UMA HEURÍSTICA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM MÚLTIPLAS VIAGENS

Daniel Martins da Silva, Yuri Abitbol de Menezes Frota
Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria 156 – Bloco E – 3° andar, São Domingos, CEP: 24210-240, Niterói, RJ
{danielmartins, yuri}@ic.uff.br

Anand Subramanian
Departamento de Engenharia de Produção – Universidade Federal da Paraíba
Centro de Tecnologia, Campus I – Bloco G, Cidade Universitária, 58051–970, João Pessoa, PB
anand@ct.ufpb.br

RESUMO

O Problema de Roteamento de Veículos (PRV) é um problema clássico da otimização onde há um depósito possuindo ofertas de produtos e um conjunto de clientes cujas demandas necessitam ser atendidas. O objetivo do problema é atender as demandas dos clientes minimizando os custos dos deslocamentos. O Problema de Roteamento de Veículos com Múltiplas Viagens (PRVMV) é uma extensão do PRV que se diferencia do problema original por permitir que um veículo atenda mais de uma rota durante um período de planejamento. Este trabalho propõe um algoritmo de duas fases para solucionar o PRVMV. Primeiro, um procedimento baseado na metaheurística *Iterated Local Search* (ILS) combinada com o método *Random Variable Neighborhood Descent* (RVND) gera conjuntos de rotas que depois são atribuídas a veículos utilizando-se uma heurística inspirada na concepção do *Bin Packing Problem* (BPP). Resultados computacionais são apresentados para um conjunto de instâncias clássicas da literatura.
PALAVRAS CHAVE. PRV com Mútiplas Viagens, Iterated Local Search, Bin Packing.

Áreas Principais: Metaheurística, Logística e Transportes, Otimização Combinatória.

ABSTRACT

The Vehicle Routing Problem (VRP) is a classical optimization problem where there is a depot with an available supply of goods and a set customers whose demands must be satisfied. The objective is to meet the customers demands minimizing the travel costs. The Vehicle Routing Problem with Multiple Trips (VRPMT) is an extension of the VRP where the vehicle is allowed to be assigned to multiple routes in a given time horizon. This work proposes a two-phase algorithm for the VRPMT. Firstly, procedure based on the *Iterated Local Search* (ILS) metaheuristic combined with a *Random Variable Neighborhood Descent* method generate sets of routes that are later assigned to vehicles by using a heuristic inspired on the *Bin Packing Problem* (BPP) concept. Computational experiments are reported for a set of well-known instances from the literature.
KEYWORDS. VRP with Multiple Trips, Iterated Local Search, Bin Packing.

Main areas: Metaheuristic, Logistics and Transport, Combinatorial Optimization.

1 Introdução

O Problema Roteamento de Veículos com Múltiplas Viagens (PRVMV) é uma derivação do problema de roteamento de veículos (PRV) clássico, que tem como objetivo minimizar os custos relacionados com transportes, utilizando uma frota de veículos homogênea e limitada a partir de um depósito, para atendimento de um conjunto de clientes. Para obter uma solução viável do problema, certas restrições devem ser respeitadas: os veículos não podem ultrapassar o seu limite de carga; o início e o fim de cada rota tem como ponto de partida e retorno o depósito; os clientes são atendidos por apenas um único veículo e este por sua vez atende apenas uma rota durante um período de planejamento.

No PRVMV a última restrição do PRV é modificada permitindo múltiplas viagens e com isso, um veículo pode atender mais que uma rota durante um período de planejamento. Este problema também é conhecido na literatura como *Vehicle Routing Problem with Multiple Trips* (VRPMT) pertencente à classe NP-difícil, isto é, não há algoritmos que obtenham soluções ótimas em tempo polinomial, por este motivo tornou-se um campo atrativo para utilização de heurísticas e metaheurísticas.

O PRVMV tem grande importância na área de transporte e logística das empresas, pois nem sempre a duração das rotas termina juntamente com a jornada de trabalho, ocasionando ociosidade dos veículos. Estudos relacionados com melhor aproveitamento desses recursos são de vital importância na gestão logística das empresas.

Na literatura, o PRVMV tem sido solucionado de diversas maneiras. Inicialmente recebeu a atenção de Hommes *et al*. (1989), onde os autores realizaram estudos empíricos numa indústria chamada *Biscuits Burton Ltd.*. Foi observado que com a relaxação da restrição de exclusividade de atendimento do veículo durante um período de planejamento, o número de veículos utilizados foi reduzido de 21 para 19, significando uma redução de custo de 5% nas entregas.

Fleischmann (1990) propõe uma heurística de *Bin Packing* com uma estratégia de *Best Fit Decreasing* (BFD) para agrupar as rotas. Taillard *et al*. (1995) fazem uso da ideia de Fleischmann combinada com a metaheurística Busca Tabu (BT). Brandão e Mercer (1998) abordam novamente o sistema de distribuição da empresa *Biscuits Burton Ltda* adicionando restrições extras como janela de tempo de atendimento aos clientes, frota heterogênea, contratação de veículos no caso de insuficiência e limites legais de jornada de trabalho. Resultados mostraram que o algoritmo superou a programação manual utilizada pela empresa em cerca de 20%.

