METAHEURÍSTICA GRASP E VNS APLICADA AO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM BACKHAULS E FROTA HETEROGÊNEA FIXA

Marcelus Xavier Oliveira[1], Marcone Jamilson Freitas Souza[2], Sérgio Ricardo de Souza[1], Dayanne Gouveia Coelho[1], Puca Huachi Vaz Penna[3]

[1]Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Av. Amazonas, 7675, CEP 30510-000, Belo Horizonte - MG, Brasil

[2]Departamento de Computação – Universidade Federal de Ouro Preto (UFOP)
Campus Universitário, Morro do Cruzeiro, CEP 35.400-000, Ouro Preto - MG, Brasil

[3]Instituto do Noroeste Fluminense de Educação Superior – Universidade Federal Fluminense (UFF)
Santo Antônio de Pádua, Rio de Janeiro, Brasil

marcelusxavier@gmail.com, marcone@iceb.ufop.br, sergio@dppg.cefetmg.br

dayagc@gmail.com, ppenna@ic.uff.br

Resumo. *Este artigo apresenta uma solução do Problema de Roteamento de Veículos com Backhauls e Frota Heterogênea Fixa (PRVBFHF) via metaheurísticas GRASP e VNS. Neste problema, os clientes são divididos em clientes de entrega e clientes de coleta. Os clientes de entrega recebem produtos vindos do depósito, enquanto os clientes de coleta são os em que se coleta produtos para serem transportados até o depósito. O problema consiste em atender a todos os clientes, iniciando-se pelos de entrega, seguidos pelos de coleta, respeitando-se as capacidades dos veículos que os atendem. Para resolvê-lo, combina-se GRASP com RVND e o procedimento PFIH de construção de rotas. O VNS é usado como estratégia de refinamento. O algoritmo foi avaliado a partir de experimentos computacionais em um conjunto de instâncias da literatura e os resultados encontrados validam sua eficiência.*
PALAVRAS-CHAVE: *Problema de Roteamento de Veículos com Bakhauls e Frota Heterogênea Fixa, GRASP, PFIH, VNS.*

Abstract. *This paper presents a solution to the Heterogeneous Fixed Fleet Vehicle Routing Problem with Backhauls (HFFVRPB) using GRASP and VNS metaheuristics. In this problem, customers are divided into linehauls and backhauls customers. Linehauls customers receive products coming from the depot, while in backhauls clients products are collected and transported to the depot. The problem consists in to serve all customers, starting with the linehauls customers and, in the sequel, by the backhauls customers, respecting the vehicle capacities that serve them. The GRASP and RVND are combined for solving the problem, with the PFIH procedure for route construction. The VNS metaheuristic was used to refine a solution. The proposed algorithm was evaluated from computational experiments with a set of instances from the literature and the results have showed its effectiveness.*
KEYWORDS: *Heterogeneous Fixed Fleet Vehicle Routing Problem with backhauls, HFFVRPB, GRASP, PFIH, VNS.*

1 Introdução

No presente trabalho, estuda-se uma extensão do Problema de Roteamento de Veículos (PRV) clássico, conhecida como Problema de Roteamento de Veículos com *Backhauls* e Frota Heterogênea Fixa (PRVBFHF). Neste problema, o conjunto de clientes é particionado em dois subconjuntos, caracterizados por coleta e entrega de produtos. Os clientes *Linehaul* (de entrega) são aqueles que recebem uma determinada quantidade de produto oriunda de um único depósito. Os clientes *Backhaul* (de coleta) são aqueles que possuem mercadorias com destino ao depósito. A restrição principal do PRVBFHF está relacionada à obrigatoriedade de que, em cada rota, todos os clientes de entrega devem ser atendidos antes de se atender um cliente de coleta. Em adicional, os veículos utilizados possuem capacidades distintas e a quantidade de veículos é limitada.

Vários trabalhos encontrados na literatura estudam e desenvolvem métodos para resolver o PRVB. Os trabalhos de Casco et al. (1988), Deif e Bodin (1984) e Goeschalckx e Jacobs-Blecha (1989), por exemplo, propõem métodos heurísticos, baseados no algoritmo desenvolvido por Clarke e Wright (1963), para resolver este problema. Casco et al. (1988) resolvem o PRVB criando, primeiramente, rotas *linehauls* pelo método de *savings*, para, em seguida, inserir rotas *backhauls*. Deif e Bodin (1984) propõem uma heurística para resolver o PRVB, que é uma extensão da heurística de Clarke e Wright (1963), em que os clientes *backhauls* só são visitados quando todas as rotas dos clientes *linehauls* forem traçadas. Dessa forma, as rotas são divididas nos conjuntos dos clientes *backhauls* e dos clientes *linehauls* e organizadas separadamente. O algoritmo desenvolvido por Goeschalckx e Jacobs-Blecha (1989) contém duas fases: na primeira, é gerada uma solução inicial viável e, na segunda, é feita uma melhora na solução inicial gerada por meio de heurísticas de busca local.

