PARETO CLUSTERING SEARCH APLICADO AO PROBLEMA DE CARREGAMENTO DE CONTÊINERES EM NAVIOS

Eliseu Junio Araújo

Antônio Augusto Chaves

Luiz Leduino de Salles Neto

Anibal Tavares de Azevedo

Universidade Federal de São Paulo – UNIFESP, Instituto de Ciência e Tecnologia
Rua Talin, 330, CEP 12231-280, São José dos Campos, SP, Brasil
araujo.eliseu; antonio.chaves@unifesp.br

RESUMO

O Problema de carregamento de contêineres em navios (CLPP, do inglês *Container ship Loading Planning Problem*) é um problema importante que surge em operações de contêineres em terminais portuários. Este problema consiste em determinar como organizar os contêineres em um navio de tal forma a minimizar o número de movimentos necessários para descarregar e carregar os contêineres e a instabilidade do navio em cada porto. O CLPP é considerado NP-difícil. Neste trabalho, o método hibrido Pareto Clustering Search (PCS) é implementado para resolver o CLPP e encontrar uma boa aproximação para a Fronteira de Pareto. O PCS procura combinar metaheurísticas e heurísticas de busca local, e a intensificação é realizada somente em regiões promissoras. Os resultados computacionais consideram instâncias trabalhadas na literatura para mostrar a eficiência do PCS.

PALAVRAS-CHAVE. Problema de carregamento de contêineres em navios, Problema biobjetivo, Pareto Clustering Search.

ÁREA. Metaheurísticas.

ABSTRACT

The Container ship Loading Plan Problem (CLPP) is an important problem that appears in the seaport container terminal operations. This problem consists of determining how to organize the containers in a ship in order to minimize the number of movements necessary to unload and load the container ship and the instability of the ship in each port. The CLPP is well known to be NP-hard. In this paper, the hybrid method Pareto Clustering Search (PCS) is proposed to solve the CLPP and get a good approximation to the Pareto Front. The PCS aims to combine metaheuristics and local search heuristics, and the intensification is performed only in promising regions. Computational results considering instances available in the literature are presented to show the efficacy of the PCS.

KEYWORDS. Container ship Loading Plan Problem, bi objective problem, Pareto Clustering Search.

MAIN AREA. Metaheuristics.

1. Introdução

Um terminal portuário depende de uma adequada programação na movimentação de contêineres para que sua eficiência seja mantida e até melhorada. Para tanto, o processo de carregamento de navios merece atenção especial, pois através de seu planejamento adequado, haverá redução do tempo demandado nessa atividade, como consequência diminuindo os custos operacionais. A estiva e o plano de carregamento associado são determinados por dois critérios fundamentais: número mínimo de remanejamento necessário nos diversos portos de destino e a estabilidade do navio (Avriel *et al*., (2000); Wilson e Roach, (2000); Ambrosino *et al*., (2006)).

O primeiro critério tem como base o fato de que diversos navios possuem uma estrutura celular, conforme observado na Figura 1, e os contêineres precisam ser carregados de modo a formarem pilhas verticais, o que traz como consequência, em muitos casos, a necessidade de movimentar os contêineres que se encontram nas partes superiores das pilhas para que os contêineres inferiores sejam descarregados. Este tipo de movimento recebe o nome de remanejamento. Tal atividade se inicia com o navio vazio. No primeiro porto começa a ser carregado. Nos portos sequentes, ele é carregado e descarregado. Por fim deve ser totalmente descarregado no último porto de sua sequência de visitas.

O segundo critério tem como base o fato de que há a necessidade de que conforme a disposição dos contêineres no navio, a instabilidade do navio seja minimizada, ou seja, que a distância do centro de massa real do navio com a disposição dos contêineres ao centro gravitacional seja mínima.

Logo, a partir desse contexto, advém o problema de carregamento de navios em contêineres (CLPP, do Inglês *Container ship Loading Planning Problem*), tratado neste trabalho, que tem por objetivos minimizar ao mesmo tempo o número de movimento dos contêineres e a instabilidade do navio, assim o tornando um problema biobjetivo.

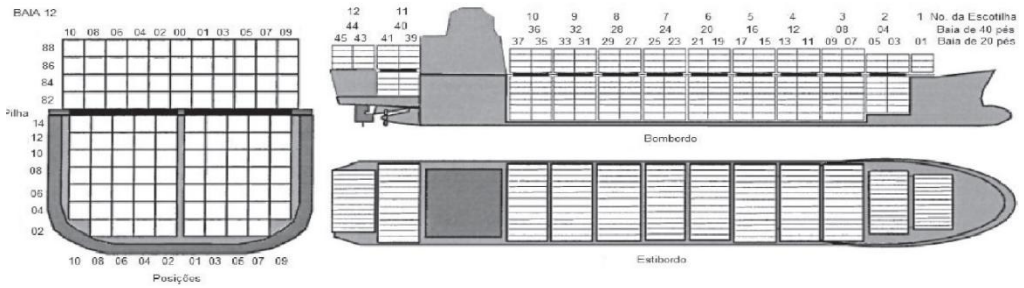

Figura 1 – Estrutura celular de um navio. Fonte: WILSON e ROACH (2000).

Solucionar um problema multiobjetivo consiste em minimizar e/ou maximizar simultaneamente um conjunto de critérios (objetivos).