Petch e Salhi (2003) desenvolveram uma heurística construtiva multi-fase que cria uma solução inicial para o PRV utilizando a heurística clássica de Clarke e Wright (1964) conhecida como heurística das economias. Foi criado um processo de diversificação da solução gerada por este método, fazendo uso de uma heurística denominada de *Tour Partition*. Para melhorar a solução, foram utilizadas heurísticas de refinamento (Exchange, 2-opt e 3-opt) e agrupamento de rotas com o objetivo de minimizar as penalidades utilizando o processo *Bin Packing*.

Zhao *et al*. (2002) utilizaram um modelo de programação linear inteira mista e um algoritmo de BT para gerar soluções para o PRVMV. Mais recentemente, Olivera e Viera (2007) empregaram um algoritmo de memória adaptativa enquanto Salhi e Petch (2007) desenvolveram um algoritmo genético híbrido que não utiliza a representação binária do cromossomo.

O objetivo deste trabalho é integrar o algoritmo ILS-RVND (Penna *et al*., 2011), que consiste em uma combinação entre a metaheurística *Iterated Local Search* (ILS) e método *Random Variable Neighborhood Descent* (RVND), com a heurística BFD para resolver o *Bin Packing Problem* (BPP).

O texto a seguir está organizado em sete seções incluindo esta introdução. A Seção 2 define o PRVMV. A Seção 3 apresenta a formulação matemática do PRVMV. A Seção 4 mostra as medidas de invialibilidade. A Seção 5 descreve o algoritmo proposto. A Seção 6 contém os resultados obtidos. Por fim, a Seção 7 apresenta as considerações finais.

2 Definição do Problema

O PRVMV pode ser definido sobre um grafo não-orientado $G = (V, E)$, em que $V = \{0, 1, ..., n\}$ representa os vértices do grafo, (assumimos que 0 é o depósito) e os nós $i \in V \setminus \{0\}$ representam os clientes, cada qual com sua demanda q_i. Cada aresta $E = \{(i, j)|i, j \in V, i < j\}$ tem um custo associado c_{ij} e um tempo t_{ij}, ambos não-negativos. A frota de veículos $K = \{1, ..., m\}$ é fixa, onde m representa o número de veículos, cada qual com sua capacidade homogênea Q. Define-se, também, um horizonte de tempo T que determina o período de planejamento.

Seja $r = (v_0, v_1, ..., v_{n(r)+1})$ uma rota, onde $v_0 = v_{n(r)+1} = 0$ e $n(r)$ é definido com o número de clientes visitados em r. Considere as seguintes notações:

- $q_r = \sum_{i=1}^{n(r)} q_{v_i}$ é a demanda coberta pela rota (cada vértice tem demanda q).
- $c_r = \sum_{i=0}^{n(r)} c_{v_i, v_{i+1}}$ é o custo da rota (c_{ij} é o custo do arco ij).
- $t_r = \sum_{i=0}^{n(r)} t_{v_i, v_{i+1}}$ é a duração da rota.

Tais elementos do problema caracterizam o PRVMV que tem como principal objetivo construir um conjunto de rotas, e atribuí-las a uma frota de veículos, minimizando o custo total de deslocamento e satisfazendo as seguintes condições:

1. Cada rota deve iniciar e terminar no depósito;
2. Cada cliente é visitado exatamente por uma rota;
3. O somatório das demandas dos clientes pertencentes a uma rota não deve exceder a capacidade do veículo Q ($q_r \leq Q$);
4. Cada veículo pode atender uma ou mais rotas, desde que a duração total das rotas não ultrapasse o horizonte de tempo.

Pode-se agora definir uma solução para o PRVMV como uma família de conjuntos de rotas $s = (R_1, ..., R_m)$ onde cada R_k contém as rotas do veículo $k \in K$. Define-se, também, R como o conjunto de todas as rotas viáveis, representada pela união de todos R_k (i.e. $R = \bigcup_{k \in K} R_k$). Além disso, o custo de uma solução $f(s) = \sum_{r \in R} c(r)$ é definido como o somatório dos custos de cada rota. Por fim, considere o símbolo $t_k = \sum_{r \in R_k} t_r$ como sendo a duração total das rotas atríbuidas ao veículo k.

3 Formulação matemática

A formulação matemática do PRVMV como um problema de programação inteira, introduzida por Olivera e Viera (2007) é apresentada pelas expressões (1) a (4) a seguir. Nessa formulação X_r^k assume o valor de 1 se a rota r está na solução e é atribuída ao veículo k, caso contrário a variável assume o valor de 0. O coeficiente a_{ir} tem o valor 1 se a rota $r \in R$ visitou o cliente $i \in V \setminus \{0\}$ e 0, caso contrário.

$$PRVMV = Min \sum_{k \in K} \sum_{r \in R} c_r X_r^k \quad (1)$$

$$s.a. \quad \sum_{k \in K} \sum_{r \in R} a_{ir} X_r^k = 1, \forall i \in V \setminus \{0\} \quad (2)$$

$$\sum_{r \in R} t_r X_r^k \leq T, \forall k \in K \quad (3)$$

$$X_r^k \in \{0,1\}, \forall k \in K, \ \forall r \in R \quad (4)$$

A formulação matemática do $PRVMV$ tem uma função objetivo (1) que busca minimizar o custo total do deslocamento. As restrições (2) determinam que cada cliente deve pertencer a exatamente uma rota. As restrições (3) asseguram que a duração das rotas atribuídas a cada veículo, não deve exceder o período de planejamento. As restrições (4) definem o domínio das variáveis.

