SEQUENCIAMENTO DE CAMINHÕES PARA UM SISTEMA DE *CROSS DOCKING* ATRAVÉS DA META-HEURÍSTICA *SCATTER SEARCH*

Evandro José da Silva
Programa de Pós-Graduação em Engenharia de Transportes
Escola Politécnica da USP
Av. Prof. Almeida Prado, Travessa 2, n° 83, São Paulo - SP
evandro.jose@ymail.com

Cláudio Barbieri da Cunha
Programa de Pós-Graduação em Engenharia de Transportes
Escola Politécnica da USP
Av. Prof. Almeida Prado, Travessa 2, n° 83, São Paulo - SP
cbcunha@usp.br

RESUMO

Uma estratégia para a redução dos custos com transporte de cargas é o emprego do conceito de *cross docking*, segundo o qual um conjunto de veículos troca cargas diretamente, utilizando no máximo um estoque temporário, que tipicamente não ultrapassa um dia. A eficiência dessa operação depende, entre outros fatores, do sequenciamento dos caminhões, apontado na literatura como um problema NP-difícil. O presente trabalho propõe uma formulação matemática linear para o problema e apresenta uma meta-heurística de busca dispersa (*scatter search*) como método de solução. Esses métodos são aplicados a 35 instâncias de teste nas quais a quantidade de carga trocada entre os caminhões é um dado de entrada. Através de experimentos computacionais demonstra-se a limitação da resolução exata frente ao aumento do número de caminhões envolvidos. Por outro lado, demonstra-se o desempenho da meta-heurística implementada, capaz de fornecer respostas de boa qualidade em tempos computacionais atrativos.

PALAVRAS CHAVE: Scatter search. Cross docking. Sequenciamento.

ABSTRACT

As to reduce transportation costs, *cross docking* is an appealing strategy, by which a set of transportation vehicles exchange their shipments using a temporary storage that often does not exceed one day. An efficient *cross docking* operation depends upon an adequate scheduling of trucks involved but this is not a trivial task since the problem is NP-hard. This work addresses a *cross docking* scheduling problem by means of a *scatter search* metaheuristic and proposes a linear mathematical model. The methods are applied to 35 test instances in which the shipments exchanged by the trucks are an input data. The computational experiments carried out show that the optimization model becomes unsuitable as the number of trucks involved increases. On the other hand, the implemented meta-heuristic demonstrates an attractive performance, by reaching good solutions in a relatively low CPU time.

KEYWORDS. Scatter search. Cross docking. Scheduling.

1. Introdução

Em algumas indústrias os custos logísticos são fatores centrais do negócio. Em outras, o tempo que a mercadoria despende entre produtores e consumidores deve ser o menor possível. Em ambos os casos a técnica de *cross docking* pode trazer grandes benefícios. Tal técnica consiste na troca direta de mercadorias entre veículos de transporte, tipicamente caminhões, utilizando no máximo estoques temporários, que em geral não superam um dia. Um sistema de *cross docking* pode incluir também atividades produtivas complementares como pesagem, troca de embalagem, endereçamento, classificação, para citar algumas. Ainda, a técnica pode contribuir para o processo de consolidação de cargas, de forma especial para os operadores logísticos focados em carga fracionada.

Com efeito, adicional competência operacional é exigida na coordenação dessa troca de mercadorias, desde a sincronização da cadeia produtiva envolvida até a operação propriamente dita. Na operação, uma das questões que surge é o sequenciamento dos caminhões destinados a trocar cargas entre si. O sequenciamento, por sua vez, demonstra uma complexidade oriunda da natureza combinatória do problema, que tende a inviabilizar a resolução por métodos exatos.

O presente trabalho propõe um modelo matemático e apresenta uma implementação da meta-heurística de busca dispersa ou *scatter search (SS)* aplicados ao problema de sequenciamento de caminhões na operação de *cross docking*, considerando uma doca de carga e uma de descarga. Esta abordagem está organizada da seguinte maneira: Na seção 2 apresenta-se uma breve revisão da literatura referente ao tema. Na seção 3 o problema é devidamente definido e um modelo matemático linear é proposto. Na seção 4 apresenta-se a meta-heurística implementada. Na seção 5 discutem-se os resultados obtidos com o *scatter search* e com a programação linear. A seção 6 apresenta as conclusões e recomendações para trabalhos futuros.

2. Revisão de Literatura

Qiu *et al.* (2009) colocam a atratividade do *cross docking* para a cadeia produtiva de alimentos congelados. Segundo os autores, as instalações necessárias à distribuição desse tipo de produto possuem custos elevados e assim, a minimização dos estoques pode trazer grandes benefícios. Além disso, a redução do manuseio contribui para a manutenção da qualidade do produto. Os autores chamam atenção para o fato de que o *cross docking* exige uma infra-estrutura própria, além de uma maior coordenação na cadeia de suprimentos, onde, por sua vez, as tecnologias de informação são fundamentais.

Vários autores trataram o problema do sequenciamento de caminhões em um terminal de *cross docking*. Yu e Egbelu (2008) apresentam nove diferentes heurísticas de sequenciamento comparando-as entre si, e com métodos enumerativos. Essas heurísticas renderam resultados próximos aos ótimos para algumas das instâncias analisadas e bons resultados em outras. Já Vahdani e Zandieh (2009) comparam diferentes meta-heurísticas para o problema do sequenciamento em *cross docking*: algoritmo genético, busca tabu, *simulated annealing*, *eletromagnetism-like algorithm* e *variable neighbourhood search*. Os autores aplicaram técnicas de projeto de experimentos (*design of experiments - DOE*) na calibração das meta-heurísticas, que foram comparadas com as heurísticas apresentadas por Yu e Egbelu (2008) produzindo resultados melhores, porém em tempos computacionais maiores. O problema tratado por Vahdani e Zandieh (2009) considera as restrições e a função objetivo do modelo matemático apresentado por Yu e Egbelu (2008). Tal modelo busca a minimização do tempo total da operação de *cross docking (makespan)*, a partir do ajuste da sequência de caminhões de recebimento e envio e da designação das cargas trocadas entre esses caminhões. Nesse modelo existem apenas duas docas: uma de carga e uma de descarga.

