# UM ALGORITMO BASEADO EM ITERATED LOCAL SEARCH PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS PERIÓDICO

Pablo Luiz Araújo Munhoz[1],
Luiz Satoru Ochi[1],
Marcone Jamilson Freitas Souza[2]

[1]Instituto de Computação – Universidade Federal Fluminense(UFF)
{pmunhoz,satoru}@ic.uff.br

[2]Departamento de Computação – Universidade Federal de Ouro Preto (UFOP)
marcone@iceb.ufop.br

## 1. Introdução

O Problema de Roteamento de Veículos (PRV) é um dos problemas mais estudados na área de Otimização, tanto por sua complexidade combinatória, quanto por ter diversas aplicações em problemas reais. Tem como objetivo minimizar os custos de transporte relacionados ao atendimento de um conjunto de clientes a partir de uma frota de veículos homogêneos, localizados em um depósito. Neste modelo básico, um conjunto de rotas, com início e término no depósito, deve ser gerado de modo que cada cliente tenha sua demanda atendida por um único veículo em uma única visita, respeitando as restrições de capacidade do veículo.

O PRVP, diferentemente do PRV Clássico, é um modelo de roteamento de veículos onde os clientes necessitam de visitas com uma certa frequência, seja para reabastecimento de estoque, ou coleta de produtos ou resíduos. Assim, há a necessidade de definir primeiramente quais clientes serão atendidos em cada dia, para assim realizar o roteamento desses clientes escolhidos, respeitando o número de veículos disponíveis por dia de planejamento, e também a restrição de capacidade que cada veículo possui.

O PRVP pode ser definido em um grafo completo $G=(N,A)$, com custos não negativos dos arcos conhecidos $c_{ij}$, $\forall (i,j) \in A$; um período de planejamento de $|D|$ dias, indexado por $d$; um nó depósito, indexado por $i = 0$; um conjunto de nós clientes $N_c = N\backslash\{0\}$, onde cada nó $i \in N_c$ tem uma demanda total $q_i$ para cada dia do planejamento e requer um número fixo de visitas $f_i$ ; e um conjunto de $K$ veículos, cada um com capacidade $Q$.

Cada cliente $i$ possui uma Lista de Padrões de Visita (LPV), que representa quais as combinações de dias do período são válidas para esse cliente, respeitando o número de visitas $f_i$ ;Caso o cliente esteja em dias que não respeitem a LPV, esta solução é considerada inviável. A Tabela 1 ilustra um exemplo de uma LPV, onde consideramos um planejamento de $|D| = 5$ dias (Segunda à Sexta).

| Cliente | $f_i$ | Padrões |
|---|---|---|
| 1 | 3 | {Seg, Ter, Sex}, {Seg, Qua, Sex} |
| 2 | 1 | {Seg}, {Ter}, {Qua}, {Qui}, {Sex} |
| 3 | 2 | {Ter, Qua}, {Qua, Qui}, {Qui, Sex} |

Tabela 1: Exemplo de LPV

Nessa Tabela são apresentados três clientes (1, 2 e 3) que possuem uma frequência de visita de 3, 1 e 2 dias, respectivamente. Podemos observar que o primeiro cliente possui dois possíveis Padrões de Visita, podendo ser visitado na {Seg, Ter, Sex} ou {Seg, Qua, Sex}. O

segundo cliente é mais flexível e pode ser visitado em qualquer dia da semana, enquanto o terceiro cliente possui três opções de Padrões de Visita, podendo ser visitado {Ter, Qua}, {Qua, Qui} ou {Qui, Sex}. A LPV é um dado de entrada do problema e qualquer outra combinação de visitas de clientes que não esteja em um dos Padrões definidos por ela é considerada inviável.

## 3. Revisões bibliográficas

Os trabalhos mais recentes na literatura para a resolução do PRVP utilizam métodos baseados em heurísticas e metaheurísticas. Os principais trabalhos serão apresentados brevemente a seguir.

Em Russell (1979) são propostas duas heurísticas de construção e uma de refinamento. Uma das heurísticas construtivas é baseada no Algoritmo de Economias de Clarke e Wright com modificações para que o método somente gere soluções viáveis. É proposta, então, uma heurística de refinamento que otimiza as rotas e também os períodos da solução.

Em Christofides (1984) é introduzida uma definição de distribuição dos custos, e uma heurística baseada em trocas é utilizada para minimizar os custos dessa distribuição para o problema. A distribuição dos custos é representada pela troca do PRV vinculado a cada dia do PRVP por um Problema da P-Mediana, ou por um Problema do Caixeiro Viajante (PCV).

