UMA HEURÍSTICA BASEADA EM ILS PARA O PROBLEMA DE LOCALIZAÇÃO-ROTEAMENTO CAPACITADO

Gustavo Rezende Carvalho
Luiz Satoru Ochi
Aline de Paula Nascimento
Lucídio dos Anjos Formiga Cabral
Instituto de Computação - Universidade Federal Fluminense (UFF)
Instituto de Computação

RESUMO

O PLR (Problema de Localização-Roteamento) é a combinação de dois problemas combinatoriais de elevada complexidade computacional (NP-Difícil): o Problema de Localização de Facilidades e o Problema de Roteamento de Veículos. Este trabalho propõe uma heurística utilizando conceitos da meta-heurística ILS (*Iterated Local Search*) com busca local VND (*Variable Neighborhood Descent*) para a solução do PLR de dois níveis com capacidade. O algoritmo foi testado para um conjunto de instâncias presentes da literatura obtendo resultados computacionais competitivos.

PALAVRAS CHAVES: Metaheuristica, Roteamento de Veículos, Localização de Facilidades.

Área Principal: Meta-heurísticas

ABSTRACT

The LRP (Location-Routing Problem) can be defined as combination of two combinatorial problems with high computational complexity (NP-Hard): FLP (Facility Location Problem) and VRP (Vehicle Routing Problem). This paper proposes a hybrid heuristic using Iterated Local Search algorithm with Variable Neighborhood Descent as a local search to solve the capacity two levels LRP. The algorithm was tested with benchmarks from the literature obtaining competitive results.

KEYWORDS: Metaheuristics Vehicle Routing Problem, Location Problem

Main Area: Metaheuristics

1. Introdução

O problema de localização-roteamento de dois níveis com restrições de capacidade (PLRC) se define pela decisão de localizar centros de distribuição/coleta (depósitos) associados a um conjunto de clientes que deverão ser atendidos por suas devidas demandas em uma rota, respeitando a capacidade dos veículos e dos depósitos. Os veículos devem retornar aos depósitos onde iniciam suas rotas. Com isso, pode-se imaginar vários cenários reais aplicados a este problema, como a localização de fábricas, centros de coleta de lixo, prestadores de serviços, lojas, aplicações militares etc. Existem várias organizações hierárquicas para este tipo de problema, porém a mais abordada na literatura é a de dois níveis. Em Laporte (1988) é descrito um sistema de localização composto por três níveis. No primeiro nível situam-se os serviços primários habitualmente compostos por unidades produtivas (hospitais, centros de coleta de lixo, aeroportos etc). No segundo nível encontram-se os serviços secundários, que representam locais de atividades intermediárias tais como armazéns, centros de distribuição, terminais de contentores etc, designados neste texto por depósitos. O terceiro nível é associado aos clientes.

Considerado NP-Difícil (Barreto, 2004), vários algoritmos exatos e heurísticos foram

propostos para a resolução do problema. Na literatura, podem ser encontradas algumas abordagens exatas como o método *branch-and-price* (Akca, 2008), *branch-and-cut* (Contardo, 2010; Belenguer, 2011) e o *set-partitioning* (Baldacci, 2011). Entretanto, os métodos exatos tendem a encontrar dificuldades para achar a solução ótima para instâncias deste problema com mais de 100 clientes. Neste contexto, também foram propostas heurísticas para obtenção de soluções de boa qualidade em um tempo computacional aceitável, como os métodos gulosos de agrupamento, (Barreto, 2007; Sahraeian, 2009), Algoritmos Meméticos (Duhamel, 2008) e baseados em vizinhanças (Hemmelmayer, 2011). Outra alternativa abordada pela literatura são os métodos híbridos, como por exemplo o LRGTS (*Lagrangean Relaxation-Granular Tabu Search*) proposto por Prins (2007).

Uma versão preliminar de um algoritmo baseado na metaheuristica *Iterated Local Search* - ILS foi apresentada em (Carvalho, 2011) para o PLRC com resultados satisfatórios para o banco de teste constituído de 19 instâncias, algumas obtidas da literatura (Or, 1976, Perl, 1983) e outras adaptadas de instâncias relacionadas com o PRV (Gaskell, 1967, Christofides e Eilon, 1969, Min et AL., 1992, Daskin, 1995). Uma versão melhorada deste algoritmo é aqui proposto com modificações que permitiram obter resultados melhores e mais robustos tanto para este banco de teste quanto para o banco de teste proposto por Prins et al. (2004). Esse segundo banco de teste contém 30 instâncias e o número de clientes variam entre 20 e 200 e o número de depósitos de 5 a 10, além disso, possui custo para cada veículo utilizado.

O artigo está organizado da seguinte forma: na sessão 2, descreve-se brevemente o problema quanto a suas restrições e objetivos; na sessão 3, é descrito o algoritmo proposto denotado por ILS-RVND; na sessão 4, apresentam-se os resultados computacionais obtidos utilizando *benchmarks* disponíveis na literatura; finalmente, na sessão 5, encontram-se as conclusões.

2. Definição do problema

O PLRC pode ser definido na estrutura de um grafo. Sendo $G = \{V, E\}$ um grafo onde V é formado pela reunião dos subconjuntos de vértices C e D, tal que $C = \{0,...,n\}$ representam os clientes e $D = \{n+1,...,n+d\}$ são os depósitos i de capacidade w_i e custo de ativação d_i. Cada aresta $\{i, j\} \in E$ possui custo não negativo c_{ij} e cada cliente k possui uma demanda associada q_k. Sendo $E = \{1,..,v\}$ o conjunto de veículos homogêneos de capacidade Q e custo R. O PLRC consiste em localizar um conjunto variável de depósitos e um conjunto de rotas de maneira que: (i) todas as demandas devem ser satisfeitas; (ii) respeite as capacidades dos veículos e dos depósitos; (iii) o cliente seja visitado por apenas um veículo e apenas uma vez; (iv) o veículo retorne ao mesmo depósito de partida da rota; (v) minimize os custos de localização, frota e roteamento envolvidos.

