ISSN 2175-6295 Rio de Janeiro- Brasil, 12 e 13 de agosto de 2010

COMPARAÇÃO DE HEURÍSTICAS NA RESOLUÇÃO DO PROBLEMA DE CARREGAMENTO E DESCARREGAMENTO DE CONTÊINERES EM TERMINAIS PORTUÁRIOS VIA REPRESENTAÇÃO POR REGRAS

Aníbal Tavares de Azevedo (UNESP) anibal@feg.unesp.br

Cassilda Maria Ribeiro (UNESP) cassilda@feg.unesp.br

Nayara Melissa Reis de Deus (UNESP) nayara.ml@gmail.com

Fábio Mascagna Bittencout Lima (UNESP) fabim_bittencourt@hotmail.com

Resumo: Neste artigo apresentamos uma nova representação para o problema de carregamento de contêineres em terminais portuários (PCCTP) que consiste em determinar como carregar e descarregamento um conjunto de contêineres de um navio porta-contêiner (containership), respeitando restrições operacionais relacionadas aos contêineres e a estrutura do navio e visando a minimização do número de movimentos. Como este problema é NP-Completo (AVRIEL et al. 2000) é aconselhável a utilização de métodos heurísticos. São propostos e comparados os desempenhos de dois métodos empregados para resolver o PCCTP: um algoritmo genético e um Beam Search. A representação das soluções aqui apresentado tem a grande vantagem de ser bastante compacta e assegurar a geração de soluções factíveis, além de possibilitar a inserção de conhecimento do tomador de decisões.

Palavras-chave: Carregamento de Contêineres em Navios; Algoritmo Genético; Beam Search.

Abstract: This paper presents a new representation for the Container Ship Stowage Problem (CSSP). 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. Although this problem is NP-Complet, here is presented and compared a genetic algorithm and a beam search method to solve this problem. The method presented uses a very compact representation of the solution that ensures all solutions are feasible.

Keywords: Container Ship Stowage; Genetic Algorithm; Beam Search.

1. Introdução

A eficiência de um terminal portuário especializado em movimentação de contêineres depende da ordenação e agilidade do processo de lidar com os contêineres, especialmente durante o processo de carregamento dos navios. A estiva e o plano de carregamento associado são determinados fundamentalmente por dois critérios: estabilidade do navio e o número mínimo de remanejo 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 alguns contêineres para descarregar outros posicionados na parte inferior da pilha. A este tipo de movimento denomina-se de remanejamento.

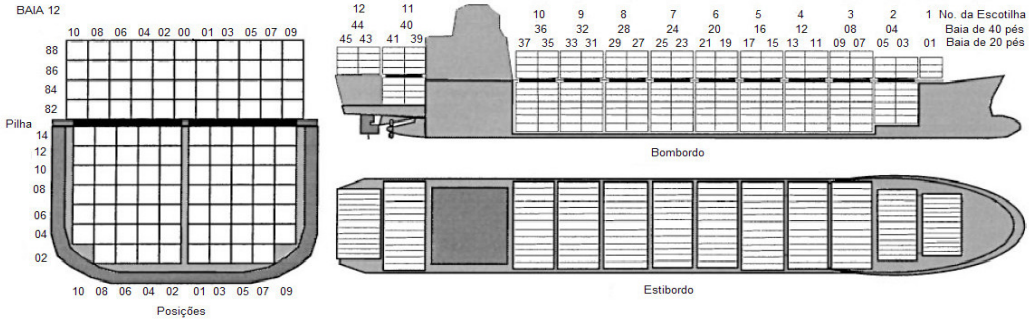

Figura 1: Estrutura celular de um navio. Fonte: WILSON e ROACH (2000).

A situação de remanejamento ocorre com freqüência, uma vez que a ordem de carregamento não é conhecida quando as cargas chegam ao pátio do terminal. No entanto, mesmo quando esta informação é disponibilizada a tempo, o arranjo ideal de contêineres na área de armazenamento é praticamente impossível de ser obtido devido à chegada aleatória de diversas outras cargas. Mais especificamente, o problema de carregamento de contêineres em terminais portuários (PCCTP) consiste na minimização do tempo de carregamento e posterior descarregamento dos contêineres em terminais portuários em um navio porta-contêiner (*containership*), respeitando restrições operacionais relacionadas aos contêineres e ao navio. (AVRIEL et al.,2000) mostrou que este problema é NP-Completo.