Nesse contexto, na otimização multiobjetivo não é encontrada apenas uma solução que otimize todos os objetivos, mas uma variedade delas onde nenhuma solução é melhor que a outra em todos os objetivos e uma melhora em um dos objetivos pode ser conseguida unicamente em detrimento de pelo menos um dos outros objetivos. Portanto, o termo "otimizar" significa encontrar soluções com todos os valores dos objetivos que não podem ser melhorados simultaneamente (Arroyo, 2002). Tais soluções são denominadas soluções Pareto-ótimas. Esta abordagem se deve ao fato de que em certos contextos encontramos objetivos conflitantes em que quando um objetivo é melhorado pode ocorrer a deterioração de um outro que também precisa ser otimizado, como neste problema em que a melhora do número de remanejamentos pode implicar na piora da estabilidade do navio. No contexto de navegação marítima, o nível de agitação da maré, por exemplo, pode ditar também qual dos objetivos é mais relevante. Se uma maré se apresenta agitada, o objetivo de minimizar a instabilidade do navio se sobrepõe a de remanejamentos. Caso contrário, a de movimentação de contêineres se torna mais importante.

O CLPP pertence à classe de problemas NP-difícil (Avriel e Penn, 1993). Dentre os métodos utilizados destacam-se o uso de heurísticas (Avriel et al., 1998; Ambrosino et al., 2006; Kaisar, 2006) e de metaheurísticas, seja de forma única (Dubrovsky et al., 2002; Imai et al., 2006) ou

compondo métodos híbridos (Wilson e Roach 1999; Ambrosino et al., 2009).

Para o CLPP multiobjetivo foi proposto o método PCS (*Pareto Clustering Search*) que consiste em combinar adequadamente metaheurísticas e heurísticas de busca local aplicando-as em regiões consideradas promissoras do espaço de busca de soluções detectadas por algum método de agrupamento. Neste trabalho realizou-se um estudo do método *Clustering Search* (CS) (Chaves, 2009), e de sua aplicação ao CLPP para, por fim, desenvolver um algoritmo PCS para o CLPP. Após o desenvolvimento, foram realizados testes computacionais com dados de Azevedo *et al.* (2013) apresentando resultados satisfatórios.

Este trabalho é organizado como se segue. A seção 2 descreve as ideias básicas do CS. A seção 3 introduz a nova proposta, descrevendo em detalhes o PCS aplicado ao CLPP e os resultados computacionais são apresentados na seção 4. Por fim, as conclusões são explanadas na seção 5.

2. Clustering Search

O Clustering Search (CS) (Oliveira *et al*., 2013) é um método híbrido que busca combinar metaheurísticas e heurísticas de busca local, em que a busca é intensificada somente em regiões do espaço de busca que merecem atenção especial (regiões promissoras). O CS introduz uma inteligência e prioridade para a escolha de soluções para aplicar a busca local, ao invés de escolher aleatoriamente ou aplicar busca local em todas as soluções. Consequentemente, é esperado uma melhora no processo de convergência com uma diminuição no esforço computacional, pois há uma aplicação mais racional das heurísticas.

O CS se atém a localizar áreas de busca promissora construindo-as em *clusters*. Um *cluster* é definido por um centro, *c*, que é, geralmente, inicializado aleatoriamente e, posteriormente, tende progressivamente a pontos promissores no espaço de busca. O número de clusters, *NC*, deve ser fixado a priori.

O CS pode ser explicitado em quatro partes conceitualmente independentes: a metaheurística (SM), o componente de agrupamento (IC), um módulo de análise (AM) e a busca local (LS). A Figura 12 apresenta um fluxograma conceitual do CS.

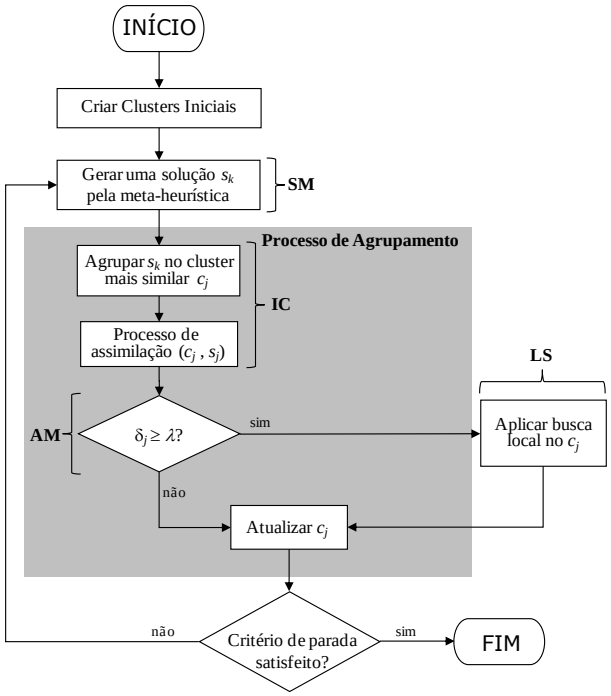

Figura 2 - Componentes do CS
Fonte: Oliveira *et al.* (2013)

O componente SM pode ser implementado por qualquer algoritmo de otimização que gera soluções diversificadas do espaço de busca. Ela trabalha como um gerador de soluções, explorando o espaço de busca através de uma manipulação de um conjunto de soluções, de acordo com sua estratégia de busca específica.

O componente IC procura reunir soluções similares dentro de grupos, mantendo um centro de *cluster* representativo delas. Uma métrica de distância, Δ, é definida, a priori, permitindo uma medida de similaridade para o processo de agrupamento. Por exemplo, em um problema de otimização combinatória, a similaridade pode ser definida como o número de movimentos necessários para alterar uma solução até o centro do *cluster* (Oliveira e Lorena, 2007).