3. Algoritmo Proposto

O algoritmo proposto é baseado na meta-heurística *Iterated Local Search* (ILS), que basicamente inicia com heuristica de construção seguido de uma busca local. Para encontrar outros ótimos locais, após a etapa da busca local é efetuada uma perturbação sobre a solução encumbente e esta solução perturbada passa por um critério de aceitação antes de retornar ao procedimento de busca local (Lourenço, 2002). Atualmente a literatura nos tem mostrado que as melhores versões das heuristicas ILS são aquelas que a busca local é efetuada por uma estrutura do tipo VNS ou VND (Penna *et.al*. 2012a & 2012b, Mine *et.al*. 2011).

Neste contexto, no ILS aqui proposto, a fase de busca local é efetuada por um algoritmo do tipo VND (Método de Descida em Vizinhança Variável – *Variable Neighborhood Descent*) proposto por Mladenović e Hansen (1997). Tal método é essencialmente caracterizado por realizar mudanças sistemáticas de estruturas de vizinhança pertencentes a um conjunto $N = \{N_1, N_2, N_3,..., N_r\}$, de forma determinística. A vizinhança $k' = k + 1 \in N$ é acionada apenas se não houver melhora nas k vizinhanças pesquisadas anteriormente. Estruturas de vizinhanças clássicas

são utilizadas: *shift*, *swap* (1,1), *swap* (2,1), *swap* (2,2), e uma adaptação do movimento *cross*. Além disso, o ILS aqui proposto é do tipo *multi-start* com sementes aleatórias para decidir qual construção será realizada em cada iteração, a ordem das buscas entre vizinhanças e as perturbações. Suas perturbações utilizam movimentos *shift* e *swap* entre depósitos e entre clientes. Podem-se destacar os seguintes procedimentos, que serão explanados em maior detalhes a seguir: estimar a frota, estimar os depósitos, construção, perturbações e RVND (VND com sequenciamento aleatório de vizinhanças). O pseudocódigo do procedimento é indicado no Algoritmo 1.

Algoritmo 1. ILS-RVND(Instância)

```
1.  Estima_Frota(Instância)
2.  depositosEstimados ← Estima_Depositos(Instância)
3.  s0 s1 s2 ← 0
4.  Para depositosEstimados < Instancia.Depositos faça
5.      Melhora ← False
6.      Para i ← 0 até i < MAX_ITER faça
7.          Construção(Instancia, s0)
8.          RVND(Instancia, s0)
9.          s1 ← s0
10.         Se f(s0) < f(s2) faça
11.             s2 ← s0
12.             Melhora ← True
13.         Para j ← 0 até j < MAX_ILS1 faça
14.             Perturba1(Instancia, s0)
15.             RVND(Instancia, s0)
16.             Se f(s0) < f(s1) faça
17.                 s1 ← s0
18.                 j ← 0
19.                 Se f(s0) < f(s2) faça
20.                     s2 ← s0
21.                     Melhora ← True
22.             Senão faça
23.                 s0 ← s1
24.     s0 ← s2
25.     Para j ← 0 até j < MAX_ILS2 faça
26.         Perturba2(Instancia, s0)
27.         RVND(Instancia, s0)
28.         Se f(s0) < f(s1) faça
29.             s1 ← s0
30.             j ← 0
31.             Se f(s0) < f(s2) faça
32.                 s2 ← s0
33.         Senão faça
34.             s0 ← s1
35. Se !Melhora Retorna (s2)
```

3.1 Estimações de Frota e Depósitos

O procedimento para estimar o número mínimo de veículos e depósitos para atender a demanda é bastante simples. Para calcular a frota, a demanda total é dividida pela capacidade do veículo e utiliza-se o teto do resultado da expressão 1. A única diferença para calcular os depósitos é que se utiliza o depósito de maior capacidade para o cálculo. Esses dados servirão de entrada para a fase de construção.

$$Frota = \frac{\sum q_i}{Q} \quad (1)$$

3.2 Construção

A fase de construção é onde se formam as soluções a serem refinadas. Foram utilizadas duas estratégias gulosas: a inserção paralela e a inserção alternada e duas estratégias baseada em *First Fit Decreasing*. Em cada iteração do algoritmo é sorteada qual das construções será utilizada.

3.2.1 Inserção Paralela

O procedimento começa com o sorteio dos depósitos de onde cada rota partirá e, após realizar todas as entregas, retornará. Este sorteio depende de dois dados: a estimativa de depósitos e a capacidade dos veículos. Baseando-se na estimativa de depósitos, é sorteada uma quantidade, dentre aqueles disponíveis. Em seguida tais depósitos são distribuídos entre as rotas, onde não pode ser designado para um número maior que teto entre a divisão da capacidade do depósito e a capacidade dos veículos. Um cliente é sorteado para cada rota para garantir a geração de diferentes soluções iniciais.

Definido os depósitos para cada rota, verifica-se entre cada cliente ainda não inserido nas rotas qual a melhor rota e posição para inseri-lo usando o critério custo. Definido o mais barato, insere-se o cliente na rota e o procedimento é novamente realizado para os clientes restantes. Se sobrar clientes que não puderam ser inseridos devidos a restrições de capacidade, um depósito é sorteado e uma nova rota é criada. O pseudocódigo do procedimento é apresentado no Algoritmo 2.

