**INGEPRO – Inovação, Gestão e Produção**
Outubro de 2010, vol. 02, nº. 10
ISSN 1984-6193
www.ingepro.com.br

# Resolução do Problema de Carregamento e Descarregamento de Contêineres em Terminais Portuários via Algoritmo Genético

Anibal Tavares de Azevedo <atanibal@gmail.com>
Cassilda Maria Ribeiro <cassilda@feg.unesp.br>
Nayara Melissa Reis de Deus <nayara.ml@gmail.com>

*Resumo: Neste artigo é apresentado um Algoritmo Genético 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 Genético para resolver esse problema. O algoritmo apresentado utiliza uma representação bastante compacta da solução do problema assegurando que elas sejam factíveis.*
*Palavras-chave: Algoritmo Genético; Carregamento de Contêiner em Navios; Otimização Combinatória*

# Solving the Container Ship Stowage Problem by Genetic Algorithm

*Abstract: This paper proposes a Genetic Algorithm 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 Genetic Algorithm method to solve this problem. The method presented uses a very compact representation of the solution and ensures that all solutions are feasible.*
*Keywords: Genetic Algorithm; Container Ship Stowage; Combinatorial Optimization.*

## 1. Introdução

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.

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.

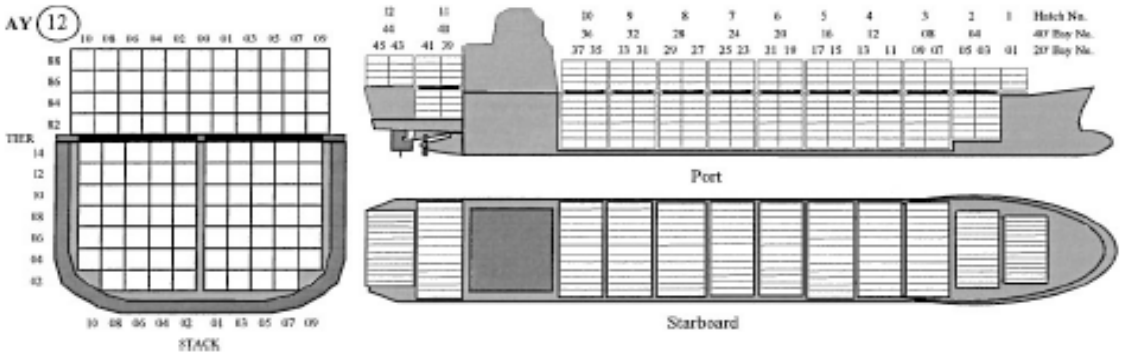


FIGURA 1: Estrutura celular de um navio. Fonte: WILSON e ROACH (2000).

Neste artigo será apresentado um Algoritmo Genético para a solução do PCCTP, ou seja para a minimização do tempo de carregamento e posterior descarregamento dos contêineres em terminais portuários. (AVRIEL et al.,2000) mostrou que este problema é NP-Completo. Na seção 2 é apresentado o problema a ser resolvido bem como sua formulação matemática. Na seção 3 é apresentada a representação por regras utilizada na implementação do algoritmo genético. Na seção 4 é apresentado o algoritmo genético e seus detalhes de implementação, e na seção 5 são apresentados os resultados computacionais obtidos e na seção 6 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. Nos navios tem uma estrutura celular (vide Figura1) 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. Em geral cada baia tem capacidade para alocar quarenta contêineres de vinte pés. 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).

INGEPRO – Inovação, Gestão e Produção
Outubro de 2010, vol. 02, nº. 10
ISSN 1984-6193
www.ingepro.com.br

O problema PCCTP que será resolvido aqui consiste em reduzir ao máximo possível o número de re-alocações dos contêineres para um certo número de portos N. Define-se re-alocação como sendo o descarregamento temporário de contêineres, da pilha de contêineres, com a finalidade de descarregar, num terminal portuário p, um contêiner que está na parte inferior da pilha. Isto é necessário porque os contêineres que estão numa pilha só podem ser acessados pelo topo. Então um contêiner que está no meio da pilha só pode ser descarregado num determinado porto *p se* os contêineres que estão acima dele forem removidos. 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 e PENN, 1993) e (BOTTER e BRINATI, 1992) e (AVRIEL at al , 1998).