Tendo em vista a complexidade do PCCTP, neste artigo será apresentada uma nova representação para a solução do PCCTP, bem como uma comparação entre dois métodos heurísticos que empregam esta representação: um algoritmo genético e um Beam Search. Na seção 2 é apresentada a formulação binária e a por regras do PCCTP. Nas seções 3 e 4 são descritos os métodos implementados: algoritmo genético e Beam Search. Na seção 5 são apresentados os resultados computacionais obtidos e as conclusões.

2. Apresentação do Problema

Um navio porta contêiner tem sua capacidade 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. Os navios têm uma estrutura celular (vide Figura 1) onde são alojados os contêineres. Essas células são agrupadas por seções ou baias (em inglês *bays*) e os contêineres são empilhados nessas seções formando pilhas verticais. Então uma baia é um agrupamento de células, com capacidade de se empilhar um certo número de contêineres. A baia tem então 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 colunas numeradas c= 1, 2, ..., C (coluna 1 é a primeira coluna da

esquerda).

O problema PCCTP que será resolvido aqui consiste em reduzir ao máximo possível o número de remanejamentos de contêineres para um número de portos N. A seguir será apresentada a formulação deste problema como sendo um problema de programação linear inteira com variáveis binárias 0-1. Esta formulação respeita as restrições operacionais relacionadas aos contêineres, navio e pátio do terminal portuário e aparece de modo mais detalhado em (AVRIEL at al , 2000).

2.1 Modelo Matemático

Considere um navio de transporte de 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 da 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 ao porto 1 completamente vazio e sequencialmente 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 $i \geq j$.

A formulação de programação linear inteira do PCCTP é dada pelas Eqs. (1)-(6).

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

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

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

$$y_i(r,c) - y_i(r+1,c) \geq 0 \qquad \begin{array}{l} i = 1,\cdots,N-1, r = 1,\cdots,R-1; \\ c = 1,\cdots,C; \end{array} \qquad (4)$$

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

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

onde: a variável binária $x_{ijv}(r,c)$ é definida de forma 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.

Similarmente, a variável $y_i(r,c)$ possui valor 1 se saindo do porto i o

compartimento (r, c) for ocupado por um contêiner; caso contrário assume valor 0. A função objetivo da Eq. (1) fornece o custo total de movimentação dos contêineres (assumindo que a movimentação de um contêiner possui um custo unitário e igual para todos os portos) em todos os portos. A restrição (2) é a restrição de conservação de fluxo, 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.

Infelizmente, o tamanho que o problema assume com a formulação dada pelas Eqs. (1)-(6) é proibitivo para problemas reais e só pode ser resolvido para problemas pequenos. Além disso, em (AVRIEL & PENN , 1993) é demonstrado que o PCCTP é um problema NP-Completo, justificando o emprego de heurísticas para encontrar boas soluções. Em particular, serão empregados Algoritmos Genéticos.

Outro problema é relacionado com a questão da representação da solução por meio de variáveis binárias. A formulação (1)-(6) é tal que para se representar uma solução de uma instância com R=6, C = 50, P = 30 serão necessárias $R*C*P^3$ variáveis $x_{ijv}(r,c)$, ou seja, 900000 variáveis x, e R*C*P variáveis $y_i(r,c)$, ou seja, 9000 variáveis y. Ou seja, um total de 909000 variáveis para representar uma única solução.

2.2 Representação por Regras

Aqui será apresentada uma alternativa a representação de soluções via variáveis binárias para a resolução PCCTP. Esta representação tem a vantagem de ser compacta e de assegurar que todas as soluções geradas pelo método sejam factíveis. Para tanto, serão empregados dois conceitos: a representação da ocupação do navio através de uma matriz, e a modificação que esta matriz pode sofrer em função das operações de descarregamento e carregamento do navio.