O PRVP é resolvido em Cordeau (1997) por uma abordagem baseada na metaheurística Busca Tabu. Esse método também é utilizado pelos autores para resolver o Problema de Roteamento de Veículos Periódico com Multiplos Depósitos e o Problema do Caixeiro Viajante Periódico. Os movimentos utilizados consistem em mover um cliente de uma rota para outra dentro do mesmo período, ou atribuir um novo padrão de visita a um cliente.

O trabalho desenvolvido em Vidal (2012) foi o primeiro a utilizar métodos de buscas locais mais agressivos visando a melhoria dos resultados da heurística. Foi proposta uma adaptação da metaheurística Algoritmos Genéticos, utilizando buscas locais e um mecanismo de diversificação adaptativo da população. Além de utilizar nove estruturas de vizinhança clássicas para o PRV, é definida uma nova estrutura de vizinhança específica para o PRVP, denominada *Pattern Improvement*.

Em Cordeau (2012) foi proposta um abordagem paralela utilizando conceitos das metaheurística *Iterated Local Search* (ILS) e Busca Tabu, denominada *Parallel Iterated Tabu Search*. Após a geração de uma solução inicial utilizando o algoritmo GENI, o ILS realiza melhorias na solução baseando-se na alternância entre duas fases, uma de busca local e outra de perturbação da solução. Esse algoritmo foi paralelizado, e melhores soluções foram obtidas.

## 4. Metodologia

Neste trabalho é proposto um algoritmo heurístico, denominado ILS-PVND, para a resolução do PRVP. Este algoritmo é baseado na metaheurística *Iterated Local Search*, e nos métodos *Variable Neighborhood Descent* (VND) e *Pattern Improvement*.

### 4.1. Representação de uma solução

Uma solução para o PRV é representada por uma combinação de clientes, numerados de 1 a $n$, e separados em $k$ partições, com $n$ representando o número de veículos utilizados. Assim, em um problema com $n = 15$ clientes e $k = 3$ veículos, uma possível solução pode ser representada por: [ [ 2 , 7 , 12 , 10 , 15 , 4 ] , [ 14 , 5 , 1 , 9 ] , [ 6 , 13 , 8 , 11 , 3 ] ]. Em cada

uma das partições, o primeiro elemento da lista representa o primeiro cliente a ser visitado pelo veículo a partir do depósito. A ordem de visitas segue a ordem dos elementos da lista. Chegando ao último elemento, o veículo retorna ao depósito. Uma representação dessa solução pode ser vista na Figura 1.

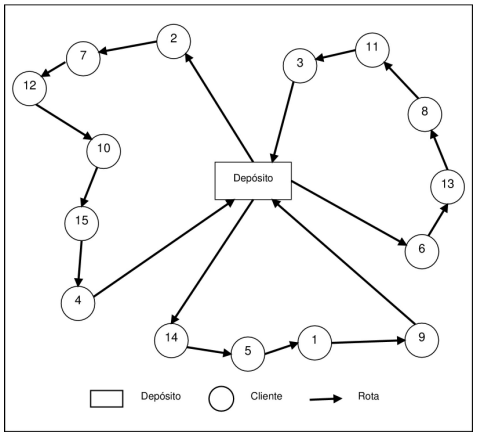


Figura 1: Representação do PRV

Para representar uma solução do PRVP, adiciona-se mais uma dimensão, referente ao período de planejamento de |*D*| dias, à representação do PRV. Adicionando-se então essa nova dimensão, os clientes são distribuídos entre os dias, de acordo com suas LPV, e em cada dia tem-se um PRV.

Assim, em um problema com $n$ = 16 clientes, $k$ = 3 veículos e |*D*| = 3 dias, uma possível solução pode ser representada por: [ { [ 8 , 1 ] , [ 9 , 6 , 2 , 3 ] , [ 5 , 7 , 4 ] } , { [ 8 , 10 ] , [ 15 , 11 ] , [ 12 , 9 , 4 ] } , { [ 8 , 16 , 6 ] , [ 14 , 7 , 13 ] } ]. Cada conjunto de partições, unidos por {...}, representa um dia do período de planejamento, e em cada dia, temos uma representação de um PRV Clássico. Uma representação dessa solução pode ser vista na Figura 2.

