Universidade Federal Fluminense

IGOR MACHADO COELHO

Contribuições para o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas

NITERÓI-RJ

2011

Universidade Federal Fluminense

IGOR MACHADO COELHO

Contribuições para o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Algoritmos e Otimização.

Orientador:
Prof. Luiz Satoru Ochi, D.Sc.

Co-Orientador:
Prof. Marcone Jamilson Freitas Souza, D.Sc.

NITERÓI-RJ

2011

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

C672 Coelho, Igor Machado.
Contribuições para o problema de roteamento de veículos de rota única com entregas obrigatórias e coletas seletivas / Igor Machado Coelho. – Niterói, RJ : [s.n.], 2011.
50 f.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2011.
Orientador: Luiz Satoru Ochi.

1. Algoritmo. 2. Problema de roteamento de veículo. 3. Metaheurística. I. Título.

CDD 005.136

Contribuições para o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas

Igor Machado Coelho

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Algoritmos e Otimização.

Aprovada por:

Prof. Luiz Satoru Ochi, D.Sc. / IC-UFF
(Presidente) / (Orientador)

Prof. Marcone Jamilson Freitas Souza, D.Sc. / DECOM-UFOP

Prof[a]. Isabel Cristina Mello Rosseti, D.Sc. / IC-UFF

Prof. Ricardo Farias, Ph.D. / COPPE-UFRJ

Niterói, 01 de Abril de 2011.

À minha família, aos meus amigos.

Agradecimentos

A meus pais, João e Gessi, sem vocês nada disso seria possível!

A meus irmãos Bruno, Vitor, Mateus e Vinicius, sempre presentes não importa a distância.

À minha namorada Cristiane, por todo o apoio e carinho durante o desenvolvimento deste trabalho.

Ao Mário, Sabir, Pablo e Marcos, pela boa amizade e momentos de muita diversão na OptHouse.

Às meninas do 302 e galera do Edifício Rocha, pela amizade e bons momentos.

Ao Satoru *sensei*, pelos ensinamentos de Pesquisa e da Vida.

Ao Marcone, pela amizade e exemplo de dedicação.

Amigos de Ouro Preto e Mariana, sempre no coração.

A Niterói, pelos momentos de lazer proporcionados.

À CAPES, pela bolsa concedida.

Aos companheiros do IC-UFF e do projeto OptFrame, porque Pesquisa é algo que não se faz sozinho.

Resumo

Este trabalho trata do Contribuições para o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas (PRVRUEOCS). Neste problema entregas devem ser feitas a uma série de clientes, mas também existem itens a serem coletados se possível, dependendo da receita gerada por estas coletas. Aplicações práticas podem ser encontradas em vários contextos de logística reversa em que clientes retornam bens de volta para o depósito, como em logística postal. O PRVRUEOCS pertence à classe NP-difícil, uma vez que ele pode ser reduzido ao clássico Problema do Caixeiro Viajante quando nenhum cliente necessita de serviço de coleta. Para resolvê-lo, propõe-se um algoritmo heurístico híbrido, denominado HGVNS, inspirado na metaheurística *General Variable Neighborhood Search* combinada com uma geração de solução inicial por meio de métodos exatos. O algoritmo proposto foi testado com um conjunto de problemas-teste da literatura e se mostrou eficiente na resolução do problema em questão.

Palavras-chave: Contribuições para o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas, *General Variable Neighborhood Search*.

Abstract

This work addresses the Single Vehicle Routing Problem with Deliveries and Selective Pickups (SVRPDSP). In this problem deliveries have to be made to a set of customers, but there are also pickup goods to be collected depending on the revenue generated by them. Practical applications arise in a number of reverse logistics contexts in which customers return goods back to the depot, as in postal logistics. The SVRPDSP belongs to the NP-hard class, once it can be reduced to the classic Traveling Salesman Problem when there aren't any pickups to be made. To solve this problem, we propose a hybrid heuristic algorithm, named HGVNS, inpired on the metaheuristic General Variable Neighborhood Search combined with an initial solution generation by means of exact methods. The proposed algorithm was tested with a set of instances available in literature and showed to be efficient in the resolution of the problem at hand.

