# UM ESTUDO DA HEURÍSTICA *LARGE NEIGHBORHOOD SEARCH* PARA O PROBLEMA DE COLETA E ENTREGA COM JANELA DE TEMPO

**Viviane Junqueira de Moraes**
Universidade Federal de Ouro Preto – Departamento de Engenharia de Produção
Campus Morro do Cruzeiro, Ouro Preto, MG – Brasil - 35400-000
vivianejmoraes@gmail.com

**Gustavo Peixoto Silva**
Universidade Federal de Ouro Preto – Departamento de Computação
Campus Morro do Cruzeiro, Ouro Preto, MG – Brasil - 35400-000
gustavo@iceb.ufop.br

## RESUMO

Este trabalho aborda o Problema de Coleta e Entrega com Janela de Tempo (*Pick-up Delivery Problem with Time Window*), o qual requer que uma frota de veículos atenda os pedidos dos clientes com o menor custo. Os pedidos requerem um veículo para carregar uma quantidade de produtos em um local e entregá-los em outro. Todos os pedidos devem ser atendidos respeitando a capacidade dos veículos e as janelas de tempo dos clientes. O PCEJT é resolvido com a técnica de Busca em Vizinhança de Grande Porte – BVGP (*Large Neighborhood Search*) que trabalha com múltiplas vizinhanças obtidas pela combinação de diferentes métodos de "destruição" e "reparo" da solução corrente. Esta estratégia permite fugir de ótimos locais e aumenta sua capacidade de percorrer todo o espaço de soluções. A heurística foi testada com uma série de problemas *benchmark* disponíveis na internet e largamente utilizados na literatura.

**PALAVRAS CHAVE. Problema de Coleta e Entrega com Janelas de Tempo, Heurísticas, Busca em Vizinhança de Grande Porte.**

## ABSTRACT

This paper addresses the Pickup and Delivery Problem with Time Windows – PDTW, which requires that a fleet of vehicles meets the demands of customers at the lowest cost. Each request needs a vehicle to take a number of products in one location and deliver it in another. All requests must be attended while respecting vehicle capacity and customer time windows. The PDTW is solved with the Large Neighborhood Search, which works with multiple neighborhoods obtained by combining different methods of ruin and recreate from the current solution. This strategy allows to get away from local optima and increases their ability to go through the whole solution space. The heuristic was tested with a series of benchmark problems available on the Internet and widely used in the literature.

**KEYWORDS. Pick Delivery Problem with Time Windows, Heuristics, Large Neighborhood Search.**

## 1. Introdução

Os setores de logística ligados ao transporte de produtos, matérias-primas, bens e pessoas lidam com situações em que os veículos devem ser roteados de modo a percorrer uma série de locais para a coleta ou entrega de mercadorias ou a prestação de serviços. Para isso, as empresas procuram constantemente reduzir os custos fixos e variáveis. Ou seja, com a aquisição de veículos e os gastos com combustíveis, manutenção, motoristas e horas extras. Desta forma torna-se fundamental utilizar métodos de otimização para definir as rotas de tal forma que utilizem o menor número possível de veículos e que percorram a menor distância necessária para atender aos pedidos.

O problema de roteamento de veículos (*Vehicle Routing Problem* - VRP) requer a construção de um conjunto de rotas de custo mínimo para atender um determinado número de pedidos, utilizando uma frota limitada de veículos que iniciam e terminam em um depósito. O problema de roteamento de veículos com janelas de tempo (*Vehicle Routing Problem with Time Windows*-VRPTW) considera janelas de tempo, que são os prazos admissíveis para coleta/ entrega e ainda para o veículo sair e retornar ao depósito. O problema de coleta e entrega com janela de tempo (*Pickup and Delivery Problem with Time Windows* - PDPTW) é uma generalização do VRPTW, que tem como objetivo construir rotas válidas, que satisfaçam os pedidos de coletar produtos/pessoas em um ponto e entregá-los em outro ponto, satisfazendo as restrições de capacidade dos veículos, janelas de tempo, acoplamento e precedência dos pedidos.

Para o PDPTW é fornecido um número de pedidos e um número de veículos capacitados. Um pedido consiste em coletar uma mercadoria em um determinado local e entregá-lo em outro. Duas janelas de tempo são assumidas para cada pedido: um intervalo de tempo para a coleta da mercadoria e outro que especifica a sua entregue. Um veículo nunca deve chegar a um local após o horário final da janela de tempo, mas pode chegar antes do início da janela. Neste caso ele deve aguardar o horário estabelecido para iniciar a operação. Tempos de serviço relacionados com a carga e descarga também são associados a cada pedido.

O local de coleta de um pedido deve ser sempre visitado antes do local de entrega, garantindo a relação de precedência. Além disso, deve ser respeitada a relação de acoplamento, ou seja, a coleta e entrega de um pedido devem ser inseridas na mesma rota. Cada veículo tem uma capacidade limitada, e inicia e finaliza sua jornada de trabalho em um depósito, chamado terminal de início e de fim. O terminal de início pode ser diferente do final para um veículo e dois veículos podem ter terminais diferentes. Cada veículo possui o horário determinado para partir do terminal de início e o horário máximo em que deverá chegar ao terminal de fim. O horário estipulado para início da jornada do veículo deve ser respeitado, mesmo que introduza um tempo de espera no primeiro local visitado. Como o número de veículos é limitado e a frota pode ser heterogênea, podem existir situações em que um pedido não pode ser atendido por determinados veículos, devido às dimensões deste ou necessidade de um compartimento refrigerado, por exemplo. Este pedido é chamado pedido especial. Quando algum pedido não pode ser atendido por nenhum veículo, ele é colocado no banco de pedidos e, por exemplo, veículos extras são contratados para atendê-lo (Ropke e Pisinger, 2006b).