Toth e Vigo (1999) propõem uma heurística que utiliza a informação dada pelo relaxamento lagrangeano para obter limites inferiores e resolver o PRVB por algoritmos exatos. No trabalho de Osman e Wassan (2002), é utilizado uma metaheurística Busca Tabu, que alcança soluções melhores que em Toth e Vigo (1999), mas com um custo computacional maior. Brandão (2006) apresenta um novo algoritmo Busca Tabu para o PRVB, capaz de encontrar as melhores soluções através de uma versão em que obtém soluções iniciais de pseudo-limites inferiores. Gajpal e Abad (2009) utilizam o algoritmo Colônia de Formigas para resolver o PRVB. Nesse trabalho, utiliza-se formigas artificiais para construir uma solução com base em informações obtidas a partir de soluções geradas anteriormente. Zachariadis e Kiranoudis (2011) propõem uma metaheurística para o Problema de Roteamento de Veículo com *Backhaul* baseada em busca local.

São poucos os trabalhos encontrados na literatura que tratam do Problema de Roteamento de Veículos com *Backhaul* e Frota Heterogênea Fixa (PRVBFHF). Assim, Tütüncü (2010) gerou um conjunto de instâncias para este problema, com o intuito de validar o algoritmo por ele proposto. Essas instâncias combinam dois conjuntos de instâncias da literatura associadas ao Problema de Roteamento de Veículos com Frota Heterôgena Fixa (PRVFHF) e o Problema de Roteamento de Veículos com *Backhauls* (PRVB). O algoritmo proposto por Tütüncü (2010) para resolver o PRVFHF e uma extensão deste problema, chamada de Problema de Roteamento de Veículos *Backhaul* com Frota Heterogêna Fixa (PRVBFHF), utiliza um procedimento de programação de pesquisa denominado GRAMPS, que opera dentro de um sistema de apoio à decisão. O autor utiliza um sistema semelhante ao utilizado em Tütüncü et al. (2009), com modificações na interface gráfica

do usuário. Os resultados encontrados mostram que a abordagem visual por ele proposta pode obter bons resultados em um tempo razoável.

O presente artigo apresenta uma proposta para a resolução do PRVBFHF utilizando uma adaptação do algoritmo VND utilizando o algoritmo GRASP, para construir uma solução, e o algoritmo VNRD como busca local. O restante deste artigo está estruturado como segue: a próxima seção descreve o Problema de Roteamento de Veículo *Backhauls* com Frota Heterogênea apresentando as definições e restrições de interesse deste trabalho; a Seção 3 apresenta o algoritmo proposto; a Seção 4 apresenta os resultados computacionais alcançados pelo algoritmo desenvolvido aplicado a instâncias da literatura e, por fim, a Seção 5 apresenta as conclusões obtidas a respeito do desenvolvimento realizado nesta pesquisa.

2 Problema de Roteamento de Veículos com *Backhauls* e Frota Heterogênea

O Problema de Roteamento de Veículos com *Backhauls* (PRVB) consiste em determinar um conjunto de rotas, com o menor custo, que atenda primeiramente, em cada rota, todos os clientes *linehauls*, e posteriormente, os clientes *backhauls*. A Figura 1 apresenta um exemplo de rota para o PRVB.

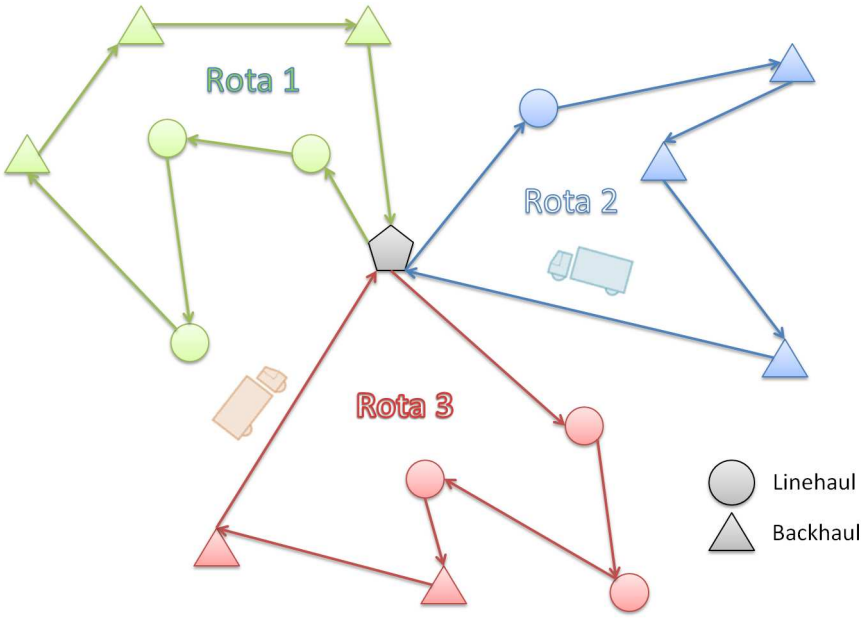

Figura 1. Exemplo de uma rota para o PRVB. O veículo indica o sentido da rota. Observe que clientes *linehaul* são atendidos primeiramente e, só em seguida deve-se atender os clientes *backhaul*.