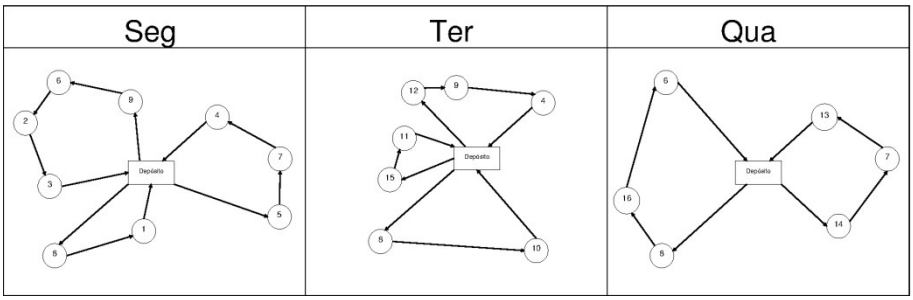


Figura 2: Representação do PRVP

## 4.2. Função de avaliação

Uma solução $\boldsymbol{s}$ é avaliada por uma função *f(s)* apresentada na Equação abaixo:

$$\boldsymbol{f(s) = \sum_{d \in D} \sum_{(i,j) \in A} c_{ij} x_{ij} + f(inv)}$$

em que:

a) A: conjunto dos arc (*i,j*) , com *i, j* ∈ *N*;

b) D: conjunto de dias do período de planejamento;

c) $\boldsymbol{c_{ij}}$: custos de transporte entre os clientes *i* e *j*;

d) $\boldsymbol{x_{ijd}}$: indica se o arco (*i,j*) ∈ A está sendo usado ($\boldsymbol{x_{ijd}} = \mathbf{1}$) ou não ($\boldsymbol{x_{ijd}} = 0$) na solução para o dia *d*.

A parcela de inviabilidade *f(inv)* foi calculada de acordo com a Equação abaixo:

$$\boldsymbol{f(\mathrm{inv}) = \mathrm{w} \sum_{\mathrm{d} \in \mathrm{D}} \sum_{\mathrm{k} \in \mathrm{K}} \max\left(0, \mathrm{Q'_{kd}} - \mathrm{Q}\right)}$$

em que:

a) *D*: conjunto de dias do período de planejamento;

b) *K*: conjunto de veículos disponíveis para cada dia;

c) $\boldsymbol{Q'_{kd}}$: a capacidade utilizada do veículo *k* no dia *d*;

d) *Q*: a capacidade de transporte dos veículos;

e) *w*: peso dado a cada unidade de sobrecarga de capacidade.

Dessa forma, para cada rota de cada dia, é calculado se houve um estouro da capacidade máxima do veículo. Caso positivo, um peso *w* é multiplicado por cada unidade que ultrapassou *Q*. Caso contrário, não é atribuída nenhuma penalidade.

### 4.3. Geração de uma solução inicial

A partir dos dados do problema, foi utilizado um método baseado na Heurística Construtiva de Inserção Mais Barata para gerar a solução inicial do problema abordado. Esse método é proposto nesse trabalho, e sua ideia é beneficiar os clientes que possuem uma frequência de visita ($f_i$) maior, pois são os que terão menos opções de inserção nos períodos. Esse método busca gerar soluções viáveis, mas caso não seja possível, uma relaxação da capacidade do veículo é utilizada.

Buscando a geração de soluções iniciais de melhor qualidade, utilizou-se também o Princípio da Otimalidade Próxima (*Proximate Optimality Principle*, Glover (1997)) durante o processo construtivo. Esse princípio afirma que boas soluções em um nível estão próximas de boas soluções em um nível adjacente. Assim, a solução parcialmente gerada é submetida a uma busca local durante a construção. Esse procedimento de busca local é executado a cada $\boldsymbol{\delta}$% do total de clientes serem inseridos.

**Algoritmo 1:** GeraçãoSoluçãoInicial$(N, D, K, \delta)$

```
1  início
2      Seja s_0 uma solução com |D| períodos e |K| rotas vazias
3      L ← ∅
4      para f = f_max até 1 faça
5          Grupo_f ← ClientesComFrequencia(f, N)
6          embaralha(Grupo_f)
7          insert(Grupo_f, L) // insere os cliente do Grupo_f em L
8      fim
9      enquanto L ≠ ∅ faça
10         c ← first(L) //o primeiro elemento de L é atribuído à c
11         taxaQ ← 1.0
12         repita
13             inseriu ← false
14             inseriu ← InserçãoMaisBarataPorPadrões(c, s_0, taxaQ)
15             se (¬ inseriu) então
16                 taxaQ = taxaQ + 0.1 //relaxação de Q
17             fim
18         até (inseriu)
19         remove(c, L) // Remove o cliente c da lista L
20         // Princípio da Otimalidade Próxima
21         se (δ% do total de clientes inseridos ) então
22             s_0 ← BuscaLocal(s_0)
23         fim
24     fim
25 fim
26 retorna s_0
```