Miao *et al.* (2009) apresentam um problema de designação de caminhões a docas em um sistema de *cross docking* com restrições temporais. Nesse trabalho, o modelo exato, resolvido através do pacote CPLEX, é comparado às meta-heurísticas busca tabu e algoritmo genético. Verificou-se que nas instâncias de médio e grande porte o modelo exato é desvantajoso e até impraticável, enquanto as meta-heurísticas alcançam bons resultados em tempos computacionais adequados para solução de problemas reais. O modelo matemático proposto por Miao *et al.*

(2009) considera um terminal com múltiplas docas, janelas de tempo e capacidade do sistema de *cross docking* em relação ao número total de caminhões processados simultaneamente. O modelo busca minimizar uma função custo que considera a distância física entre as docas dos caminhões que transferem carga entre si mais uma penalidade. Tal penalidade está associada à impossibilidade de dois caminhões trocarem carga entre si, devido às janelas de tempo.

Finalmente cabe mencionar a contribuição apresentada em Boysen e Fliendner (2009) que tratam o assunto *cross docking* sob a ótica da pesquisa, em especial os problemas de sequenciamento de caminhões. Além de uma análise cronológica da pesquisa na área, os autores propõem uma sistematização dos conceitos envolvidos, ao apresentarem diferentes possibilidades de função objetivo, ambiente operacional e forma de interação entre os caminhões e o terminal.

3. Definição do problema e modelagem matemática

O problema objeto do trabalho pressupõe um conjunto de premissas necessárias à sua delimitação, as quais são essenciais para o desenvolvimento dos métodos propostos. O estudo trata a situação na qual dois conjuntos distintos de caminhões trocam suas cargas através de um terminal constituído por duas docas, uma de carga e outra de descarga. Um dos conjuntos (*inbound*) agrega os caminhões que chegam carregados ao terminal, os quais deverão repassar toda sua carga ao longo do processo de transferência. O outro conjunto (*outbound*) é constituído pelos caminhões vazios, destinados a receber o carregamento. Cada caminhão pertence a somente um conjunto e cada conjunto utiliza somente uma doca (carga/descarga). Uma vez alocado na respectiva doca, o caminhão só poderá sair vazio (caso pertença ao conjunto de recebimento - *inbound*) ou carregado (caso pertença ao conjunto de envio – *outbound*).

No início do processo de transferência todos os caminhões envolvidos estão disponíveis e é conhecida a quantidade de carga Q_{ij} que cada caminhão i do conjunto de recebimento R repassa a um ou mais caminhões j do conjunto de envio S. A carga só pode ser carregada ou descarregada quando o caminhão estiver alocado à sua respectiva doca: assim no máximo dois caminhões podem transferir carga simultaneamente, sendo um *inbound* e um *outbound*. Um caminhão do conjunto de recebimento poderá repassar sua carga diretamente ao caminhão alocado na doca de envio ou poderá destiná-la a caminhões *outbound* futuros, neste caso descarregando para o estoque temporário. De forma análoga, um caminhão da sequência de envio poderá receber carga diretamente de um caminhão alocado na doca *inbound* ou de caminhões *inbound* prévios, nesse caso carregando a partir do estoque intermediário. Admite-se que cada unidade de carga repassada de forma direta gera uma unidade de custo. Caso essa mesma unidade de carga utilize o estoque intermediário, ela gerará um custo de duas unidades. Diante dessas premissas, busca-se um sequenciamento de cada um dos conjuntos de caminhões, de forma a minimizar o custo, ou seja, a quantidade de carga transferida através do estoque intermediário.

Para maior clareza é útil a exemplificação através de uma instância de tamanho reduzido. A Figura 1 relaciona o número de unidades de carga que cada caminhão de recebimento i transfere para os caminhões de envio j. Por exemplo, o caminhão de recebimento i=4 transfere 31 unidades para o caminhão do conjunto de envio representado por j=1 e 5 unidades para aquele representado por j=3.

Q_{ij} =

		Envio (j)		
		1	2	3
Recebimento (i)	1	14	5	0
	2	0	32	15
	3	0	0	24
	4	31	0	5

Figura 1 – Matriz Q_{ij}

O que dificulta a passagem direta da carga é a restrição de que cada caminhão só pode ser alocado na sua respectiva doca uma vez durante o processo de transferência e, estando alocado, deve carregar ou descarregar toda a carga. Isso faz com que alguns caminhões do conjunto de *inbound* precisem descarregar toda sua carga no estoque intermediário a fim de permitir o posicionamento do caminhão seguinte na doca *inbound*.

Adotando-se as sequências *inbound:* 1-3-4-2 e *outbound*: 1-3-2 para o exemplo da Figura 1, ter-se-á um *cross docking* com as características ilustradas na Figura 2.

Caminhão *Inbound*	Caminhão *Outbound*	Carga transferida diretamente	Estoque intermediário para *Outbound* 1	Estoque intermediário para *Outbound* 2	Estoque intermediário para *Outbound* 3
1	1	14		5	
3	1				24
4	1	31			
4	3	5			
2	3	15			
2	2	32			

Figura 2 – *Cross docking* segundo sequências arbitrárias para uma instância de tamanho reduzido.

Com essas sequências, 29 unidades de carga passaram através do estoque intermediário e o custo resulta em: 14 + 31 + 5 + 15 + 32 + 2 x (5 +24) = 155.

