# UMA HEURÍSTICA EFICIENTE BASEADA EM BUSCA LOCAL ITERADA PARA O PROBLEMA DE LOCALIZAÇÃO-ROTEAMENTO

**Gustavo Rezende Carvalho**
**Anand Subramanian**
**Luiz Satoru Ochi**
Instituto de Computação - Universidade Federal Fluminense (UFF)
Instituto de Computação

**Lucídio dos Anjos Formiga Cabral**
Departamento de Informática - Universidade Federal da Paraíba (UFPB)
Departamento de Informática

**RESUMO**
Este trabalho propõe uma heurística híbrida utilizando conceitos da meta-heurística ILS (Busca Local Iterada - *Iterated Local Search*) com busca local VND (Metodo de Descida em Vizinhança Variável - *Variable Neighborhood Descent*) para a solução do Problema de Localização-Roteamento (PLR) de dois níveis com capacidade. O PLR é a combinação de dois problemas de elevada complexidade computacional (NP-Difícil): o Problema de Localização de Facilidades (Facility Location Problem – FLP) com o Problema de Roteamento de Veículos (Vehicle Routing Problem - VRP) o que, por conseqüência o classifica também como da classe NP-Difícil. O algoritmo foi testado em cima de instâncias presentes na literatura obtendo resultados competitivos.

**ABSTRACT**
This paper proposes a hybrid heuristics using ILS (Iterated Local Search) metaheuristics concepts with VND (Variable Neighborhood Descent) local search to solve the capacity two levels LRP (Location-Routing Problem). The LRP can be defined as combination of two hard problems with high computational complexity (NP-Hard): FLP (Facility Location Problem) with the VRP (Vehicle Routing Problem); consequently, LRP is NP-Hard. The algorithm was tested with literature benchmarks obtaining competitive results.

## 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.

Pelo fato do PLRC ser definido muitas vezes como a combinação do problema de localização de facilidades com o problema de roteamento de veículos (Srivastava, 1990), não é aconselhável definir a localização dos depósitos sem levar em consideração a otimização das rotas associadas. A simples escolha de uma boa localização dos depósitos não acarreta, necessariamente, em uma diminuição dos custos globais envolvidos. Podemos citar vários casos onde uma escolha aparentemente boa de localização não acarreta em uma boa solução:

escolher um depósito de custo baixo, porém mal localizado ou utilizar um número de depósitos inferior ou superior ao de agrupamentos ótimos poderá acarretar em um alto custo para as rotas. Vários autores defendem as premissas (Webb, 1968; Sussams,1969; Marks e Stricter,1971; Eilon, 1971; Rand, 1976; Geoffrion e Powers ,1980; Perl e Daskin,1985; Laporte, 1988; Wu et al.,2002) e, em destaque o trabalho de Salhi e Rand (1989) sobre o impacto de ignorar as rotas durante a localização dos depósitos.

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 *branch-and-price* (Akca, 2008) e o *branch-and-cut* (Contardo, 2010; Belenguer, 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) e os meméticos (Duhamel, 2008). Outra alternativa abordada pela literatura são os métodos híbridos, como por exemplo o LRGTS (*Lagrangean Relaxation-Granular Tabu Search*) proposto por Prins (2007).

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 completo 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 $\{i, j\} \in E$ tem 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 $Q$. 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 e roteamento envolvidos.

## 3. ALGORITMO PROPOSTO

O algoritmo é baseado na meta-heurística *Iterated Local Search* (ILS), que a partir de uma solução ótima local encontrada por um algoritmo, ao invés de reiniciar o procedimento e encontrar uma solução completamente nova, aplica repetidamente uma busca local às soluções iniciais obtidas através de perturbações das soluções ótimas (Lourenço, 2002).

No ILS proposto, a fase de busca local é efetuada por um 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. As estruturas de vizinhança utilizadas são: *shift, swap*

(1,1), *swap* (2,1), *swap* (2,2), e uma adaptação do movimento *cross*. Além disso, o ILS é *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 *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. O pseudocódigo do procedimento é indicado na Figura 1.