Algoritmo 2. Insercao_Paralela(instância, s)

```
1.  Sorteia_Depositos(s)
2.  Sorteia_Clientes(s)
3.  Enquanto Lista_Clientes ≠ 0 faça
4.      Para cada cliente faça
5.          Para cada rota faça
6.              Para cada posição na rota faça
7.                  custo ← Calcula_Insercao(rota, posição, cliente)
8.                  Guarda_Melhor_Insercao(custo, rota, posição, cliente)
9.              Insere_cliente(custo, rota, posição, s, cliente)
10.     Se não consegue inserir todos os clientes faça
11.         Adiciona_Rota(s)
12. Retorna(s)
```

3.2.2 Inserção Alternada

O procedimento de construção de inserção alternada possui semelhanças com a inserção paralela. A diferença se apresenta durante a fase de inserção dos clientes nas rotas. Esta fase se difere da construção paralela pelo fato de necessariamente inserir um cliente para cada rota de forma alternada, ou seja, priorizando o balanceamento das rotas quanto ao número de clientes.
O pseudocódigo do procedimento é apresentado no Algoritmo 3.

Algoritmo 3. Insercao_Alternada(Instância, s)

```
1.  Sorteia_Depositos(s)
2.  Sorteia_Clientes(s)
3.  Enquanto Lista_Clientes ≠ 0 faça
```

```
4.      Para cada rota faça
5.          Para cada cliente faça
6.              Para cada posição na rota faça
7.                  custo ← Calcula_Insercao(Instância, rota, posição, cliente)
8.                  Guarda_Melhor_Insercao(Instância, custo, rota, posição, cliente)
9.          Insere_cliente(custo, rota, posição, s, cliente)
10.     Se não consegue inserir todos os clientes faça
11.         Adiciona_Rota(s)
12. Retorna(s)
```

3.2.2 First Fit Decreasing

Os procedimentos de construção baseados em *First Fit Decreasing* são utilizados para construir soluções com o menor número de depósitos possíveis para instâncias com carga apertada. Para determinar quando esta construção será utilizada, um cálculo sobre a folga da carga é utilizada. Esse cálculo é feito utilizando o somatório da demanda dos clientes dividido pela multiplicação da capacidade do depósito de maior capacidade pelo teto da divisão da demanda total pela capacidade do depósito de maior capacidade (observar expressão 2). Esta equação retorna valores entre 0 e 1.

$$FOLGA = \frac{\sum q_i}{w_x * teto(\sum q_i \div w_x)}$$ onde w_x é a capacidade do depósito de maior capacidade. (2)

Quando o valor retornado por este cálculo for maior ou igual a 0,85, utilizamos as construções baseadas em *First Fit Decreasing*. Foram implementadas duas versões. Na primeira o critério de escolha dos depósitos a serem sorteados entre as rotas são determinados por suas capacidades. Quanto maior, mais chances de utilizarmos nas rotas. Escolhido os depósitos, tenta-se inserir os clientes de maior demanda na primeira rota, depois na segunda e assim por diante.
Na segunda versão, muito semelhante à primeira, muda-se o critério de escolha dos depósitos. Os depósitos escolhidos são os de menor custo e o critério de inserção também é a carga. Assim, o procedimento é parecido com a construção paralela. O pseudocódigo do procedimento é apresentado no Algoritmo 4.

Algoritmo 4. Insercao_FFD(instância, s)

```
1.  Sorteia_Depositos(s)
2.  Sorteia_Clientes(s)
3.  Enquanto Lista_Clientes ≠ 0 faça
4.      Para cada cliente faça
5.          Para cada rota faça
6.              Para cada posição na rota faça
7.                  carga ← Calcula_Insercao(rota, posição, cliente)
8.                  Guarda_Melhor_Insercao(carga, rota, posição, cliente)
9.              Insere_cliente(custo, rota, posição, s, cliente)
10.     Se não consegue inserir todos os clientes faça
11.         Adiciona_Rota(s)
12. Retorna(s)
```

3.3 RVND

A diferença entre o RVND para o VND definido por Mladenović e Hansen (1997) e o implementado neste algoritmo é que a escolha das vizinhanças não é determinística. Utiliza-se uma semente aleatória para determinar a ordem em que as vizinhanças serão executadas. Caso a vizinhança sorteado seja bem sucedido, buscas intra-rota são realizadas nas rotas modificadas, caso contrário, a vizinhança é removida da lista da busca entre rotas distintas. O pseudocódigo do procedimento é apresentado no Algoritmo 5.

Algoritmo 5. RVND(Instância, s)

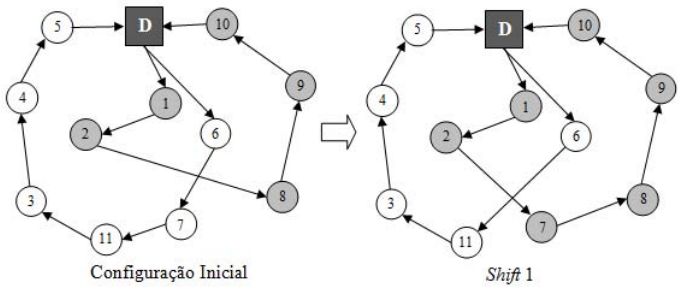
```
1.  Lista_Vizinhança[];
```

2. **Enquanto** Lista_Vizinhanca não estiver vazia **faça**
3. Sorteia_Vizinhança(Lista_Vizinhança)
4. **Se** vizinhança não for bem sucedida **faça**
5. Apague_Vizinhanca(Lista_Vizinhança)
6. **Senão** BuscaIntraRota**(**Instância, s**)**
7. **Retorna**(s)

Estão presentes no algoritmo duas vizinhanças do tipo *shift*:

- *Shift* 1: consiste em remover um cliente de uma rota e inserir em outra. Na Figura 5.a pode-se observar que o cliente "7" foi inserido na outra rota entre os clientes "2" e "8";
- *Shift* 2: consiste em remover dois clientes de uma rota e inserir em outra. Na Figura 5.b, os clientes "7" e "11" foram inseridos na outra rota entre os clientes "2" e "8".