O objetivo deste problema é minimizar a distância total percorrida, o tempo gasto por cada veículo e o número de pedidos que permanecem no banco de pedidos. O PDPTW é um problema do tipo *NP-hard*, por se tratar de um caso particular do problema do caixeiro viajante. Ele conta com um conjunto de restrições tanto de ordem legal quanto da operação da empresa, o que o torna de grande complexidade. A necessidade de resolver problemas reais desta natureza, que na maioria das vezes tem grandes dimensões, em um tempo computacional razoável ressalta a importância da utilização de métodos heurísticos eficientes.

Diferentes trabalhos sobre o problema de coleta e entrega com janelas de tempo são encontrados na literatura, os quais apresentam os mais diversos métodos de resolução e situações reais. Para resolver este problema, são encontrados tanto métodos heurísticos quanto métodos exatos. Nanry e Barnes (2000) foi um dos primeiros trabalhos a apresentar uma metaheurística para o

PDPTW com frota homogênea e um único depósito. Os autores apresentam uma abordagem de Busca Tabu Reativa (*Reactive Tabu Search - RTS*) que combina diferentes movimentos em vizinhanças com o mesmo padrão. Para testar a metaheurística, os autores criaram instâncias com até 100 pedidos a partir de um conjunto padrão do VRPTW proposto por Solomon (1987). O RTS mostrou-se eficiente para resolver um conjunto de problemas práticos de tamanho considerável e com múltiplos veículos.

Li e Lim (2001) usaram uma metaheurística híbrida, combinando a *Simulated Annealing* e a Busca Tabu. O método foi testado em nove casos maiores do que os de Nanry e Barnes (2000), sendo que eles consideram 56 instâncias com base no VRPTW proposto por Solomon (1987). Lim *et al*. (2002) aplicaram otimização local do tipo *squeaky wheel* e busca local para o PDPTW. Esta heurística também foi testada no conjunto de problemas propostos por Li e Lim (2001).

Vários métodos de geração de coluna para o PDPTW têm sido propostos, tanto do tipo exato quanto heurístico. Dumas *et al.* (1991) apresentam um algoritmo exato para resolver o PDPTW referente ao transporte de mercadorias com veículos homogêneos e múltiplos depósitos. O algoritmo proposto utiliza um esquema de geração de colunas no qual é usado o problema de caminho mínimo com restrição de recursos para gerar as colunas do problema principal. Os autores desenvolveram também um método *branch-and-bound*, capaz de resolver problemas com até 55 pedidos.

Xu *et al*. (2003) consideraram um PDPTW com várias restrições práticas incluindo múltiplas janelas temporais, compatibilidade e restrições de tempo máximo de condução. O problema foi resolvido usando uma heurística de geração de colunas e testado com instâncias com até 500 pedidos.

Recentemente, Zachariadis *et al*. (2010) trataram o problema de roteamento de veículos com coletas e entregas simultâneas, que considera que os clientes exigem a coleta e a entrega simultânea de serviços. Os autores utilizaram uma abordagem metaheurística, que introduz um algoritmo de memória adaptativa que guarda e combina características promissoras da solução para gerar soluções de alta qualidade, conduzindo a busca para diversas regiões do espaço de soluções. A metaheurística proposta foi testada para instâncias entre 50 e 400 clientes e foram obtidas soluções de alta qualidade com um esforço computacional limitado.

A literatura mostra que há uma grande variedade de problemas relacionados ao PDPTW assim como métodos de resoluções propostos para a sua resolução. Isso mostra o interesse e a importância de estudar novas técnicas de busca em vizinhança de grande porte e suas combinações com diferentes metaheurísticas de otimização.

Neste trabalho foi implementada a heurística de busca em vizinhança de grande porte (*Large Neighborhood Search* - LNS), introduzida por Shaw (1997), que trabalha com múltiplas vizinhanças definidas implicitamente através da utilização combinada de diferentes métodos de destruição e reconstrução da solução corrente.

O artigo está organizado assim: A seção 2 apresenta o problema tratado neste trabalho e a sua formulação matemática. Na seção 3 é apresentado o método de busca LNS e as heurísticas de destruição e reconstrução da solução corrente. Os experimentos computacionais são reportados e discutidos na seção 4. Por fim, na seção 5 são apresentadas as conclusões e trabalhos futuros.

## 2. Modelo Matemático

Nesta seção é apresentado o modelo matemático descrito por Ropke e Pisinger (2006a), o qual se baseia no modelo proposto por Desaulniers *et al*. (2002). Como o PDPTW é resolvido heuristicamente, o modelo não é apresentado na forma linear inteira.

Considere o PDPTW contendo $n$ pedidos e $m$ veículos. O problema é definido em um grafo onde $P = \{1,...,n\}$ é o conjunto de nós de coleta e $D = \{n+1,...,2n\}$ é o conjunto de nós de entrega. O pedido $i$ é representado pelos nós $i$ e $i+n$. Sendo $K$ o conjunto de todos os veículos, então $|K| = m$. Um veículo pode não ser capaz de atender qualquer pedido devido à restrição de capacidade. Então $K_i$ é o conjunto dos veículos capazes de servir ao pedido $i$, e $P_k \subseteq P$ e $D_k \subseteq D$ são os conjuntos de

coletas e entregas, respectivamente, que podem ser servidos pelo veículo $k$. Assim, para todo $i$ e todo $k$ temos que $k \in K_i \Leftrightarrow i \in P_k \wedge i \in D_k$. Pedidos onde $K_i \neq K$ são chamados pedidos especiais. Define-se ainda $N = P \cup D$ e $N_k = P_k \cup D_k$. Sejam $\tau_k = 2n + k$, $k \in K$, e $\tau'_k = 2n + m + k$, $k \in K$ os nós que representam, respectivamente, o terminal de início e o terminal de fim do veículo $k$. O grafo $G=(V,A)$ consiste dos nós $V = N \cup \{\tau_1,\ldots,\tau_k\} \cup \{\tau'_1,\ldots,\tau'_k\}$ e dos arcos $A = V \times V$. Para cada veículo, temos um subgrafo $G_k = (V_k, A_k)$, onde $V_k = N_k \cup \{\tau_k\} \cup \{\tau'_k\}$ e $A_k = V_k \times V_k$. Para cada aresta $(i, j) \in A$ é atribuida uma distância $d_{ij} \geq 0$ e um tempo de viagem $t_{ij} \geq 0$. Supõe-se que as distâncias e os tempos são não negativos, ou seja, $d_{ij} \geq 0$, $t_{ij} \geq 0$ e que os tempos devem satisfazer a desigualdade triangular $t_{ij} \leq t_{il} + t_{li}$ para todos os $i,j,l \in V$.