**Algoritmo 1.** ILS-RVND(Instância)

```
1.  Estima_Frota(Instância)
2.  Estima_Depositos(Instância)
3.  s0, s1, s2 ← 0
4.  Para  i ← 0 até i < MAX_ITER faça
5.      Construção(Instancia, s0)
6.      RVND(Instancia, s0)
7.      s1 ← s0
8.      Se s0 < s2 faça
9.          s2 ← s0
10.     Para i ← 0 até i < MAX_ILS faça
11.         Pertuba(Instancia, s0)
12.         RVND(Instancia, s0)
13.         Se s0 < s1 faça
14.             s1 ← s0
15.             Se s0 < s2 faça
16.                 s2 ← s0
17.             i ← 0
18.         Senão faça
19.             s0 ← s1
20. Retorna(s2)
```

**Figura 1:** Algoritmo ILS-RVND

### 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. Em cada iteração do algoritmo é sorteada qual das duas 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. Caso o problema tenha como estimativa que um depósito tem capacidade para suprir toda a demanda, 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 o menor custo para inseri-lo. 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 na Figura 2.

**Algoritmo 2.** Insercao_Paralela(instância, s)

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

**Figura 2:** Algoritmo do procedimento Inserção Paralela

*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. Verifica-se para a primeira rota qual é a inserção de cliente mais barata e se insere o cliente se possível. Depois se repete o procedimento para a segunda rota e assim por diante até que todos os clientes estejam inseridos. Caso não seja possível inserir todos os clientes devido às restrições de capacidade, uma nova rota é inserida assim como está descrito na inserção paralela. O pseudocódigo do procedimento é apresentado na Figura 3.

**Algoritmo 3.** Insercao_Alternada(Instância, s)

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

**Figura 3:** Algoritmo do procedimento Inserção Alternada

### 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. O pseudocódigo do procedimento é apresentado na Figura 4.

**Algoritmo 4.** RVND(Instância, s)

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

**Figura 4:** Algoritmo do procedimento RVND

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 após os clientes "2" e "8";
- *Shift* 2: consiste em remover dois clientes de uma rota e inserir em outra. Na Figura 5.b, os cliente "7" e "11" foram inseridos na outra rota entre os clientes "2" e "8".

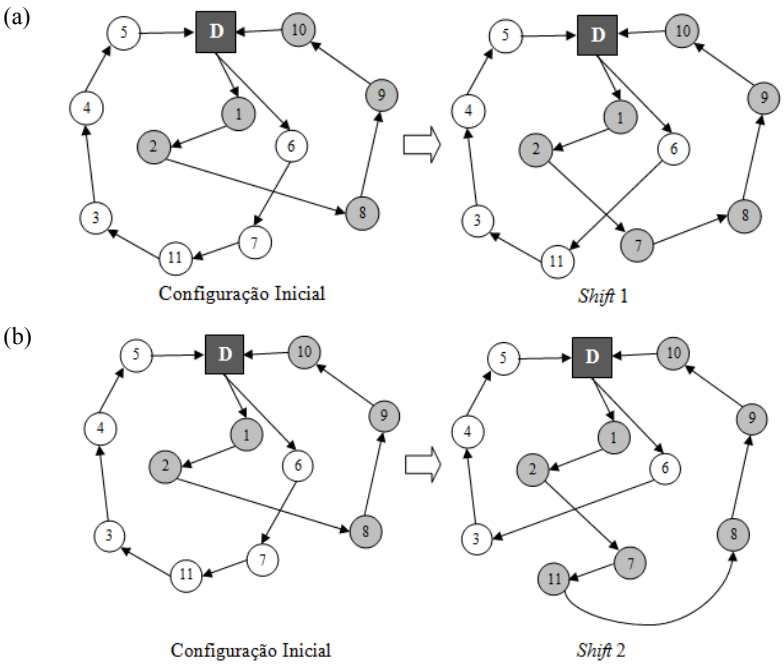


**Figura 5:** Vizinhança *Shift* 1 e *Shift* 2

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 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 de uma rota com dois clientes de outra. Na Figura 6.c os clientes "1" e "2" são trocados pelos clientes "6" e "7".