(a)



(b)

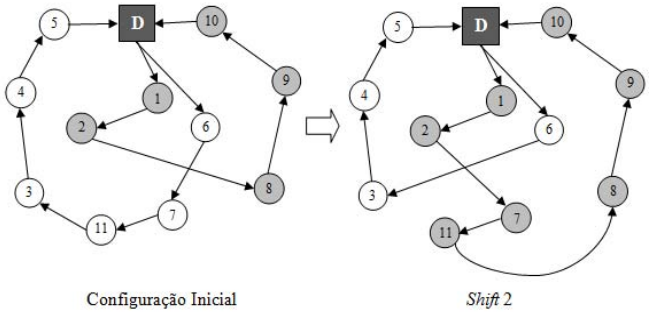
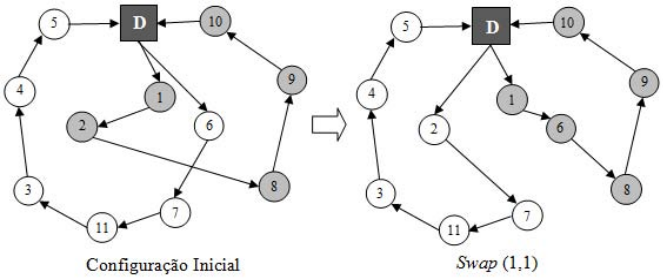


Figura 5: Vizinhança *Shift* 1 e *Shift* 2

Existem três vizinhanças do tipo *swap*:

- *Swap* (1,1): consiste em trocar um cliente de uma rota com um cliente de outra. Na Figura 6.a, o cliente "2" troca de lugar com o cliente "6";
- *Swap* (2,1): consiste em trocar dois clientes consecutivos de uma rota com um cliente de outra Na Figura 6.b os clientes "6" e "7" são trocados pelo cliente "2";
- *Swap* (2,2): consiste em trocar dois clientes consecutivos de uma rota com dois clientes de outra. Na Figura 6.c os clientes "1" e "2" são trocados pelos clientes "6" e "7".

(a)



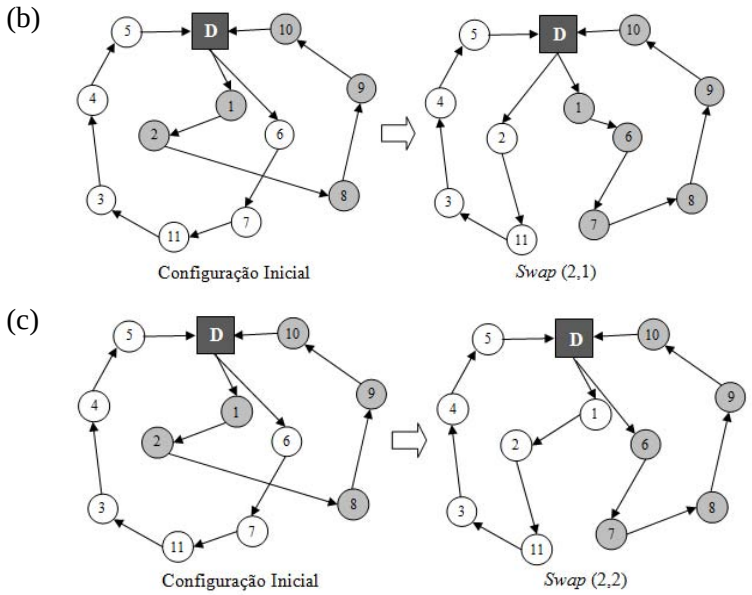


Figura 6: Vizinhança *Swap* (1,1), *Swap* (2,1) e *Swap* (2,2)

Além de um movimento de *cross*:

- *Cross*: O arco entre os clientes adjacentes c_1 e c_2, pertencentes a uma rota r_1, e o arco entre clientes c_3 e c_4 de uma rota r_2 são removidos. Em seguida, um arco é inserido entre c_1 e c_4 e outro entre c_3 e c_2. Caso as rotas possuam depósitos diferentes, os arcos que retornam ao depósito são corrigidos. Na Figura 7 as arestas entre os clientes "2" e "8" e entre os clientes "7" e "11" são removidas e as arestas entre os clientes "2" e "11" e entre os clientes "7" e "8" são inseridos.

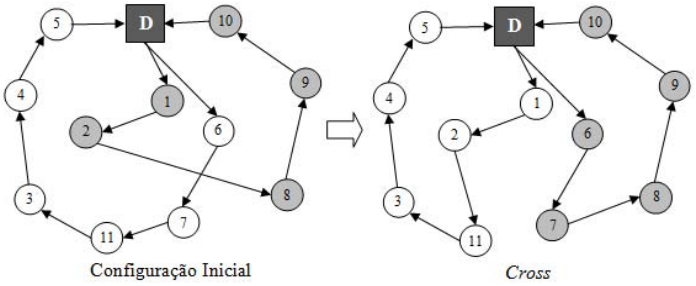


Figura 7: Vizinhança *cross*

As buscas Intra-Rotas são realizadas nas rotas que sofreram modificações pelas vizinhanças entre rotas. Os métodos utilizados são:

- *Or-Opt 1,2 e 3:* Proposto por Or (1976), onde um, dois ou três clientes consecutivos são removidos e posteriormente inseridos em outra posição;
- *2-opt:* Um par de arcos é removido e outro par é inserido*;*
- *Permutação*: permutação entre dois clientes.