4 Medidas de Inviabilidade

Devido ao alto grau de complexibilidade do PRVMV, nem sempre as metaheurísticas e os métodos exatos são capazes de encontrar soluções viáveis. Assim sendo, faz-se necessária a utilização de certas métricas para mensurar a qualidade das soluções inviáveis. Taillard *et al.* (1995) propõem três maneiras de quantificar esta inviabilidade: *Overtime* (Horas extras), *Longest Tour Ratio* e *Penalized Cost*.

- *Overtime* - ocorrre toda vez que a duração das rotas de um veículo ultrapassa o horizonte de planejamento do veículo. Este critério é definido por:

$$\text{Ot} = \sum_{k=1}^{m} [max(0, t_k - T)]. \quad (5)$$

- *Longest Tour Ratio* - este outro critério consiste na razão entre a maior duração t_k para todo $k \in K$, sobre o período de planejamento T.

$$\text{LTR} = \frac{max_{k \in K}(t_k)}{T}. \quad (6)$$

- *Penalized Cost* - é definido como a adição de uma penalidade P_γ ao custo da solução. Esta penalidade é o custo adicional empregado pela violação do período de planejamento, isto é, o *Overtime* total multiplicado por um parâmetro γ. Logo, define-se o custo do *Penalized Cost* PC_γ como :

$$PC_\gamma = f(s) + P_\gamma \quad (7)$$

onde, $P_\gamma = (\gamma)(Ot)$.

5 Algoritmo Proposto

O algoritmo proposto reúne componentes da metaheurística ILS (Lourenço *et al.*, 2002), implementado usando uma abordagem multi-start, e do procedimento Randomized Variable Neighborhood Descent (RVND), utilizado na fase de busca local. O algoritmo pode ser visto de forma simplificada no Algoritmo 1, cujos parâmetros são utilizados para determinar o número máximo de iterações ($MaxIter$) e o número máximo de pertubações consecutivas sem melhoras ($MaxIterILS$).

A cada iteração (linha 5-27) uma nova solução (conjunto de rotas) é gerada. No laço principal do ILS (linha 8-19) há uma tentativa de aprimoramento da solução s efetuando uma varredura no espaço de solução usando o RVND (linha 9). Caso o critério de aceitação da linha 10 ou 13 seja satisfeito é atualizado a nova melhor solução corrente e reiniciado o contador $iterILS$. No final o algoritmo retorna a melhor solução encontrada s^* em todas as iterações. Nas seções a seguir, serão explicados os principais componentes do Algoritmo 1.

Algoritmo 1 ILS-RVND-BFD

1: ILS-RVND-BFD($MaxIter$, $MaxIterILS$)
2: $CarregaInstancia()$;
3: $rota \leftarrow EstimaNumeroRotas()$;
4: $f^* \leftarrow \infty$; $P_\gamma s^* \leftarrow \infty$;
5: **para** $i := 1, ..., MaxIter$ **faça**
6: $s \leftarrow GeraSolucaoInicial(rota)$;
7: $s^{'} \leftarrow s$; $iterILS \leftarrow 0$;
8: **enquanto** $iterILS \leq MaxIterILS$ **faça**
9: $s \leftarrow RVND(s)$;
10: **se** $P_\gamma(s) = 0$ and $P_\gamma(s^{'}) = 0$ and $f(s) < f(s^{'})$ **então**
11: $s^{'} \leftarrow s$; $iterIls \leftarrow 0$;
12: **senão**
13: **se** $P_\gamma(s) < P_\gamma(s^{'})$ **então**
14: $s^{'} \leftarrow s$; $iterIls \leftarrow 0$;
15: **fim se**
16: **fim se**
17: $s \leftarrow Pertubacao(s^{'})$;
18: $iterILS \leftarrow iterILS + 1$;
19: **fim enquanto**
20: **se** $P_\gamma(s^{'}) = 0$ and $f(s^{'}) < f^*$ **então**
21: $s^* \leftarrow s^{'}$; $f^* \leftarrow f(s^*)$; $P_\gamma s^* \leftarrow 0$;
22: **senão**
23: **se** $P_\gamma(s^{'}) < P_\gamma s^*$ **então**
24: $s^* \leftarrow s^{'}$; $P_\gamma s^* \leftarrow P_\gamma(s^*)$;
25: **fim se**
26: **fim se**
27: **fim para**
28: return s^*;
29: end ILS-RVND-BFD