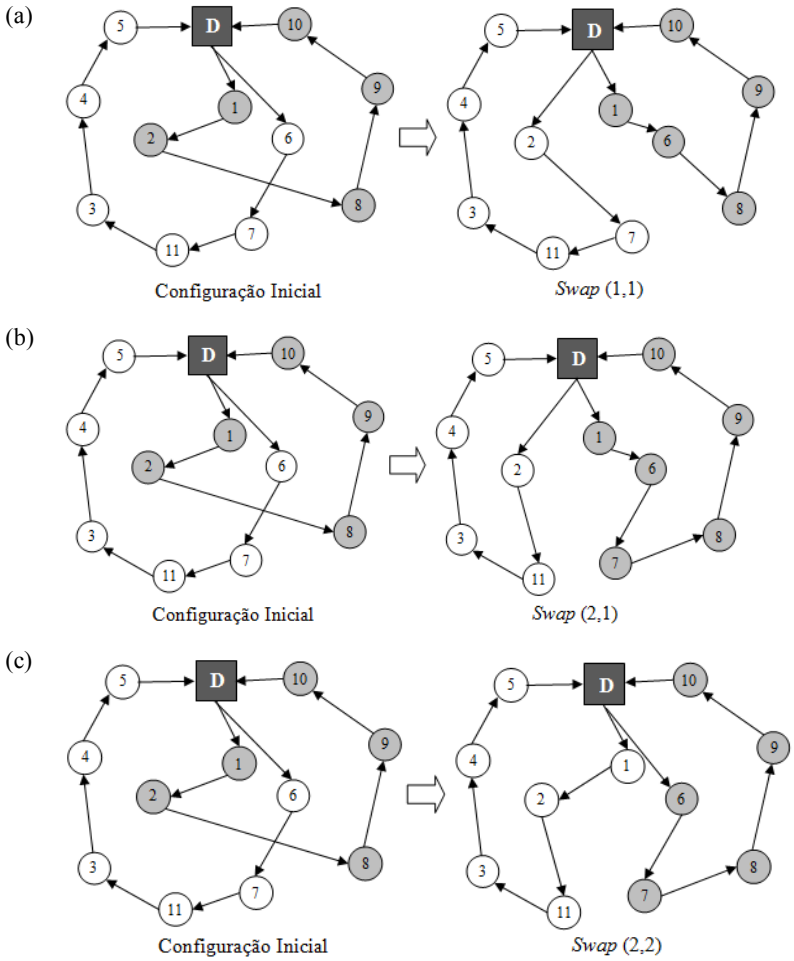


**Figura 6:** Vizinhança *Swap* (1,1), *Swap* (2,1) e *Swap* (2,2)

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.

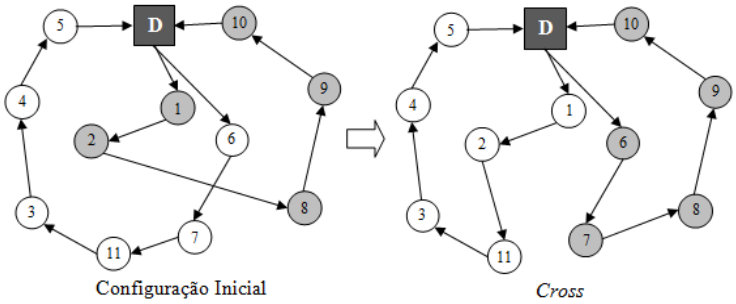


**Figura 7:** Vizinhança *cross*

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.

**Algoritmo 6.** BuscaIntraRota(Instância, s)

```
1.  Lista_Vizinhança_IR[ ]
2.
3.  Enquanto Lista_Vizinhanca_IR não estiver vazia faça
4.        Sorteia_Vizinhança(Lista_Vizinhança_IR)
5.        Se vizinhança não for bem sucedida faça
6.              Apague_Vizinhanca(chave)
7.  Retorna(s)
```

**Figura 7:** Algoritmo do procedimento Busca Intra Rotas

### 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, pode-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;
- *MultiSwap:* Sorteia-se dois clientes de rotas distintas e realiza-se a permuta;
- *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 "$D_1$".