O processo de assimilação é aplicado no centro mais próximo c_j, considerando a nova solução gerada s_k. O método *Path-relinking* (Glover *et al.*, 2000) pode ser usado para gerar um série de pontos, memorizando o melhor ponto avaliado para ser o novo centro.

O componente AM examina cada *cluster*, em intervalos regulares, indicando um provável *cluster* promissor. Uma densidade de *cluster*, $\delta_{j,}$ é uma medida que indica o nível de atividade dentro do *cluster j*. Para simplificar, δ_j pode contar o número de soluções geradas pelo SM e agrupada em c_j. Quando δ_j atinge um limitante λ, indicando que certo padrão de soluções estão sendo gerados por SM, a região deste *cluster* é melhor investigada para acelerar o processo de convergência.

Por último, o componente LS é um módulo de pesquisa interno que provém a exploração de um região supostamente promissora, representada por um *cluster*, intensificando a busca nessa região, aplicando uma busca local no centro desse *cluster*.

3. Pareto Clustering Search aplicado ao CLPP

3.1 Representação por regras

Nesta subseção será apresentada a representação proposta por Azevedo *et al.* (2013) desenvolvida para a solução CLPP. Esta representação tem a vantagem de ser compacta e de assegurar que todas as soluções geradas pelo método sejam factíveis.

Para reduzir as operações de remanejamento é necessário estabelecer regras para o carregamento e descarregamento de contêineres em cada porto que leve em consideração a relação existente entre as duas operações. Para tanto foram criadas oito regras, sendo seis para o carregamento (Re1, Re2, Re3, Re4, Re5 e Re6) e duas para o descarregamento (Rs1 e Rs2) que devem ser combinadas e ao se realizar tal combinação, aplicá-la no porto corrente. Desta forma, obtemos 12 regras combinadas de carregamento e descarregamento. Para cada combinação é atribuído um número, sendo esse seu ID. A Tabela 1 mostra a descrição de cada uma das regras de carregamento e descarregamento.

Tabela 1- *Descrição das* regras de carregamento e descarregamento

Regra	**Descrição**
Re1	Esta regra preenche o navio por baia, começando da primeira até a última linha, da esquerda para a direita, colocando na parte inferior da pilha de cada baia as cargas cujo destino é mais distante.
Re2	Esta regra preenche o navio por linha começando da primeira até a última baia, da esquerda para a direita, colocando na parte inferior da pilha de cada linha as cargas cujo destino é mais distante.
Re3	O navio será preenchido por baia, da direita para a esquerda, colocando na parte inferior da pilha de cada baia as cargas cujo destino é mais distante.
Re4	Esta regra é o espelho da regra Re2, isto é, o navio será preenchido por linha, da direita para a esquerda, colocando na parte inferior da pilha de cada baia as cargas cujo destino é mais distante.
Re5	O carregamento do navio é feito por baia 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 ao longo das baias (preenchendo camadas). A linha θ_p é calculada pegando-se a função teto, resultante da soma do total de contêineres que estavam no navio e foram embarcados nos portos anteriores, menos a quantidade de contêineres que serão desembarcados no porto atual, mais a quantidade de contêineres que a serem embarcados no porto atual, dividido pelo número de baias do navio.
Re6	Esta regra também é o espelho da regra Re5. Ela faz o preenchimento do navio 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 ao longo das baias (preenchendo camadas). A linha θ_p é calculada de modo idêntico ao calculado na regra Re5.
Rs1	Nesta regra quando o navio chega a um porto, são removidos todos os contêineres cujo destino é o porto corrente e todos os contêineres que estão acima dos contêineres do porto corrente.
Rs2	Nesta regra quando o navio chega ao porto, todos os contêineres são removidos para permitir que todas as pilhas sejam reordenadas por alguma regra de carregamento a ser aplicada posteriormente.

Uma solução para o CLPP pode ser representada por um vetor com N elementos (N é o número de portos). Cada elemento é uma regra de carregamento e descarregamento. Cada posição *i* do vetor representa um porto, ou seja, em todos os portos que possuem contêineres para serem colocados a bordo do navio ou que necessitam serem deixados lá haverá remanejamentos (no primeiro porto, o navio está vazio, por isso há somente regra de entrada e no porto final, ele é totalmente esvaziado, assim necessita unicamente de regra de saída). Por exemplo, para o caso representado na Figura 3, na posição *i* = 2 do vetor de ID de regras será aplicado o par com id = 9 das regras, que é representado pelas regras de entrada e saída, Re5 e Rs1, respectivamente.

Regras de entrada	Re2	Re5	-
Regras de saída	-	Rs1	Rs2
ID da regra	4	9	2

Figura 3 - Representação de uma solução do CLPP

3.2 Função Objetivo

O termo $\phi_1(x)$ relativo ao 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 é dado pela Eq. (1).

$$\phi_1(x) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{v=i+1}^{j-1} \sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{d=1}^{D} x_{ijv}(r,c,d) \qquad (1)$$

A variável binária $x_{ijv}(r,c,d)$ é definida de forma que assume o valor 1 se existir um contêiner no compartimento *(r, c, d)* 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, d)* entende-se a linha *r*, a coluna *c* e a baia *d* no compartimento de carga do navio. Em cada um dos N portos é realizada a soma dos movimentos demandados pela aplicação da regra de carregamento e descarregamento determinada no porto corrente.

O cálculo da instabilidade em um porto representa o desvio quadrático entre o centro de massa e o centro geométrico do navio após realizado o carregamento. Assim quanto menor o número obtido do cálculo maior será a estabilidade. Esta formulação matemática é apresentada em Azevedo *et al.* (2013) e é calculada em cada porto da rota do navio.