## 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).

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

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

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

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

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

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

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

Uma alternativa a representação de soluções via variáveis binárias para a resolução PCCTP é a representação por regras. 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 4 regras, sendo 2 para o carregamento (Rc) e 2 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 *j*.

| **Regra k usada no porto j** | **Regra de carregamento** | **Regra de descarregamento** |
|---|---|---|
| 1 | Rc1 | Rd1 |
| 2 | Rc1 | Rd2 |
| 3 | Rc2 | Rd1 |
| 4 | Rc2 | 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.

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 2 | 2 | 0 |
| 3 | 3 | 3 | 3 |

} B

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 $\theta_p$. A linha $\theta_p$ 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).

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

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 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.

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 3 | 3 | 4 | 0 |
| 3 | 3 | 4 | 5 |
| 3 | 3 | 3 | 4 |

} B

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). A aplicação conjunta desta regra com a regra Rc1 gera a matriz B tal como mostrada na Figura 6(b).

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 5 | 5 | 2 | 2 |
| 3 | 2 | **3** | **5** |
| 2 | **3** | **3** | **4** |

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

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 5 | 3 | **3** | **5** |
| 5 | **3** | **3** | **4** |

FIGURA 6(b)**:** Depois da aplicação da regra **Rd1** e **Rc1.**

**Regra Rd2:** Nesta regra quando o navio chega ao porto *p*, todos os contêineres são removidos para permitir que todas as pilhas possam ser reordenadas. Suponha, por exemplo, que ao se chegar no porto 2 a matriz de Ocupação seja a da Figura 7(a). A aplicação desta regra em conjunto com Rc1 gera a matriz de ocupação tal 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 | 3 |
| 5 | 5 | 5 | 4 |

FIGURA 7(b)**:** Matriz B depois de **Rd2** e **Rc1.**

As vantagens do emprego de regras são:

- A facilidade de incorporar conhecimento prévio do decisor sob a forma de regras.
- As regras só podem produzir matrizes de Ocupação factíveis, facilitando e garantindo, a obtenção de soluções factíveis por métodos heurísticos (neste trabalho um Algoritmo Genético).
- 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 por exemplo a utilizada em (AVRIEL et all., 2000).

A última vantagem é particularmente importante, pois indica que 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, T] = descarregar(rd, B, i)
       nmov ← nmov + aux
     fim
     Se (i < N-1)
       [aux, B, T] = 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 Be T atualizadas. |
| 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 e T atualizadas. |