Devido a estrutura celular dos navios, tal como ilustrado na Figura 1, pode-se representar o navio por meio de uma matriz, denominada de matriz de ocupação B, que fornece a quantidade de espaços disponíveis e a localização dos contêineres no navio em cada porto. Em suma, a matriz B fornece o estado do navio em um porto i. Ou seja, cada elemento da matriz B_{rc} representa o estado de uma célula (r,c), isto é se B_{rc}=0 significa que a celula (r,c) está vazia e se B_{rc}=j significa que a celula contém um contêiner cujo destino é o porto j. Assim, no Exemplo da Figura 2, o elemento (1,1) é igual a 5 significando que neste local existe um contêiner que será descarregado no porto 5. De modo análogo, o elemento (4,4)=0 significa que esta célula está vazia. Lembrando que a linha 4 representa o topo da pilha de carregamento e a linha 1 representa a parte inferior da pilha.

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

B

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

Um navio, representado pela matriz B, que chega em um porto i, terá que realizar duas operações:

(i) Descarregamento dos contêineres cujo destino é o porto i.

(ii) Carregamento de contêineres cujo destino são os portos i+1, ..., N.

As duas operações modificam a matriz B e para serem realizadas é necessário

despender um certo número de movimentos. É importante observar que o modo como o navio é carregado em um porto i pode influenciar no número de movimento necessários para se descarregar contêineres nos portos i+1, ..., N. Por exemplo, se um navio, cuja matriz B corresponde aquela apresentada na Figura 2, chega no porto 2, então, se faz necessário descarregar, primeiramente, os contêineres cujo destino é o porto 2, ou seja os contêineres localizados nas posições (3,1), (3,2), (2,3) e (3,4). Observe, porém, que o descarregamento do contêiner (2,3) só pode ser realizado se o contêiner localizado em (3,3) também for realizado, embora este contêiner tenha como destino o porto 3. Todo movimento de descarregamento que resulte em um movimento adicional de carregamento é denominado de remanejamento e é desejável que o remanejamento seja tanto menor quanto o possível.

O aspecto mais importante da abordagem atual é que o navio é representado uma matriz B e que esta matriz sofre duas operações de modificação a cada porto i: descarregamento e carregamento. Observe que existem diversas possibilidades de estratégias para realizar tanto o carregamento como o descarregamento e que uma estratégia sob a forma de algoritmo é denominada de regra de carregamento e descarregamento. A combinação de uma regra de descarregamento com outra de carregamento é denominada de regra para o navio no porto i, ou simplesmente, regra.

Para ilustrar o potencial deste enfoque foram criadas 8 regras, sendo 2 para o carregamento (Rc) e 4 para o descarregamento (Rd). A combinação de uma regra de carregamento com uma de descarregamento fornece a regra k para o porto i.

Tabela 1: Regras *k* a serem utilizadas em cada porto *i*.

Regra k usada no porto i	Regra de carregamento	Regra de descarregamento	Regra k usada no porto i	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

Observe na Tabela 1 que a regra 2 foi obtida utilizando a regra Rd2 para o descarregamento dos contêineres e a regra Rc1 para o carregamento.

A aplicação destas regras em cada porto j vai atualizar a Matriz de Ocupação B no porto i. Vale lembrar que inicialmente a matriz B está com todos seus elementos iguais a zero (sem contêineres) e ela começa a ser preenchida no porto 1. Para melhor ilustrar a utilização das regras, será utilizada a matriz de transporte T, que fornece a quantidade de contêineres que devem ser embarcados em cada porto i com destino a cada porto j, tal como dado na Figura 3. A capacidade e as dimensões adotadas para o navio são as mesmas apresentadas na Figura 2. É suposto que o navio está no porto 1 para as regras de carregamento e porto 2 para as regras de descarregamento.

	D2	**D3**	**D4**	**D5**
O1	2	5	0	0
O2	0	2	3	1
O3	0	0	2	2
O4	0	0	0	1

Figura 3: Matriz de transporte T.

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 Aplicação desta regra considerando a matriz T da Figura 3 e que o navio se encontra no porto 1, resultará na matriz B da Figura 4.