5.1 Procedimento EstimaNumeroRotas

Este procedimento, cujo pseudocódigo é apresentado no Algoritmo 2, é aplicado para determinar o número de rotas que deve ser gerado a cada solução inicial. Inicialmente assume-se que a variável $rota$ é igual a 1. Na linha 3 é criada uma lista de candidatos (LC) com todos os clientes. Enquanto a lista não for vazia (linha 6-16) é feito o cálculo do custo $g(i)$ $\forall i \in LC$ que representa o custo de inserir o cliente i na rota mais próxima, de acordo com um critério de Inserção Mais Próxima (IMP). O menor custo de inserção (g^{min}) associado ao cliente n deve ser adicionado na rota s^{rota}. Caso a rota ultrapasse a capacidade de carga (Q) do veículo, uma nova rota é gerada.

5.2 Construção da Solução Inicial

Para obter uma solução inicial do PRVMV, primeiramente é preciso encontrar uma solução que satisfaça o PRV. O Algoritmo 3 cria esta solução sendo que, em princípio, em cada

Algoritmo 2 EstimaNumeroRotas

```
1: EstimaNumeroRotas()
2: rota ← 1;
3: LC = V \ {0}
4: s_rota ← i ∈ LC selecionado aleatoriamente
5: Atualiza LC
6: enquanto LC ≠ {} faça
7:    g(i) é o custo associado i ∈ LC usando o critério IMP.
8:    g^min ← min{ g(i)|i ∈ LC }
9:    n ← cliente associado g^min
10:   s_rota ← s_rota ∪ {n}
11:   se demanda da s_rota > Q então
12:      rota ← rota + 1
13:      s_rota ← k ϵ LC selecionado aleatoriamente
14:   senão
15:      Atualiza LC
16:   fim se
17: fim enquanto
18: return rota
19: end EstimaNumeroRotas
```

rota é colocado um cliente da lista LC escolhido aleatoriamente. No laço principal (linha 8 - 14), foi desenvolvida uma estratégia de inserção paralela que utiliza novamente o critério de inserção mais próxima. Caso a solução gerada alcance uma inviabilidade, i.e. existe um cliente $i \in LC$ onde a sua inserção em qualquer rota R de s induzirá $q_R > Q$, uma nova rota é adicionada e o algoritmo retorna para linha 2.

5.3 Random Variable Neighborhood Descent

O Algoritmo 4 apresenta o pseudocódigo do procedimento RVND, que consiste em uma adaptação do *Variable Neighborhood Descent*, proposto por Mladenovic e Hansen (1997). Inicialmente a lista LV é inicializada com as estruturas de vizinhança Inter-Rotas (linha 2). Em cada execução do laço principal (3-12) uma vizinhança N é escolhida aleatoriamente (linha 4) e então, o movimento que gera um menor custo e que não aumente a penalidade (calculado através do algoritmo BFD descrito na seção 5.7) é armazenado em $s^{'}$ (linha 5). Em caso de melhora, a solução $s^{'}$ passa pelo procedimento *IntraRota* e a lista LV é reinicializada com todas as vizinhanças Inter-Rotas (linha 8). Caso contrário, a estrutura de vizinhança N é removida da lista LV.

5.4 Vizinhanças Inter-Rotas

Nas vizinhanças inter-rotas ocorrem movimentos entre duas rotas distintas. As buscas locais descriminadas abaixo têm como objetivo principal o aprimoramento da solução corrente encontrada. Vale ressaltar que ao encontrar um movimento que diminua o custo de deslocamento, este é submetido a uma avaliação, caso ocorra aumento de penalidade (P_γ) esse movimento é descartado. As vizinhanças Inter-Rotas são descritas a seguir:

- **Shift(1,0)** - consiste em transferir um cliente de uma rota para outra.
- **Shift(2,0)** - consiste em transferir dois clientes consecutivos de uma rota para outra.
- **Swap(1,1)** - permutação entre dois clientes pertencentes a rotas diferentes.
- **Swap(2,1)** - dois clientes consecutivos de uma rota são permutados com um cliente de outra rota.

Algoritmo 3 Gera Solução Inicial

1: GeraSolucaoInicial($rota$)
2: $LC = V \setminus \{0\}$
3: $s = \{R_1, ..., R_{rota}\}$
4: **para** $r := 1, ..., rota$ **faça**
5: $R_r \leftarrow i \in LC$ selecionado aleatoriamente
6: Atualiza LC
7: **fim para**
8: **enquanto** $LC \neq \{\}$ e existir pelo menos um cliente $i \in LC$ que pode ser adicionando a s **faça**
9: $g(i)$ é o custo associado $i \in LC$ usando o critério IMP.
10: $g^{min} \leftarrow \min\{ g(i) | i \in LC \}$
11: $n \leftarrow$ cliente associado g^{min} e $R^{min} \leftarrow$ rota associada g^{min}
12: $R^{min} \leftarrow R^{min} \cup \{n\}$
13: Atualiza LC
14: **fim enquanto**
15: **se** $s = Inviavel$ **então**
16: $rota = rota + 1$
17: Go to linha 2
18: **fim se**
19: return s
20: end GeraSolucaoInicial