3.4 Perturbações

Como já foi apresentada anteriormente, a meta-heurística ILS aplica repetidamente uma busca local às soluções iniciais obtidas através de perturbações das soluções ótimas locais previamente visitadas. No algoritmo proposto, as perturbações são movimentos de esforço computacional pequeno, todos baseadas em movimentos de busca-local. O critério de aceitação

do movimento de perturbação é que ele não pode gerar uma solução inviável. Como conseqüência, podem-se gerar soluções com custos piores que os das soluções já encontradas, porém permite que o algoritmo escape de mínimos locais. Foram quatro movimentos utilizados, sendo eles:

- *MultiShift:* Sorteia-se um cliente pertencente a qualquer rota e insere-o em uma outra qualquer. Se difere do movimento *Shift 1* por aceitar soluções piores após sua execução;
- *MultiSwap:* Sorteia-se dois clientes de rotas distintas e realiza-se a permuta. Se difere do movimento *Swap (1,1)* por aceitar soluções piores após sua execução;
- *DepotSwap:* Sorteia-se uma rota e troca-se o depósito; Na Figura 8.a pode-se observar a troca do depósito de uma das rotas.
- *AllDepotSwap:* Sorteia-se as rotas que saem de um mesmo depósito e troca-se seu depósito de partida. Na Figura 8.b pode-se observar que todas as rotas do depósito "D_0" são transferidas para o depósito "D_1".

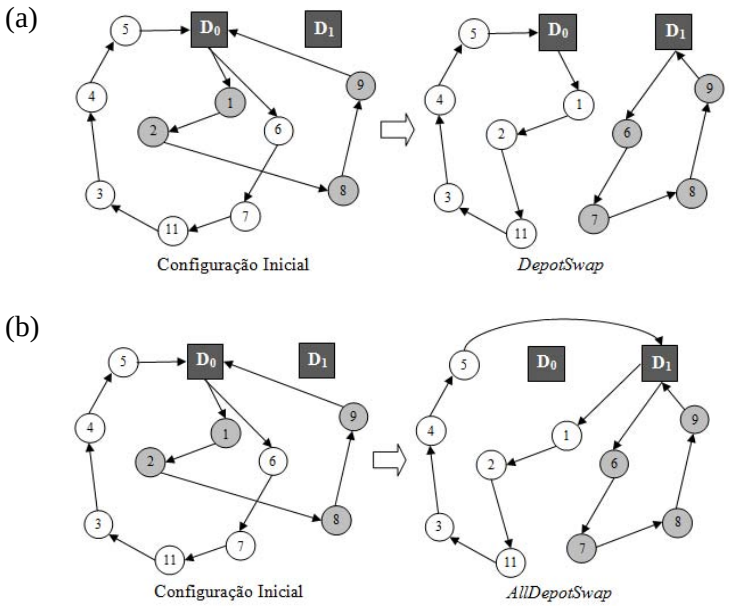


Figura 8: Perturbações entre Depósitos

As perturbações não possuem a mesma probabilidade de ocorrer. As perturbações entre depósitos possuem 50% menos chance de ocorrer que as entre clientes (procedimento Perturba1 – linha 14 do Algoritmo 1). Durante o refinamento final, realiza-se apenas os movimentos *Multishift* e *MultiSwap* com a mesma probabilidade de ocorrência (procedimento Perturba2 – linha 26 do Algoritmo 1).

4. Resultados

Como já foi citado anteriormente, os experimentos foram realizados em dois bancos de testes. Um deles é composto por 19 instâncias, algumas obtidas da literatura (Or, 1976, Perl, 1983) e outras adaptadas de instâncias relacionadas com o PRV (Gaskell, 1967, Christofides e Eilon, 1969, Min et AL., 1992, Daskin, 1995). O outro banco de teste, proposto por Prins et al. (2004), contém 30 instâncias e o número de clientes variam entre 20 e 200 e o número de depósitos de 5 a 10, além disso, possui custo para cada veículo utilizado. Os resultados são comparados com os resultados disponíveis na literatura (Prins, 2007; Barreto, 2007; Akca, 2008; Duhamel, 2008; Sahraeian, 2009; Contardo, 2010; Baldacci, 2011; Belenguer, 2011; Hemmelmayer, 2011). Para avaliar o algoritmo, duas comparações foram utilizadas:

- Comparam-se as melhores soluções encontradas pelo algoritmo e a média dos resultados em 10 execuções com as melhores soluções encontradas na literatura;
- Comparam-se as melhores soluções encontradas pelo algoritmo e a média dos resultados em 10 execuções com as melhores soluções da heurística ALNS proposta por Hemmelmayer et al. (2011), com a finalidade de comparar uma relação entre tempo e qualidade da solução.

O algoritmo proposto foi implementado na linguagem de programação C utilizando o compilador GCC e foi executado em um Intel Core I7 com frequência de 2.96 GHz com 8GB de memória RAM e sistema operacional Ubuntu 64 bits Professional Edition. A parametrização do campo "MAX_ITER" é 16 vezes o número de depósitos disponíveis. Para "MAX_ILS1" é duas vezes o número de clientes. Por "MAX_ILS2" possuir perturbações mais leves e por ser executado poucas vezes foi parametrizado com 32 vezes o número de clientes vezes o número de depósitos. Vários testes foram realizados para parametrização destes valores levando em conta o tempo de execução em relação as outras heurísticas. A expressão 2 apresenta como o *gap* utilizado nas tabelas foi calculado. $Campo_1$ e $campo_2$ são os valores a serem comparados. Exemplo: "gap/MRL" nas Tabelas 1 e 2 é o *gap* entre "MRL" ($campo_1$) com "Melhor Solução do ILS-RVND"($campo_2$)