Em cada porto *i*, *i* = 1, 2, 3, ..., *N* será calculado uma média dos valores da instabilidade após o carregamento no porto *i* e porto *i*-1, com exceção do porto 1.

A formulação matemática do cálculo da instabilidade é mostrada pela Eq.(2), porém mais detalhada pelas Eq.(3)-(5):

$$\phi_2(x) = (xm - R/2)^2 + (ym - C/2)^2 + (zm - C/2)^2 \qquad (2)$$

Onde:

$$xm = \left(\sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{d=1}^{D} (y_i(r,c,d) \bullet (r - 0.5)) \Big/ \sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{d=1}^{D} y_i(r,c,d) \right) \qquad (3)$$

$$ym = \left(\sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{d=1}^{D} (y_i(r,c,d) \bullet (c - 0.5)) \Big/ \sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{d=1}^{D} y_i(r,c,d) \right) \qquad (4)$$

$$zm = \left(\sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{d=1}^{D} \left(y_i(r,c,d) \bullet (d - 0.5) \right) \right) \Big/ \left(\sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{d=1}^{D} y_i(r,c,d) \right) \qquad (5)$$

3.3 Pareto Simulated Annealing

O Pareto *Simulated Annealing* (PSA) (Duh e Brown, 2006) é um método multiobjetivo que se utiliza de uma busca local probabilística. Ele é usado como gerador de soluções (componente SM do CS) para o PCS. O PSA inicia seu processo a partir de um conjunto de soluções inicias aleatórias. O PSA abordado é populacional, ou seja, não é tratada somente uma solução corrente, mas um conjunto de soluções de número fixo ao longo das iterações. É criado um ciclo de iterações que gera aleatoriamente, para cada solução corrente *x* da população um único vizinho *y*.

Quando *y* domina ou é igual a *x*, então a nova solução *y* é aceita. Quando a nova solução *y* é considerada pior que a solução corrente *x*, a solução *y* é aceita com probabilidade menor que 1, que decai ao longo do ciclo do algoritmo. Há também a possibilidade de *y* ser indiferente a *x*. Existem várias regras de agregação para tratar estas situações. Uma destas regras, utilizada na implementação do algoritmo, é definida pela seguinte expressão:

$$P(x, y, T, \Delta) = \min\{1, \exp(\sum_{j} \lambda_j (f_j(x) - f_j(y)) / T\} \qquad (6)$$

Em que:

T é a temperatura;
x é a solução corrente;
y é a nova solução gerada a partir de x;
$\Delta = (\lambda_1, \ldots, \lambda_n)$ é o vetor de pesos; e
f_j é a função objetivo j.

Primeiramente T assume um valor elevado T_0 e após certo número de iterações a temperatura decai gradativamente por uma razão de resfriamento α, tal que $T_k = \alpha * T_{k-1}$, onde $0 < \alpha < 1$.

Para cada solução gerada existe um vetor de pesos associada a ela. Estes vetores são gerados aleatoriamente e depois são modificados iteração por iteração. Desta forma, estes vetores permitem influenciar a direção da busca no espaço objetivo para uma solução gerada em particular. O vetor de pesos associados com cada solução gerada *x* é modificado na tentativa de aumentar a probabilidade de mover *x* na direção do vizinho mais próximo *x'*. Para isso é feito um incremento dos pesos quando *x* é melhor que *x'* e um decremento dos pesos quando *x* é pior que *x'*. Essas alterações nos pesos são feitas através de um fator β, onde $\beta > 1$ e β é próximo de 1.

O PSA efetua uma exploração da fronteira de Pareto, encontrando um grande número de soluções eficientes (não dominadas).

3.4 Processo de agrupamento

O primeiro passo do método PCS é criar *clusters* iniciais, cada *cluster* contém uma solução que representa sua localização no espaço de busca, o centro c_i. Inicialmente, o valor de volume v_i é o mesmo para todos os *clusters*, sendo v_i definido com valor unitário ($v_i = 1$).

Para que os centros dos *clusters* representem diferentes regiões do espaço de busca, foi utilizado um método de criação dos centros baseado na diversidade máxima entre eles.

A medida de diversidade entre dois *clusters* é dada pela distância entre os dois centros desses *clusters*. Seja *N* um conjunto de *n* soluções aleatórias, *d(i, j)*, a distância entre as soluções *i* e *j* de *N*, e *M* um conjunto diverso com *m* soluções ($m < n$). O método para criar os centros dos *clusters* iniciais seleciona *m* soluções não dominadas entre si de *N*, de forma a maximizar a soma das distâncias entre essas soluções, distância essa calculada pela distância de Hamming (Hamming, 1950) que consiste, neste trabalho, em avaliar a diferença entre o vetor das regras dos centros dos *clusters* e o vetor das regras da solução a ser agrupada. A diferença entre os vetores é considerada maior à medida que cada posição *i* de ambos os vetores for diferente.

Neste trabalho, o PCS inicia sua busca por soluções através da meta-heurística multiobjetivo PSA (Duh e Brown, 2006), então agrupa-se as soluções geradas pelo PSA no *cluster* mais

similar, de acordo com a distância de Hamming entre o vetor das regras dos centros dos *clusters* e o vetor das regras da solução a ser agrupada. Para atualizar o centro do *cluster* é realizada a assimilação por caminhos, utilizando o método *Path-relinking* (PR) (Glover, 1996) e usa-se um método de busca local para intensificar a busca no centro do cluster.

Como estrutura de vizinhança do PSA é usada uma troca aleatória de uma das regras do vetor de regras da solução. É escolhida de forma aleatória uma regra nova que será adicionada a um porto selecionado também de forma aleatória, ou seja, será escolhida uma nova regra de carregamento e outra de descarregamento.