Algoritmo 4 RVND

1: RVND(s)
2: Inicializa uma Lista (LV) com as vizinhanças Inter-Rotas
3: **enquanto** $LV \neq \{\}$ **faça**
4: Escolhe uma vizinhança $N \in LV$ aleatoriamente
5: $s^{'} \leftarrow$ Encontre o melhor movimento da vizinhança N em s que não ocasione aumento de penalidade
6: **se** $f(s^{'}) < f(s)$ **então**
7: $s \leftarrow IntraRota(s^{'})$
8: Reinicializa LV
9: **senão**
10: Remove N da lista LV
11: **fim se**
12: **fim enquanto**
13: return s
14: end RVND

- **Swap(2,2)** - dois clientes consecutivos de uma rota são permutados com dois cliente consecutivos de outra rota.
- **Cross** - consiste na remoção de dois arcos, sendo um de cada rota, e inserção de outros de modo a gerar uma nova solução.

5.5 Vizinhanças Intra-Rotas

As estruturas de vizinhanças Intra-Rotas utilizadas neste trabalho são *Or-opt*, *Reinserção*, *Or-opt3*, *2-opt* e *Exchange*. Todos estes movimentos acontecem na mesma rota.

- **Reinserção** - mudança de um cliente de uma posição para outra posição da rota.
- **Or-opt2** - realocação de dois clientes adjacentes de uma posição para outra posição da rota.
- **Or-opt3** - três clientes adjacentes são removidos e inseridos em uma outra posição.
- **2-opt** - remoção de dois arcos não adjacentes enquanto outros dois arcos são adicionados de tal forma que uma nova rota é gerada.
- **Exchange** - permutação de dois clientes.

O pseudocódigo do procedimento *IntraRota* está esquematizado no Algoritmo 5. Este procedimento difere do anterior por não reinicializar a lista de vizinhanças ($LV^{'}$).

Algoritmo 5 IntraRota

```
1:  IntraRota(s)
2:  Inicializa uma Lista (LV') com as vizinhanças Intra-Rotas
3:  enquanto LV' ≠ {} faça
4:     Escolhe uma vizinhança N' ∈ LV' aleatoriamente
5:     s' ← Encontre o melhor movimento da vizinhança N' em s
6:     se f(s') < f(s) então
7:        s ← s'
8:     senão
9:        Remove N' da lista LV'
10:    fim se
11: fim enquanto
12: return s
13: end IntraRota
```

5.6 Mecanismo de Perturbação

Mecanismos de perturbação permitem a movimentação de cliente (s), sem que haja verificação de redução de custo, com a finalidade de diversificar uma solução. Neste trabalho foram utilizadas três estruturas de perturbação escolhidas aleatoriamente em cada iteração do laço principal do ILS que são as seguintes:

- **Random-Shift(1,1)** - consiste em transferir um cliente de uma rota r_1 para uma rota r_2 e transferir um cliente de r_2 para r_1, ambas as transferências efetuadas aleatoriamente.
- **Random-Swap(1,1)** - consiste na aplicação do movimento Swap(1,1) de forma aleatória.
- **Random-Swap(2,2)** - consiste na aplicação do movimento Swap(2,2) de forma aleatória.

5.7 Bin Packing Problem

O BPP aplicado ao PRVMV consiste em atribuir um conjunto de rotas a um conjunto fixo de veículos, no qual cada rota tem uma duração e o tempo de trabalho dos veículos é limitado. Consequentemente, a soma das viagens atribuídas a um veículo não pode ultrapassar a sua jornada de trabalho. Caso ocorra esta violação no horizonte de tempo do veículo é calculado o *overtime*.

Para solucionar o BPP foi aplicada uma heurística gulosa denominada *Best Fit Decreasing* (BFD), onde as viagens (rotas) são ordenadas em ordem não-crescente de duração. A BFD aloca a viagem no veículo $k \in K$ de maior t_k, mas que tenha tempo livre o suficiente para armazená-la. Caso nenhum dos veículos já utilizados tiver capacidade de tempo suficiente para acomodar a viagem, um novo veículo é usado. No entanto, caso o número limite de veículos seja atingido, uma nova viagem é atríbuida ao veículo que produza menor penalidade. A complexidade do algoritmo é $O(nlogn)$.

6 Experimentos computacionais

O algoritmo proposto ILS-RVND-BFD foi desenvolvido na linguagem de programação C++ e os testes foram efetuados em um notebook com microprocessador AMD Turion X2 Ultra Dual-Core Mobile ZM-80 2.10 GHz com 4 GB de memória RAM, utilizando o sistema operacional Ubuntu Linux 10.04 Kernel 2.6.32-33.