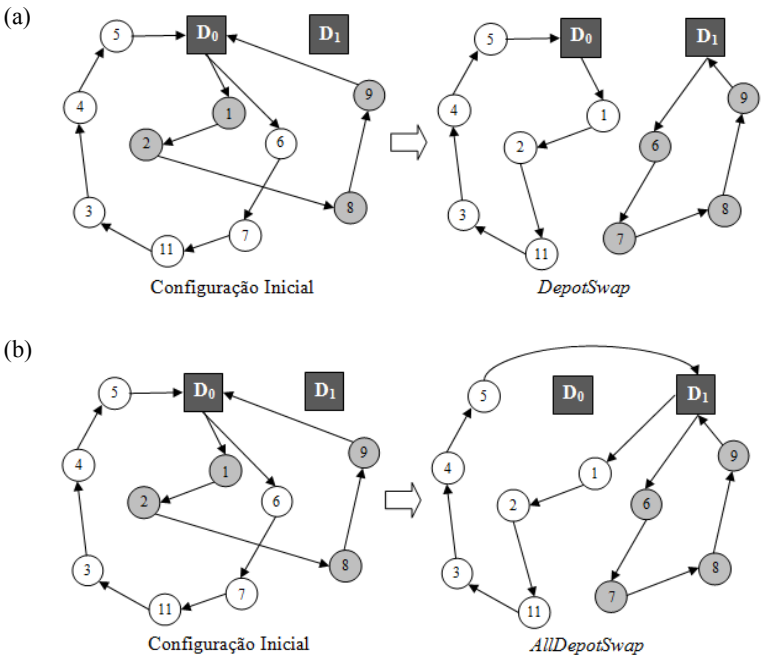


**Figura 8:** Perturbações entre Depósitos

## 4. RESULTADOS

Os experimentos foram realizados considerando 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). Foi então comparado com os resultados disponíveis na literatura (Prins, 2007; Barreto, 2007; Akca, 2008; Duhamel, 2008; Sahraeian, 2009; Contardo, 2010; Belenguer, 2011). As instâncias estão disponibilizadas em Barreto (2003).

Para avaliar o algoritmo, três comparações foram utilizadas:

- Comparam-se as melhores soluções encontradas pelo algoritmo e a média dos resultados em 20 execuções com as melhores soluções encontradas na literatura;
- Comparam-se as melhores soluções encontradas pelo algoritmo e a média dos resultados em 20 execuções com as melhores soluções das heurísticas propostas por Barreto et. al (2007), pois é o autor que apresenta as melhores soluções para as instâncias de Or (1976) e Perl (1983);
- Comparam-se as melhores soluções encontradas pelo algoritmo e a média dos resultados em 20 execuções com as melhores soluções da heurística *Lagrangean Relaxation-Granular Tabu Search* (LRGTS) proposta por Prins et. al (2007), 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 Athlom 3500+ 2.00 GHz com 2GB de memória RAM e sistema operacional Ubuntu 64 bits Professional Edition. A parametrização do campo "MAX_ITER" é duas vezes o número de depósitos disponíveis. Para "MAX_ILS" é duas vezes o número de depósitos disponíveis vezes o número de clientes. Vários testes foram realizados para parametrização destes valores levando em conta o tempo de execução em

relação a outras heurísticas. A expressão 2 apresenta como o *gap* utilizado nas tabelas foi calculado.

$$gap = 100 - \frac{(100 / campo_1)}{campo_2} \quad (2)$$

Na Tabela 1, pode-se observar a comparação entre os melhores resultados da literatura e o algoritmo ILS-RVND. O campo "Instância" contém o nome da instância com o nome do Autor que propôs, ano publicado, número de clientes e número de depósitos. O campo "MSC" representa os melhores resultados da literatura. O campo "Custo" é a melhor solução obtida pelo ILS-RVND em 20 execuções. "Média" é o custo médio das 20 execuções. Apresentam-se dois *gaps*: "gap/Custo" é o *gap* entre o campo "MSC" e o campo "Custo"; gap/Média é o *gap* entre o "MSC" e a "Média" do ILS-RVND. Utilizando o "gap/Média" como parâmetro, de 19 instâncias, e considerando o valor 0,01 empate, o ILS-RVND possui seis melhores soluções e empata em cinco. No parâmetro "gap/Custo", sete soluções foram melhoradas e empata em 10.