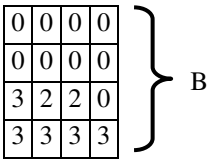

Figura 4: Matriz de Ocupação no porto 1, após a aplicação da **regra Rc1**

Regra Rc2: Esta regra faz o preenchimento da matriz de ocupação B em um porto *p* preenchendo cada coluna até a linha θ_p. A linha θ_p é calculada pelo 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 diretamente na matriz de transporte T, através da equação (7).

$$\theta_p = \frac{\sum_{i=1}^{p} \sum_{j=p+1}^{N} T_{ij}}{C} \tag{7}$$

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 de colunas da matriz de ocupação no porto *p*. Seja por exemplo um navio com capacidade para 16 contêineres e atendimento a 5 portos. Suponha agora que no porto 2, a matriz de transporte seja a da Figura 3. Neste caso tem-se:

$$\theta_2 = (T_{13}+T_{14}+T_{15}+T_{23}+T_{24}+T_{25})/4 \quad \theta_2 = (5+0+0+2+3+1)/4 \quad \theta_2 = 3.$$

Supondo agora que a matriz de ocupação **B** no porto 2 seja a Figura 4. Aplicando-se a regra **Rc2**, a matriz de Ocupação **B** será preenchida até a linha 3, tal como mostrado na Figura 5.

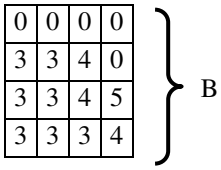

Figura 5: Matriz de Ocupação **B** no porto 2, após a aplicação da regra Rc2.

Regra Rd1: Nesta regra quando o navio chega a um porto *i*, são removidos todos os contêineres cujo destino é i e todos os contêineres que estão acima dos contêineres do porto *i* e cujos destinos são os portos *i+1, ..., N*. 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 B ficaria como mostrada na Figura 6(b).

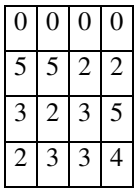

Figura 6(a): Antes da aplicação da regra **Rd1.**

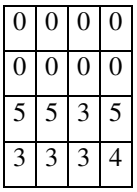

Figura6(b): Depois da aplicação da regra **Rd1.**

Regra Rd2: Nesta regra quando o navio chega ao porto *p*, todos os contêineres são removidos e todas as pilhas são reordenadas. Suponha por exemplo que ao se chegar no porto 2 a matriz de Ocupação seja a da Figura 7(a). De acordo com esta regra a matriz de ocupação ficaria como mostrada na Figura 7(b).

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

Figura 7(a): Matriz B, antes de **Rd2.**

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

Figura7(b): Matriz B depois de **Rd2.**

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.**

As vantagens do emprego de regras são (A) A facilidade de incorporar conhecimento prévio do decisor sob a forma de regras, (B) As regras só podem produzir matrizes de ocupação factíveis, facilitando e garantindo, a obtenção de soluções factíveis por métodos heurísticos e (C) 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). Ou seja, a representação por regras fornece uma representação mais compacta da solução do problema. Por exemplo, para se representar uma solução por meio de variáveis binárias para uma instância com R=6, C = 50, P = 30 serão necessárias $R*C*P^3$ variáveis $x_{ijv}(r,c)$, ou seja, 900000 variáveis x, e R*C*P variáveis $y_i(r,c)$, ou seja, 9000 variáveis y. Ou seja, um total de 909000 variáveis para representar uma única solução. Já empregando a representação por regras o número de variáveis para se representar uma única solução corresponde ao número de portos menos um, ou seja, neste caso, 29. Tal economia em termos de representação só é possível, pois a avaliação de uma solução *s* combina avaliação das regras com a simulação do estado do navio ao longo dos portos após a aplicação de cada regra tal como dado na Figura 8.

```
Avaliação de Solução
Início
  i ← 1,   nmov ← 0
  inicializar(B,T)
  enquanto (i < N ) faça
  Início
      [rc, rd] = extrairRegras(s(i))
      Se (i > 1)
        [aux, B] = descarregar(rd, B, i)
        nmov ← nmov + aux
      fim
      Se (i < N-1)
        [aux, B] = carregar(rc, B, T, i)
        nmov ← nmov + aux
      fim
       i ← i + 1
       retornar nmov
  fim
Fim
```

Figura 8: Algoritmo para avaliação de uma solução por meio de regras.