Cabe comentar que nenhum sequenciamento resultará inviável, pois na pior das hipóteses, todos os caminhões do conjunto de recebimento descarregariam no depósito intermediário e os caminhões do conjunto de envio seriam carregados a partir desse estoque.

Essas definições levaram à formulação de um modelo matemático linear, descrito adiante, capaz de relacionar as duas sequências ao custo que elas implicam. Mas antes da apresentação do modelo, é interessante uma discussão sobre a natureza das transferências de carga realizadas entre os dois conjuntos de caminhões. Na instância tratada acima, a primeira carga a utilizar o estoque intermediário foi aquela repassada do caminhão *inbound* 1 para o *outbound* 2, ou seja, $Q_{12} = 5$. Tal carga precisou passar para o estoque intermediário devido ao fato de que o caminhão *outbound* j=1 não estava completo e precisava da carga de caminhões posicionados após i=1 na sequência de recebimento. Com isso, qualquer caminhão *inbound* posicionado antes do último caminhão a transferir carga para j=1 precisaria descarregar no estoque intermediário toda a carga destinada a caminhões *outbound* posteriores a j=1.

Essas idéias permitem uma generalização: dado um par de caminhões i e j, se j receber carga de algum caminhão posicionado após i, o caminhão i deverá descarregar no estoque intermediário toda a carga destinada a qualquer caminhão *outbound* posicionado após j. Essa mesma lógica, vista de uma forma alternativa, mostra que, dado um par de caminhões i e j, eles não poderão trocar carga diretamente entre si, caso qualquer caminhão posicionado após i envie carga para qualquer caminhão posicionado antes de j. Essa lógica aplicada à instância do exemplo, mostraria que i=1 não pode transferir diretamente para j=2 porque i=3 envia carga para j=1 e assim i=1 precisa esvaziar antes que i=3 fosse alocado na doca. De uma forma bem mais intuitiva seria o mesmo que dizer que um caminhão *outbound* que já passou pela doca de carregamento não poderia ter recebido carga de um caminhão *inbound* que ainda não tivesse sido alocado na doca de descarga.

A partir das relações discutidas acima, foi possível formular um modelo alternativo aos pesquisados na literatura, nos quais o tempo é uma variável fundamental para a modelagem. Isso porque o modelo aqui proposto é capaz de tratar a possibilidade de transferência direta de cargas entre uma dupla de caminhões quaisquer verificando somente a posição relativa entre os caminhões *inbound* e *outbound* na sequência. Ou seja, não é necessária a utilização de uma variável relacionada ao tempo de chegada e saída dos caminhões nas docas. Assim, o modelo aqui proposto é orientado ao objetivo específico de minimizar a quantidade de carga transferida através do estoque intermediário.

Tal formulação tem como dados de entrada as quantidades de carga a serem transferidas entre os caminhões de recebimento e envio. Essas quantidades entram no modelo conforme uma matriz de duas dimensões simbolizada por Q_{ij}, onde i representa o caminhão do conjunto de recebimento e j se refere ao caminhão do conjunto de envio. Já a matriz binária K_{ij} possui coeficiente 1 ca so i envie carga para j e 0 no caso contrário. O número total de caminhões de recebimento é $|R|$ e o número de caminhões de envio é $|S|$.

As variáveis de decisão do modelo são PIN_{im}, $POUT_{jn}$ POS_IN_i, POS_OUT_j, a_{im}, b_{jn}, z_{ij}, sendo que POS_IN_i, POS_OUT_j são variáveis inteiras e as demais são binárias. A variável PIN_{im} fornece a posição de um dado caminhão *i* na sequência de *inbound*, por exemplo, $PIN_{57} = 1$ significa que o caminhão 5 ocupa a sétima posição na sequência de recebimento. A variável $POUT_{jn}$ tem função idêntica à PIN_{im}, porém na sequência de envio. Já POS_IN_i assume o valor da posição de um dado caminhão *i* na sequência de recebimento; por exemplo, $POS_IN_8 = 3$ indica que o caminhão 8 ocupa a terceira posição da sequência de recebimento. A variável POS_OUT_j é análoga à POS_IN_i , porém é aplicável à sequência de envio. Já a variável binária a_{im} assume valor 1 se o caminhão *i* precede o caminhão *m* na sequência de recebimento. Por sua vez, a variável b_{jn} admite 1 se o caminhão *j* precede o caminhão *n* na sequência de envio.

Finalmente z_{ij} recebe 1 caso o caminhão *inbound* *i* e o caminhão *outbound* *j* não possam ser alocados simultaneamente às suas respectivas docas; nesse caso *i* e *j* não podem transferir carga de forma direta. Com base na lógica descrita, formulou-se o modelo apresentado a seguir, o qual é baseado em variáveis inteiras e binárias:

$$Minimizar \quad \sum_{i=1}^{R}\sum_{j=1}^{S} Q_{ij} \times z_{ij} \tag{1}$$

Sujeito a:

Restrições de sequência:

$$\sum_{m=1}^{R} PIN_{im} = 1 \qquad \forall \quad i \in R \tag{2}$$

$$\sum_{i=1}^{R} PIN_{im} = 1 \qquad \forall \quad m \in R \tag{3}$$

$$\sum_{n=1}^{S} POUT_{jn} = 1 \qquad \forall \quad j \in S \tag{4}$$

$$\sum_{j=1}^{S} POUT_{jn} = 1 \qquad \forall \quad n \in S \tag{5}$$

$$POS_IN_i = \sum_{m=1}^{R} PIN_{im} \times m \qquad \forall \quad i \in R \tag{6}$$

$$POS_OUT_j = \sum_{n=1}^{S} POUT_{jn} \times n \qquad \forall \quad j \in S \tag{7}$$

$$POS_IN_m - POS_IN_i + 1 \leq a_{im} \times M \qquad \forall \quad i,m \in R, \quad i \neq m \tag{8}$$

$$POS_IN_m - POS_IN_i \geq M(a_{im} - 1) \qquad \forall \quad i,m \in R, \quad i \neq m \tag{9}$$

$$POS_OUT_n - POS_OUT_j + 1 \leq b_{jn} \times M \qquad \forall \quad j,n \in S, \quad j \neq n \tag{10}$$

$$POS_OUT_n - POS_OUT_j \geq M(b_{jn} - 1) \qquad \forall \quad j,n \in S, \quad j \neq n \tag{11}$$