Em cada iteração do PCS, uma solução s_k é agrupada no *cluster* mais próximo *j*. O volume de tal *cluster*, δ_j, é acrescentado em uma unidade e o centro c_j deverá ser atualizado por um processo de assimilação.

O processo de assimilação usa o método *Path-relinking*. O procedimento inicia-se através do cálculo da diferença entre o centro c_j e a solução s_k, $\Omega(c_j,s_k)$, ou seja, o conjunto dos movimentos necessários para atingir s_k de c_j. O caminho das soluções é gerado, ligando c_j e s_k. Em cada passo, o procedimento move-se e examina todos os movimentos $m \in \Omega(c_j,s_k)$ da solução corrente *s* e seleciona aquele com melhor custo, ou seja, que melhor domina as outras soluções, aplicando então o melhor movimento à solução *s*. O conjunto de movimentos disponíveis é atualizado. O procedimento termina quando 50% das soluções no caminho foram analisadas. O novo centro c_j é a solução mais eficiente ao longo do caminho. Depois de realizar a assimilação, é preciso realizar uma análise do volume δ_j, verificando se este *cluster* pode ser considerado promissor. Um *cluster* é considerado promissor quando seu volume atinge o limiar λ. O valor de λ foi definido como 15.

Em cada iteração, o PCS é responsável por controlar a fronteira de Pareto e atualizá-la.

3.5 Procedimentos de Busca Local

Caso um *cluster* seja considerado promissor, uma busca local é aplicada no centro do *cluster*. Como componente de Busca Local (componente LS) foi utilizado o método VND (Mladenovic e Hansen, 1997). São usadas duas heurísticas no VND:

- Heurística *2-troca First*;
- Heurística de troca de regra em uma determinada posição.

A heurística *2-troca First* consiste em percorrer o vetor de regras e realizar uma troca entre duas regras localizadas em posições diferentes do vetor. Assim, o vetor é percorrido a partir da posição i=1 e vai até a posição N-1, em que N é o tamanho do vetor. A cada posição i, o vetor é percorrido da posição j=i+1 até N, e é realizada uma troca entre as regras encontradas nas posições i e j. Caso essa troca resulte em uma solução que domine a anterior, ela então é assumida como solução corrente, ou seja, o vetor de regras é atualizado com o novo vetor resultante da troca. A exploração das trocas continua a partir da nova solução.

A heurística de troca de regra consiste em escolher a melhor posição (porto) para se realizar uma troca de regra e realizar a substituição pela melhor regra, ou seja, a posição i que trocada pela regra r produz a solução que domina as demais soluções encontradas.

4. Resultados computacionais

O PCS para o CLPP foi codificado em Java e os experimentos foram conduzidos em um PC com processador Intel core i5, 2.5 GHz e memória de 6 GB de RAM sob plataforma Ubuntu.

Com o objetivo de verificar a eficiência do PCS, fez-se a comparação das soluções obtidas somente através da meta-heurística PSA e das soluções dadas pelo PCS.

Nos testes computacionais foram realizadas comparações de resultados com o trabalho de Azevedo *et al.* (2013). No trabalho citado foi realizado um estudo buscando otimizar os objetivos (número de movimentos e instabilidade) separadamente através da aplicação da metaheurística *Simulated Annealing*, atribuindo pesos para os objetivos e considerando apenas um ou outro objetivo em cada execução do algoritmo ($\alpha = 1$ e $\beta = \alpha - 1$).

Para testar o modelo proposto foram utilizadas as instâncias-teste de Azevedo *et al.* (2013). Nas instâncias testes utilizadas, o navio possui 50 colunas, 6 linhas e 5 baias. As instâncias de 1-3 possuem 10 portos, as de 4-6 possuem 15 portos, as de 7-9 possuem 20 portos, as de 10-12 possuem 25 portos e as de 13-15 possuem 30 portos. Para cada instância, há uma matriz de transporte especifica e com número de contêineres variável.

Foram utilizadas duas métricas para medir o desempenho do PCS:

- Métrica de Cobertura: Sejam P e Q dois conjuntos de soluções, esta métrica calcula a proporção de soluções de Q que são dominadas pelas soluções de P (Zitzler, Deb e L.thiele, 2000). Este valor é denotado por *SC(P,Q)*(do inglês, *Set Coverage*) e é expressado da seguinte maneira:

$$SC(P,Q) = \frac{|\{i \in P |\ \exists\ \ j \in Q\ \ e\ \ i\ \ domina\ \ j\}|}{|Q|}$$

  Quando *SC(P,Q)* = 1 todas as soluções de Q são dominadas por soluções de P. Se *SC(P,Q)* = 0, então nenhuma solução de Q é dominada pelas soluções em P. É possível notar que *SC(P,Q)* não é necessariamente igual a *SC(Q,P)*. Então, deve se calcular ambos os valores para saber a proporção de soluções de P que dominam as soluções de Q e vice-versa.
- Métrica de Espalhamento Máximo: Esta métrica retorna a extensão máxima das soluções no conjunto Q (Zitzler, Deb e L.thiele, 2000). É definida da seguinte forma:

$$ME = \sqrt{\sum_{m=1}^{M} (max_{i=1}^{|Q|} f_m^i - min_{i=1}^{|Q|} f_m^i)^2}$$

  Em que M é quantidade de objetivos. Um maior valor para ME significa uma melhor cobertura do espaço (neste caso, espaço de objetivos).