Keywords: Single Vehicle Routing Problem with Deliveries and Selective Pickups, General Variable Neighborhood Search.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

1 Introdução

O Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas (PRVRUEOCS) (em inglês, *Single Vehicle Routing Problem with Deliveries and Selective Pickups* [1], também encontrado na literatura como *Single Vehicle Routing Problem with Unrestricted Backhauls* [2]) é uma variação do Problema de Roteamento de Veículos (PRV) clássico [3].

O PRV clássico pode ser definido da seguinte maneira: dado um conjunto N de clientes, cada qual com uma demanda d_i e uma frota de veículos com capacidade Q, estabelecer os trajetos de custo mínimo a serem percorridos pelos veículos, de forma a atender completamente a demanda dos clientes.

No PRVRUEOCS entregas devem ser efetuadas a uma série de clientes em uma rota única, mas também existem itens a serem coletados se possível, dependendo da receita gerada por estas coletas.

O PRVRUEOCS pertence à classe NP-difícil, uma vez que ele pode ser reduzido ao Problema do Caixeiro Viajante (PCV) [4] quando nenhum cliente necessita de serviço de coleta. Assim, as abordagens mais sugeridas na literatura para problemas de elevada complexidade são baseadas em metaheurísticas como *Iterated Local Search* [5], *Variable Neighborhood Search* [6], GRASP [7] e Busca Tabu [8].

1.1 Motivação

O PRV é amplamente estudado na literatura devido a sua dificuldade de resolução e alta aplicabilidade na área de logística.

O PRVRUEOCS tem muitas aplicações em contextos de logística reversa onde clientes retornam bens de volta para o depósito, como em entrega/coleta de garrafas ou logística

postal. Porém, embora o PRVRUEOCS seja um problema com muitas aplicações potenciais é um problema ainda pouco explorado na literatura comparado ao PRV clássico e outras variantes.

1.2 Objetivos do trabalho

Este trabalho tem como objetivo geral o desenvolvimento de um algoritmo eficiente de otimização, baseado na metaheurística *General Variable Neighborhood Search* (GVNS) [9], para resolver o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas.

1.2.1 Objetivos específicos

São os seguintes os objetivos específicos:

1. Fazer uma revisão de literatura do PRVRUEOCS e de suas técnicas de solução;
2. Desenvolver um algoritmo heurístico híbrido baseado na metaheurística GVNS combinada com uma geração de solução inicial integrada com resolvedores exatos;
3. Avaliar o desempenho do algoritmo desenvolvido com os resultados da literatura.

1.3 Estrutura do trabalho

O presente trabalho está dividido em cinco capítulos, incluindo esta introdução.

O Capítulo 2 apresenta a definição e a literatura existente do problema abordado neste trabalho, o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas. Apresenta-se também uma descrição da metaheurística *General Variable Neighborhood Search*.

No Capítulo 3 é apresentado o algoritmo proposto, denominado HGVNS, para resolver o PRVRUEOCS. O método HGVNS é explicado em detalhes bem como a representação de uma solução do problema, a função de avaliação, as estruturas de vizinhança utilizadas para percorrer o espaço de soluções e os métodos para geração de soluções iniciais.

No Capítulo 4 são apresentados e analisados os resultados computacionais e no Capítulo 5 são apresentadas as conclusões e apontadas as sugestões para trabalhos futuros.

2 Descrição e Literatura do Problema Abordado

O Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas (PRVRUEOCS), ou *Single Vehicle Routing Problem with Deliveries and Selective Pickups* [1], é uma variante do Problema de Roteamento de Veículos (PRV) clássico. Este problema também pode ser encontrado na literatura como Problema de Roteamento de Veículos de Rota Única com *Backhauls* Irrestrito, ou *Single Vehicle Routing Problem with Unrestricted Backhauls* [2].