O PRVB pode ser formulado por meio de um grafo completo, em que cada vértice representa um cliente. Neste trabalho, o PRVB é definido da seguinte forma: seja $G = (V, E)$ um grafo não direcionado, em que V é o conjunto de vértices e $E = \{(i, j)/i, j \in V\}$ é o conjunto de arestas. Para cada aresta (i, j), é associado um custo (distância) positivo c_{ij} para cada $i, j \in V$, tal que $i \neq j$ e $c_{ii} = +\infty$ para cada $i \in V$. Os clientes *linehaul* e *backhaul* são representados, respectivamente, pelos subconjuntos $L = \{1, 2, \cdots, N\}$ e $B = \{N + 1, N + 2, \cdots, N + M\}$, sendo $N + M$ o número total de clientes. O depósito é representado pelo vértice 0. Desse modo, $V = \{0\} \cup L \cup B$. A frota consiste em K veículos idênticos, de mesma capacidade Q. Cada cliente $i \in L \cup B$

requer uma determinada demanda q_i para ser entregue ($i \in L$) ou para ser coletada ($i \in B$). A quantidade máxima de veículos é dada por $k \geq \max\{K_L, K_B\}$, em que K_L e K_B são, respectivamente, o número mínimo de veículos necessários para atender todos os clientes *linehauls* e *backhauls*. Dessa forma, resolver o PRVB consiste em encontrar um conjunto de rotas, com início e fim no depósito, com um custo total mínimo, que atenda às exigências de demanda e coleta dos clientes, e, além disso, respeitando a capacidade dos veículos. O custo é calculado pela soma dos custos das arestas da solução, respeitando as seguintes restrições:

(i) cada veículo realiza apenas um percurso, ou seja, percorre apenas uma rota;
(ii) cada rota começa e termina no depósito;
(iii) em uma rota, a demanda total dos clientes *linehaul* e *backhaul*, separadamente, não pode exceder a capacidade do veículo;
(iv) os clientes *linehaul* devem ser atendidos antes dos clientes *backhaul* (restrição de precedência) e nenhuma rota pode ser formada apenas por clientes *backhaul*. Assim, o custo $c_{ji} = c_{0j} = Y$, para cada $j \in B$ e cada $i \in L$, sendo Y uma constante de valor alto;
(v) cada cliente é visitado apenas uma vez por um dos veículos, e, nesta única visita, sua demanda deve ser atendida.

Neste trabalho, no entanto, é tratado o Problema de Roteamento de Veículos com *Backhaul* e Frota Heterogênea Fixa(PRVBFHF), que se diferencia do PRVB descrito acima por ter o acréscimo de restrições do Problema de Roteamento de Veículos com Frota Heterogênea (PRVFH). Assim, para a formulação do PRVBFHF, são adicionadas, às restrições do PRVB, restrições sobre a frota, que passa a ser composta por m diferentes tipos de veículos, com $M = \{1, \ldots, m\}$. Para cada tipo $k \in M$, existem m_k veículos disponíveis, cada um com uma capacidade Q_k. Ou seja, duas fortes distinções são presentes, quais sejam, a heterogeneidade da frota de veículos, bem como a limitação tanto do número total dos mesmos quanto do número de veículos de cada tipo existente na frota.

3 Algoritmo Proposto

Nesta seção, são descritas a representação de uma solução; a metodologia de construção da solução inicial para o PRVBFHF; a função de avaliação utilizada; a estrutura de vizinhança para explorar o espaço de soluções do problema; e os algoritmos propostos para solucionar o PRVBFHF.

3.1 Representação de uma solução

Cada solução do PRVBFHF pode ser representada por um vetor $s = [r_1, r_2, \cdots, r_n]$, que constitui um conjunto de n rotas. Para cada posição deste vetor, ou seja, para cada rota r_i, são atribuídos um veículo, com suas devidas características, e um conjunto de clientes $[C_1, \cdots, C_m]$ que devem ser atendidos.

3.2 Estruturas de Vizinhança

A vizinhança de uma solução s é o conjunto de soluções $N(s)$, em que cada solução $s' \in N(s)$ é obtida a partir de um movimento feito na solução corrente s. Para melhor explorar o espaço de soluções do problema, foram utilizadas seis tipos de movimentos:

(i) *Shift(1,0)*: consiste na transferência de um cliente de uma rota para outra rota;

(ii) *Shift(2,0)*: semelhante ao *Shift(1,0)*, porém consiste na realocação de dois clientes de uma mesma rota para outra rota;
(iii) *Swap(1,1)*: consiste na troca de um cliente i de uma rota r_a com um cliente j de uma outra rota r_b;
(iv) *Swap(2,1)*: consiste na troca de dois clientes i e t de uma rota r_a com um cliente j de uma outra rota r_b;
(v) *Swap*: consiste em trocar dois clientes de uma mesma rota;
(vi) *Reinsertion*: consiste em remover um cliente i de uma rota e reinseri-lo em uma outra posição da mesma rota;