Para eliminar os subdeslocamentos e facilitar a modelagem da restrição de precedência, é assumido que $t_{i,n+i} + s_i > 0$. Cada nó $i \in V$ tem um tempo de serviço $s_i$ e uma janela de tempo $[a_i, b_i]$. O tempo de serviço representa o tempo necessário para carga ou descarga, e a janela de tempo indica quando a visita a um determinado local deve começar. Uma visita ao nó $i$ só pode ser realizada dentro do intervalo de tempo $[a_i , b_i]$. Um veículo pode chegar antes do tempo $a_i$, mas deve aguardar até este horário para iniciar sua atividade de carga ou descarga.

Para cada nó $i \in N$, tem-se que $l_i$ é a quantidade de bens que devem ser carregados no veículo no nó, $l_i \geq 0$ para $i \in P$, e $l_i = - l_{i-n}$ para $i \in D$. A capacidade do veículo $k \in K$ é denotada por $C_k$. Quatro tipos de variáveis de decisão são utilizadas no modelo matemático: *a*) $x_{ijk}$ para $i, j \in V$, $k \in K$ é uma variável binária que assume o valor 1 se o arco $(i, j)$ for usado pelo veículo $k$ e 0 caso contrário; *b*) $S_{ik}$ para $i \in V$, $k \in K$ é uma variável inteira não negativa que indica quando o veículo $k$ começa o serviço no local $i$; *c*) $L_{ik}$, $i \in V$, $k \in K$ é uma variável inteira não negativa que indica a quantidade de mercadorias no veículo $k$ depois de ter servido o nó $i$; *d*) $Z_i$ , $i \in P$, é uma variável binária que assume valor 1 se o pedido $i$ for colocado no banco de pedidos e 0 caso contrário. $S_{ik}$ e $L_{ik}$ estão bem definidas somente quando o veículo $k$ realmente visita o nó $i$. O modelo matemático fica:

$$\min \propto \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ijk} + \beta \sum_{k \in K} \left(S_{\tau'_k,k} - a_{\tau_k}\right) + \gamma \sum_{i \in P} Z_i \quad (1)$$

Sujeito a:

$$\sum_{k \in K_i} \sum_{j \in N_k} x_{ijk} + z_i = 1 \quad \forall i \in P \quad (2)$$

$$\sum_{j \in V_k} x_{ijk} - \sum_{j \in V_k} x_{j,n+1,k} = 0 \quad \forall k \in K, \forall i \in P_k \quad (3)$$

$$\sum_{j \in P_k \cup \{\tau'_k\}} x_{\tau_k,j,k} = 1 \quad \forall k \in K \quad (4)$$

$$\sum_{i \in D \cup \{\tau_k\}} x_{\tau'_k,j,k} = 1 \quad \forall k \in K \quad (5)$$

$$\sum_{i \in V_k} x_{ijk} - \sum_{i \in V_k} x_{jik} = 0 \quad \forall k \in K, \forall j \in N_k \quad (6)$$

$$S_{ik} + s_i + t_{ij} - S_{jk} \leq M\left(1 - x_{ijk}\right), \forall k \in K, \forall (i,j) \in A_k, M \gg 0 \quad (7)$$

$$a_i \leq S_{ik} \leq b_i \quad \forall k \in K, \forall i \in V_k \quad (8)$$

$$S_{ik} \leq S_{n+1,k} \quad \forall k \in K, \forall i \in P_k \quad (9)$$

$$L_{ik} + l_i - L_{jk} \leq M\left(1 - x_{ijk}\right), \forall k \in K, \forall (i,j) \in A_k, M \gg 0 \quad (10)$$

$$L_{ik} \leq C_k \quad \forall k \in K, \forall i \in V_k \quad (11)$$

$$L_{\tau_k k} = L_{\tau'_k k} = 0 \quad \forall k \in K \quad (12)$$

$$x_{ijk} \in \{0,1\} \quad \forall k \in K, \forall (i,j) \in A_k \quad (13)$$

$$z_i \in \{0,1\} \quad \forall i \in P \quad (14)$$

$$S_{ik} \geq 0 \quad \forall k \in K, \forall i \in V_k \quad (15)$$

$$L_{ik} \geq 0 \quad \forall k \in K, \forall i \in V_k \quad (16)$$

A função objetivo minimiza a soma ponderada da distância total percorrida pelos veículos, o somatório do tempo total gasto por cada veículo e o número de pedidos colocados no *banco de pedidos*. O tempo total gasto por cada veículo é definido como o horário em que o veículo chegou ao terminal final menos o horário de início da jornada, que é dado a priori, multiplicado pelos respectivos pesos $\propto$, $\beta$ e $\gamma$