Os símbolos e funções empregados no algoritmo da Figura 8 são descritos a seguir:

i	- variável contadora que indica o porto atual da simulação.
N	- número total de portos.
s	- Vetor tal que o elemento s(i) contém a regra k a ser aplicada no porto i e modificar a matriz B adequadamente.
nmov	- número de movimentos realizados para carregar ou descarregar o navio ao longo dos N portos.
B	- matriz de ocupação que indica o estado do navio em cada porto i.
rc	- variável que contém o nome da regra de carregamento a ser aplicada.
rd	- variável que contém o nome da regra de descarregamento a ser aplicada.
inicializar	- função que preenche a matriz B com valores iguais a zero.
extrairRegras	- função que define a correspondência entre a regra k contida em s(i) com as regras de descarregamento e carregamento a serem armazenadas nas variáveis rd e rc, respectivamente.
descarregar	- função que aplica a regra de descarregamento contida em rd na matriz B no porto i, e retorna o número de movimentos realizados e B atualizada.
carregar	- função que aplica a regra de carregamento contida em rc na matriz B no porto i, e retorna o número de movimentos realizados e B atualizada.

É importante observar que o Algoritmo da Figura 8 facilita a aplicação de métodos heurísticos na resolução PCCTP tais como Algoritmos Genéticos e Beam Search como destacado a seguir.

3. Algoritmo Genético

Os algoritmos genéticos foram introduzidos por Holland em 1975 (HOLLAND, 1975). Eles baseiam-se nos mecanismos existentes na genética e na teoria da evolução natural, onde as populações têm uma evolução aleatória. Fazendo a associação entre cada individuo e as possíveis soluções do problema. O método parte de uma solução e a seguir ele provoca uma evolução aleatoria da mesma com intuito de se obter soluções melhores para o problema. Os algoritmos genéticos são uma classe de métodos de busca de propósito geral, pois tenta estabelecer um equilíbrio entre dois objetivos aparentemente conflitantes: o aproveitamento das melhores soluções e a exploração do espaço de busca (*exploitation x exploration*). Por esta razão o Algoritmo Genético pode ser aplicado a problemas combinatórios tal como PCCTP cuja dimensão inviabiliza a solução por métodos exatos.

A Figura 9, a seguir, mostra a estrutura geral do Algoritmo Genético. Um Algoritmo Genético mantém uma população de indivíduos $A(t) = \{A_1^t, ..., A_n^t\}$ na iteração (geração) t e cada indivíduo representa um candidato à solução do problema em questão. Na implementação computacional aqui utilizada, o indivíduo é representado através da linha A_i, da matriz A. Cada solução A_i^t é avaliada e produz alguma medida de adaptação, ou fitness, então, uma nova população é formada na iteração t+1 pela seleção dos indivíduos mais adaptados. Alguns indivíduos da população são submetidos a um processo de alteração por meio de operadores genéticos para formar novas soluções. Existem transformações unárias m_i (mutação) que criam novos indivíduos através de pequenas modificações de atributos em um indivíduo (m_i: $A_i \rightarrow A_i$), e transformações de ordem superior c_j (crossover), que criam novos indivíduos através da combinação de dois ou mais

indivíduos (c_j: $A_j \times ... \times A_k \rightarrow A_j$). Após um número de gerações, a condição de parada deve ser atendida, seja porque existe, na população, um indivíduo que represente uma solução aceitável para o problema, seja porque o número máximo de gerações foi atingido.

```
Algoritmo Genético
Início
  t ← 0
  inicialize A(t)
  avalie A(t)
  Enquanto (não condição de parada) faça
  início
      t ← t + 1
      selecione A(t) a partir de A(t-1)
      altere A(t)
      avalie A(t)
fim
Fim
```

Figura 9: Estrutura geral de um algoritmo genético.

O Algoritmo Genético deste artigo combina a representação de soluções com o esquema de regras descrito na Seção 3. Com isto os indivíduos são codificados em uma notação compacta que sempre fornece soluções factíveis. Esta estratégia permite o tratamento de problemas de maior porte em tempo computacional razoável. Maiores detalhes acerca implementação do Algoritmo Genético podem ser observadas em (AZEVEDO et al.,2009).

4. Beam Search

O Beam Search foi usado pela primeira vez pela comunidade de Inteligência Artificial para tratar problemas de reconhecimento de fala (LOWERRE, 1976) e mais recentemente aplicado em problemas de seqüenciamento da produção (DELLA CROCE E T'KINDT, 2002). O algoritmo do Beam Search é um método do tipo Enumeração Implícita para resolver problemas de Otimização Combinatória. 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, isto é, somente alguns poucos nós são selecionados para serem analisados; o tempo de execução do método 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 ao ***i-ésimo*** Porto.