3.3 Função de Avaliação

O objetivo do PRVB é minimizar o custo total do deslocamento, ou seja, a distância total percorrida. Desse modo, a função de avaliação é dada pela expressão:

$$f(s) = \sum_{k \in P} \sum_{(i,j) \in E} c_{ijk} x_{ijk} \tag{1}$$

que calcula a distância total percorrida, sendo P o conjunto de veículos disponíveis; E o conjunto de arestas (i, j), com $i, j \in V \backslash \{0\}$; c_{ijk} é a distância entre um cliente i e um cliente j vezes o custo para a utilização do veículo k; e x_{ijk} é uma variável de decisão binária, que assume valor 1 a aresta $(i, j) \in E$ é utilizada pelo veículo k; e 0, caso contrário.

3.4 Algoritmo GRASP Proposto

Greedy Randomised Adaptive Serch Procedure, ou apenas GRASP, é um metaheurística proposta por Feo e Resende (1995), que se divide em uma fase de construção, em que a solução é gerada elemento a elemento, e a fase de Busca Local, apta a pesquisar os ótimos locais referentes à solução construída. Assim, a cada iteração do método, suas respectivas fases são executadas e, ao término, um resultado é apresentado. A melhor solução encontrada ao longo de todas as iterações realizadas pelo GRASP é retornada. o pseudocódigo do algoritmo é apresentado no Algoritmo 1.

Algoritmo 1: *GRASP*

```
Entrada: f(.), g(.), N(.), GRASPmax,s
Saída: s
1  início
2      f* ← ∞;
3      para Iter=1 até GRASPmax faça
4          Construção( g(.), α, s);
5          Buscalocal (f(.), N(.), s);
6          se (f(s) < f*) então
7              s* ← s;
8              f* ← f(s);
9          fim
10     fim
11     s ← s*;
12     Retorne s;
13 fim
```

3.4.1 Fase de Construção

Devido à restrição de atendimento prioritário aos clientes de entrega (*Linehaul*) em relação aos clientes de coleta (*Backhaul*) imposta pelo PRVBFHF, os clientes são divididos em duas outras listas separadas pelo tipo, $C_{Linehaul}$ e $C_{Backhaul}$, para evitar a combinação entre tipos de clientes diferentes. Assim, a fase de construção, mostrada no Algoritmo 2, é aplicada a cada uma das listas separadamente. Por fim, para a construção das rotas, a partir das listas de clientes geradas, é utilizada a heurística PFIH, proposta por Solomon (1987). Este procedimento é exposto pela linha 4 do Algoritmo 2.

Algoritmo 2: Construção da Solução

Entrada: $g(.)$, α, $C_{Linehaul}$, $C_{Backhaul}$, veículos K
Saída: s
1 **início**
2 $L_{Linehaul} \leftarrow ConstrucaoLista(g(.), C_{Linehaul}, \alpha, K)$;
3 $L_{Backhaul} \leftarrow ConstrucaoLista(g(.), C_{Backhaul}, \alpha, K)$;
4 $s \leftarrow PFIH(L_{Linehaul}, L_{Backhaul}, K)$
5 Retorne s;
6 **fim**

A construção da lista é vista no Algoritmo 3 e, a cada iteração deste algoritmo, uma lista l é construída, elemento a elemento.

Algoritmo 3: Construção da Lista

Entrada: $g(.)$, C α, K
Saída: s
1 **início**
2 $s \leftarrow \emptyset$;
3 Inicialize o conjunto C de candidatos;
4 **enquanto** $C \neq \emptyset$ **faça**
5 $g(t_{min}) = min\ \{g(t) \mid t \in C\}$;
6 $g(t_{max}) = max\ \{g(t) \mid t \in C\}$;
7 $LRC = \{t \in C \mid g(t) \leq g(t_{min}) + \alpha\,(g(t_{mam}) - g(t_{min}))\}$;
8 Selecione, aleatoriamente, um elemento $t \in LRC$;
9 $l \leftarrow l \cup \{t\}$;
10 Atualize o conjunto C de candidatos;
11 **fim**
12 Retorne l;
13 **fim**

O algoritmo *ConstrucaoLista* consiste em colocar os possíveis candidatos a serem incluídos na solução em uma lista C de candidatos, segundo um critério guloso, que avalia o benefício do candidato ser inserido na solução. Este benefício é dado pela função de avaliação, calculada pela expressão:

$$C_i = -\alpha d_{0i} + \gamma \left[\left(\frac{p_i}{360} \right) d_{0i} \right] \quad \forall i \in C \tag{2}$$

devida a Solomon (1987), sendo:

- α = 0,7 e γ = 0,3: parâmetros definidos empriricamente em Solomon (1987);
- d_{0i} é a distância entre o depósito e o cliente i;
- p_i é o ângulo polar do cliente i em relação ao depósito;

Em seguida, é criada uma Lista Restrita de Candidatos LRC, cujo número de elementos é determinado pelo parâmetro α. A LRC é formada pelos melhores elementos da lista C. Assim, a lista l é construída escolhendo-se, aleatoriamente, um elemento da LRC.