Os experimentos computacionais foram realizados em um subconjunto de nove problemas-testes amplamente conhecidos da literatura, a saber: CMT-1, CMT-2, CMT-3, CMT-4, CMT-5, CMT-11 e CMT-12 adaptados de Christofides *et al.* (1979) e F-11 e F-12 adaptados de Fischer (1994), totalizando 104 instâncias. Na execução de cada um desses problemas foram considerados dois horizontes de tempo diferentes $T_1 = \{(1.05 * z^*)/m\}$ e $T_2 = \{(1.10 * z^*)/m\}$ onde z^* é o valor encontrado pela solução VRP obtido pelo algoritmo de Rochat e Taillard (1995) e m é o número de veículos. Para todos os problemas testes assume-se que $c_{ij} = t_{ij}$.

6.1 Resultados Detalhados

Para obter uma solução, o algoritmo ILS-RVND-BFD necessita de dois parâmetros, *MaxIter* e *MaxIterILS*, parâmetros descritos na Seção 5. Estes foram determinados empiricamente e os valores adotados foram os seguintes: $MaxIter = 40$ e $MaxIterILS = n + 10 \times rotas$, onde n é o número de clientes. Além disso, na execução do ILS-RVND-BFD foram adicionadas duas condições. A primeira ocorre quando a penalidade $P_\gamma = 0$ e $MaxIter > 10$ ocasionando na interrupção do algoritmo. Essa restrição tem como objetivo minimizar o tempo computacional para instâncias fáceis de encontrar soluções viáveis a partir das primeiras iterações. A segunda condição é aplicada quando $MaxIter = 38$, fazendo com que o número de rotas seja duplicado, com o intuito de diversificação da solução inicial.

Para definir a qualidade das soluções viáveis, foi utilizada a métrica GAP que pode ser representada pela razão entre a solução do ILS-RVND-BFD sobre melhor custo do VRP.

$$GAP_s = 100 * \left(\frac{f(s)}{z^*} - 1 \right). \tag{8}$$

Na Tabela 1 é apresentado o resultado médio de cinco execuções do ILS-RVND-BFD em comparação com o algoritmo de Olivera e Viera (2007), que apresenta os melhores resultados para soluções viáveis na literatura. Cada linha analisa uma instância do PRV sobre os dois horizontes de tempos T_1 e T_2 acompanhados de seus relativos GAPs, além disso, as seguintes características são reportadas:

- Viável/Total: o número de soluções encontradas viáveis e o número total de instâncias avaliadas;

- GAP: é a média do GAP considerando apenas os problemas onde foram encontrados soluções viáveis;
- A última linha apresenta o total de soluções viáveis encontradas (Viável/Total) e a Média Total do GAP.

Tabela 1: Resultado médio considerando o melhor GAP para cada instância

		ILS-RVND-BFD				Olivera e Viera (2007)			
		T_1		T_2		T_1		T_2	
Problema	n	Viável/Total	GAP	Viável/Total	GAP	Viável/Total	GAP	Viável/Total	GAP
CMT-1	50	2/4	**0.80**	4/4	**2.62**	2/4	**0.80**	4/4	**2.62**
CMT-2	75	6/7	**0.47**	7/7	**0.52**	6/7	0.72	7/7	0.95
CMT-3	100	6/6	**0.41**	6/6	**0.35**	6/6	0.87	6/6	0.47
CMT-4	150	8/8	**1.04**	8/8	**0.62**	7/8	1.16	8/8	1.33
CMT-5	199	10/10	**1.56**	10/10	**1.33**	10/10	2.74	10/10	2.15
CMT-11	120	5/5	**0.70**	5/5	**0.11**	5/5	1.52	5/5	0.34
CMT-12	100	5/6	0.65	6/6	**0.18**	5/6	**0.63**	6/6	**0.18**
F-11	71	2/3	**1.84**	3/3	**1.67**	2/3	2.10	3/3	1.8
F-12	134	3/3	**0.00**	3/3	**0.01**	3/3	0.48	3/3	0.64
Média Total		47/52	**0.83**	52/52	**0.82**	46/52	1.37	52/52	1.19

O algoritmo ILS-RVND-BFD apresenta 99 soluções viáveis, uma a mais que o algoritmo de Olivera e Viera (2007). Já os algoritmos de Brandão e Mercer (1998), Taillard *et al.* (1995), Petch e Salhi (2003) e Salhi e Petch (2007) reportaram 89, 86, 76 e 72 soluções viáveis, respectivamente. Na Média Total, a metaheurística empregada na resolução do PRVMV apresenta resultados melhores na ordem de 39% com relação ao período de planejamento T_1 e 31% em comparação a T_2.

6.2 Resultados Detalhados para Instâncias Inviáveis

Os melhores resultados encontrados pelo algoritmo durante cinco execuções para as soluções não viáveis são apresentados na Tabela 2, onde há um comparativo com os algoritmos de melhores resultados na literatura: Olivera e Viera (2007), Brandão e Mercer (1998) e Taillard *et al.* (1995). Para cada instância são avaliados os Custos $f(s)$, *Overtime* Ot e *Penalized Cost* PC_γ. Além disso, a última linha da tabela apresenta as diferenças (%) entre as médias de cada algoritmo em relação ao ILS-RVND-BFD.