(D.2) Os nós que estão no **nível 1** da árvore são chamados de nós semente pois 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 dois movimentos: movimento de descarregamento e movimento de descarregamento das (***i*-1**) atribuições anteriores.

Tendo em vista que a árvore possui **N-1** níveis **(D.1)**, onde **N-1** é o número de portos, uma solução completa, com a atribuição de ***m*** regras à todos os **N-1** portos só será obtida ao se definir as atribuições até o nível **N-1** da árvore.

O algoritmo utilizado para se construir a árvore é dado na Figura 10.

Beam Search
Inicio
nível ← 0
Enquanto (**nível < N-1**) faça:
nível ← nível + 1
Criar os nós deste novo nível , um nó para cada regra.
Para cada nó criado no **nível *i***, faz-se:
- Armazenar a identificação da regra e do porto;
- Calcular e armazenar o número de movimentos de descarregamento e carregamento relativa à atribuição efetuada no nó.
- Calcular e armazenar o custo parcial da solução, isto é, o número de movimentos de descarregamento e carregamento relativa à atribuição efetuada do nó semente até o nó criado;

fim
Se (**nível < N-1**) **então**
Para cada nó criado considera-se os custos calculados até então, e acha-se uma **solução gulosa** para o problema, através de uma **busca em profundidade** na árvore, partindo deste nó.
fim
Conserve na árvore, neste nível, os nós que geraram as β melhores soluções gulosas.
Descarte os demais nós.
fim

Figura 10: Estrutura geral do Beam Search para o PCCTP.

Maiores detalhes acerca do Beam Search empregado em (RIBEIRO et al.,2009).

5. Resultados obtidos e Conclusões

Para testar o algoritmo, foram geradas automática e aleatoriamente 15 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*, pois a matriz de transporte é factível se:

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

De acordo com (AVRIEL et al. 2000) é possível gerar três tipos de matriz de transporte: 1-Média, 2-Longa, 3- Curta. As instâncias foram classificadas de acordo com a quantidade de portos a serem percorridos e o tipo de matriz de transporte. Para todas as instâncias apresentadas neste artigo, foi suposto que o navio possui as seguintes dimensões: R = 6 e C = 50, resultando em uma capacidade máxima de 300 contêineres.

Na Tabela 3, a coluna Portos indica o número de portos de uma instância, Tipo Matriz ao tipo de matriz conforme descrito anteriormente, e os valores contidos na coluna MIN são os limitantes inferiores para o número total de movimentos a serem realizados ao longo do percurso do navio. Esse valor é obtido multiplicando-se por dois o valor do somatório de T_{ij} calculado de acordo com a Equação (8). A sigla AG denota os resultados do algoritmo genético, descrito na Seção 4.1, para seleção elitista, uma taxa de mutação de 15%, uma taxa de Crossover de 80%, tamanho da população com 50 indivíduos e um máximo de 1000 gerações. A sigla BS denota os resultados obtidos para um Beam Search com largura de busca β = número de regras = 8**.** Para cada instância são apresentados dois valores que, para o caso do algoritmo genético, são valores médios produzidos após dez rodadas ou ainda, para o caso do Beam Search, foram obtidos com uma execução:

• F.O representa o valor da função objetivo em termos de número médio do total de movimentos realizados pelo navio ao percorrer todos os portos;

• Tempo é a média do tempo computacional gasto, em segundos.

Os resultados da Tabela 2 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.

Tabela 2: Resultados do Algoritmo Genético e do Beam Search.

Instância	Portos	Tipo Matriz	MIN	AG		BS	
				FO	Tempo[s]	FO	Tempo[s]
1	10	1	1322	1438.0	147.43	1500	2.95
2	10	2	750	1236.0	143.14	1236	2.37
3	10	3	2282	**2288.0**	151.72	2292	2.71
4	15	1	1580	**2308.0**	217.30	2312	8.14
5	15	2	778	**1194.4**	210.03	1254	7.83
6	15	3	3024	**3028.0**	217.65	3208	8.30
7	20	1	1990	2810.2	285.07	**2678**	20.26
8	20	2	784	1059.2	271.72	**1026**	17.91
9	20	3	4880	4880.0	294.91	4880	22.14
10	25	1	1664	2237.2	343.82	**2112**	36.34
11	25	2	944	1673.0	331.44	**1538**	32.92
12	25	3	5492	**5536.8**	364.24	5546	41.50
13	30	1	2262	3235.0	415.66	**3104**	66.94
14	30	2	1030	**1975.4**	404.37	2004	63.20
15	30	3	6380	6384.0	428.99	6384	70.36