A restrição (2) garante que cada local de coleta é visitado ou que o pedido correspondente é colocado no banco de pedidos. A restrição (3) atende às restrições de precedência e de acoplamento. As restrições (4) e (5) garantem que os veículos deixem seus terminais de início e chegam a seus respectivos terminais de fim. E juntamente com restrição (6), garantem que os pares consecutivos entre $\tau_k$ e $\tau'_k$ (terminais de inicio e fim) são formados para cada $k \in K$. As restrições (7) e (8) garantem que $S_{ik}$ está definida corretamente ao longo dos caminhos e que as janelas de tempo são obedecidas, para $M$ suficientemente grande. Estas restrições também impedem a formação de subrotas. A restrição (9) garante que cada coleta ocorra antes da respectiva entrega. As restrições (10), (11) e (12) garantem que a carga variável é definida corretamente ao longo dos caminhos e que as restrições de capacidade dos veículos são respeitadas, para $M$ suficientemente grande. As demais restrições se referem ao tipo e a não negatividade das variáveis de decisão

## 3. Large Neighborhood Search – LNS

A heurística de busca em vizinhança de grande porte (LNS) foi introduzida por Shaw (1997). Esta heurística é similar à heurística destrói e reconstrói (*Ruin and Recreate* - RR) proposta por Schrimpf *et al*. (2000). De acordo com Ropke and Pisinger (2006a), a estrutura LNS pode ser considerada uma variante da busca em vizinhança variável (*Variable Neighborhood Search* - VNS) descrita em Hansen e Mladenovic (1999). A principal diferença é que a VNS opera em um tipo de vizinhança com tamanho variável, enquanto a LNS opera com diferentes estruturas de vizinhanças. Shaw (1997, 1998) e Bent e Van Hentenry (2004) utilizaram a heurística LNS para o VRPTW e obtiveram bons resultados. Foi também aplicada ao PDPTW por Bent e Van Hentenry (2003).

**Função LNS** (solução $s$, $q \in$ N)

1. solução $s_{melhor} := s$ ;
2. **Repita**
3. $s' := s$ ;
4. remova $q$ pedidos de $s'$;
5. reinsira os $q$ pedidos removidos resultando em $s'$;
6. **se** $s'$ for melhor do que $s_{melhor}$ **então**
7. $s_{melhor} := s'$;
8. **se** $s'$ satisfaz os critérios de aceitação **então**
9. $s := s'$;
10. **até** que o critério de parada seja satisfeito
11. **retorna** $s_{melhor}$ ;

Algoritmo 1- Heurística LNS

No pseudocódigo da heurística LNS, apresentado no Algoritmo 1, o parâmetro $q \in \{0, ..., n\}$ determina o alcance da busca, pois representa o número de pedidos que deverão ser removidos. O algoritmo assume que uma solução inicial viável é gerada por alguma heurística construtiva simples. Na linha 4, $q$ pedidos são removidos da solução por uma heurística destrutiva e na linha seguinte estes pedidos são reinseridos na solução corrente por uma heurística construtiva. Quando a solução resultante $s'$ atende ao critério de aceitação, a nova busca é feita a partir desta. Quando $s'$ é melhor que a melhor solução encontrada até o momento então $s_{melhor}$ é atualizada com a solução $s'$. Este processo é repetido até que um critério de parada seja satisfeito. Neste trabalho a heurística é interrompida após um determinado tempo de processamento.

### 3.1 Heurísticas

Diferentes heurísticas para remoção (ou destruição) de soluções têm sido propostas. Neste trabalho foram implementadas três heurísticas propostas por Ropke e Pisinger (2006a): remoção aleatória, remoção da pior posição e remoção Shaw. As duas últimas possuem um parâmetro $p$, que determina o grau de aleatoriedade da heurística. A seguir são detalhadas estas heurísticas.

#### 3.1.1 Remoção Aleatória

O algoritmo de remoção aleatória simplesmente seleciona $q$ pedidos de forma aleatória e os remove da solução.

#### 3.1.2 Remoção da Pior Posição

O algoritmo de remoção da pior posição procura selecionar os pedidos que parecem ter sido colocados numa posição errada na solução, gerando um custo maior. O custo de um pedido $i$ atendido por algum veículo na solução $s$, é definido como $c(i, s) = f(s) - f\_i(s)$, onde $f\_i(s)$ é o custo da solução sem o pedido $i$. O pedido não é incluído no banco de pedidos, mas removido completamente para posterior reinserção na próxima etapa. Esta heurística procura eliminar os pedidos com alto custo $c(i, s)$ segundo o grau de aleatoriedade dado pelo parâmetro $p$. Isto é feito para evitar situações onde os mesmos pedidos são removidos várias vezes.

**Função** Remoção Pior Posição ($q \in N$, $p \in R_+$)

1 **enquanto** $q > 0$ **faça**
  Agrupar: L = pedidos $i$ planejados, em ordem decrescente de custo $c(i, s)$;
3  escolhe um número aleatório $y$ no intervalo [0,1);
4  pedido: $r = L[y^p|L|]$;
5  remove $r$ da solução $s$;
6  $q = q - 1$;
7 **fim enquanto**

Algoritmo 2 - Heurística Remoção da Pior Posição

Os pedidos são colocados em uma lista L, em ordem decrescente do valor do custo $c(i, s)$. Em cada iteração, um número aleatório $y$ é escolhido no intervalo [0, 1). Então, o pedido $r = L[y^p | L |]$ é removido da solução e os custos $c(i, s)$ e os custos de todas as rotas restantes são atualizados. O grau de randomização é controlado pelo parâmetro $p$ denominado $p_{pior}$. Este procedimento é repetido até que $q$ pedidos sejam removidos da solução.