Inicialmente uma solução com rotas vazias é criada e uma lista *L* utilizada para armazenar os clientes para inserção é definida como vazia. Após essa etapa de inicialização, os clientes são separados em grupos por frequência, em ordem decrescente. Os clientes pertencentes à cada grupo são embaralhados e adicionados à lista de clientes para inserção *L*. Ao embaralhar os clientes, garantimos que para cada geração inicial gerada a partir do mesmo problema-teste, novas soluções serão geradas. Após esse passo, a lista *L* conterá todos os clientes em ordem decrescente de frequência. Cada cliente pertencente a lista *L* será então adicionado sequencialmente a solução utilizando o Método de Inserção Mais Barata, buscando não exceder a capacidade *Q* do veículo. Caso não seja possível inserir o cliente, uma relaxação da capacidade do veículo é aplicada. Após o cliente ser inserido na solução inicial ele é removido

da lista *L*, e caso o critério de busca seja satisfeito, o Princípio da Otimalidade Próxima é aplicado. O processo se repete até que não haja mais nenhum cliente na lista. Esse método garante que todos os clientes sejam inseridos respeitando suas Listas de Padrões de Visita.

### 4.4. Estruturas de vizinhança

Para explorar o espaço de soluções, foram utilizadas 4 estruturas de vizinhança clássicas Intra-Rotas e 6 estruturas de vizinhança clássicas Inter-Rotas, todas pertencentes ao Problema de Roteamento de Veículos Clássico.

#### 4.4.1. Intra-rotas

Nas Estruturas de Vizinhança Intra-Rotas os movimentos aplicados são sempre entre clientes que pertençam à mesma rota. As 4 estruturas de vizinhança utilizadas foram:

- *Exchange*: há uma permutação entre dois clientes *i* e *j*;
- *1Or-opt*: há uma realocação de um cliente *i* para uma nova posição na rota;
- *2Or-opt*: há uma realocação de dois clientes consecutivos *i* e *j* para uma nova posição na rota;
- *2-opt*: dois arcos da solução, não adjacentes, são removidos e outros dois são inseridos formando uma nova solução onde há uma inversão na ordem de visitação dos clientes entre esses dois arcos.

#### 4.4.2. Inter-rotas

Nas Estruturas de Vizinhança Inter-Rotas os movimentos são aplicados entre clientes que pertencem à rotas diferentes. As 6 estruturas de vizinhança utilizadas foram:

- *Swap(1,1)*: há uma permutação entre um cliente *i* da rota *r1* e um cliente *i'* da rota *r2*;
- *Swap(2,1)*: há uma permutação entre dois clientes adjacentes *i* e *j* da rota *r1* e um cliente *i'* da rota *r2*;
- *Swap(2,2)*: há uma permutação entre dois clientes adjacentes *i* e *j* da rota *r1* e dois clientes adjacentes *i'* e *j'* da rota *r2* . A permutação é testada fazendo as quatro combinações possíveis entre os arcos *(i ,j)* e *(i' ,j')*;
- *Shift(1,0)*: há uma realocação de um cliente *i* da rota *r1* para uma nova posição na rota *r2* ;
- *Shift(2,0)* há uma realocação de dois clientes adjacentes *i* e *j* da rota *r1* para uma nova posição na rota *r2*. Assim como no movimento *Swap(2,1)* a realocação é testado com os arcos *(i ,j)* e *(j, i)*;
- *Cross*: são escolhidos dois arcos, *(i ,j)* pertencente à rota *r1* , e outro *(i' ,j')* pertencente à rota *r2* . Esses arcos são removidos e dois novos arcos *(i ,j')* e *(i' ,j)* são adicionados.

### 4.5. Buscas locais

Para realizar as buscas locais para o PRVP, foram utilizas duas heurísticas. A primeira, denominada *Periodic* VND, explora as características do Problema de Roteamento de Veículos Clássico presentes no PRVP, e a outra, denominada *Pattern Improvement*, explora a característica de periodicidade do PRVP.

#### 4.5.1. Estruturas auxiliares de dados

Visando auxiliar as heurísticas de Busca Local, três Estruturas Auxiliares de Dados (EAD) foram utilizadas: *carga[][]*, *padraoCliente[]* e *statusDia[]*. Cada uma dessas estruturas armazena em si informações úteis sobre as rotas da solução corrente do Algoritmo.