**Tabela 1**: Comparação da melhor solução conhecida com o ILS-RVND

| | | | ILS-RVND | | | | |
|---|---|---|---|---|---|---|---|
| Instância | Capacidade | MSC | Custo | Média | gap/MSC | gap/Média | Tempo |
| Christofides69-50x5 | 160 | 565,60* | 565,65 | 573,59 | 0,01 | 1,39 | 3,03 |
| Christofides69-75x10 | 140 | 861,60 | 869,11 | 882,37 | 0,86 | 2,35 | 48,22 |
| Christofides69-100x10 | 200 | 833,40 | 833,42 | 841,31 | 0,00 | 0,94 | 174,64 |
| Daskin95-88x8 | 9000000 | 355,80 | 355,78 | 356,61 | -0,01 | 0,23 | 71,32 |
| Daskin95-150x10 | 8000000 | 44011,70 | 44083,06 | 44558,62 | 0,16 | 1,23 | 641,98 |
| Gaskell67-21x5 | 6000 | 424,90* | 424,89 | 424,89 | 0,00 | 0,00 | 0,17 |
| Gaskell67-22x5 | 4500 | 585,10* | 585,1 | 585,10 | 0,00 | 0,00 | 0,20 |
| Gaskell67-29x5 | 4500 | 512,10* | 512,1 | 518,83 | 0,00 | 1,30 | 0,62 |
| Gaskell67-32x5A | 8000 | 571,70 | 562,22 | 562,22 | -1,69 | -1,69 | 0,75 |
| Gaskell67-32x5B | 11000 | 504,30* | 504,32 | 504,32 | 0,00 | 0,00 | 0,79 |
| Gaskell67-36x5 | 250 | 460,40* | 460,37 | 460,37 | -0,01 | -0,01 | 1,24 |
| Min92-27x5 | 2500 | 3062,00* | 3062,01 | 3078,83 | 0,00 | 0,55 | 0,49 |
| Min92-134x8 | 850 | 5809,00 | 5851,79 | 5941,66 | 0,73 | 2,23 | 317,12 |
| Perl83-12x2 | 140 | 204,00* | 203,97 | 203,97 | -0,01 | -0,01 | 0,01 |
| Perl83-55x15 | 120 | 1136,20 | 1112,79 | 1112,97 | -2,10 | -2,09 | 52,59 |
| Perl83-85x7 | 160 | 1656,90 | 1623,14 | 1626,24 | -2,08 | -1,89 | 46,48 |
| Perl83-318x4 | 25000 | 580680,20 | 569664,86 | 574548,34 | -1,93 | -1,07 | 2773,37 |
| Perl83-318x4 | 8000 | 747619,00 | 685806,43 | 703111,83 | -9,01 | -6,33 | 2361,30 |
| Or76-117x14 | 150 | 12474,20 | 12296,93 | 12319,94 | -1,44 | -1,25 | 544,63 |

Na Tabela 2, pode-se observar a comparação entre os resultados obtidos pelas heurísticas apresentadas por Barreto (2007) e o algoritmo ILS-RVND. O campo "Custo" representa os melhores resultados obtidos pelos algoritmos. O campo "D." representa o número de depósitos utilizados na melhor solução e "R.", o número de veículos. Apresentamos dois gaps: "gap/Custo" é o gap entre os melhores custos dos algoritmos; gap/Média é o gap entre o

"Custo" do LRGTS e a "Média" do ILS-RVND. Utilizando o gap/Média como parâmetro, o ILS-RVND possui melhores soluções em 17 e empata em uma. Os resultados se repetem para o parâmetro gap/Custo.