Restrição de simultaneidade:

$$z_{ij} \geq a_{im} + b_{nj} \times K_{mn} - 1 \qquad \forall \quad i\ e\ m \in R, \quad j\ e\ n \in S \tag{12}$$

Demais restrições:

$z_{ij} \in \{0,1\}$; $a_{im} \in \{0,1\}$; $b_{nj} \in \{0,1\}$; $PIN_{im} \in \{0,1\}$; $POUT_{jn} \in \{0,1\}$;
$POS_IN_i \in N$; $POS_OUT_j \in N$

A função objetivo (1) considera o custo oriundo das unidades de carga transferidas através do estoque intermediário. As restrições (2) e (3) do modelo garantem que cada caminhão *inbound* só assuma uma posição e que cada posição *inbound* esteja associada a somente um caminhão. Com isso, para cada *i* existirão ($|R|$-*1*) variáveis PIN_{im} nulas e somente uma igual a 1; e o mesmo vale para cada *m*. As restrições (4) e (5) têm função análoga na sequência de envio. Já a restrição (6) faz com que a variável POS_IN_i assuma o valor da posição de *i* na sequência de *inbound*. A restrição (7), de forma similar, garante que a variável POS_OUT_j assuma o valor da posição do caminhão *j* na sequência de *outbound*.

As restrições (8) e (9) fazem com que a variável binária a_{im} assuma valor 1, se e somente se, o caminhão *i* precede o caminhão *m* na sequência de recebimento. As restrições (10) e (11) são análogas, conferindo valor 1 a todo b_{jn} onde *j* precede *n* na sequência de envio. Finalmente, a restrição (12) garante que z_{ij} assuma valor 1 caso qualquer caminhão posicionado após *i* na sequência de recebimento transfira carga para qualquer caminhão posicionado antes de *j*.

Um detalhe a lembrar é que o custo de uma dupla de sequências será a soma entre a função objetivo apresentada e o número total de unidades de carga transferidas - ***UCT***. Dessa forma o custo unitário da carga que passa direto será um e da carga que passa pelo estoque intermediário será dois, pois esta será somada tanto ao *UCT* quanto à função objetivo do modelo, onde a carga possui o coeficiente de custo 1 implícito. Como o *UCT* é fixo para uma dada instância, ele não influencia o modelo matemático. O *UCT* é dado por $\sum_i \sum_j Q_{ij}$.

Se o modelo fosse aplicado ao exemplo tratado anteriormente, conforme as sequências *inbound:* 1-3-4-2 e *outbound*: 1-3-2, as seguintes variáveis binárias resultariam não nulas: PIN_{11}, PIN_{24}, PIN_{32}, PIN_{43}, $POUT_{11}$, $POUT_{23}$, $POUT_{32}$, a_{12}, a_{13}, a_{14}, a_{34}, a_{32}, a_{12}, b_{12}, b_{13}, b_{32}, z_{12}, z_{13}, z_{32}, z_{33}, z_{42}. Já as variáveis inteiras não nulas seriam: $POS_IN_1 = 1$, $POS_IN_2 = 4$, $POS_IN_3 = 2$, $POS_IN_4 = 3$, $POS_OUT_1 = 1$, $POS_OUT_2 = 3$, $POS_OUT_3 = 2$.

Percebe-se que o número de variáveis é expressivo, tendo-se em vista que a instância tem um tamanho bastante reduzido. Isso se deve à característica combinatória do problema, sendo que são possíveis R! x S! combinações de sequência, o que dificulta a resolução por métodos exatos.

4. Método de solução baseado em *scatter search* (*SS*)

O *SS* é uma meta-heurística populacional evolutiva na qual é mantido um conjunto de soluções de grande qualidade e um conjunto de soluções de elevada diversidade. A combinação entre essas soluções produz novas soluções, as quais podem ser melhores e substituir as antigas. Diferentemente do algoritmo genético, o SS se baseia em procedimentos sistemáticos ao invés do recurso da aleatoriedade. Além disso, a população é em geral, consideravelmente menor. Para maiores detalhes ver Martí *et al.* (2006).

Antes da demonstração dos métodos que compõe a meta-heurística implementada, é conveniente a discussão da forma pela qual se **representa a solução**. Trata-se de um vetor onde a sequência de recebimento é contida nas primeiras R posições e a sequência de envio nas últimas S posições. Exemplo: 1-3-4-2:1-3-2. Essa representação de solução tem implicações sobre o método de combinação, pois a simples união de partes de duas soluções não garante a viabilidade do novo vetor, uma vez que não é admissível nem a repetição de números em uma sequência, nem a omissão. Por essa razão foram empregados dois **métodos de combinação** distintos e complementares (MC1 e MC2), adaptados a partir do cruzamento PMX – *partially matched crossover*. Para maiores detalhes sobre o método PMX ver Sahu e Tapadar (2008). A adaptação fica por conta dos pontos de cruzamento, definidos na metade de cada uma das sequências de recebimento e envio. A diferença entre o MC1 e o MC2 está na porção da sequência preservada: O MC1 conserva a segunda metade da solução1 (PR) e o MC2 não altera a primeira metade da solução1 (PR). Onde PR e SEG se referem à qual solução tem prioridade na combinação, ou seja, qual solução repassa um trecho da sequência sem alterações. A Figura 3 ilustra esses métodos, sendo que: as células com fundo claro representam os trechos copiados da solução1 (PR) ; em fundo escuro e fonte clara estão os trechos copiados da solução2 (SEG); e as células com preenchimento hachurado representam os caminhões inseridos através do método de mapeamento, típico do cruzamento PMX.