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

## 4. Algoritmos Genéticos

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 tem uma evolução aleatória. Fazendo a associação entre cada individuo e as possiveis soluções dos problema. O método parte de uma solução e a seguir ele provoca uma evolução aleátoria 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, segundo (MICHALEWICZ, 1996).

```
Programa Evolutivo
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.

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 coluna $A_i$, 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.

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.

### 4.1 Detalhes da Implementação do Algoritmo Genético

Para a implementação do Algoritmo Genético foram definidos os seguintes componentes que o constituem, isto é:

**(A) Estruturas de dados utilizadas para codificar um indivíduo:**

A primeira coisa a ser definida é como cada indivíduo da população irá representar uma solução para o problema PCCTP. Uma representação compacta da solução pode ser realizada através de um vetor *S* cujo elemento *s*[*j*] de valor igual a *k* indica qual regra foi utilizada para o porto *j* tal como dado na Figura 3. Então, no exemplo da Figura 9, tem-se que no porto 1, foi utilizada a regra 4, no porto 2 a regra 3 e assim sucessivamente.

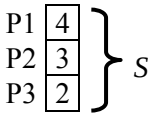


FIGURA 10**:** Codificação do algoritmo genético para relacionar indivíduos e soluções.

Deste modo, uma população com b indivíduos (soluções), pode ser armazenada em uma matriz A de dimensão (N-1*b), onde N é o número de portos e b, representa a quantidade de melhores soluções que serão guardadas. De modo geral pode-se adotar b como sendo igual a cem. Então, cada coluna da matriz representa uma solução (indivíduo) para o problema PCCTP. Assim, cada elemento A [*i,s*] da matriz A, representa a decisão tomada de se aplicar a regra k no porto *i* para o indivíduo (solução) *s* . Então se A [1,1] =3, significa que a regra 3 foi utilizada no porto 1 para o indivíduo 1, se A [2,1]=1 significa que a regra 1 foi utilizada no porto 2 pelo indivíduo 1, e assim por diante. Observe então que cada coluna de A é formada por vetores *S*. A Figura 11, a seguir, mostra a matriz A, para um exemplo com quatro portos, três regras e duas soluções guardadas. Note que o porto 4 não precisa ser representado na matriz A, pois este é o ultimo porto, então, a regra utilizada não irá interferir no resultado.

A

| | Solução 1 | solução 2 |
|---|---|---|
| Porto 1 | 3 | 2 |
| Porto 2 | 1 | 1 |
| Porto 3 | 2 | 2 |

FIGURA 11: Representação das b melhores soluções através de matriz.

**(B) Considerações das restrições**

O conjunto de regras de carregamento e descarregamento descritas na seção 3 é construído de modo a garantir que o estado da matriz B de Ocupação, obtida em cada porto, é factível. Cabe ao Algoritmo Genético apenas definir a seqüência de regras que fornece o menor número de remanejamento dos contêineres ao longo dos N portos, isto é o Algoritmo Genético fornece como resultado a matriz A (vide Figura 11) que contém quais regras de carregamento e descarregamento serão utilizadas em cada em cada porto. De posse da matriz A obtém-se facilmente o número total de movimentos e assim define-se de forma indireta os valores que as variáveis $x_{ijv}$ assumem para cada solução.

**(C) Função de Avaliação (Fitness):**

A função de avaliação ou Fitness é responsável por estabelecer uma relação de ordem entre os indivíduos de uma população para uma dada geração t. Assim, o Fitness deve ser tal que os indivíduos mais aptos tenham maior Fitness e isto deve gerar um valor menor para a função objetivo dada pela Eq. (1). Dessa forma, seja $A_i$ o *i-ésimo* indivíduo de uma população, que representa uma solução, seu Fitness será dado por (8):

$$Fit(A_i) = 1/(1 + f(A_i)) \qquad (8)$$

onde: $f$ é a função de avaliação definida na Eq. (1) e corresponde ao número de movimentos de carregamento e descarregamento de contêineres associado à aplicação de uma seqüência de regras, armazenada no vetor $A_i$, para os *N-1* portos.

**(D) Seleção de indivíduo para a próxima geração:**

O processo de seleção a ser empregado é do "*Roulette Wheel"*, ou seja, é realizado um sorteio aleatório onde a probabilidade $Q(A_i)$ de sorteio do *i-ésimo* indivíduo, de uma população com *b* indivíduos, para participar da próxima geração é diretamente relacionado ao seu Fitness tal como dado por (9):

$$Q(A_i) = Fit(A_i) / \sum_{i=1}^{b} (Fit(A_i)) \quad (9)$$

O melhor indivíduo de todas as gerações sempre é escolhido para a próxima geração.

**(E) Operadores de Crossover:**

Os operadores de Crossover consistem na tentativa de se gerar novos indivíduos para a população combinando as informações de gerações anteriores contidas em seus indivíduos. A definição de qual dos dois operadores será empregado é realizada através de um sorteio aleatório em que cada um dos operadores tem 50% de chance de ser escolhido.