3.4.2 Fase de busca local

O procedimento de busca local da metaheurística GRASP (linha 5 do Algoritmo 1) é feita pelo método *Randomized Variable Neighborhood Descent* - RVND, proposto em Subramanian et al. (2010). Esta heurística é uma variação do método *Variable Neighborhood Descent* – VND, proposto em Mladenovic e Hansen (24). A diferença entre esses algoritmos está na aleatoriedade da ordem das vizinhanças utilizadas para explorar o espaço de soluções. No RVND, a cada chamada ao método, a ordem de escolha das estruturas de vizinhanças disponíveis é alterada aleatoriamente. O pseudocódigo deste procedimento é apresentado no Algoritmo 4.

Algoritmo 4: RVND

```
   Entrada: f(·), N(·), r, s
   Saída: s
1  início
2      Seja r o número total de diferentes estruturas de vizinhança;
3      Seja L_r a lista que contém as diferentes estruturas de vizinhança;
4      L_r ← embaralha(L_r)
5      k ← 1;
6      enquanto k ≤ r faça
7          p ← L_r(k);
8          Encontre o melhor vizinho s' ∈ N^(p)(s);
9          se (f(s') < f(s)) então
10             s ← s';
11             k ← 1;
12             senão
13                 k ← k + 1;
14             fim
15         fim
16     fim
17     Retorne s;
18 fim
```

É importante destacar o funcionamento da lista L_r descrito na linha 3 do Algoritmo 4. A lista L_r é composta pelas estruturas de vizinhanças disponíveis, por exemplo, $L_r = \{1, 2, 3, \cdots, k\}$. Na linha 4 do RNVD, esta lista é embaralhada ($L_r = \{3, 1, 2, \cdots, k\}$) e utilizada na linha 7, em que, dado o valor de k, atribui-se a p o conteúdo da posição k na lista L_r.

3.5 Algoritmo VNS Proposto

O algoritmo *Variable Neighborhood Search (VNS)*, proposto por Mladenovic e Hansen (24), consiste em um método de busca local que visa explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança na solução corrente, afim de descobrir um vizinho do espaço de busca o qual seja promissor para que se encontre melhores resultados. Diferentemente de outras metaheurísticas baseadas em busca local, o VNS não percorre uma trajetória, mas, diferentemente, o método explora, gradativamente, vizinhanças mais afastadas da solução corrente como forma de evitar confinamento em ótimos locais. O pseudocódigo do método VNS é apresentado pelo Algoritmo 5.

Algoritmo 5: VNS

```
   Entrada: s_0, IterSemMelhora
1  início
2      Seja s_0 uma solução inicial;
3      Seja r o número de estruturas de vizinhaça do método GeracaoVizinho();
4      s ← s_0;
5      enquanto iter < IterSemMelhora faça
6          k ← 1;
7          enquanto k < r faça
8              s' ← GeracaoVizinho(s, k);
9              s'' ← RVND(s');                    {Busca Local}
10             se f(s'') < f(s) então
11                 s ← s'';
12                 k ← 1;
13                 iter ← 0;
14             fim
15             senão
16                 iter ← iter + 1;
17                 k ← k + 1;
18             fim
19         fim
20     fim
21     Retorne s;
22 fim
```

O VNS parte de uma solução inicial s_0 gerada pelo algoritmo GRASP e, inicialmente, da primeira estrutura de vizinhança $k \leftarrow 1$ gera um vizinho aleatório s' dentro da vizinhança $N^{(k)}(s)$, linha 8. Posteriormente, é aplicado em s' o procedimento de Busca Local RVND, mostrado no Algoritmo 4, gerando um outro vizinho s''. Se s'' for melhor que a solução s corrente, então a busca continua a partir de s'' e da primeira estrutura de vizinhança $N^{(1)}(s)$. Caso contrário, permanece a solução corrente s e a busca prossegue a partir da próxima estrutura de vizinhança $N^{(k+1)}(s)$. O algoritmo termina quando a condição de parada, que é o número de iterações sem melhora, for satisfeita.

3.5.1 Mecanismos de Geração de um Vizinho

Para gerar um vizinho na solução corrente s, o VNS seleciona, através do parâmetro $k = 1, 2$, um dos dois mecanismos a seguir e aplica n vezes o movimento selecionado. O valor de n é definido aleatoriamente entre 1 a 4, inclusive. Os procedimentos são:

- **Para** $k = 1 \rightarrow$ **Múltiplos Shift(1,0):** Realiza n movimentos *Shift(1,0)* (descrito na seção 3.2) sucessivamente a cada chamada desse mecanismo.
- **Para** $k = 2 \rightarrow$ **Múltiplos Swap(1,1):** Realiza n movimentos *Swap(1,1)* (descrito na seção 3.2) sucessivamente;

4 Resultados Computacionais

Os Algoritmos GRASP e VNS foram desenvolvidos utilizando-se a plataforma Java 7. Os experimentos foram feitos em um PC Intel Core 2 Duo 2.2 GHz, com 4 GB de memória RAM, em ambiente Windows 7.