		Inbound						Outbound			
PR	Solução1	3	2	4	1	6	5	2	3	1	4
SEG	Solução2	1	4	6	2	3	5	3	1	4	2
	MC1	3	4	2	1	6	5	3	2	1	4
	MC2	3	2	4	1	6	5	2	3	4	1

Pontos de cruzamento

Figura 3 – Métodos de combinação de solução.

Outro elemento importante da implementação é **o método de atualização do conjunto de referência**. Mas antes de detalhá-lo é preciso apresentar a forma segundo a qual se estabelece a diferença entre duas soluções, que corresponde ao somatório do quadrado das diferenças entre as posições que cada caminhão ocupa no vetor de solução. A Figura 4 ilustra o cálculo da distância entre as soluções:

	Inbound			Outbound	
Solução 1	3	1	2	1	2
Posição 1	2	3	1	1	2
Solução 2	2	3	1	2	1
Posição 2	3	1	2	2	1
Quad. Diferença	1	4	1	1	1
Soma Diferença	6			2	

Figura 4 – Diferença entre duas soluções

Conforme será detalhado mais adiante, a meta-heurística possui dois conjuntos de referência distintos, um para as soluções de alta qualidade ($Refset_1$) e outro para aquelas com elevada diversidade ($Refset_2$). É mantido também um conjunto *(Pool)* que armazena as soluções oriundas das combinações. A atualização do $Refset_1$ se dá a partir das soluções presentes em $Refset_1 \sqcup Pool$, segundo o pseudo-código a seguir, onde *nRefset* corresponde ao número de soluções armazenadas no $Refset_1$.

ATUALIZAÇÃO DO $REFSET_1$ ( ):
Posicionar a melhor de todas as soluções de $Refset_1 \sqcup Pool$ na primeira posição de $Refset_1$;
Para cada uma das posições do $Refset_1$ de 2 até nRefset
    Selecionar a melhor solução entre as de $Refset_1 \sqcup Pool$ ainda não inserida no novo $Refset_1$;
    Se essa solução atende critério de distância mínima em relação à solução vizinha recém inserida
        Inserir essa solução e passar para a posição seguinte do $Refset_1$;
    Senão excluir essa solução do conjunto $Refset_1 \sqcup Pool$ e voltar a **Selecionar;**

Figura 5 – Atualização do *$Refset_1$*

Este método de atualização, portanto, incorpora um critério de distância mínima da solução que se tenta inserir, em relação à outra solução recém inserida no novo conjunto. Esse método impossibilita que o $Refset_1$ possua soluções iguais, a menos que o critério de distância seja especificado como nulo. Caso nenhuma das novas soluções consiga atender ao critério de distância, a atualização não modificará o $Refset_1$. Já se a partir desta condição, o critério de distância cair levemente, as soluções mais diversas armazenadas no *Pool* tendem a entrar no $Refset_1$. A partir dessas informações percebe-se que o critério de distância entre as soluções é um importante parâmetro de busca, pois poderá favorecer a intensificação em torno de um conjunto de soluções similares ou a diversificação. De qualquer forma, a melhor solução corrente não será descartada durante a atualização. Outro método de atualização necessário diz respeito ao conjunto das soluções diversas – $Refset_2$, formalizado no pseudo-código a seguir:

```
ATUALIZAÇÃO DO REFSET2 ( ):
Para cada uma das soluções do Pool
        Calcular a distância entre essa solução e a melhor solução para a seq. de receb. e envio;
                Somar as distâncias das sequências de recebimento e envio;
        Ordenar as soluções em ordem decrescente conforme a distância, formando um ranking;
Atualizar o Refset2 substituindo suas soluções pelas nRefset primeiras do ranking;
```

Figura 6 – Atualização do *Refset₂*

Cabe ainda mencionar que a atualização dos conjuntos de referência é realizada de forma **estática**. O método de **geração de subconjuntos**, por sua vez, é detalhado a seguir, sendo que as siglas MC1 e MC2 se referem ao método de combinação e as siglas PR e SEG dizem respeito à ordem de combinação. Por exemplo, em uma combinação MC1, a solução indicada como PR transmitirá, sem modificações, os últimos elementos das suas sequências de recebimento e envio para a solução nova. Já se o método fosse o MC2 os elementos das primeiras posições seriam os mais preservados. As soluções com sigla SEG serão as menos preservadas, pois estão sujeitas ao mapeamento discutido anteriormente**.**

- Subconjunto 01: a melhor solução (PR) e a segunda melhor (SEG) – **MC2**;
- Subconjunto 02: a melhor solução (PR) e a terceira melhor (SEG) – **MC1**;
- Subconjunto 03: a melhor solução (SEG) e todas as outras do *Refset₁* (PR) **– MC1**;
- Subconjunto 04: metade das soluções do *Refset₁* (PR) são combinadas com as soluções do *Refset₂* (SEG) com posição equivalente (ex.: $Refset_1^1$ com $Refset_2^1$, $Refset_1^2$ com $Refset_2^2$, etc.). Nesse caso o método de combinação utilizado é o **MC2**.
- Subconjunto 05: a segunda metade das soluções do *Refset₁* (PR) são combinadas com as soluções do *Refset₂* (SEG) com posição equivalente (ex.: $Refset_1^4$ com $Refset_2^4$, $Refset_1^5$ com $Refset_2^5$, etc.). Nesse caso o método de combinação utilizado é o **MC1**.