**(E.1) Crossover OX:**

Primeiro é necessário selecionar aleatoriamente dois indivíduos $A_1$ e $A_2$ de uma geração t. A seguir é sorteado número $\delta$ aleatório inteiro dentro do intervalo [1, *N*-2]. Note que o vetor de soluções possui *N-1* elementos. Após isso, os genes de $A_1$ que estão nas posições $\delta$+*i*, até *N-1* são trocados com os genes de $A_2$ que também estão nas posições $\delta$+*i* até *N-1*, de modo que esta troca gere dois novos indivíduos $nA_1$ e $nA_2$ que poderão participar da próxima geração.

**(E.2) Crossover CE:** Este operador consiste em primeiro empregar o Crossover OX para obter dois novos indivíduos $\mathbf{nA_1}$ e $\mathbf{nA_2}$. O segundo passo é, para cada um dos novos indivíduos, escolher dois elementos $\mathbf{nA_1[}i\mathbf{]}$ e $\mathbf{nA_1[j]}$. Todo elemento do vetor de solução, cujo valor é igual ao valor contido em **i** será trocado pelo valor contido em **j** e vice-versa. O mesmo é realizado para $\mathbf{nA_2}$.

**F) Operador de Mutação:**

O operador de mutação simplesmente realiza a mutação de um gene do indivíduo $A_i$ de acordo com uma probabilidade dada por (10):

$$pm_{it} = 0.5 * \left( \frac{Max\left(Fit(A_j^t)\right) - Fit(A_i^t)}{Max\left(Fit(A_j^t)\right) - \overline{Fit(A_j^t)}} \right) \quad (10)$$

Onde: $Fit(A_i^t)$é o fitness do indivíduo $A_i^t$ na geração t, $Max\left(Fit(A_j^t)\right)$ é o maior valor de fitness na geração t e $\overline{Fit(A_j^t)}$ é a média do fitness da população na geração t.

## 5 Resultados Obtidos

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 $\theta_p$ dado pela equação (7) deve ser menor ou igual a R para todo porto *p*, isto porque a matriz de transporte é factível se:

$$\sum_{i=1}^{p} \sum_{j=p+1}^{N} T_{ij} \leq R \times C \text{ para todo p= 1, ..., N} \quad (11)$$

De acordo com (AVRIEL et al. 1998) 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 2 o índice da linha corresponde ao número de portos da instância e o índice da coluna o tipo da matriz de transporte da instância. Os valores contidos na Tabela 2 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 (11). A Tabela 3 mostra os resultados do algoritmo genético, descrito na Seção 4, para uma taxa de mutação de 15%, uma taxa de Crossover de 80%, tamanho da população com 50 indivíduos e um máximo de 1000 gerações. Para cada instância são apresentados dois valores médios produzidos após dez rodadas do algoritmo genético:

- F.O representa o valor da função objetivo em termos de número médio do total de movimentos realizados pelo navio para percorrer todos os portos;
- Tempo é a média do tempo computacional gasto, em segundos, na dez execuções do algoritmo genético.

Os resultados da Tabela 3 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:** Número mínimo de movimentos.

| | **Tipo de Matriz de Transporte** | | |
|---|---|---|---|
| **Portos** | **1** | **2** | **3** |
| **10** | 1322 | 750 | 2282 |
| **15** | 1580 | 778 | 3024 |
| **20** | 1990 | 784 | 4880 |
| **25** | 1664 | 944 | 5492 |
| **30** | 2262 | 1030 | 6380 |

TABELA 3**:** Resultados do algoritmo genético.

| **Portos** | **1** | | **2** | | **3** | |
|---|---|---|---|---|---|---|
| | **F.O** | **Tempo[s]** | **F.O** | **Tempo[s]** | **F.O** | **Tempo[s]** |
| **10** | 1438.00 | 147.43 | 1236.00 | 143.14 | 2288.00 | 151.72 |
| **15** | 2308.00 | 217.30 | 1194.40 | 210.03 | 3028.00 | 217.65 |
| **20** | 2810.20 | 285.07 | 1059.20 | 271.72 | 4880.00 | 294.91 |
| **25** | 2237.20 | 343.82 | 1673.00 | 331.44 | 5536.80 | 364.24 |
| **30** | 3235.20 | 415.66 | 1975.40 | 404.37 | 6384.00 | 428.99 |

A primeira observação importante acerca dos resultados da Tabela 3 é sobre o tempo computacional gasto pelo algoritmo genético. É importante observar que para a formulação dada pelas equações (1)-(6) as instâncias com 30 portos são problemas com 909000 variáveis binárias (30 portos × 6 linhas × 50 colunas). Para estas instâncias o algoritmo genético consegue produzir boas soluções em torno 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.

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, 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), o Algoritmo Genético apresentou 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.

## 6 Conclusões e Trabalhos Futuros

Este artigo apresentou pela primeira vez uma nova representação de soluções para o problema de carregamento de contêineres em terminais portuários (PCCTP). Esta nova representação permite reduzir a quantidade de informações necessárias para se representar uma solução de um vetor de tamanho $R*C*( P^3+P)$ 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, portanto esta representação 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 planejador no sistema computacional.

Por fim foram aplicados Algoritmos Genéticos 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 a 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.

BOTTER, R. C. ; BRINATI, M. A. . Stowage Container Planing: a model for getting optimal solution. In: *International Conference of Computer Applications in the Automation of Shipyard Operation and Ship Design*, 1992, Rio de Janeiro, 1991. p. 217-229.

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

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

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.