Seja PSA o conjunto de soluções dadas pela meta-heurística PSA e PCS o conjunto de soluções dadas pelo PCS, a Tabela 2 apresenta os resultados obtidos.

Tabela 2 - Resultados por métricas de desempenho

Instância	***SC*(P_{PSA},P_{PCS})**	***SC*(P_{PCS}, P_{PSA})**	***ME*(P_{PSA})**	***ME*(P_{PCS})**
1	0	1	8225	5091
2	0	1	7799	5010
3	0	1	3970	693
4	0	1	11198	10652
5	0	1	13916	6579
6	0	1	5267	2779
7	0	1	18605	10511
8	0	1	11472	11161
9	0	1	8449	2704
10	0	1	19696	21985
11	0	1	20388	9102
12	0	1	8972	5740
13	0	1	27685	27042
14	0	1	24055	11826
15	0	1	12725	1564

Através dos dados na Tabela 2 nota-se que o PCS possui convergência melhor que o PSA. Em todos os casos, as soluções do PSA são dominadas por soluções do PCS conforme verificado pela métrica SC. Entretanto, o PSA possui melhor cobertura do espaço de objetivos, pois os valores de *ME* para o PSA são maiores do que os valores de *ME* para o PCS, exceto pela instância 10.

As Tabelas 3 e 4 apresentam a comparação do SA de Azevedo *et al.* (2013) com as melhores soluções obtidas pelo PSA e PCS em suas fronteiras de Pareto, considerando os objetivos de

número mínimo de movimentos e menor estabilidade. A coluna *SA* representa a solução encontrada em Azevedo *et al.* (2013). Considere também *PSA* e *PCS* as melhores soluções dadas pelos métodos, Mov. = número de movimentos, Inst. = número para avaliação de instabilidade e Tempo (s), o tempo demandado pelo algoritmo no processo (o artigo de Azevedo *et al.* (2013) não apresenta os tempos computacionais). A Tabela 3 apresenta a comparação dessas soluções priorizando o número de movimentos. A Tabela 4 apresenta a comparação dessas soluções priorizando a instabilidade.

Tabela 3 - Comparação de resultados para número de movimentos

	SA		PSA			PCS		
Instancia	*Mov.*	*Inst.*	*Mov.*	*Inst.*	*Tempo(s)*	*Mov.*	*Inst.*	*Tempo(s)*
1	**7068**	507	8094	14	17,4	7072	**21**	121
2	**4208**	492	4236	42	19,5	4228	**46**	139,1
3	**17088**	583	17206	26	16,9	17108	**32**	118,8
4	10420	221	11482	21	26,5	**10096**	**20**	418,1
5	5082	532	7557	37	21,5	**4954**	**37**	335,4
6	24998	595	25322	31	24,4	**24962**	**98**	388,6
7	10749	202	14934	13	27,2	**10600**	**33**	607,1
8	5458	451	8055	22	32,1	**5182**	**35**	739,7
9	32632	1000	32824	195	27,2	**32614**	**103**	754,3
10	11590	200	18004	11	42,7	**11352**	**39**	1788,2
11	5430	561	13817	53	34	**5428**	**376**	1476,5
12	44078	543	44988	64	41,4	**43855**	**79**	1745,2
13	**12146**	183	23662	36	43,5	12226	**41**	2671
14	5246	168	13721	16	48,3	**5164**	**41**	2990
15	54454	519	56586	74	41,6	**53824**	**124**	2512,3
média	16709,8	450,5	20032,5	43,7	30,9	16577,7	75,0	1120,4

Na Tabela 3, percebe-se que as soluções dadas pelo PSA quanto ao número de movimentos são piores que as soluções de Azevedo *et al.* (2013) nas instâncias 1-15, porém em todas elas, o valor de instabilidade do PSA é melhor que os encontrados na literatura quando se procura priorizar o número de movimentos.

Na Tabela 3, também se percebe que as soluções dadas pelo PCS quanto ao número de movimentos são próximas as soluções de Azevedo *et al.* (2013) nas instâncias 1-3 e 13 e melhores nas demais instâncias, e em todas elas, o valor de instabilidade do PCS é melhor que os encontrados no trabalho citado quando se procura priorizar o número de movimentos.

Tabela 4 - Comparação de resultados para instabilidade

	SA		PSA			PCS		
Instancia	*Mov.*	*Inst.*	*Mov.*	*Inst.*	*Tempo(s)*	*Mov.*	*Inst.*	*Tempo(s)*
1	11122	14	10520	7	17,4	**8406**	**7**	121
2	**6672**	11	10008	7	19,5	6907	**7**	139,1
3	17554	92	17754	7	16,9	**17370**	**7**	118,8
4	**13910**	30	15576	5	26,5	14427	**4**	418,1
5	8098	34	14939	12	21,5	**7999**	**10**	335,4
6	25896	154	30489	14	24,4	**25624**	**13**	388,6
7	16396	13	19326	6	27,2	**16326**	**4**	607,1
8	10022	14	17446	13	32,1	**8038**	**12**	739,7
9	33378	575	36155	55	27,2	**33301**	**50**	754,3
10	**14674**	12	25524	8	42,7	15306	**7**	1788,2
11	9456	106	20550	36	34	**8941**	**29**	1476,5
12	**45120**	218	48532	22	41,4	45300	**20**	1745,2
13	18180	17	27267	10	43,5	**15542**	**9**	2671
14	**8604**	13	24921	10	48,3	10999	**7**	2990
15	55754	299	67474	32	41,6	**54915**	**26**	2512,3
média	19655,7	106,8	25765,4	16,3	30,9	19293,4	14,1	1120,4

Na Tabela 4, percebe-se que as soluções dadas pelo PSA priorizando a avaliação de instabilidade são melhores que as soluções de Azevedo *et al.* (2013) nas instâncias 1-15, e em todas elas, exceto para a instância 1, o número de movimentos do PSA é maior que os encontrados na literatura, quando se procura priorizar a avaliação de instabilidade.