**Tabela 2:** Comparação das melhores soluções das heurísticas propostas por Barreto com o ILS-RVND

| | | Barreto | ILS-RVND | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instância | Capacidade | Custo | Custo | D. | R. | Média | gap/Custo | gap/Média |
| Christofides69-50x5 | 160 | 582,70 | 565,65 | 2 | 5 | 573,59 | -3,01 | -1,59 |
| Christofides69-75x10 | 140 | 886,30 | 869,11 | 3 | 11 | 882,37 | -1,98 | -0,45 |
| Christofides69-100x10 | 200 | 889,40 | 833,42 | 2 | 8 | 841,31 | -6,72 | -5,72 |
| Daskin95-88x8 | 9000000 | 384,90 | 355,78 | 2 | 6 | 356,61 | -8,18 | -7,93 |
| Daskin95-150x10 | 8000000 | 46642,70 | 44083,06 | 3 | 11 | 44558,62 | -5,81 | -4,68 |
| Gaskell67-21x5 | 6000 | 435,90 | 424,89 | 2 | 4 | 424,89 | -2,59 | -2,59 |
| Gaskell67-22x5 | 4500 | 591,50 | 585,1 | 1 | 3 | 585,10 | -1,09 | -1,09 |
| Gaskell67-29x5 | 4500 | 512,10 | 512,1 | 2 | 4 | 518,83 | 0,00 | 1,30 |
| Gaskell67-32x5 | 8000 | 571,70 | 562,22 | 1 | 4 | 562,22 | -1,69 | -1,69 |
| Gaskell67-32x5 | 11000 | 511,40 | 504,32 | 1 | 3 | 504,32 | -1,40 | -1,40 |
| Gaskell67-36x5 | 250 | 470,70 | 460,37 | 1 | 4 | 460,37 | -2,24 | -2,24 |
| Min92-27x5 | 2500 | 3062,00 | 3062,01 | 2 | 4 | 3078,83 | 0,00 | 0,55 |
| Min92-134x8 | 850 | 6238,00 | 5851,79 | 3 | 11 | 5941,66 | -6,60 | -4,99 |
| Perl83-12x2 | 140 | 204 | 203,97 | 1 | 2 | 203,97 | -0,01 | -0,01 |
| Perl83-55x15 | 120 | 1136,2 | 1112,79 | 3 | 10 | 1112,97 | -2,10 | -2,09 |
| Perl83-85x7 | 160 | 1656,9 | 1623,14 | 3 | 11 | 1626,24 | -2,08 | -1,89 |
| Perl83-318x4 | 25000 | 580680,2 | 569664,86 | 4 | 8 | 574548,34 | -1,93 | -1,07 |
| Perl83-318x4 | 8000 | 747619 | 685806,43 | 4 | 26 | 703111,83 | -9,01 | -6,33 |
| Or76-117x14 | 150 | 12474,2 | 12296,93 | 3 | 5 | 12319,94 | -1,44 | -1,25 |

Na Tabela 3, pode-se observar a comparação entre os resultados obtidos pelos algoritmos LRGTS (Prins, 2007) e ILS-RVND. O "Tempo" é o tempo de execução em segundos e o "Tempo M." é o tempo médio das 20 execuções do algoritmo. Além do "gap/Custo" e o "gap/Média", esta tabela também apresenta o *gap* entre o "Tempo" e o "Tempo M.", "gap/Tempo". Utilizando o gap/Média como parâmetro, de 13 instâncias, o ILS-RVND possui melhores soluções em cinco e empata em uma. No parâmetro "gap/Custo", apresenta melhores soluções em seis e empata em duas. Como os problemas foram rodados em processadores de gerações próximas, o algoritmo LRGTS é considerado, aparentemente, 45% mais rápido que o ILS-RVND.

## 5. CONCLUSÕES

O algoritmo ILS-RVND demonstra resultados competitivos em cima do banco de testes disponível em Barreto (2003), além de apresentar novos limites superiores para as maiores instâncias. Quanto ao tempo de execução superior ao algoritmo LRGTS, pode ser diminuído através de estruturas que impeçam que algumas buscas sejam realizadas repetidas vezes desnecessariamente.

