	726.01	0.01
31	6x150	1	10	3222	3640	34.14	12.97
32	6x150	2	10	2238	2274	23.76	1.61
33	6x150	3	10	5882	5882	26.77	0.00
34	6x150	1	15	4562	4878	114.73	6.93
35	6x150	2	15	2212	3446	131.15	55.79
36	6x150	3	15	7672	7672	84.34	0.00
37	6x150	1	20	5470	6316	294.72	15.47
38	6x150	2	20	3000	4194	292.70	39.80
39	6x150	3	20	13190	13190	259.60	0.00
40	6x150	1	25	5516	6006	560.40	8.88
41	6x150	2	25	2868	3294	436.54	14.85
42	6x150	3	25	16864	16864	560.49	0.00
43	6x150	1	30	7130	7538	1025.74	5.72
44	6x150	2	30	2104	2710	754.69	28.80
45	6x150	3	30	21342	21342	1116.36	0.00

Tabela 2: Resultados obtidos com o Beam Search

A primeira observação importante acerca dos resultados da Tabela 2 é sobre o tempo computacional gasto pelo Beam Search. 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 27000 variáveis inteiras. Para estas instâncias o Beam Search consegue produzir soluções com um tempo computacional em torno de 965 segundos. Levando-se em conta que este valor foi obtido com a implementação em Matlab, espera-se então que futuras implementações em linguagem C venham a reduzir ainda mais esse tempo. Observe agora, na coluna 7 (DP) 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, isto é o valor de DP é igual a zero. 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. O desvio percentual (DP) ficou, em média, em torno de

10%, donde se pode concluir que o conjunto de regras adotadas é bastante eficiente para este tipo de matriz. No caso das instâncias com matriz de transporte do tipo longa distância (tipo 2) o desvio percentual da solução ficou, em média, em torno de 31%. Estes resultados indicam a necessidade de se incorporar ao sistema um número maior de regras que levem em consideração a arrumação dos contêineres que permanecerão um longo período de tempo dentro do navio.

6. Conclusões e Trabalhos Futuros

Este artigo tem duas inovações, a saber: a primeira delas é que ele apresenta uma nova maneira de se representar às soluções para o problema de carregamento de contêineres em terminais portuários (PCCTP). Esta nova representação permite reduzir consideravelmente o número de variáveis do problema reduzindo assim a quantidade de informações necessárias para se representar uma solução. 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 e as regras garantem que as soluções obtidas são sempre factíveis.

A segunda inovação é que a utilização de regras de carregamento e descarregamento permite a incorporação, no sistema computacional, do conhecimento do profissional responsável por esse serviço no porto. Esta incorporação é feita através da criação e implantação de novas regras. Esta maneira de representar o PCCTP em conjunto com a aplicação de regras, é uma alternativa promissora na resolução deste problema, até porque assim como se podem acrescentar novas regras, também se podem utilizar outras heurísticas, como por exemplo, os algoritmos genéticos. Futuramente pretende-se fazer uma abordagem tridimensional da matriz de ocupação do PCCTP para se levar em conta o centro de massa do navio.

Agradecimentos ao suporte financeiro do CNPq e da FAPESP a este trabalho.

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.

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

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.

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.