#### 3.1.3 Remoção Shaw

Esta heurística foi proposta por Shaw (1997, 1998) e foi ligeiramente modificada por Ropke e Pisinger (2006a) para se adequar ao PDPTW. A ideia central é remover os pedidos que são semelhantes. Dessa forma fica razoavelmente fácil embaralhar os pedidos semelhantes e, assim, criar soluções melhores. A similaridade de dois pedidos $i$ e $j$ é definida usando uma medida de parentesco $R(i, j)$, que é definida como:

$$R(i,j) = \varphi\left(d_{A(i),A(j)} + d_{B(i),B(j)}\right) + \chi\left(\left|T_{A(i)} - T_{A(j)}\right| + \left|T_{B(i)} - T_{B(j)}\right|\right) + \psi\left|l_i - l_j\right| + \omega\left(1 - \frac{|K_i \cap K_j|}{\min\{|K_i|,|K_j|\}}\right) \quad (17)$$

Onde os termos ponderados $\varphi$, $\chi$, $\psi$ e $\omega$, medem, respectivamente, a distância, a conexão temporal, compara a demanda de capacidade dos pedidos e garante que dois pedidos têm uma medida de parentesco alta se apenas poucos ou nenhum veículo é capazes de servir ambos. $A(i)$ e $B(i)$ denotam os locais de coleta e entrega do pedido $i$, e $T_i$ indica o momento no qual a localização $i$ é visitada.

Quanto menor for $R(i, j)$, mais relacionados são os dois pedidos. O procedimento para a remoção de pedidos é mostrado no pseudocódigo do Algoritmo 3.

**Função** Remoção Shaw ($q \in N, p \in R_+$)

```
1   Pedidos: r = pedido selecionado aleatoriamente de s;
2   conjunto de pedidos: D = { r };
3   enquanto |D| < q faça
4      r = pedido selecionado aleatoriamente de D;
5      agrupar: L = todos os pedidos de s que não estão em D;
6      classificar L de modo que
7         i < j  ⇒ R(r, L[ i ]) < R(r, L[ j ]);
8         Escolher um número aleatório y do intervalo [0,1);
9         D = D ∪ {L[ y^p|L|]};
10  fim enquanto
11 remova os pedidos D de s;
```

Algoritmo 3 - Heurística de Remoção de Relacionados

O procedimento inicialmente escolhe um pedido aleatório para remover e nas iterações seguintes ele escolhe os pedidos que são semelhantes aos pedidos já removidos. Um parâmetro deterministico $p \geq 1$ introduz alguma aleatoriedade na seleção dos pedidos. Um valor baixo de $p$ corresponde a uma maior aleatoriedade.

### 3.2 Heurísticas de Inserção

Para a inserção dos pedidos são utilizadas heurísticas simples e rápidas. Todas as heurísticas recebem um número de rotas parciais e o número dos pedidos que foram removidos pelas heurísticas destrutivas e tentam reinseri-los na solução. As heurísticas construtivas utilizadas são classificadas como paralelas, ou seja, constroem várias rotas ao mesmo tempo.

#### 3.2.1 Heurística Gulosa Básica

Esta heurística procura inserir o pedido na posição de menor custo. Ela realiza no máximo $n$ iterações, uma vez que insere um pedido em cada iteração. $\Delta f_{i,k}$ representa a mudança do valor da função objetivo ocorrido pela inserção do pedido $i$ na rota $k$, na posição que minimiza o valor da função objetivo. Se não for possível inserir o pedido $i$ na rota $k$, então $\Delta f_{i,k} = \infty$. Definimos $c_i$ como $c_i = min_{k \in K}\{ \Delta f_{i,k} \}$. Em outras palavras, $c_i$ é o custo de inserir o pedido $i$ na sua melhor posição dentre todas as rotas. Esta posição é denotada como *posição de custo mínimo*. Por fim, é escolhido o pedido $i$ que minimiza $c_i$, ou seja, $\min_{i \in U} c_i$ ,o qual é inserido em sua posição de custo mínimo. Este processo continua até que todos os pedidos tenham sido inseridos ou que nenhum outro pedido possa ser inserido. Um problema desta heurística é que ela muitas vezes adia a colocação de pedidos difícieis (com $c_i$ muito elevado) para as últimas iterações, onde não há muitas oportunidades para a inserção de pedidos, porque muitas das rotas já estão completas.

#### 3.2.2 Heurística de Arrependimento

As heurísticas de arrependimento foram propostas por Potvin e Rousseau (1993). Elas baseiam-se na heurística gulosa, porém incorporam uma espécie de avaliação do tipo "olhar à frente" quando vai selecionar um pedido para ser inserido. O valor do arrependimento para cada pedido é calculado como segue: seja $x_{ik} \in \{1,..., m\}$, uma variável que indica a rota para a qual o pedido $i$ tem o *k-ésimo* menor custo de inserção, ou seja, $\Delta f_{i,Xik} \leq \Delta f_{i,Xik'}$ para $k \leq k'$. Usando esta notação, podemos expressar $c_i$ como $c_i = \Delta f_{i,Xi1}$. Na heurística de arrependimento define-se um valor de arrependimento $c_i^*$ como $c_i^* = \Delta f_{i,Xi2} - \Delta f_{i,Xi1}$. Em outras palavras, o valor de arrependimento é a diferença entre o custo

de inserir o pedido na sua melhor rota e na sua segunda melhor rota. Em cada iteração a heurística de arrependimento escolhe inserir o pedido *i* que maximiza $\max_{i \in U} c_i^*$. O pedido é inserido na posição de custo mínimo. A heurística pode ser estendida de forma natural, definindo uma classe de heurísticas de arrependimento. A heurística do *k-arrependimento*, onde *k* representa o número de rotas a serem consideradas, é a heurística de construção que, em cada passo, escolhe para inserir o pedido *i* que maximiza:

$$\max_{i \in U} \left\{ \sum_{j=2}^{k} \left( \Delta f_{i,x_{ij}} - \Delta f_{i,x_{i1}} \right) \right\} \quad (18)$$

Se algum pedido não puder ser inseridos em pelo menos *m* - *k* + 1 rotas, então são escolhidos aqueles que podem ser inseridos no menor número de rotas. O pedido é inserido na sua posição de custo mínimo. A grosso modo, pode-se dizer que a heurística de arrependimento para *k* > 2 pesquisa o custo de inserção de 1 pedido nas *k* melhores rotas e o insere na rota cuja diferença do custo entre inserir na melhor rota e nas *k* - 1 melhores rotas é o maior.