É importante observar, na Tabela 2, que para a formulação dada pelas equações (1)-(6) as instâncias com 30 portos são problemas com 909000 variáveis inteiras (30 portos, 6 linhas e 50 colunas). Para estas instâncias o algoritmo genético consegue produzir boas soluções em menos de 7 minutos. Pode-se observar também que, de forma geral, um aumento de 5 no número de portos a serem percorridos, de uma instância para outra, produz em média um aumento de mais ou menos 1 minuto e 10 segundos no tempo computacional gasto pelo Algoritmo Genético. Por exemplo, em instâncias com 10 portos leva-se 1 minuto e 20 segundos para se obter uma solução, ao passo que em instâncias com 15 portos leva-se, em média, 2 minutos e 20 segundos. Espera-se ainda, que futuras implementações em linguagem C venham a reduzir o tempo computacional de solução pelo algoritmo genético, bem como permitam a comparação com outros métodos com base no tempo computacional.

O Beam Search consegue soluções melhores que as do Algoritmo Genético em um tempo computacional significativamente menor (cerca de 50 vezes menor para as instâncias com 10 portos e 6 vezes menor para as instâncias com 30 portos). O único, porém, é que em termos de qualidade de solução existe uma ligeira vantagem para o Algoritmo Genético, embora não seja possível determinar o melhor método para o PCCTP.

Os resultados indicam que para as instâncias em que a matriz de transporte é do tipo curta distância (tipo 3) as regras são bastante adequadas e produzem resultados muito próximos do limite inferior, e chegam mesmo a sempre atingir este limite independentemente do método aplicado, como no caso para 20 portos.

Já para as instâncias em que a matriz de transporte é do tipo média distância (tipo 1)

e longa distância (tipo 2), tanto o algoritmo genético como o Beam Search apresentaram soluções cujo número de movimentos é bem maior que o limitante inferior. Estes resultados indicam a necessidade de se incorporar ao sistema um número maior de regras que levem em consideração a arrumação de contêineres que permanecerão um longo período de tempo dentro do navio.

Este artigo apresentou pela primeira vez uma comparação de métodos que empregam a representação de soluções por regras para o PCCTP. Esta nova representação permite reduzir a quantidade de informações necessárias para se representar uma solução de um vetor de tamanho R×C×P para um vetor de tamanho P-1. Sabendo-se que R×C expressa o número de contêineres que podem ser armazenados em um navio e P o número de portos esta representação, portanto, representa uma enorme economia em tempo computacional. Para tanto, a representação emprega regras para definir como será carregado e descarregado o navio. Estas regras garantem o uso de movimentos factíveis no carregamento e descarregamento dos navios, produzindo, portanto, sempre soluções factíveis. A utilização de regras permite, também, uma incorporação mais fácil do conhecimento do decisor no sistema computacional e o emprego de heurísticas. Por fim foram aplicados algoritmos genéticos e Beam Search em conjunto com a representação que emprega regras e se mostraram uma alternativa promissora na resolução deste problema.

Agradecimentos

Este trabalho contou com o suporte financeiro da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

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. Container ship stowage problem, *Computers and Industrial Engineering*, v. 25, p. 271-274, 1993.

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

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

AZEVEDO, A.T., RIBEIRO, C. M., DEUS, N.M.R., Resolução do Problema de Carregamento e Descarregamento de Contêineres em Terminais Portuários via Algoritmo Genético, XVI Simpósio de Engenharia de Produção, p. 1-12, 2009.

DELLA CROCE, F.; T'KINDT**,** V.**,** 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, 2002.

HOLLAND, J. H., Adaptation in natural and artificial systems. The University of Michigan Press, 1975.

LOWERRE, B. T., *The HARPY Speech Recognition System*, PhD. thesis, Carnegie-Mellon University, USA, 1976.

MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd edition, Springer-Verlag, 1996.

RIBEIRO, C. M., AZEVEDO, A.T., LIMA, F.M.B., Resolução do Problema de Carregamento e Descarregamento de Contêineres em Terminais Portuários via Beam Search, XVI Simpósio de Engenharia de Produção, p. 1-12, 2009.

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.