No PRVRUEOCS entregas devem ser efetuadas a uma série de clientes, mas também existem itens a serem coletados se possível, dependendo da receita gerada por estas coletas. O PRVRUEOCS pertence à classe NP-difícil, uma vez que ele pode ser reduzido ao Problema do Caixeiro Viajante (PCV) quando nenhum cliente necessita de serviço de coleta.

Para ilustrar o PRVRUEOCS, seja a Figura 2.1. Nela, um depósito é representado por um retângulo, enquanto os clientes são representados por círculos. Cada cliente tem demandas de entrega e de coleta, representados pelo número no lado esquerdo e direito de cada círculo, respectivamente. As entregas são obrigatórias, enquanto as coletas são opcionais. Os benefícios por fazer uma coleta são representados por estrelas ao lado de cada cliente. Uma seta ligando o lado esquerdo de um círculo ao lado direito indica que é feita entrega seguida de coleta.

Nessa solução apresentada, o veículo parte do depósito com 170 unidades, percorre uma distância de valor 80 e visita o primeiro cliente entregando 40 unidades. O veículo agora carrega 130 unidades. O segundo cliente é visitado, com a entrega de 30 unidades seguido da coleta de 20 unidades, com um benefício de valor 50. No terceiro cliente pode-se perceber que é feita uma entrega e nenhuma coleta. No quarto e quinto clientes são feitas entregas e coletas conjuntas. Antes de retornar ao depósito o veículo passa no

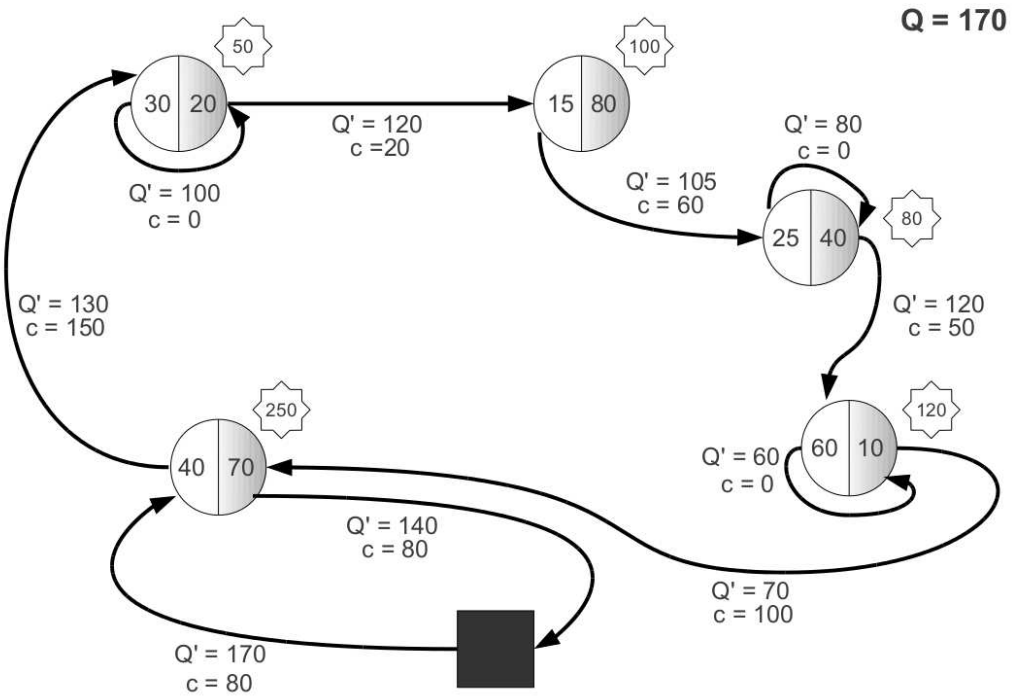

Figura 2.1: O Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas

primeiro cliente novamente para fazer uma coleta de 70 unidades, com um benefício de valor 250.