## 4. Experimentos Computacionais

Para testar a heurística LNS utilizou-se os dados construídos por Li and Lim (2001) com até 100 pedidos, disponíveis em SINTEF (http://www.sintef.no/Projectweb/TOP/PDPTW/Li--Lim-benchmark/100-customers/). Nestes dados os veículos partem e retornam a um único depósito. Todos os pedidos devem ser obrigatoriamente atendidos, e consequentemente, não podem ser colocados no banco de pedidos. Além disso, todos os veículos podem atender todos os pedidos. As instâncias propostas por Li e Lim (2001), são classificativas em 3 tipos. No tipo R, todas as localizações estão distribuidas uniformemente. Nas instâncias do tipo C, as localizações estão distribuídas em 10 grupos geográficos e no tipo RC, 50% das localizações são alocadas em 10 grupos e os outros 50% são distribuídos uniformemente. Neste trabalho foram selecionados 3 conjunto de dados de cada tipo.

Para a definição dos parâmetros, forma utilizados os melhores resultados obtidos e sugeridos por Ropke and Pisinger (2006a). A heurística de remoção Randômica não tem parâmetros, a heurística de remoção Shaw é controlada por cinco parâmetros, $\varphi$, $\chi$, $\psi$, $\omega$ e *p*, enquanto a heurística de remoção da Pior Posição é controlada por apenas um parâmetro, $p_{pior}$. Os valores são dados pelo vetor ($\varphi$, $\chi$, $\psi$, $\omega$, *p*, $p_{pior}$) = (9, 3, 2, 5, 6, 3). Os parâmetros da função objetivo, $\propto$, $\beta$ e $\gamma$, foram considerados como ($\propto$, $\beta$, $\gamma$) = (1,1,1000), evitando assim que pedidos fossem colocados no banco de pedidos.

Os testes computacionais foram realizados com a combinação entre as três heurísticas de remoção e as duas heurísticas de reconstrução da solução. Cada par remoção-reconstrução constituiu uma versão do LNS de para resolver o problema. As seis versões do LNS foram executadas 10 vezes para cada instância, sendo que cada execução durou 1 hora de processamentgo. Os experimentos foram realizados em um computador Intel i7, com 12GB de memória RAM, Windows 7, linguagem de programação C++ no programa Code::Blocks 12.11.

A maioria dos resultados na literatura apresentados para as instâncias de Li e Lim (2001), consideram apenas o número de veículos e a distância total percorrida. Porém para poder avaliar a função objetivo proposta neste artigo, foram considerados todos os atributos da função objetivo dada pela expressão (1) do modelo matemático. Desta forma, torna-se inviável comparação dos resultados obtidos neste trabaho com aqueles os melhores resultados disponibilizados pelo SINTEF, os quais consideram apenas o número de veículos e da distância total percorrida.

A solução viável inicial empregada nestes testes foi criada usando um método de inserção sequencial, que constrói uma rota por vez até que todos os pedidos sejam planejados. O critério para seleção dos pedidos considera a minimização da mesma função objetivo dada pela expressão (1).

Para facilitar a visualização dos resultados, estes foram agrupados combinando cada uma das três heurísticas de remoção com as duas heurísticas de inserção. Desta forma são apresentadas três tabelas de resultados, sendo que a Tabela 1 se refere à Remoção Randômica-Inserção Gulosa e

Remoção Randômica-Inserção de Arrependimento. Na Tabela 2 são apresentados os resultados da Remoção Shaw-Inserção Gulosa e Remoção Shaw-Inserção de Arrependimento e finalmente, na Tabela 3, os resultados das combinações Remoção Pior Posição-Inserção Gulosa e Remoção Pior Posição- Inserção de Arrependimento.

**Tabela 1.** Combinação Remoção Randômica com as Inserções Gulosa e de Arrependimento.

| Remoção Randômica | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Inserção Gulosa | | | | | Inserção Arrependimento | | | |
| **Problema** | **FO** | **Distância Total** | **Tempo Espera** | **Total de Veículos** | **FO** | **Distância Total** | **Tempo Espera** | **Total de Veículos** |
| LC101 | 10.657,87 | 828,94 | 0,00 | 10 | 10.657,87 | 828,94 | 0,00 | 10 |
| LC102 | 12.472,51 | 1.538,92 | 394,67 | 14 | 12.472,51 | 1.538,92 | 394,67 | 14 |
| LC103 | 13.265,25 | 1.796,51 | 672,24 | 17 | **13.171,09** | 1.812,27 | 546,54 | 17 |
| LR101 | 5.228,90 | 1.667,68 | 893,55 | 19 | 5.228,90 | 1.667,68 | 893,55 | 19 |
| LR102 | 5.353,81 | 1.831,32 | 691,17 | 20 | **5.345,18** | 1.841,28 | 662,61 | 20 |
| LR103 | 5.393,89 | 1.971,73 | 450,42 | 20 | 5.393,89 | 1.971,73 | 450,43 | 20 |
| LRC101 | 5.210,69 | 1.864,68 | 481,33 | 17 | **5.114,64** | 1.867,85 | 378,93 | 16 |
| LRC102 | **5.485,71** | 2.156,21 | 173,29 | 17 | 5.495,05 | 2.092,84 | 309,37 | 18 |
| LRC103 | 5.268,52 | 2.037,92 | 192,68 | 18 | **5.244,61** | 2.026,92 | 190,77 | 18 |

Os resultados da Tabela 1 mostram que as duas inserções chegaram ao mesmo resultado em quatro problemas. Para os demais, problemas, a combinação da Remoção Randômica com a Inserção de Arrependimento produziu resultados superiores em quatro problemas e a Inserção Gulosa em apenas um problema.