É possível observar que a diferença média do custo $f(s)$ de roteamento do algoritmo ILS-RVND-BFD apresenta um acréscimo maior do que os demais, entretanto o maior acréscimo é de apenas 3,53% com relação ao algoritmo de Taillard *et al.* (1995). Porém, o objetivo principal do algoritmo ILS-RVND-BFD para instâncias inviáveis é a redução do *Overtime* (horas extras). Esta métrica, em relação aos algoritmos de Olivera e Viera (2007), Brandão e Mercer (1998) e Taillard *et al.* (1995), obtém melhores resultados, apresentando uma redução na média da diferença de 11.71%, 39.53% e 67.14% respectivamente. Usando $\lambda = 2$ o custo penalizado do ILS-RVND-BFD é apenas maior 0,14% do que o algoritmo de Busca Tabu de Taillard *et al.* (1995), porém, vale ressaltar que quanto maior o valor cobrado pelas horas extras, maior será o valor do PC, consequentemente melhorando os resultados do ILS-RVND-BFD.

A Tabela 3 mostra um comparativo da razão da maior rota sobre o período de planejamento (LTR) para soluções inviáveis. O ILS-RVND-BFD iguala-se aos melhores da literatura em três, das cinco instâncias em que não foram encontradas soluções viáveis.

6.3 Comparativo de Tempos Computacionais

O comparativo realizado com relação ao tempo não é uma tarefa trivial, pois cada autor tem uma maneira diferente de codificar seus algoritmos e os programas são executados em máquinas de configurações e modelos distintos. Tais fatores influenciam drasticamente no tempo computacional de execução de cada programa. Para criar-se um comparativo mais justo foi utilizado

Tabela 2: Comparação de total overtime (Ot) e penalized cost (PC) para soluções Inviáveis

	ILS-RVND-BFD			Olivera e Viera (2007)			Brandão e Mercer (1998)			Taillard *et al.* (1995)		
Prob.|m	$f(s)$	Ot	PC_λ	$f(s)$	Ot	PC_λ	$f(s)$	Ot	PC_λ	$f(s)$	Ot	PC_λ
CMT-1|3	556.22	**6.54**	**569.30**	558.82	7.29	573.40	556.34	9.70	575.73	**533.00**	23.24	579.48
CMT-1|4	555.08	**6.63**	568.34	547.10	8.49	**564.07**	547.10	8.49	**564.07**	**546.29**	9.49	565.27
CMT-2|7	868.15	**1.38**	**870.91**	873.40	1.86	877.11	870.07	13.25	896.57	**843.60**	17.35	878.29
CMT-12|6	852.99	**3.81**	860.61	852.99	**3.81**	860.61	**819.56**	14.14	847.85	**819.56**	12.96	**845.48**
F-12|3	256.07	1.84	259.75	256.85	1.85	260.54	254.40	**1.54**	257.47	**244.60**	6.36	**257.31**
Média da Diferença%				0.02	-11.71	-0.19	1.19	-39.53	-0.17	3.53	-67.14	0.14

Tabela 3: Comparação de Longest tour ratio(LTR) para soluções Inviáveis

Prob.|m	ILS-RVND-BFD	Olivera e Viera (2007)	Brandão e Mercer (1998)	Taillard *et al.* (1995)	Petch e Salhi (2003)	Salhi e Petch (2007)
CMT-1|3	1.032	**1.024**	1.041	1.115	1.026	1.030
CMT-1|4	**1.027**	**1.027**	**1.027**	**1.027**	1.085	1.056
CMT-2|7	**1.009**	**1.009**	1.088	1.073	1.064	1.102
CMT-12|6	**1.014**	**1.014**	1.072	1.064	1.029	1.029
F-12|3	1.020	1.020	**1.011**	1.075	1.020	1.020
Média	1.020	1.019	1.048	1.071	1.045	1.047
Média da Diferença%		0,16	-2,53	-4,64	-2,28	-2,49

o relatório de velocidade em *Milion of Floatin-Point Operations per Second* (Mflop/s) de Dongarra (2011) que apresenta a velocidade de processamento de vários computadores. Caso o processador não seja encontrado, assume-se a velocidade de um processador com características semelhantes.

Pelo relatório de Dongarra (2011), assume-se que as velocidades dos computadores em Mflop/s são: 172 MFlop/s - Ultra Enterprise 450 dual 300MHz , 15 MFlop/s - SGI Indigo2 Extreme R400/100MHz, 38 MFlop/s - Pentium Pro 200MHz, e 732 MFlop/s - AMD Athlon XP 2200+ 1.8GHz. No caso do processador AMD Turion X2 Ultra Dual-Core 2.10 GHz utilizado neste trabalho, obteve-se uma velocidade de 1385 MFlops/s.