Com essa forma proposta de geração de subconjuntos, evita-se que ocorram muitas perdas dos fragmentos de sequência, pois existe uma alternância na designação de PR e SEG às soluções, além de uma alternância dos métodos de combinação. Esses dois métodos são complementares, pois enquanto o MC1 preserva os primeiros caminhões das sequências, o MC2 preserva a segunda metade. Também, a melhor solução é a que mais participa das combinações.

Já o **método de melhoria** é aplicado a cada iteração sobre duas soluções: a terceira melhor do *Refset₁* e a melhor solução oriunda da combinação corrente. Esse método é demonstrado na Figura 7, seguindo uma lógica aleatória e um certo número de repetições(conforme o parâmetro *n_tentaivas*). Ao longo da implementação foram testadas diferentes propostas de melhorias seguindo heurísticas inspiradas no modelo matemático. Essas heurísticas, no entanto, se mostraram falhas para algumas das instâncias analisadas, o que prejudicou sua robustez mediante os testes estatísticos. Notou-se que as relações múltiplas entre os elementos das duas sequências é um complicador ao sucesso de um método de melhoria de caráter sistemático.

```
MELHORIA ( ):
Sorteio de uma posição da sequência de recebimento (i);
Para cada posição da sequência de recebimento (m);
    Trocar os caminhões de posição m e i;
    Se a troca melhorar o custo aceitar, senão desfazer;
Para x=1 até n_tentativas:
    Sorteio de uma posição da sequência de envio (j);
    Para cada posição da sequência de envio (n);
            Trocar os caminhões de posição n e j;
            Se a troca melhorar o custo aceitar, senão desfazer;
```

Figura 7 – Método de melhoria

Outro método implementado e importante na meta-heurística, é o **método de diversificação**. Ele foi adaptado a partir das idéias apresentadas em Glover (1998) no qual uma solução inicial dá origem a uma solução diversificada. O método se baseia em uma forma de embaralhamento da solução, segundo um critério de intercalação dos valores.

Para a criação de um conjunto inicial de soluções, foi utilizada uma **heurística construtiva.** Sobre essa heurística aplica-se o método de diversificação, gerando uma nova solução. Sobre essa nova solução aplica-se novamente o método de diversificação gerando a terceira solução e assim sucessivamente até completar o *Pool*. A heurística construtiva segue os passos abaixo:

1. Criar uma ordenação decrescente dos caminhões de envio, com base na carga total destinada a cada um deles, dando origem à sequência de envio:
2. Selecionar o primeiro caminhão da sequência de envio;
3. Enquanto a sequência de recebimento não estiver completa, inserir os caminhões de recebimento com maior quantidade total de carga e que troquem carga com o caminhão de envio selecionado. Se não existirem mais caminhões de recebimento que troquem carga com o caminhão de envio selecionado, selecionar o próximo caminhão de envio.

Em resumo, a meta-heurística proposta executa os seguintes passos:

I. Criação de uma solução inicial, e de um *Pool* de soluções diversificadas;
II. Criação dos conjuntos de referência $Refset_1$ e $Refset_2$;
III. Combinação entre as soluções dos conjuntos de referência;
IV. Aplicação do método de melhoria;
V. Atualização dos conjuntos de referência;
VI. Se o critério de parada foi atingido, finalizar, senão voltar ao passo III.

O **critério de parada** atende a duas condições: tempo e número de iterações, sendo considerada a primeira a ser atingida. O limite de tempo é de 18 minutos, atuando somente nas instâncias de tamanhos 100x50 e 200x100. Já o limite de iterações (*lim_iter)* obedece a uma lógica que visa evitar que o *scatter search* seja interrompido precocemente ou continuado de forma desnecessária. Para tanto, um número mínimo de 600 iterações é especificado, sendo que esse limite vai sendo aumentado de acordo com a distância máxima verificada entre as melhorias consecutivas (*max_it_s_m*). Tal variável (*max_it_s_m*) é multiplicada ao parâmetro *busca* e adicionada a *lim_iter* cada vez que uma solução de melhor qualidade é encontrada. O parâmetro *busca* serve para prolongar o número de iterações. Caso o número de iterações seja maior do que 1500, o parâmetro *lim_iter* não é atualizado e a meta-heurística se aproxima do final.

Calibração

O processo de intensificação e diversificação do *scatter search* depende muito de um ajuste adequado dos parâmetros de busca. Para tanto, implementou-se um conjunto de rotinas capazes de ajustar esses parâmetros ao longo da meta-heurística. Entre os parâmetros ajustados automaticamente estão o critério de parada, discutido anteriormente e a distância mínima entre as soluções que entram no $Refset_1$. Esse critério de distância (*dist*) entre as soluções é variado à medida que fica mais difícil encontrar uma solução de melhor qualidade.

Esse ajuste é realizado conforme um ciclo pré-definido segundo o qual o critério inicia nulo e vai aumentando até certo limite após *n* iterações sem melhoria. Ao atingir um ponto de distância máxima (*dist_max*) o critério volta a se tornar nulo. Com isso, no início da meta-heurística, novas soluções são admitidas com facilidade no $Refset_1$. À medida que essas soluções se tornam muito parecidas e suas combinações infrutíferas, o critério de distância aumentado força a entrada de soluções de menor qualidade e maior diversidade no $Refset_1$. A conseguinte diminuição de *dist* leva novamente a um processo de intensificação onde soluções de alta qualidade são combinadas à soluções equilibradas em termos de diversidade e qualidade.

Os demais parâmetros foram calibrados através de simulações em instâncias de tamanho pequeno (10x5, 12x6, 14x7 e 20x10). Esses parâmetros são: tamanho dos conjuntos de referência (*nRefset*), valor máximo do distância entre soluções (*dist_max*), número de repetições do método de melhoria sob uma dada solução (*n_tentativas*), número adicional de iterações a executar

(*busca*). Os valores adotados foram os seguintes: *nRefset* = 6, *dist_max* =$R^2/4$, *n_tentativas* = *S/2* e *busca* = *2*.