Para realizar os experimentos computacionais, foi utilizado o conjunto de problemas-teste proposto em Tütüncü (2010). Esse autor gerou o conjunto de problemas-teste do PRVBFHF utilizando dois conjuntos de instâncias de versões distintas do PRV. O

primeiro conjunto corresponde a instâncias do Problema de Roteamento de Veículos com Frota Heterogênea e Fixa (PRVFHF), propostas em Taillard (1996), enquanto o segundo corresponde a instâncias geradas por Toth e Vigo (1999) para o PVRB. A Tabela 1 apresenta as características das instâncias obtidas a partir da combinação desses dois conjuntos de instâncias. As colunas 3, 4 e 5 representam, respectivamente, o número total de clientes n, a quantidade de clientes *linehauls* L e a quantidade de clientes *backhauls* B para cada instância. As colunas 6 a 20 apresentam, para cada tipo m de veículo (A, B, C, DeE), os valores Q e c, que indicam, respectivamente, a capacidade e o custo variável associados ao veículo k e a quantidade de veículos disponíveis desse tipo, como nas instâncias de Taillard (1996).

Tabela 1. Problemas-teste de (Tütüncü,2010)

Problema	Instância PRVB	n	L	B	Tipos de Veículos														
					A			B			C			D			E		
					Q	c	K	Q	c	K	Q	c	K	Q	c	K	Q	c	K
PRVBFHF1	eil22.62	50	25	25	50	1.0	2	100	1.6	2	160	2.0	1						
PRVBFHF2	eil22.63	50	34	16	50	1.0	2	100	1.6	3	160	2.0	1						
PRVBFHF3	eil22.64	50	40	10	50	1.0	2	100	1.6	3	160	2.0	1						
PRVBFHF4	eil22.62	50	25	25	40	1.0	1	80	1.6	2	140	2.1	2						
PRVBFHF5	eil22.63	50	34	16	40	1.0	1	80	1.6	3	140	2.1	2						
PRVBFHF6	eil22.64	50	40	10	40	1.0	2	80	1.6	3	140	2.1	2						
PRVBFHF7	eil22.65	75	37	38	50	1.0	2	120	1.2	1	200	1.5	1	350	1.8	1			
PRVBFHF8	eil22.66	75	50	25	50	1.0	2	120	1.2	2	200	1.5	1	350	1.8	1			
PRVBFHF9	eil22.67	75	60	15	50	1.0	3	120	1.2	2	200	1.5	2	350	1.8	1			
PRVBFHF10	eil22.65	75	37	38	20	1.0	3	50	1.3	3	100	1.9	2	150	2.4	2	250	2.9	1
PRVBFHF11	eil22.66	75	50	25	20	1.0	4	50	1.3	3	100	1.9	2	150	2.4	2	250	2.9	1
PRVBFHF12	eil22.67	75	60	15	20	1.0	4	50	1.3	4	100	1.9	2	150	2.4	2	250	2.9	1
PRVBFHF13	eil22.77	100	50	50	100	1.0	1	200	1.4	2	300	1.7	1						
PRVBFHF14	eil22.78	100	67	33	100	1.0	3	200	1.4	2	300	1.7	1						
PRVBFHF15	eil22.79	100	80	20	100	1.0	3	200	1.4	3	300	1.7	1						
PRVBFHF16	eil22.77	100	50	50	100	1.0	2	140	1.7	2	200	2.0	2						
PRVBFHF17	eil22.78	100	67	33	100	1.0	4	140	1.7	3	200	2.0	2						
PRVBFHF18	eil22.79	100	80	20	100	1.0	4	140	1.7	4	200	2.0	2						

Os algoritmos propostos foram executado 30 vezes para cada instância. O GRASP foi executado com valores de $\alpha = 0,2$, ou 20% de aleatoriedade, e, como critério de parada, utilizou-se o número de iterações igual a 50. O critério de parada do VNS foi o número de iterações sem melhora, dado por $IterSemMelhora = 50$.

A Tabela 2 apresenta os resultados obtidos pela aplicação dos algoritmos propostos na resolução do Problema de Roteamento de Veículos com *Backhauls* e Frota Heterogênea Fixa, utilizando-se as instâncias descritas na Tabela 1. A primeira coluna mostra a instância usada, as colunas 2 e 3, 4 e 5, apresentam os resultados encontrados para as 30 execuções do algoritmo GRASP e VNS, respectivamente, na forma de melhor resultado e média dos resultados. Deve-se ressaltar que o VNS inicia a partir da solução gerada pelo GRASP. As colunas 4 e 7, mostram o tempo médio de cada execução por método. Na última coluna, é ilustrado o tempo médio total, representando a soma dos tempos de ambos os algoritmos, em segundos.