**Tabela 2.** Combinação Remoção Shaw com as Inserções Gulosa e de Arrependimento.

| Remoção Shaw | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Inserção Gulosa | | | | | Inserção Arrependimento | | | |
| **Problema** | **FO** | **Distância Total** | **Tempo Espera** | **Total de Veículos** | **FO** | **Distância Total** | **Tempo Espera** | **Total de Veículos** |
| LC101 | 10.657,87 | 828,94 | 0,00 | 10 | 10.657,87 | 828,94 | 0,00 | 10 |
| LC102 | 12.472,51 | 1.538,92 | 394,67 | 14 | 12.472,51 | 1.538,92 | 394,67 | 14 |
| LC103 | 13.302,09 | 1.702,33 | 897,43 | 17 | **13.203,24** | 1.769,47 | 664,30 | 16 |
| LR101 | 5.228,90 | 1.667,68 | 893,55 | 19 | 5.228,90 | 1.667,68 | 893,55 | 19 |
| LR102 | 5.353,81 | 1.831,32 | 691,17 | 20 | **5.345,18** | 1.841,28 | 662,61 | 20 |
| LR103 | **5.383,85** | 1.968,83 | 446,19 | 20 | 5.393,89 | 1.971,73 | 450,43 | 20 |
| LRC101 | 5.210,77 | 1.864,75 | 481,26 | 17 | **5.114,64** | 1.867,85 | 378,93 | 16 |
| LRC102 | 5.495,05 | 2.092,84 | 309,37 | 18 | **5.487,15** | 2.155,29 | 176,57 | 17 |
| LRC103 | 5.268,52 | 2.037,92 | 192,68 | 18 | **5.255,39** | 2.037,47 | 180,46 | 18 |

A Remoção Shaw combinada com a Inserção Arrependimento produziu melhores resultados na maioria dos problemas estados. Dentre os nove problemas, ela foi superior em cinco,

empatou em três e foi pior do que quando combinada com a Inserção Gulosa em apenas um problema.

**Tabela 3.** Combinação da Remoção Pior Posição com as Inserções Gulosa e de Arrependimento.

| Remoção Pior Posição | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Inserção Gulosa | | | | Inserção Arrependimento | | | |
| **Problema** | **FO** | **Distância Total** | **Tempo Espera** | **Total de Veículos** | **FO** | **Distância Total** | **Tempo Espera** | **Total de Veículos** |
| LC101 | 10.657,87 | 828,94 | 0,00 | 10 | 10.657,87 | 828,94 | 0,00 | 10 |
| LC102 | 12.530,38 | 1.534,91 | 460,56 | 14 | **12.472,51** | 1.538,92 | 394,67 | 14 |
| LC103 | 13.336,95 | 1.731, 23 | 874,49 | 17 | **13.150,64** | 1.793,48 | 563,68 | 16 |
| LR101 | 5.228,90 | 1.667,68 | 893,55 | 19 | 5.228,90 | 1.667,68 | 893,55 | 19 |
| LR102 | 5.353,96 | 1.831,48 | 691,01 | 20 | **5.345,18** | 1.841,28 | 662,61 | 20 |
| LR103 | 5.393,89 | 1.971,73 | 450,43 | 20 | **5.376,07** | 1.981,23 | 413,61 | 19 |
| LRC101 | 5.121,38 | 1.874,59 | 372,20 | 16 | **5.114,64** | 1.867,85 | 378,93 | 16 |
| LRC102 | **5.485,71** | 2.156,21 | 173,29 | 17 | 5.495,05 | 2.092,84 | 309,37 | 18 |
| LRC103 | 5.267,44 | 2.036,98 | 193,47 | 18 | **5.246,13** | 2.027,34 | 191,45 | 18 |

A combinação da heurística de Remoção da Pior Posição com a Inserção de Arrependimento foi superior à combinação com a Inserção Gulosa em seis problemas, empatou em dois e foi inferior em um único problema.

Comparando os valores da FO, é possível verificar que a combinação Remoção da Pior Posição com a Inserção de Arrependimento foi aquela alcançou os melhores resultados o maior número de vezes, ou seja, para quatro problemas. Em segundo lugar ficou a combinação da Remoção Randômica com Inserção de Arrependimento. É importante ressaltar que, em alguns problemas, mais de uma combinação obteve o melhor resultado.

**Tabela 4.** Valor da função objetivo de todas as versões da heurística LNS.

| | Remoção Randômica | | Remoção Shaw | | Remoção da Pior Posição | |
|---|---|---|---|---|---|---|
| **Inserções** | **Gulosa** | **Arrependimento** | **Gulosa** | **Arrependimento** | **Gulosa** | **Arrependimento** |
| **Problema** | **FO** | **FO** | **FO** | **FO** | **FO** | **FO** |
| LC101 | 10.657,87 | 10.657,87 | 10.657,87 | 10.657,87 | 10.657,87 | 10.657,87 |
| LC102 | 12.472,51 | 12.472,51 | 12.472,51 | 12.472,50 | 12.530,38 | 12.472,51 |
| LC103 | 13.265,25 | 13.171,09 | 13.302,09 | 13.203,24 | 13.302,09 | **13.150,64** |
| LR101 | 5.228,90 | 5.228,90 | 5.228,90 | 5.228,90 | 5.228,90 | 5.228,90 |
| LR102 | 5.353,81 | **5.345,18** | 5.353,81 | **5.345,18** | 5.353,96 | **5.345,18** |
| LR103 | 5.393,89 | 5.393,89 | 5.383,85 | 5.393,89 | 5.393,89 | **5.376,07** |
| LRC101 | 5.210,69 | **5.114,64** | 5.210,77 | **5.114,64** | 5.121,38 | **5.114,64** |
| LRC102 | **5.485,71** | 5.495,05 | 5.495,05 | 5.487,15 | **5.485,71** | 5.495,05 |
| LRC103 | 5.268,52 | **5.244,61** | 5.268,52 | 5.255,39 | 5.267,44 | 5.246,13 |

Observando os resultados obtidos nos experimentos computacionais, é possível verificar que a Inserção de Arrependimento é mais eficiente do que a Inserção Gulosa, independentemente da heurística de remoção aplicada. Por outro lado, comparando o valor da FO para os diferentes problemas, a Remoção da Pior Posição foi capaz de atingir a melhor solução o maior número de vezes do que as outras heurísticas de remoção. Desta forma, verifica-se, neste experimento, que a combinação da Remoção da Pior Posição com a Inserção de Arrependimento foi mais eficiente do que as demais combinações.