Na Tabela 4, também se percebe que as soluções dadas pelo PCS quanto a avaliação de instabilidade são melhores que as soluções de Azevedo *et al.* (2013) nas instâncias 1-15, e nas instâncias 2, 4, 10, 12 e 14, o número de movimentos do PCS é maior que os encontrados no trabalho citado e nas restantes é menor quando se procura priorizar a avaliação de instabilidade, porém melhores que os valores encontrados pelo PSA tanto em valor de instabilidade, quanto em número de movimentos.

Observando as Tabelas 3 e 4 percebe-se que o tempo de execução do PCS é maior que o PSA, isto ocorre, pois o PCS possui o mesmo procedimento do PSA além de outros métodos. Portanto, esse aumento no tempo de execução já era esperado. Contudo, notou-se nas execuções que o PCS converge mais rapidamente as melhores soluções, mesmo tendo o PSA em seu processo, logo o método se mostra eficiente.

Os gráficos apresentados na Figura 4 mostram as soluções obtidas pelo PCS e PSA para as instâncias 1, 4, 12, e 15 (os demais gráficos das outras instâncias são similares a esses). O eixo das abscissas representa o número de movimentos. O eixo das coordenadas representa a avaliação de instabilidade.

Observando os gráficos é possível visualizar os resultados mostrados nas Tabelas 2 e 3. Nota-se que as soluções geradas pelo PCS possuem menores valores para o número de movimentos (fo1, eixo das abscissas) e menores valores para a instabilidade (fo2, eixo das coordenadas) do que as soluções geradas pelo PSA, e portanto, o PCS gera soluções melhores que o PSA.

5. Conclusão

Neste trabalho foi proposto um algoritmo baseado no método *Clustering Search* (CS) para resolver o Problema de Carregamento de Contêineres em Navios (CLPP), sendo um método multi-objetivo. Logo é realizado uma alteração no CS para aplicá-lo a esse caso, modificando ele para o PCS (Pareto *Clustering Search*). A ideia do PCS é evitar a aplicação de heurísticas de busca local em todas as soluções geradas por uma metaheurística. O PCS detecta regiões promissoras no espaço de busca e aplica busca local somente nessas regiões. Detectar regiões promissoras é uma alternativa interessante, prevenindo a aplicação indiscriminada de heurísticas.

Neste trabalho foram usados dois métodos: uma metaheurística (Pareto *Simulated Annealing* (PSA)) e um método híbrido (Pareto *Clustering Search* (PCS)). A metaheurística PSA foi usada como geradora de soluções para o PCS. A metaheurística também foi colocada em execução, produzindo seus resultados independentemente do PCS e é usado para comparação com os resultados produzidos pelo PCS.

Este trabalho reporta resultados encontrados por PCS e PSA sobre instâncias de Azevedo *et al.* (2011) para o CLPP. O PCS apresentou as melhores soluções quando comparadas as soluções retornadas pelo PSA e as encontradas na literatura. Os resultados mostraram que o PCS é competitivo para resolver o problema de carregamento de contêineres em navios. Porém, o tempo computacional do PCS ainda se mostra muito elevado em relação ao PSA.

Como trabalhos futuros, propõe-se testar outras metaheurísticas para aplicação em conjunto com o PCS, desenvolver outras heurísticas de busca local e estudar novas técnicas de agrupamento de soluções. Além de aplicar o PCS em outros problemas multi-objetivos.