- *carga[][]*:esta estrutura guarda o valor da carga utilizada de cada veículo *k* para cada dia *d* (*carga[d][k]*). Essa informação é utilizada para que um movimento inviável, que exceda a capacidade *Q* do veículo, seja descartado antes de ser aplicado. Assim movimentos que gerariam soluções inviáveis são evitados, não sendo aplicados desnecessariamente;

- *padraoCliente[]*:esta estrutura guarda em qual Padrão de Visita o cliente se encontra. Assim, quando há troca de padrões de visita de um cliente, e este precisa ser removido, não é necessário que toda a solução seja percorrida à procura dele, bastando acessar os dias em que ele realmente se encontra;

- *statusDia[]*:esta estrutura guarda se houve alguma modificação na solução no dia *d*. Assim, caso *statusDia[d]* = 1, alguma mudança ocorreu no dia *d* e 0, caso contrário. Esta estrutura faz que somente dias que sofreram algum tipo de alteração passem por Buscas Locais, evitando que dias que não sofreram alterações tentem ser explorados exaustivamente sem que nenhuma melhora possa ser conseguida.

### 4.5.2. *Periodic* vnd

A heurística *Variable Neighborhood Descent* (VND), Mladenovic (1997), é um método de refinamento que consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada (SOUZA, 2011a).

As trocas de estruturas de vizinhança são classicamente feitas seguindo uma ordem pré-estabelecida. No presente trabalho optou-se por utilizar uma ordem aleatória de exploração, que além de simplificar o algoritmo, pois como são utilizadas 10 estruturas de vizinhança para o problema, há 10! permutações possíveis e não seria viável estabelecer a ordem mais adequada. A utilização dessa técnica obteve bons resultados nos problemas tratados por Penna (2011) e Souza (2011b) e é denominada *VND with random neighborhood ordering* (RVND).

Como podemos considerar cada dia do PRVP como um PRV Clássico, a heurística RVND é utilizada, em cada um desses dias, com todas as Estruturas de Vizinhança do PRV descritas na Seção 4.4. aplicadas diretamente ao PRVP sem que adaptações tenham sido feitas. Essa estratégia de executar o RVND em cada dia foi denominada *Periodic* VND (PVND).

### 4.5.3 *Pattern improvement*

A heurística *Pattern Improvement* (PI) foi proposta por Vidal (2011) e visa explorar a característica de periodicidade do PRVP. Para entendermos o seu funcionamento, tomaremos $\overline{\boldsymbol{p}}$ o padrão de visita do cliente $\boldsymbol{i}$ na solução corrente. O método PI é executado sobre todos os clientes, tomados em ordem aleatória, e computa, para cada cliente *i* e padrão $p \in LPV$, $\boldsymbol{\theta(i,p)} = \sum_{l \in p} \boldsymbol{\vartheta(i.l)}$, o custo mínimo que satisfaz aos requisitos de visita do cliente *i* de acordo com o padrão de visita $\boldsymbol{p}$. Definimos $\boldsymbol{\vartheta(i.l)}$ como o custo da inserção mais barata do cliente *i* no dia *l*. Caso o padrão de visita exista de tal modo que $\boldsymbol{\theta(i,p)} < \theta(\boldsymbol{i},\overline{\boldsymbol{p}})$, então todas as visitas do cliente *i* são removidas da solução corrente, e as novas visitas são adicionadas na melhor posição encontrada pela busca nos dias $l \in p$. A busca termina quando todos os clientes forem considerados e nenhuma mudança ocorrer na solução corrente.

**Algoritmo 2:** PI($s$, padraoCliente[ ])

```
1  início
2      alterado ← false
3      repita
4          s' ← s // Fazendo uma cópia da solução corrente
5          L ← {1, 2, 3, ..., n} //Criando uma lista com todos os clientes
6          embaralha(L)
7          alterado ← false
8          // Enquanto não verificar todos os clientes
9          enquanto L ≠ ∅ faça
10             c ← first(L) //o primeiro cliente de L é atribuído à c
11             removeCliente(c, s', padraoCliente[c])
12             taxaQ ← 1.0
13             repita
14                 inseriu ← false
15                 inseriu ← InserçãoMaisBarataPorPadrões(c, s', taxaQ)
16                 se (¬ inseriu) então
17                     taxaQ = taxaQ + 0.1 //relaxação de Q
18                 fim
19             até (inseriu)
20             remove(c, L) // Remove o cliente c da lista L
21             //Verificando se a inserção modificou a solução corrente
22             se (s' ≠ s) então
23                 alterado ← true
24             fim
25         fim
26         s ← s' //Atualizando a solução corrente
27     até (alterado = false)
28     Atualiza o statusDia[ ] de acordo com as mudanças nos dias
29 fim
30 retorna s
```

### 4.6. Algoritmo proposto ILS-PVND

O ILS, Lourenco (2003), é uma metaheurística que procura focar a busca não no espaço completo de soluções, mas num pequeno subespaço definido por soluções que são ótimas locais de um determinado mecanismo de otimização. De acordo com Lourenco (2003), o

sucesso do ILS é centrado no conjunto de amostragem de ótimos locais, juntamente com a escolha da Busca Local, das Perturbações e do Critério de Aceitação.