## 5. Conclusão

Este trabalho trata do PDPTW e apresenta uma implementação da heurística LNS combinando os procedimentos de remoção do tipo Randômico, Shaw, w da Pior Posição com os de inserção Gulosa e de Arrependimento. Os testes computacionais mostraram que a combinação da remoção da Pior Posição com a inserção de Arrependimento produziu os melhores resultados.

A minimização do número de veículos é utilizada na literatura como prioridade no problema de roteamento de veículos. A implementação apresentada neste trabalho tem como objetivo minimizar todos os termos que compõem a função objetivo, ou seja, a distância percorrida, o tempo total de operação dos veículos e o número de veículos utilizados. O número de veículos encontrados se aproximou dos melhores resultados registrados no SINTEF somente em alguns casos. Para o problema LC101, todas as versões chegaram à melhor solução da literatura, pois neste caso o tempo de espera é igual a zero, o que torna a função objetivo do modelo considerado igual à função objetivo utilizada na literatura. Percebe-se portanto, que o tempo de espera da solução é considerável e não foi desprezado. Para os demais problemas, as soluções encontradas são piores do que as melhores soluções conhecidas.

Na literatura, os casos Li e Lim (2001) são resolvidos usando a heurística LNS, com $\alpha$ igual a 1 e $\beta$ igual e zero na função objetivo, portanto os melhores resultados disponíveis na literatura servem como referência, mas não podem ser comparados diretamente com os resultados obtidos neste artigo. A propósito, a implementação apresentada foi capaz de obter a melhor para o problema LC101.

Devido à complexidade de atender às janelas de tempo do PDPTW, acredita-se que uma solução inicial de melhor qualidade facilite heurística de busca a encontrar melhores resultados. Acredita-se que utilizando um procedimento de busca local, após o procedimento destrói-reconstrói, que primeiro minimiza o número de veículos e depois minimiza a distância percorrida, pode produzir melhores resultados com a heurística proposta.

## Agradecimentos

Os autores agradecem ao CNPq, à FAPEMIG e à UFOP pelo apoio recebido no desenvolvimento deste trabalho.

## Referências

**Bent, R.; P. Van Hentenryck**. (2003) A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Lecture Notes in Computer Science*, Vol. 2833. *Proc. Internat. Conf. Constraint Programming (CP-2003)*, p. 123–137.

**Bent, R.; P. Van Hentenryck** (2004) A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Sci.* vol. 38(4), p. 515–530.

**Desaulniers, G.; Desrosiers, J.; Erdmann, A.; Solomon, M. M.; Soumis, F**. (2002) The VRP with pickup and delivery. P. Toth, D. Vigo, eds. *The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications*, v. 9, p. 225–242, SIAM, Philadelphia.

**Dumas, Y.; Desrosiers, J.; Soumis, F**. (1991) The pickup and delivery problem with time windows. *European. Journal of Operations Research,* v. 54. p. 7–22.

**Hansen, P.; Mladenovic, N**. (1999) An introduction to variable neighborhood search. *In*: S. Voss et al. (Eds.), *Meta-Heuristics, Advances and Trends in Local Search Paradigms for Optimization*, p. 433-458, Kluwer, Boston.

**Li, H.; Lim, A**. (2001) A metaheuristic for the pickup and delivery problem with time windows. *ICTAI-2001, 13th IEEE Conf. Tools Artificial Intelligence*, p. 160-170, Dallas.

**Li, H.; Lim, A.; Rodrigues, B**. (2002) Solving the pickup and delivery problem with time windows using "squeaky wheel" optimization with local search. *Technical Report*, Singapore Management University, Singapore.

**Nanry, W. P.; Barnes, J. W**. (2000) Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Res. Part B,* v. 34, p. 107–121.

**Potvin, J. Y.; Rousseau, J. M**. (1993) A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, v. 66, p. 331–340.

**Ropke, S.; Pisinger, D**. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, v. 40, p. 455–472.

**Ropke, S.; Pisinger, D**. (2006b) A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, v. 171, p. 750–775.

**Schrimpf, G.; Schneider, J.; Stamm-Wilbrandt, H.; Dueck, G**. (2000) Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, v. 159, p. 139–171.

**SINTEF**. Vehicle routing and travelling salesperson problems. Disponível em (www.sintef.no/Projectweb/TOP/PDPTW/Li--Lim-benchmark/100-customers/, 4, 2013.

**Shaw, P**. (1997) A new local search algorithm providing high quality solutions to vehicle routing problems. *Technical Report*, Department of Computer Science, University of Strathclyde, Scotland.

**Shaw, P**. (1998) Using constraint programming and local search methods to solve vehicle routing problems. *Proc. CP-98 Fourth Internat. Conf. Principles Practice Constraint Programming*.

**Solomon, M. M**. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research,* v. 35, p. 254–265.

**Xu, H., Z.-L. Chen, S. Rajagopal, S. Arunapuram**. (2003). Solving a practical pickup and delivery problem. *Transportation Sci*. v. 37, p. 347–364.

**Zachariadis, E. E.; Christos D. Tarantilis, C. D.; Kiranoudis, C. T.** (2010) An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research,* v. 202, p. 401–411.