5. Experimentos computacionais

Uma vez que as premissas adotadas nesse trabalho são diferentes daquelas tratadas na literatura pesquisada, instâncias de teste aleatórias foram geradas, atendendo aos seguintes critérios: cada caminhão do conjunto de recebimento envia carga para no mínimo 1 e no máximo 5 caminhões do conjunto de envio; cada caminhão da área de envio recebe carga de no mínimo 1 caminhão da sequência de *inbound*; a carga atribuída a uma dupla *inbound:outbound* é de no mínimo uma unidade e de no máximo 19 unidades. Porém, como uma mesma dupla pode ser sorteada mais de uma vez, é possível que a carga transferida supere as 19 unidades.

Foram geradas 35 instâncias de testes, onde o conjunto de recebimento possui o dobro do número de caminhões do conjunto de envio. Essas 35 instâncias estão divididas conforme as seguintes dimensões: 10x5, 12x6, 14x7, 20x10, 50x25, 100x50 e 200x100, com 5 replicações cada.

Para os experimentos utilizou-se um microcomputador com processador Intel Core 2 Duo 2.0 GHz, com 3 GB de RAM e 32 bits. O software *Gurobi*, na versão acadêmica 3.0.0, foi utilizado para a resolução exata das instâncias menores. A meta-heurística foi implementada em linguagem C.

Foi possível obter a solução exata somente para as instâncias de tamanhos 10x5 e 12x6 através do *Gurobi*. Os problemas de dimensão 14x7 já se mostraram suficientes para esgotar a memória do computador, além de consumirem muito tempo de CPU. A Tabela 1 resume os resultados das instâncias de tamanhos 10x5, 12x6 e 14x7, obtidos a partir da meta-heurística (com cinco sementes) e do modelo exato. A última coluna da Tabela 1 compara a distância entre as soluções, através da relação: ((SS - solução exata) *100 / ótimo).

Tabela 1 – Soluções obtidas a partir do SS e do modelo exato para as instâncias de pequeno porte

Problema	UCT	Scatter Search (SS)				Solução Exata (SE)			Distância relativa da solução do SS perante SE
		Tempo médio (s)	Pior	Média	Melhor	Custo	Gap*	Tempo (s)	
10x5_1	331	1,82	488	478,8	478	478	0,00%	127	0,00%
10x5_2	275	1,36	392	365,0	363	363	0,00%	47	0,00%
10x5_3	282	1,20	392	392,0	392	384	0,00%	43	2,08%
12x6_1	311	1,44	427	388,4	378	378	0,00%	482	0,00%
12x6_2	495	1,52	706	705,2	705	705	0,00%	1.637	0,00%
12x6_3	484	1,67	723	715,8	712	712	0,44%	11.378	0,00%
14x7_1	318	1,61	412	404,8	403	397	29,10%	21.045	1,51%
14x7_2	467	2,23	678	673,4	672	669	46,40%	17.385	0,45%
14x7_3	650	1,90	981	981,0	981	963	44,80%	17.895	1,87%

* Gap em relação ao *lower bound* da relaxação linear.

Verifica-se que o SS despende muito menos tempo, sendo mais rápido na ordem de milhares de vezes. Ainda, as distâncias entre as soluções são muito pequenas, atingindo-se inclusive o ótimo global, como é o caso dos problemas 10x5_1, 10x5_2, 12x6_1 e 12x6_2.

Diante da impossibilidade de resolução dos problemas maiores de forma exata, estes foram resolvidos somente através da meta-heurística que foi comparada à heurística construtiva (HC) anteriormente mencionada. A distância entre o SS e a HC é mostrada na última coluna da Tabela 2, obedecendo à relação (HC - SS)*100/SS.

Tabela 2 – Resultados do SS e da HC para os problemas de tamanho 20x10, 50x25, 100x50 e 200x100

Problema	UCT	Scatter Search (SS)				Solução Heurística construtiva (HC)	Distância relativa da solução do SS perante HC
		Tempo médio (s)	Pior	Média	Melhor		
20x10_1	434	3,0	594	582,8	569	701	23,20%
20x10_2	452	3,7	614	602,0	597	660	10,55%
20x10_3	521	4,0	717	704,0	685	788	15,04%
50x25_1	1168	26,8	1652	1609,4	1575	1848	17,33%
50x25_2	1352	42,3	1991	1974,2	1958	2191	11,90%
50x25_3	1197	27,3	1713	1693,2	1666	2020	21,25%
100x50_1	2451	912,8	3595	3563,8	3487	4105	17,72%
100x50_2	2580	905,8	3849	3821,8	3803	4301	13,09%
100x50_3	2344	638,4	3524	3481,6	3342	4003	19,78%
200x100_1	4883	1081,4	7163	7151,0	7133	8141	14,13%
200x100_2	5095	1083,6	7613	7574,6	7544	8538	13,18%
200x100_3	5072	1083,2	7681	7665,8	7633	8354	9,45%

Uma forma alternativa de comparação entre o SS e a heurística construtiva é a padronização do custo através do UCT, conforme a Figura 8. Tal padronização segue a seguinte regra (solução/UCT -1)*100 e reflete o percentual de carga transferida através do estoque intermediário. Percebe-se que a meta-heurística é capaz de manter uma distância relativamente constante em relação à solução da heurística construtiva tanto para as instâncias pequenas, onde coincide com o ótimo global como para as instâncias maiores. Além disso, para as 5 sementes analisadas, a melhor e a pior das respostas do SS são relativamente próximas, conforme o gráfico abaixo.