O algoritmo proposto, denominado ILS-PVND, é um algoritmo *Multistart* que baseia-se na metaheurística ILS, e utiliza as componentes anteriormente definidas.

A Busca Local utilizada é uma combinação entre o PI e o PVND. Uma solução é passada para essa busca local que primeiramente executa o PI. Após essa execução, a solução gerada pelo PI é passada para o PVND para uma nova etapa de otimização. O processo se repete até que não haja mais melhora na solução corrente.

A etapa de perturbação da solução foi implementada utilizando a aplicação de um movimento aleatório de uma das seguintes Estruturas de Vizinhança: *Swap(1,1)*, *Shift(1,1)*, *Shift(2,1)* e *ShiftLPV*.

Caso haja melhora no valor da função objetivo, a melhor solução encontrada é armazenada e é reiniciado o contador de iterações sem melhora do ILS (*iterILS*). Observa-se que somente soluções de melhora foram consideradas pelo Critério de Aceitação adotado na execução do ILS. Após cada execução completa do ILS, é feito o teste que armazena a melhor solução encontrada em todas as iterações do ILS-PVND.

**Algoritmo 3:** ILS-PVND($instancia, iterMax, iterMaxILS, \delta$)

```
1  início
2      prob ← CarregaProblemaTeste(instancia)
3      s_best ← ∅
4      f_best ← ∞
5      para iter = 1 até iterMax faça
6          s_0 ← GeraçãoSoluçãoInicial(prob.N, prob.D, prob.K, δ)
7          s* ← PVND(s_0)
8          iterILS ← 0
9          enquanto iterILS < iterMaxILS faça
10             s' ← Perturbação(s*)
11             //Busca Local
12             enquanto houver melhora faça
13                 s" ← PI(s')
14                 s" ← PVND(s")
15                 s' ← s"
16             fim
17             //Critério de Aceitação
18             se (f(s') < f(s*)) então
19                 s* ← s'
20                 iterILS ← 0
21             senão
22                 iterILS ← iterILS + 1
23             fim
24         fim
25         se (f(s*) < f_best) então
26             s_best ← s*
27             f_best ← f(s*)
28         fim
29     fim
30 fim
31 retorna s_best
```

## 5. Resultados computacionais

O algoritmo desenvolvido no presente trabalho, ILS-PVND, foi desenvolvido utilizando a linguagem C++. Os testes foram executados em um computador com processador Intel Core i7 3,07 GHz, 8GB de memória RAM, sistema operacional Ubuntu 10.04. Apesar de o computador oferecer recursos *multi-core*, esses não foram utilizados.

Para testar o algoritmo foi utilizado um conjunto de 32 problemas-teste da literatura que não possuem restrição de duração da rota. Eles possuem de 50 a 417 clientes, com períodos de 2 a 10 dias. O algoritmo foi executa 10 vezes para cada problema-teste, com limite de tempo de 1 hora (3600 segundos).

Na Tabela 2 são apresentados os melhores resultados encontrados na literatura para cada um dos problemas-teste analisados e também os resultados obtidos pelo ILS-PVND. Na primeira coluna indica-se o nome do problema-teste. A segunda coluna, Lit., é feita uma compilação dos melhores resultados disponíveis na literatura. Na terceira e quarta colunas são apresentados os melhores resultados obtidos pelo ILS-PVND, e o tempo gasto, em segundos, para a obtenção desse resultado. Na quinta coluna é calculado o GAP da melhor solução encontrada pelo ILS-PVND e o melhor resultado da literatura. Na quinta e sexta colunas são apresentados os valores médios obtidos pelo ILS-PVND e também o tempo médio gasto pelo algoritmo por problema teste, respectivamente. Na sétima coluna é apresentado o GAP Médio que é calculado utilizando-se o resultado médio obtido pelo ILS-PVND e o melhor resultado da Literatura. Na penúltima coluna é apresentado o Desvio Padrão para cada um dos problemas-teste. Por fim, na última coluna é apresentado o Coefiente de Variação. Os empates nos resultados entre a metodologia proposta e a Literatura são destacados em negrito.