$$gap = 100 - \frac{(100 / campo_1)}{campo_2} \quad (3)$$

Na Tabela 1, o campo "Instância" contém o nome da instância com o nome do Autor que propôs, ano publicado, número de clientes e número de depósitos. Nas Tabelas 2 e 3, o campo "Instância" contém o nome da instância com o número de clientes, número depósitos e versão. A versão "20-5-1a" se difere da "20-5-1b" pela capacidade dos veículos, capacidades dos depósitos e custo dos depósitos. Nas tabelas 1 e 2, o campo "MRL" representa os melhores resultados da literatura. O campo "Melhor Solução" é a melhor solução obtida pelo ILS-RVND em 10 execuções. "Média" é o custo médio das 10 execuções. Apresentam-se dois *gaps*: "gap/MRL" é o *gap* entre o campo "MRL" e o campo "Melhor Solução"; gap/Média é o *gap* entre o "MRL" e a "Média" do ILS-RVND. O campo "Tempo Médio" é o tempo médio das 10 execuções do algoritmo e estão expressos em segundos. Para facilitar a visualização, os resultados ótimos conhecidos são seguidos de asterisco, os resultados em negrito significam que empatam com os valores comparados e os resultados sublinhados são melhores que os resultados aos quais são comparados.

Tabela 1: Comparação das melhores soluções conhecidas com o ILS-RVND – Banco de Teste 1

			ILS-RVND				
Instância	Capacidade	MRL	Melhor Solução	Média	gap/MRL	gap/Média	Tempo Médio(s)
Christofides69-50x5	160	**565,60***	**565,62**	**565,62**	0,00	0,00	4,97
Christofides69-75x10	140	**802,08**	845,93	848,96	5,18	5,52	48,20
Christofides69-100x10	200	**833,40***	**833,43**	834,20	0,00	0,10	242,35
Daskin95-88x8	9000000	**355,80***	**355,78**	**355,78**	0,00	0,00	111,82
Daskin95-150x10	8000000	43919,90	43923,45	43975,78	0,01	0,13	557,62
Gaskell67-21x5	6000	**424,90***	**424,90**	**424,90**	0,00	0,00	0,32
Gaskell67-22x5	4500	**585,10***	**585,11**	**585,11**	0,00	0,00	0,52
Gaskell67-29x5	4500	**512,10***	**512,10**	**512,10**	0,00	0,00	1,33
Gaskell67-32x5A	8000	**562,22***	**562,22**	**562,22**	0,00	0,00	1,53
Gaskell67-32x5B	11000	**504,30***	**504,33**	**504,33**	0,01	0,01	1,96
Gaskell67-36x5	250	**460,40***	**460,37**	**460,37**	-0,01	-0,01	2,21
Min92-27x5	2500	**3062,00***	**3062,02**	**3062,02**	0,00	0,00	0,88
Min92-134x8	850	5709,00	5719,25	5727,33	0,18	0,32	286,03
Perl83-12x2	140	**204,00***	**203,98**	**203,98**	-0,01	-0,01	0,01
Perl83-55x15	120	1112,10	**1112,06**	**1112,13**	0,00	0,00	42,03
Perl83-85x7	160	1622,25	1622,50	1623,40	0,02	0,07	39,95
Perl83-318x4	25000	580680,20	559808,84	565228,68	-3,73	-2,73	2448,41
Perl83-318x4	8000	747619,00	674392,12	678045,57	-10,86	-10,26	739,03
Or76-117x14	150	12474,20	12296,93	12302,51	-1,44	-1,40	777,15

A tabela acima permite comparar os melhores resultados da literatura e o algoritmo ILS-RVND no primeiro banco de teste, utilizando o "gap/MRL" como parâmetro, de 19 instâncias, e considerando o valor ±0,01 como empate, evidenciando que este último apresentou três novas melhores soluções e empatou em 13 resultados.

Tabela 2: Comparação das melhores soluções conhecidas com o ILS-RVND – Banco de Teste 2

		ILS-RVND				
Instância	MRL	Melhor Solução	Média	gap/MRL	gap/MLI	Tempo Médio(s)
20-5-1ª	**54793***	**54793**	**54793**	0,00	0,00	0,25
20-5-1b	**39104***	**39104**	**39104**	0,00	0,00	0,38
20-5-2ª	**48908***	**48908**	**48908**	0,00	0,00	0,24
20-5-2b	**37542***	**37542**	**37542**	0,00	0,00	0,35
50-5-1ª	**90111***	**90111**	**90111**	0,00	0,00	3,26
50-5-1b	**63242***	**63242**	63247,6	0,00	0,01	5,10
50-5-2ª	**88298***	**88298**	**88298**	0,00	0,00	3,73
50-5-2b	**67308***	**67308**	67321	0,00	0,02	6,03
50-5-2BIS	**84055***	**84055**	**84055**	0,00	0,00	5,60
50-5-2bBIS	**51822***	**51822**	51870,8	0,00	0,09	3,25
50-5-3ª	**86203***	**86203**	**86203**	0,00	0,00	3,03
50-5-3b	**61830***	**61830**	**61830**	0,00	0,00	5,51
100-5-1ª	**274814***	**275281**	275538,5	0,17	0,26	24,49
100-5-1b	213615	213654	214142,6	0,02	0,25	41,35
100-5-2	**193671***	**193671**	193849,7	0,00	0,09	23,35
100-5-2b	**157095***	157129	157162,3	0,02	0,04	41,73
100-5-3ª	**200079***	200131	200271,7	0,03	0,10	22,17
100-5-3b	**152441***	**152441**	152573,6	0,00	0,09	32,66
100-10-1a	287983	303100	310848,2	4,99	7,36	45,66
100-10-1b	231763	237902	244982,1	2,58	5,40	65,95
100-10-2a	**243590***	**243590**	243758,6	0,00	0,07	46,40
100-10-2b	203988	**203988**	204068,3	0,00	0,04	67,35
100-10-3a	250882	252992	259787,8	0,83	3,43	46,41
100-10-3b	204317	204661	204865,1	0,17	0,27	73,56
200-10-1a	477248	476822	496095,3	-0,09	3,80	400,51
200-10-1b	378065	376982	378448,5	-0,29	0,10	723,89
200-10-2a	449571	448735	449094,2	-0,19	-0,11	498,12
200-10-2b	374330	373837	374396	-0,13	0,02	677,51
200-10-3a	**469433***	471147	484970,2	0,36	3,20	400,01
200-10-3b	362817	362664	363271,8	-0,04	0,13	669,42