Na Tabela 4 há um comparativo de tempo computacional em segundos com relação aos melhores algoritmos da literatura. Cada tempo é multiplicado por um fator obtido pela razão da velocidade da máquina sobre a menor velocidade encontrada entre os computadores comparados, que são: 11 - Salhi e Petch (2007), 1 - Taillard *et al.* (1995), 2,5 - Brandão e Mercer (1998), 11 - Petch e Salhi (2003), 49 - Olivera e Viera (2007) e 92 - ILS-RVND-BFD. O algoritmo proposto obteve tempos competitivos com relação as metaheurísticas comparadas, ficando 45% acima do tempo se comparado ao algoritmo de Brandão e Mercer (1998). Quanto aos demais algoritmos de Salhi e Petch (2007) , Taillard *et al.* (1995), Petch e Salhi (2003) e Olivera e Viera (2007) obtiveram um tempo maior de 119%, 45%, 586% e 34% respectivamente.

Tabela 4: Tempo médio de CPU

Prob.	n	ILS-RVND-BFD	Salhi e Petch (2007)	Taillard *et al.* (1995)	Brandão e Mercer (1998)	Petch e Salhi (2003)	Olivera e Viera (2007)
CMT-1	50	257	175	300	150	1,188	784
CMT-2	75	550	330	420	300	3,630	1,421
CMT-3	100	976	770	1,440	600	9,108	1,323
CMT-4	150	3,564	2,271	3,060	1,500	10,824	3,332
CMT-5	199	7,310	5,319	3,960	3,750	26,994	6,125
CMT-11	120	2,971	12,448	2,700	1,500	26,730	1,372
CMT-12	100	835	498	1,380	600	1,320	1,323
F-11	71	497	1,026	1,560	150	2,838	637
F-12	134	2,425	6,426	4,500	4,800	8,910	1,519
Média		2,154	3,251	2,147	1,483	10,171	1,982

7 Conclusões e trabalhos futuros

O algoritmo proposto baseado na metaheurística *Iterated Local Search* integrado com *Random Variable Neighborhood Descent* junto com o *Best Fit Decreasing* mostrou resultados eficientes e robustos para solucionar o PRVMV. Para o conjunto de instâncias testadas, o algoritmo

ILS-RVND-BFD mostrou um melhor desempenho, encontrando uma nova solução para PRVMV e o GAP foi reduzido em 39% com relação ao período de planejamento T_1 e 31% em comparação a T_2. Com relação a soluções não viáveis, o algoritmo ILS-RVND-BFD apresentou uma redução na média da diferença de 11.71%, 39.53% e 67.14% com relação *Overtime* em comparação com os algoritmos de Olivera e Viera (2007), Brandão e Mercer (1998) e Taillard *et al*. (1995) respectivamente.

Como trabalhos futuros, pretende-se estender o algoritmo proposto para tratar outras variantes do PRVMV tal como o PRV com Múltiplas Viagens e Janela de Tempo.

Referências

Brandão, J. e Mercer, A. (1998), The multi-trip vehicle routing problem. *The Journal of the Operational Research Society*, v. 49, n. 8, p. 799–805.

Christofides, N., Mingozzi, A. e Toth, P. (1979), The vehicle routing problem. en combinatorial optimization. *N. Christofides, A. Mingozzi, P. Toth, C. Sandi, editores*, p. 315-338.

Clarke, G. e Wright, J. (1964), Scheduling of vehicles for a central depot to a number of delivery points. *Operations Research*, v. 12, p. 568–581.

Dongarra, J. J. (2011), Performance of various computers using standard linear equations software. *Computer Science Mathematics Division*.

Fischer, M. (1994), Optimal solution of vehicle routing problems using minimum k–trees. *Operations Research*, v. 42, p. 626-642.

Fleischmann, B. (1990), The vehicle routing problem with multiple use of vehicles. Working paper, Fachbereich Wirtschaftswissenschaften, Universitat Hamburg.

Hommes, P., Howe, E. e Pape, C. Burton's biscuits: a study of the load planning operation atthe new depot at risley. Dissertação de mestrado, Lancaster University, 1989.

Lourenço, H. R., Martin, O. C. e Stützle, T. *Handbook of Metaheuristics*, Capítulo Iterated Local Search, p. 321–353. Kluwer Academic Publishers, Norwell, MA, 2002.

Mladenovic, N. e Hansen, P. (1997), Variable neighborhood search. *Computers & Operations Research*, v. 24, n. 11, p. 1097 – 1100.

Olivera, A. e Viera, O. (2007), Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, v. 34, n. 1, p. 28 – 47.

Penna, P., Subramanian, A. e Ochi, L. (2011), An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, p. 1–32.

Petch, R. e Salhi, S. (2003), A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, v. 133, p. 69 – 92.

Rochat, Y. e Taillard, . D. (1995), Probabilistic intensification and diversification in local search for vehicle routing. *Journal of Heuristics*, p. 147–167.

Salhi, S. e Petch, R. (2007), A ga based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms*, v. 6, p. 591–613.

Taillard, E. D., Laporte, G. e Gendreau, M. (1995), Vehicle routing with multiple use of vehicles. *The Journal Of The Operational Research Society*, v. 47, n. 8, p. 1065–1070.

Zhao, Q., Wang, S., Lai, K. e Xia, G. (2002), A vehicle routing problem with multiple use of vehicles. *Advanced Modeling and Optimization*, v. 4, p. 21–40.