| Inst. | Lit. | Melhor | Tempo (s) | GAP | Média | Tempo Médio | GAP Médio | Desvio Padrão | Coef. Variação |
|---|---|---|---|---|---|---|---|---|---|
| p01 | 524,61 | 524,61 | 23,92 | **0,00%** | 524,61 | 32,12 | 0,00% | **0,00** | 0,00% |
| p02 | 1322,87 | 1322,87 | 42,12 | **0,00%** | 1336,76 | 53,63 | 1,05% | 6,64 | 0,50% |
| p03 | 524,61 | 524,61 | 22,28 | **0,00%** | 524,61 | 24,78 | 0,00% | **0,00** | 0,00% |
| p04 | 835,26 | 835,26 | 88,28 | **0,00%** | 842,06 | 116,30 | 0,81% | 3,95 | 0,47% |
| p05 | 2024,96 | 2037,97 | 429,16 | 0,64% | 2055,40 | 533,20 | 1,50% | 8,38 | 0,41% |
| p06 | 835,26 | 835,26 | 256,27 | **0,00%** | 838,66 | 199,46 | 0,41% | 3,44 | 0,41% |
| p07 | 826,14 | 826,14 | 212,03 | **0,00%** | 829,55 | 330,08 | 0,41% | 1,53 | 0,18% |
| p08 | 2022,47 | 2040,78 | 403,06 | 0,91% | 2053,63 | 467,17 | 1,54% | 8,29 | 0,40% |
| p09 | 826,14 | 826,14 | 262,20 | **0,00%** | 829,76 | 331,83 | 0,44% | 1,63 | 0,20% |
| p10 | 1593,43 | 1604,26 | 370,22 | 0,68% | 1624,69 | 327,15 | 1,96% | 8,99 | 0,55% |
| p11 | 770,89 | 779,29 | 700,20 | 1,09% | 791,70 | 695,71 | 2,70% | 5,17 | 0,65% |
| p12 | 1186,47 | 1202,72 | 611,14 | 1,37% | 1228,78 | 626,39 | 3,57% | 10,52 | 0,86% |
| p13 | 3462,73 | 3596,60 | 3600,00 | 3,87% | 3660,36 | 3600,00 | 5,71% | 19,94 | 0,54% |
| p14 | 954,80 | 954,81 | 3,49 | **0,00%** | 954,81 | 3,54 | 0,00% | **0,00** | 0,00% |
| p15 | 1862,60 | 1862,63 | 11,11 | **0,00%** | 1862,63 | 11,40 | 0,00% | **0,00** | 0,00% |
| p16 | 2875,10 | 2875,24 | 31,03 | **0,00%** | 2875,24 | 55,69 | 0,00% | **0,00** | 0,00% |
| p17 | 1597,75 | 1597,75 | 32,79 | **0,00%** | 1608,01 | 36,20 | 0,64% | 8,48 | 0,53% |
| p18 | 3131,09 | 3150,25 | 100,35 | 0,61% | 3158,33 | 149,83 | 0,87% | 4,55 | 0,14% |
| p19 | 4834,34 | 4846,49 | 295,59 | 0,25% | 4846,49 | 334,66 | 0,25% | **0,00** | 0,00% |
| p20 | 8367,40 | 8367,40 | 1649,73 | **0,00%** | 8367,40 | 1823,77 | 0,00% | **0,00** | 0,00% |
| p21 | 2170,61 | 2173,36 | 122,26 | 0,13% | 2182,97 | 147,72 | 0,57% | 3,20 | 0,15% |
| p22 | 4193,95 | 4231,03 | 549,91 | 0,88% | 4288,10 | 771,21 | 2,24% | 35,72 | 0,83% |
| p23 | 6420,71 | 6426,51 | 1485,95 | 0,09% | 6608,13 | 3460,42 | 2,92% | 27,23 | 0,41% |
| p24 | 3687,46 | 3687,46 | 55,68 | **0,00%** | 3716,95 | 64,72 | 0,80% | 24,68 | 0,66% |
| p25 | 3777,15 | 3781,38 | 30,48 | 0,11% | 3782,25 | 33,37 | 0,13% | 2,61 | 0,07% |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| p26 | 3794,95 | 3795,32 | 400,72 | 0,01% | 3796,43 | 596,96 | 0,04% | 1,76 | 0,05% |
| p27 | 21833,87 | 21998,78 | 361,82 | 0,76% | 22196,46 | 512,82 | 1,66% | 175,18 | 0,79% |
| p28 | 22242,50 | 22357,81 | 434,36 | 0,52% | 22412,13 | 556,47 | 0,76% | 32,45 | 0,14% |
| p29 | 22543,75 | 22667,95 | 459,00 | 0,55% | 22794,92 | 562,98 | 1,11% | 112,99 | 0,50% |
| p30 | 73875,19 | 75352,63 | 2506,87 | 2,00% | 76351,77 | 2912,84 | 3,35% | 510,91 | 0,67% |
| p31 | 76001,57 | 77145,98 | 2273,87 | 1,51% | 78313,88 | 3089,94 | 3,04% | 597,03 | 0,76% |
| p32 | 77598,00 | 78740,91 | 2379,23 | 1,47% | 80215,91 | 2970,98 | 3,37% | 628,79 | 0,78% |
| **Médias** | | | | 0,55% | | | 1,31% | | 0,36% |