**Tabela 3:** Comparação do algoritmo LRGTS com o ILS-RVND

| | LRGTS | | | | ILS-RVND | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instância | Custo | D. | R. | Tempo | Custo | D. | R. | Média | gap/Custo | gap/Média | Tempo M. | gap/Tempo |
| C69-50x5 | 586,40 | 2 | 6 | 2,4 | 565,65 | 2 | 5 | 573,59 | -3,67 | -2,23 | 3,03 | 20,77 |
| C69-75x10 | 863,50 | 3 | 10 | 10,1 | 869,11 | 3 | 11 | 882,37 | 0,65 | 2,14 | 48,22 | 79,05 |
| C69-100x10 | 842,90 | 2 | 8 | 28,2 | 833,42 | 2 | 8 | 841,31 | -1,14 | -0,19 | 174,64 | 83,85 |
| D95-88x8 | 356,40 | 2 | 6 | 17,5 | 355,78 | 2 | 6 | 356,61 | -0,17 | 0,06 | 71,32 | 75,46 |
| D95-150x10 | 44386,30 | 3 | 14 | 119,2 | 44083,1 | 3 | 11 | 44558,62 | -0,69 | 0,39 | 641,98 | 81,43 |
| G67-21x5 | 424,90 | 2 | 4 | 0,2 | 424,89 | 2 | 4 | 424,89 | 0,00 | 0,00 | 0,17 | -17,65 |
| G67-22x5 | 587,40 | 1 | 3 | 0,2 | 585,1 | 1 | 3 | 585,10 | -0,39 | -0,39 | 0,20 | 0,00 |
| G67-29x5 | 512,10 | 2 | 4 | 0,4 | 512,1 | 2 | 4 | 518,83 | 0,00 | 1,30 | 0,62 | 35,48 |
| G67-32x5A | 584,60 | 2 | 4 | 0,6 | 562,22 | 1 | 4 | 562,22 | -3,98 | -3,98 | 0,75 | 20,00 |
| G67-32x5B | 504,80 | 1 | 3 | 0,5 | 504,32 | 1 | 3 | 504,32 | -0,10 | -0,10 | 0,79 | 36,71 |
| G67-36x5 | 476,50 | 1 | 4 | 0,7 | 460,37 | 1 | 4 | 460,37 | -3,50 | -3,50 | 1,24 | 43,55 |
| M92-27x5 | 3065,20 | 2 | 4 | 0,3 | 3062,01 | 2 | 4 | 3078,83 | -0,10 | 0,44 | 0,49 | 38,78 |
| M92-134x8 | 5809,00 | 3 | 11 | 48,3 | 5851,79 | 3 | 11 | 5941,66 | 0,73 | 2,23 | 317,12 | 84,77 |

**REFERÊNCIAS BIBLIOGRÁFICAS**

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.

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.

CONTARDO, C., CORDEAU, J., GENDRON, 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 -*

EILON, S., WATSON-GANDY, C. D. T. E CHRISTOFIDES, N. (1971) *Distribution Management: Mathematical Modelling and Practical Analysis - Charles Griffin and Company Limited*, London.

FORD, L. R. e FULKERSON, D. R. (1956) *Solving the transportation problem. – Management Science 3, p. 24-32.*

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

GEOFFRION, A. M. e POWERS, R. F. (1980) *Facility location analysis is just the beginning (if you do right) – Interfaces 10,* p. 22-30.

LAPORTE, G., NOBERT, Y. e PELLETIER, P. (1983) *Hamilton Location Problems – European Journal of Operational Research 12,* p. 83-89.

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.

MARKS, D. H. e STRICTER, R. (1971) *Routing for public service vehicle. – Journal of the Urban Planning and Development Quarterly 15,* p. 261-270.

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

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, Estados Unidos da América.

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

PERL, J. e DASKIN, M. S. (1984) *A unified warehouse location-routing methodology – Journal of Business Logistics 5,* 1, p. 92-111.

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.

RAND, G. K. (1976) *Methodological choices in depot location studies – Operational Researsh Quartely 27*, 1-ii, p. 241-249.

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.

SALHI, S. e RAND, G. K. (1989) *The effect of ignoring when locating depots. – European Journal of Operational Research 39,* p. 150-156.

SRIVATAVA, R. E BENTON, W. C. (1990) *The location-routing problem: Considerations in physical distributuion system design – Computers and Operations Research 17,* 5, p. 427-435.

SUSSAMS, J. E (1969) *Industrial Logistics – Gower Press,* London.

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

WEBB, M. (1968) *Cost functions in the locations of depots for multiple-delivery journeys – Operations Research Quarterly 19,* 3, p. 311-321.

WU, T. -H., LOW, C. E BAI, J. –W. (2002) *Heuristic solutions to multi-depot location-routing problems – Computers and Operations Research 29,* p. 1393-1415.