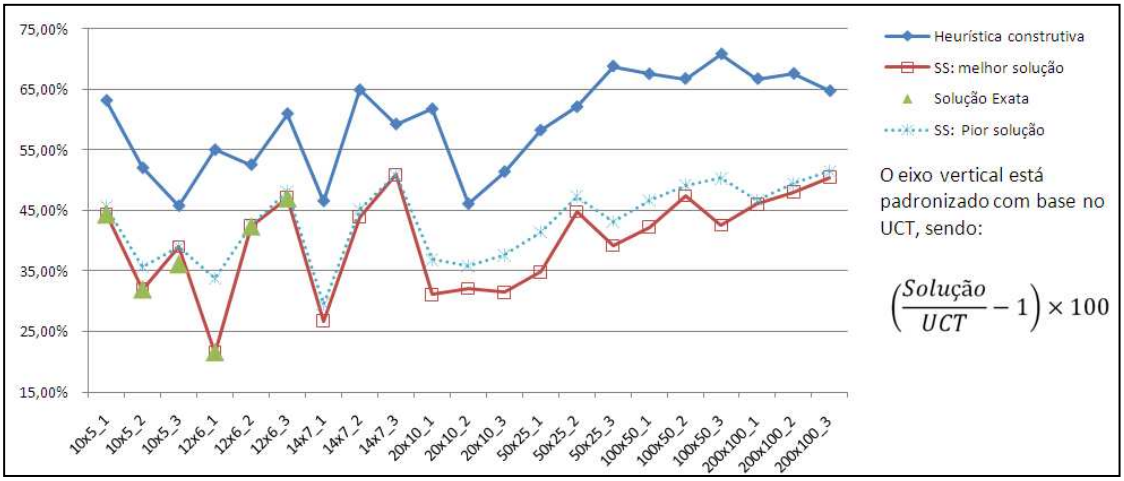

Figura 8 – Comparação entre o SS e a heurística construtiva padronizadas com base no UCT

6. Considerações finais e recomendações

Este trabalho apresentou um modelo matemático aplicável ao problema de sequenciamento de caminhões na operação de *cross docking*. O modelo

atende a todas as premissas adotadas, sendo que sua resolução no software *Gurobi* forneceu as soluções ótimas para as instâncias de pequeno porte. No entanto, a complexidade combinatória do problema limitou a utilização do modelo nas instâncias maiores, pois o tempo de processamento aumentou muito e a memória do computador se tornou um recurso limitante.

Com o propósito de obter-se soluções de boa qualidade em tempos de processamento aceitáveis, implementou-se a meta-heurística *scatter search*, que atingiu a solução ótima para cinco das instâncias de pequeno porte, em tempos computacionais muito atrativos. A fim de se

analisar a eficácia da meta-heurística nas demais instâncias, suas soluções foram confrontadas com aquelas provenientes da heurística construtiva utilizada para a construção inicial do $Refset_1$.

Essa comparação utilizou um percentual padronizado, que se baseia no *UCT* de cada instância (ver Figura 8), o qual reflete o percentual de carga transferida através do estoque intermediário. Verificou-se que a meta-heurística mantém um padrão constante de distância em relação à heurística construtiva, tanto para as instâncias menores, onde atinge o ótimo global, como para as instâncias maiores. Isso é um indício de que as soluções produzidas para os problemas de médio e grande porte são de boa qualidade. Porém seria interessante a comparação dos resultados do SS aqui implementado com outros métodos de solução, a fins de comprovar seu desempenho nos problemas de maior porte. Adicionalmente, o SS conseguiu produzir essas respostas em tempos computacionais aceitáveis, mesmo no caso das instâncias de grande porte.

Um trabalho futuro poderia aperfeiçoar a resolução do modelo exato através de uma maior delimitação do espaço de solução, seja pela incorporação de restrições adicionais, seja por uma melhor estimativa do *lower bound* durante a execução do software otimizador. Isso porque verificou-se que o *lower bound* permanece muito baixo durante parte considerável do tempo de solução. Ainda, uma exploração da estrutura do problema a partir das idéias apresentadas na seção 3, poderia levar ao desenvolvimento de uma heurística de melhoria mais elaborada, capaz de conferir maior eficiência e eficácia à meta-heurística.

Referências

Boysen, N., Fliedner, M. (2009), Cross dock scheduling: Classification, literature review and research agenda, The *International Journal of Management Science*, doi:10.1016/j.omega.2009.10.008.

Glover F. (1998) A template for scatter search and path relinking. In: Hao J-K, Lutton E, Ronald E, Schoenauer M, Snyers D (eds) Artificial evolution. Lecture notes in computer science, vol. 1363, Springer, Berlin Heidelberg New York, pp 13–54.

Martí, R., Laguna M. e Glover F.(2006), Principles of Scatter search, *European Journal of Operational Research* 169, 359-372.

Miao, Z., Lim A. e Ma H.(2009), Truck dock assignment problem with operational time constraint within docks, *European Journal of Operational Research* 192, 105-115.

Qiu, Q., Zhang, Z., Song X. e Gui S.(2009), Application Research of *Cross docking* Logistics in Food Cold-Chain Logisticas, *2009 International Conference on Information Management, Innovation Management and Industrial Engineering*, 236-240.

Sahu, A., Tapadar R.,.(2008), Solving the assignment problem using Genetic Algorithm and Simulated Annealing, *International Journal of Applied Mathematics* 36:1, IJAM_36_1_7.

Vahdani B. e Zandieh M. (2009), Scheduling trucks in cross-docking systems: Robust meta-heuristics, Computers and Industrial Engineering, doi:10.1016/j.cie.2009.06.006

Yu, W. e Egbelu, P. (2008), Scheduling of inbound and outbound trucks in *cross docking* systems with temporary storage, European Journal of Operational Research 184, 377-396.