Tabela 2: Resultados Computacionais

Ao observar os resultados apresentados na Tabela 2 quanto aos melhores resultados obtidos, verifica-se que o algoritmo proposto (ILS-PVND) consegue alcançar os resultados da Literatura em 13 dos 32 problemas-teste, e dentre os demais, em apenas dois casos o GAP para a melhor solução da Literatura ultrapassou os 2,00%. Além disso, em média, houve um GAP das melhores soluções de apenas 0,55%, o que mostra que, de modo geral, o método está muito próximo das melhores soluções obtidas pelos melhores algoritmos presentes na Literatura.

Observando-se os resultados médios, em 18 dos 32 problemas-teste, o algoritmo obteve resultados com GAP Médio abaixo de 1,0%, além de uma média dos GAPs de 1,31% em relação aos melhores resultados da Literatura.

O ILS-PVND se mostrou robusto, visto que apresentou desvios relativamente baixos, observando-se o Coeficiente de Variação. Em 21 dos 32 problemas-teste analisados, o desvio foi menor ou igual a 0,50%. No demais casos, obteve o desvio máximo de 0,86%, mostrando que o algoritmo proposto é capaz de gerar soluções com baixa variabilidade em relação à qualidade.

## 6. Conclusões

O ILS-PVND é um algoritmo *Multistart* que combina a metaheurística ILS, a heurística VND e o método PI. Para a exploração do espaço de soluções foram utilizadas Estruturas de Vizinhança típicas do Problema de Roteamento de Veículos Clássico, aplicadas dia-a-dia na solução.

Para validar o algoritmo proposto, um conjunto de 32 problemas-teste foram usados e os resultados obtidos comparados aos da Literatura. Os experimentos computacionais mostram que a abordagem proposta é competitiva, pois consegue obter bons resultados. Além disso, apresentou um baixo Coeficiente de Variação, que mostra que o algoritmo é robusto, e gera soluções com baixa variabilidade.

## Agradecimentos

Os autores agradecem às agências CNPq, FAPERJ e FAPEMIG pelo apoio ao desenvolvimento deste trabalho.

## Referências

**CHRISTOFIDES, N. & BEASLEY, J.E.** *The period routing problem.* Networks. Vol. 14, p. 237-256, 1984.

**CORDEAU, J.-F.; GENDREAU, M. & LAPORTE, G.** *A tabu search heuristic for periodic and multi-depot vehicle routing problems.* Networks. Vol. 30, p. 105-119, 1997.

**CORDEAU,J.-F. & MAISCHBERGER,M.** *A parallel iterated tabu search heuristic for vehicle routing problems.* Computers & Operations Research. Vol. 39, p. 2033-2050, 2012

**GLOVER, F. & LAGUNA, M.** *Tabu Search.* Kluwer Academic Publishers,1997.

**LOURENÇO, H. R.; MARTIN, O. C.; STUTZLE, T.** *Iterated local search*. In Handbook of Metaheuristics, F. GLOVER and G. A. KOCHENBERGER, Eds. Kluwer Academic Publishers, p. 321-353, 2003.

**MLADENOVIC, N. & HANSEN, P.** *Variable neighborhood search.* Computers & Operations Research. Vol. 24, p. 1097-1100, 1997.

**PENNA, P.; SUBRAMANIAN, A. & OCHI, L.** *An iterated local search heuristic for the heterogeneous fleet vehicle routing problem*. Journal of Heuristics, p. 1-32, 2011.

**RUSSELL, R. & IGO,W.** *An assignment routing problem.* Networks. Vol. 9, p. 1-17, 1979.

**SOUZA, M.J.F.** *Inteligência computacional para otimização.* Online, disponível em http://www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/InteligenciaComputacional.htm, acessado em 01 de Novembro de 2011a.

**SOUZA, M.J.F.; MINE, M.T.; SILVA, M.S.A.; OCHI, L.S. & SUBRAMANIAN,A.** *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 and Management. Vol. 10, p. 142-157, 2011b.

**VIDAL, T.; CRAINIC, T.; GENDREAU, M.; LAHRICHI, N. & REI, W.** *A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems.* Operations Research, 2012 (A ser publicado).