Na Tabela 2, pode-se observar a comparação entre os melhores resultados da literatura e o algoritmo ILS-RVND para o segundo banco de teste, de 30 instâncias, quando se obteve cinco novas melhores soluções, empatando em 16 e perdendo em nove resultados. O campo utilizado como parâmetro foi o "gap/MRL", considerando o valor ±0,01 como empate.

Tabela 3: Comparação do algoritmo ALNS com o ILS-RVND

	ALNS			ILS-RVND				
Instância	Melhor Solução	Média	Tempo Médio(s)	Melhor Solução	Média	Tempo Médio(s)	gap/MS	gap/Média
20-5-1a	54793	54793,00	39,00	**54793**	**54793**	0,25	0,00	0,00
20-5-1b	39104	39104,00	54,00	**39104**	**39104**	0,38	0,00	0,00
20-5-2a	48908	48908,00	38,00	**48908**	**48908**	0,24	0,00	0,00
20-5-2b	37542	37542,00	67,00	**37542**	**37542**	0,35	0,00	0,00
50-5-1	90111	90111,00	101,00	**90111**	**90111**	3,26	0,00	0,00
50-5-1b	63242	63242,00	65,00	**63242**	63247,6	5,10	0,00	0,01
50-5-2	88443	88576,80	99,00	88298	88298	3,73	-0,16	-0,32
50-5-2b	67340	67448,20	200,00	67308	67321	6,03	-0,05	-0,19
50-5-2bis	84055	84119,00	107,00	**84055**	84055	5,60	0,00	-0,08
50-5-2bbis	51822	51840,00	98,00	**51822**	51870,8	3,25	0,00	0,06
50-5-3	86203	86261,60	101,00	**86203**	86203	3,03	0,00	-0,07
50-5-3b	61830	61830,00	137,00	**61830**	**61830**	5,51	0,00	0,00
100-5-1	275636	276364,00	520,00	275281	275538,5	24,49	-0,13	-0,30
100-5-1b	214735	215059,00	1190,00	213654	214142,6	41,35	-0,51	-0,43
100-5-2	193752	193903,00	463,00	193671	193849,7	23,35	-0,04	-0,03

100-5-2b	157095	157156,60	859,00	157129	157162,3	41,73	0,02	0,00
100-5-3	200305	200495,60	454,00	200131	200271,7	22,17	-0,09	-0,11
100-5-3b	152441	152899,80	684,00	**152441**	152573,6	32,66	0,00	-0,21
100-10-1	296877	299982,40	210,00	303100	310848,2	45,66	2,05	3,50
100-10-1b	235849	240289,20	188,00	237902	244982,1	65,95	0,86	1,92
100-10-2	244740	245548,20	136,00	243590	243758,6	46,40	-0,47	-0,73
100-10-2b	204016	204494,60	261,00	203988	204068,3	67,35	-0,01	-0,21
100-10-3	253801	254882,00	202,00	252992	259787,8	46,41	-0,32	1,89
100-10-3b	205609	206175,20	224,00	204661	204865,1	73,56	-0,46	-0,64
200-10-1	480883	483204,80	752,00	476822	496095,3	400,51	-0,85	2,60
200-10-1b	378961	380538,80	1346,00	376982	378448,5	723,89	-0,52	-0,55
200-10-2	450451	451750,60	1201,00	448735	449094,2	498,12	-0,38	-0,59
200-10-2b	374751	376111,80	1349,00	373837	374396	677,51	-0,24	-0,46
200-10-3	475373	479366,60	1251,00	471147	484970,2	400,01	-0,90	1,16
200-10-3b	366902	369614,00	1137,00	362664	363271,8	669,42	-1,17	-1,75

Ao se comparar ainda os resultados obtidos pelos algoritmos ALNS (Hemmelmayer, 2011) e ILS-RVND (Tabela 3) ressaltou-se que, neste último algoritmo, 15 instâncias apresentaram melhores soluções, empataram em 12 e perderam em duas, quando utilizado o "gap/MS" (*gap* entre as melhores soluções obtidas).

5. Conclusões

O algoritmo ILS-RVND aqui proposto demonstrou ser bastante competitivo nos resultados empíricos obtidos em cima do banco de testes disponível em Barreto (2003) e Prins et al. (2004), além de apresentar oito novas soluções não conhecidas pela literatura. As modificações em relação a sua versão anterior (Carvalho, 2011) levaram a estes melhores resultados, sendo as principais modificações responsáveis: a implementação de estruturas que impeçam que algumas buscas sejam realizadas repetidas vezes desnecessariamente; duas novas construções que vieram suprir a necessidade de criar soluções com cargas apertadas; o ajuste da probabilidade das perturbações; e por último, um segundo refinamento apenas entre clientes. O algoritmo continua sobre estudo tendo em vista a criação de versões paralelas e híbridas agora conjugando metaheuristicas com métodos exatos de programação matemática (Penna *et. al* 2012a).