Nas instâncias para as quais não foram apresentados resultados computacionais, o algoritmo proposto não foi capaz de encontrar soluções factíveis pois a demanda total dos clientes *linehaul* ultrapassa a capacidade total dos veículos, violando-se, assim, a restrição de capacidade dos veículos disponíveis. Contudo, como em Tütüncü (2010), fonte

Tabela 2. Resultados obtidos pelo VNS-GRASP

Problema	Solução GRASP			Solução VNS		
	Melhor	Média	Tempo Médio (seg)	Melhor	Média	Tempo Médio (seg)
PRVBFHF1	983,02	1060,66	9,711	880	973,97	14,74
PRVBFHF2	991,59	1067,43	11,565	951,04	994,41	12,37
PRVBFHF3	–	–	–	–	–	–
PRVBFHF4	1140,93	1248,58	8,884	1062,58	1130,77	13,08
PRVBFHF5	1143,69	1207,51	10,801	1069,67	1123,04	12,12
PRVBFHF6	–	–	–	–	–	–
PRVBFHF7	1298,01	1344,00	23,27	1184,84	1264,53	51,07
PRVBFHF8	–	–	–	–	–	–
PRVBFHF9	1237,34	1289,78	32,62	1108,74	1186,70	47,83
PRVBFHF10	1788,13	1949,10	30,03	1578,3	1749,83	38,29
PRVBFHF11	1852,29	1986,86	33,78	1642,12	1766,96	35,40
PRVBFHF12	–	–	–	–	–	–
PRVBFHF13	1428,94	1471,21	81,48	1312,29	1400,26	146,37
PRVBFHF14	–	–	–	–	–	–
PRVBFHF15	1484,83	1614,27	15,56	1341,24	1484,49	86,50
PRVBFHF16	1681,66	1743,47	79,38	1447,72	1541,91	127,85
PRVBFHF17	1620,81	1695,69	92,54	1399,48	1479,91	114,99
PRVBFHF18	1672,67	1783,37	124,29	1433,55	1559,19	126,48

Tabela 3. Comparação dos resultados obtidos para o PRVBFHF

Problema	(Tütüncu, 2010)			$GAP(\%)$
	GRAMPS	ADVISER	VNS	
PRVBFHF1	1111,67	1056,44	**880**	-16,70
PRVBFHF2	1067,28	982,86	**951,04**	-3,24
PRVBFHF3	1124,14	998,22	–	–
PRVBFHF4	1094,08	1070,06	**1062,58**	-0,70
PRVBFHF5	1135,21	1127,97	**1069,67**	-5,17
PRVBFHF6	1200,58	1183,36	–	–
PRVBFHF7	1190,63	1190,63	**1184,84**	-0,49
PRVBFHF8	1211,28	1182,66	–	–
PRVBFHF9	1222,66	1203,09	**1108,74**	-7,84
PRVBFHF10	1845,75	1781,50	**1578,3**	-11,41
PRVBFHF11	2035,39	1941,74	**1642,12**	-15,43
PRVBFHF12	1945,35	1917,54	–	–
PRVBFHF13	1228,24	1227,81	1312,29	6,88
PRVBFHF14	1136,87	1109,02	–	–
PRVBFHF15	1228,56	1216,65	1341,24	10,24
PRVBFHF16	1629,47	1555,35	**1447,72**	-6,92
PRVBFHF17	1609,03	1585,30	**1339,48**	-11,72
PRVBFHF18	1618,27	1615,08	**1433,55**	-11,24

original das instâncias utilizadas no presente artigo, são apresentados resultados para essas instâncias, ou a Tabela 1 apresenta algum equívoco em relação aos dados dos veículos disponíveis ou os resultados ali postos não estão corretos.

Na Tabela 3, as colunas 2 e 3 apresentam, respectivamente, os melhores resultados obtidos pelos procedimentos GRAMPS e ADVISER, listados em Tütüncü (2010). A coluna 4 apresenta a melhor solução encontrada pelo VNS, utilizando o GRASP para a construção da solução inicial, e a última coluna mostra o *gap* % entre as soluções dadas pelo algoritmo ADVISER e as obtidas pelo algoritmo proposto, calculado na forma:

$$gap\% = \frac{MelhorVNS_i - MelhorLit_i}{MelhorLit_i} \times 100\%$$

sendo $MelhorVNS_i$ o resultado encontrado pela aplicação do algoritmo proposto e $MelhorLit_i$ o resultado obtido pelo algoritmo ADVISER, ambos aplicado à instância i.