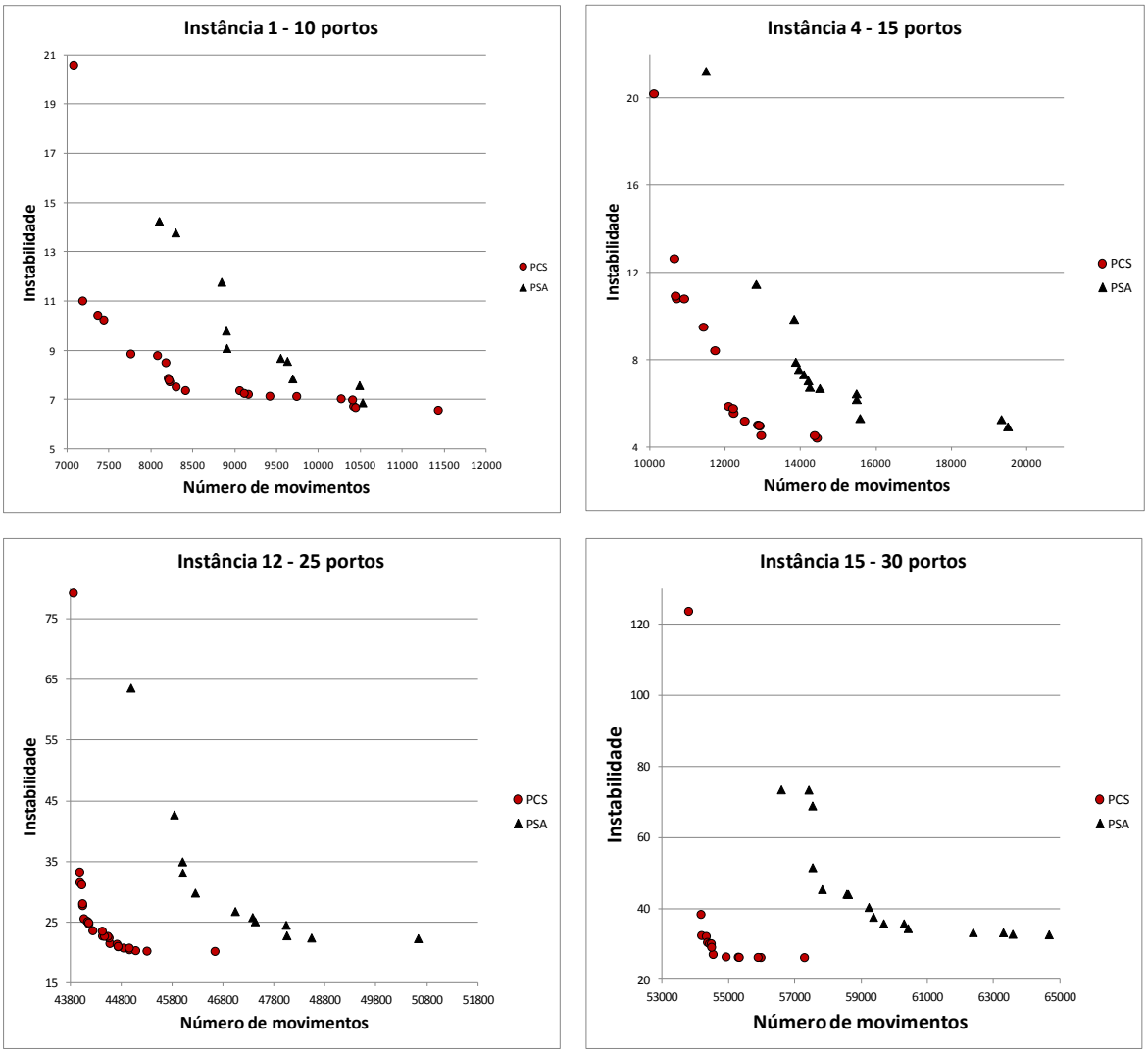

Figura 4 - Fronteira de Pareto produzida pelo PSA e PCS para as instâncias 1, 4, 12 e 15

Referências

AMBROSINO, D., ANGHINOLFI, D., PAOLUCCI, M., & SCIOMACHEN, A. (2009). **A new three-step heuristic for the master bay plan problem**. Maritime Economics and Logistics, 11, 98–120.

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

ARROYO, J. E. C. (2002) **Heurísticas e metaheurísticas para otimização combinatória multiobjetivo.** Tese (Doutorado) - Universidade Estadual de Campinas.

AVRIEL, M.; PENN, M. (1993), **Exact and approximate solutions of the container ship stowage problem**. Computers and Industrial Engineering, 25, 271–274.

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.

AVRIEL, M., PENN, M., SHPIRER, N., & WITTEBOON, S. (1998). **Stowage planning for**

container ships to reduce the number of shifts. Annals of Operations Research, 76, 55–71.

AZEVEDO, A. T.; RIBEIRO, C. M.; LEDUINO, L. S. N.; SILVA, M.P.E., SILVESTRE, M.C. (2013)**. Comparação de heurísticas para a solução do problema de carregamento e descarregamento 3D de navios via representação por regras.** In Lopes, H.S., de Abreu Rodrigues, L.C., Steiner, M.T.A., eds.: Meta-Heursticas em Pesquisa Operacional. 1 edn. Omnipax, Curitiba, PR. 271-288

CHAVES, A.A.; LORENA, L.A.N. (2010) **Clustering search algorithm for the capacitated centered clustering problem**.*Computers & Operations Research*, v. 37, p. 552-558.

DUBROVSKY, O., LEVITIN, G. & PENN, M. (2002). **A genetic algorithm with a compact solution encoding for the container ship stowage problem**. Journal of Heuristics, 8, 585–589.

DUH, J.D.; BROWN, D. G. (2006), **Knowledge-informed pareto simulated annealing for multi-objective spatial allocation.** Science Direct, p. 5-8.

GLOVER, F. (1996), **Tabu search and adaptive memory programing: advances, applications and challenges.** Interfaces in Computer Science and Operations Research, p. 1-75.

HAMMING, R. W. (1950), **Error detecting and error correcting codes. Bell System Technical Journal**, v. 26, p. 147-160.

IMAI, A., SASAKI, K., NISHIMURA, E., & PAPADIMITRIOU, S. (2006). **Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks**. European Journal of Operational Research, 171, 373–389.

KAISAR, E. (2006). **A stowage planning model for multiport container transportation**. Tese de Doutorado, Univ. of Maryland, Maryland, USA.

OLIVEIRA, A. C. M.; CHAVES, A. A.; LORENA, L.A.N. (2013), **Clustering Search**. Pesquisa operacional, v.33 (1), 105-121.

OLIVEIRA, A. C. M.; LORENA, L. A. N. (2004) **Detecting promising areas by evolutionary clustering search**. In: BAZZAN, A. L. C.; LABIDI, S. (Ed.). Advances in Artificial Intelligence - SBIA 2004. Berlin / Heidelberg: Springer, (Lecture Notes in Artificial Intelligence). p. 385-394. 30, 46

WILSON, I. & ROACH, P. (1999). **Principles of combinatorial optimization applied to container-ship stowage planning**. Journal of Heuristics, 5, 403–418.

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

ZITZLER, E.; DEB, K.; L.THIELE. (2000) **Comparison of multiobjective evolutionary algorithms: Empirical results.** Evolutionary Computation, v. 8, p. 173-195.