REFERÊNCIAS

Akca, Z., Ralphs, T. K., Berger, R. T., (2008) *A Branch-and-Price Algorithm for Combined Location and Routing Problems Under Capacity Restrictions - Working paper,* COR@L Lab, Lehigh University.

Baldacci, R., Mingozzi, A., Calvo, R. W. (2011) *An Exact Method for the Capacitated Location-Routing Problem* – Operations Search.

Barreto, S. S. (2004) *Análise e Modelização de Problema de Localização-Distribuição – Tese (Doutorado),* Universidade de Aveiro, Portugal.

Barreto, S. S. (2007) *Using clustering analysis in a capacitated location-routing problem – European Journal of Operational Researh 179,* p. 968-977.

Belenguer, J., Benavent, E., Prins, C., Prodhon, C., Calvo, R. W. (2011) *A Branch-and-Cut method for the Capacitated Location-Routing Problem – Computer & Operations Research 38,* p. 931-941.

Christofides, N., (1976) *The vehicle routing problem. R.A.I.R.O. Recherche Opérationnelle 10, 2 p. 55-70.*

Christofides, N., Eilson, S., (1969) *An algorithm for the vehicle dispatching problem - Operational Research Quarterly 20 (3),* p. 309-318.

Carvalho, G. R., Subramanian, A., Ochi, L. S., Cabral, L. A. F. (2011) *Uma Heurísitca Eficiente Baseada em Busca Local Iterada para o Problema de Localização-Roteamento - XXV Congresso de Pesquisa e Ensino em Transportes,* Belo Horizonte/MG.

Contardo, C., Cordeau, J., Gengron, B. (2010) *A Branch-and-Cut Algorithm for the*

Capacitated Location-Routing Problem – Proceedings of the VI ALIO/EURO Workshop on Applied Combinatorial Optimization. Buenos Aires, Argentina.

Daskin, M. S., (1995) *Network and Discrete Location: Models, Algorithms and Applications - John Wiley & Sons*, Inc, New York.

Duhamel C., Lacomme, P., Prins, C., Prodhon, C. (2008) – *A Memetic Approach for the Capacited Location Routing Problem.International Workshopon Metaheuristics for Logistics and Vehcile Routing, Troyes, France.*

Gaskell, T. J., (1967) *Bases for vehicle fleet scheduling - Operational Researh Quarterly 18 (3)*, p. 281-295.

Hemmelmayer, V. C., Cordeau, J., Crainic, T. G., (2011) – *An Adaptative Large Neighborhood Search Heuristic for Two-Echelon Vehicle Routing Problems Arising in City Logistics- technical Report CIRRELT-2011-42*, Université de Montréal, 2011.

Laporte, G. (1988) *Location-routing problems – Vehicle Routing: Methods and Studies*, B.L. Golden e A. A. Assad, Eds, Elsevier Sciences Publishers B. V. P. 163-197, Holanda.

Lourenço, H. R., Martin, O. C., Stutzle, T. (2002) *Iterated Local Search - Fred Glover e Gary A. Kochenberger (eds.)", Handbook of Metaheuristics*, p 321-353. Kluwer Academic Publishers, Norwell, MA.

Min, H., Current, J., Schilling D., (1992) *The multiple depot vehicle routing problem with backhauling - Journal of Business Logistics 13 (1)*, p. 259-288.

Mine, M., Silva, M. S. A., Subramanian, A., Ochi, L. S., Souza, M. J. F. (2011)**.** A hybrid heuristic based on iterated local search and Genius, for the vehicle routing problem with simultaneous pickup and delivery. International Journal of Logistics Systems Management (IJLSM) – Vol. 10(2), pp. 142 - 157.

Mladenovic, N., Hansen, P. (1997) *Variable Neighbourhood Search - Computers & Operations Research*, v. 24, n. 11, p. 1097-1100.

Or, I. (1976) *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking – Tese (Doutorado)*, Northwestern, University, EUA.

Penna, P. H. V., Subramanian, A., Ochi, L. S., Uchoa, E. (2012a) A Hybrid Algorithm for the Heterogenous Fleet Vehicle Routing Problem. European Journal of Operational Research - EJOR – ELSEVIER, Volume 221, pp: 285-295, (2012).

Penna, P. H. V., Subramanian, A., Ochi, L. S., (2012b**).** An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem. To appear in Journal of Heuristics – 2011. (aceito em 07/2011).

PERL, J., (1983) *A unified warehouse location-routing analysis – Tese (Doutorado)*, Northwestern University, Evanston, Illinois, Estados Unidos da América.

Prins, C., Prodhon, C., e Woler Calvo, R. (2004) *Nouveaux algorithmes pour le problème de localisation et routage sous contraintes de capacit_e. In Dolgui, A. And Dauzère-Pérès, S., editors, MOSIM, volume 4, pages 1115-1122. Ecole des Mines de Nantes, Lavoisier.*

Prins, C., Prodhon, C., Ruiz, A., Siriano, P., Calvo, R. W. (2007) *Solving the Capacited Location-Routing Problem by a Cooperative Lagrangean Relaxation-Granular Tabu Search Heuristic – Transportation Science*, 41, p. 470-483.

Sahraeian, R., e Nadizadeh, A., (2009) *Using greedy clustering method to solve capacitated location-routing problem – 3nd International Conference on Industrial Engineering and Industrial Management.* XIII Congreso de Ingeniena de Organizacion. p. 79-85, Barcelona-Terrassa.

Stützel, T. G. (1998) *Local Search Algorithms for Combinatorial Problems – Tese (Doutorado)*, Technische Universit¨at Darmstadt, Alemanha.