De acordo com a Tabela 2, é possível notar que o VNS fornece boas soluções para o problema em um tempo computacional baixo. Para as instâncias em que a solução está em negrito na coluna 4 da Tabela 3, o VNS obteve soluções melhores que as obtidas pelo GRAMPS e pelo ADVISER. Assim, foram encontrados, para 11 instâncias, resultados melhores que os disponíves em Tütüncü (2010). Para estas 13 instâncias, o procedimento GRAMPS encontrou uma distância total de $18016,20$; o algoritmo ADVISER apresenta um valor total de $17554,50$; já o algoritmo VNS proposto encontra 16352 como custo total. Ou seja, o valor total encontrado pelo VNS proposto é 6,85% inferior ao melhor valor total encontrado na literatura ou 9,24% melhor que o valor encontrado para o custo total vinculado à aplicação do procedimento GRAMPS.

5 Conclusões

Este trabalho tratou o Problema de Roteamento de Veículos com *Backhauls* e Frota Heterogênea Fixa por meio dos algoritmos GRASP e VNS. Na fase de construção do GRASP, uma solução do problema é gerada utilizando o método PFIH e, na fase de busca local, é aplicada a heurística de refinamento RVND. O VNS segue precisamente as descrições proposta pelo autor, com a pertubação aplicando múltiplos movimentos *Shift(1,0)* e *Swap(1,1)*.

Para avaliar o algoritmo proposto, foram utilizados problemas-teste da literatura, e os resultados comparados com o trabalho de Tütüncü (2010).

Os resultados obtidos se mostraram promissores, visto que, das 18 instâncias apresentadas em Tütüncü (2010), 5 violam a restrição de capacidade do PRVBFHF (e, portanto, não possuem solução factível) e, nas 13 restantes, foram obtidos soluções factíveis de boa qualidade. Destas 13 restantes, o algoritmo VNS proposto se mostrou superior ao de Tütüncü (2010) em 11 instâncias, ou seja, valor total encontrado pelo VNS proposto é 6,85% inferior ao melhor valor total encontrado na literatura.

Por fim, a qualidade alcançada pelos resultados mostra que o VNS, utilizando a construção gerada pelo GRASP, obteve sucesso ao resolver o problema do PRVBFHF.

Em trabalhos futuros, pretende-se testar outras estruturas de vizinhança e outras heurísticas de refinamento dentro do algoritmo VNS, além de outros mecanismos de pertubação para o mesmo.

Agradecimentos

Os autores agradecem ao CEFET-MG, à CAPES, à FAPEMIG e ao CNPq pelo apoio ao desenvolvimento do trabalho.

Referências

Brandão, J. (2006). A new tabu search algorithm for the vehicle routing problem with backhaul. *European Journal of Operational Research*, v. 173, p. 540–555.

Casco, D; Golden, B. L. e Wasil, E. A. (1988). Vehicle routing with backhauls: Models, algorithms, and case studies. Golden, B. L e Assad, A A., editors, *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in Management Science and Systems*, p. 127–147. North-Holland, Amsterdam.

Clarke, G. e Wright, J. W. (1963). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, v. 11, p. 568–581.

Deif, I. e Bodin, L. (1984). Extension of the clarke and wright algorithm for solving the vehicle routing problem with backhauling. Kidder, A., editor, *Proceedings of the Banson Conference on Software Uses in Transportation and Logistic Management*, p. 75–96, (1984).

Feo, Thomas A. e Resende, Maurício G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 9, p. 849–859.

Gajpal, Yuvraj e Abad, P. L. (2009). Multi-ant colony system (macs) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 196, p. 102–117.

Goeschalckx, M e Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *Eur*, v. 42, p. 39–51.

Mladenovic, N. e Hansen, P. (24). Variable neighborhood search. *Computers and Operations Research*, v. 1097-1100, p. 1997.

Osman, I. H. e Wassan, N. A. (2002). A reactive tabu search meta-heuristic for the vehicle routing problem with backhauls. *Journal of Scheduling*, v. 5, p. 263–285.

Solomon, Marius M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, v. 35, n. 2, p. 254–265.

Subramanian, A.; Drummond, L. M. A.; Bentes, C.; Ochi, L. S. e Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, v. 37, p. 1899–1911.

Taillard, E. D. *A Heuristic Column Generation Method for the Heterogeneous Fleet VRP*. Publication CRT-03-96, Universite Montreal., (1996).

Toth, Paolo e Vigo, Daniele. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing prolems with backhauls. *European Journal of Operational Research*, v. 113, p. 528–543.

Tütüncü, G. Yazgi. (2010). An interactive gramps algorithm for the heterogeneous fixed fleet vehicle routing problem with and without backhauls. *European Journal of Operational Research*, v. 201, p. 593–600.

Tütüncü, G. Yazgi; Carreto, Carlos A. C. e Baker, Barrie M. (2009). A visual interactive approach to classical and mixed vehicle routing problems with backhauls. *Omega - The International Journal of Management Science*, v. 37, p. 138–154.

Zachariadis, Emmanouil E. e Kiranoudis, Chris T. (2011). A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. *Expert Systems with Applications*, v. 38, n. 3, p. 2717 – 2726. ISSN 0957-4174.