Desta forma, a distância total percorrida é de: $80+150+0+20+60+0+50+0+100+80=540$. O total de benefícios é de: $50+80+120+250=500$. Logo, o custo dessa solução para o problema é de: $540-500=40$.

Podemos perceber que a coleta no primeiro cliente é muito vantajosa, porém ela não pode ser feita no mesmo momento que a entrega por não haver espaço suficiente no veículo. Vale ressaltar também que o custo final da solução pode ser **negativo**, quando o valor em benefícios coletados for superior à distância total percorrida.

2.1 Literatura

Na literatura, as abordagens mais comuns para o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas (PRVRUEOCS) são por meio de métodos de programação matemática e metaheurísticas.

O PRVRUEOCS foi proposto por [2], onde um modelo de programação matemática

é utilizado para resolver problemas-teste criados pelos autores. Em [1] o problema é novamente abordado, porém com outra nomenclatura. Um novo modelo de programação matemática é proposto para resolver outros problemas-teste criados pelos autores. Visto que o modelo não é capaz de encontrar bons resultados, estratégias heurísticas são desenvolvidas. Basicamente, 12 heurísticas de construção e refinamento são desenvolvidas, e um *gap* médio de $30,62\%$ é obtido pela melhor heurística proposta **CI5+IMP**. A partir daí 12 estratégias inspiradas na metaheurística Busca Tabu são desenvolvidas, tendo resultados diversos segundo as métricas propostas *pgap* e *cgap*. Os resultados médios encontrado pelas estratégias de Busca Tabu são: $pgap$ igual a $0,36\%$ e $cgap$ igual a $4,03\%$. Os resultados são comparados a um limite inferior do problema.

Em [2] é feita uma classificação de problemas com *backhauls* (coleta) utilizando a notação $\alpha/\beta/\gamma$ em que: α denota a quantidade de veículos, β denota o tipo de serviço de coleta e γ especifica se existe precedência de visita entre clientes de entrega/coleta. Desta forma, α pode assumir os valores: 1, para um único veículo e M, para o caso multi-veículos; β pode ser: *livre*, para coletas opcionais e *obrigatório*, para coletas obrigatórias; e γ pode ser: *prec*, quando houver restrição de precedência entre clientes de entrega sobre clientes de coleta; e *qualquer*, quando clientes puderem ser visitados em qualquer ordem. Assim, o PRVRUEOCS se enquadra como $1/livre/qualquer$.

Um problema similar ao PRVRUEOCS muito estudado na literatura é conhecido como Problema de Roteamento de Veículos com *Backhauls* (PRVB). O PRVB é uma extensão do PRV clássico, que considera clientes de entrega (*linehaul*) e de coleta (*backhaul*) e, por restrições operacionais, na rota de cada caminhão as entregas devem preceder as coletas. Cada veículo deve sair do depósito e retornar ao depósito, respeitando uma capacidade máxima estabelecida para toda a frota (frota homogênea), onde nem a soma das entregas e nem das coletas devem exceder essa capacidade. Todos os clientes (*linehaul* e *backhaul*) devem ser visitados, sendo que o problema se reduz ao PRV clássico, quando não existem clientes *backhaul*. Busca-se, então, minimizar os custos totais de roteamento. Existem problemas-teste para o problema criados por [10] e [11]. Abordagens heurísticas para o problema podem ser encontradas em [12], [13] e [14]. Este problema se enquadra como $M/obrigatório/prec$.

O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP) é uma variante do Problema do Caixeiro Viajante clássico. No PCVCP existe um prêmio associado a cada cidade e uma penalidade se a cidade não for visitada. O objetivo é minimizar a distância total percorrida e as penalidades, satisfazendo um valor mínimo em prêmios coletados.

Referências ao PCVCP podem ser encontradas em [15], [16] e [17].

Outro problema similar é o Problema de Roteamento de Veículos com Entrega e Coleta Simultânea. Neste caso, as coletas são obrigatórias e devem ocorrer na mesma visita em que a entrega é feita, diferentemente do PRVB. Referências podem ser encontradas em [18] e [19].

Com relação aos problemas-teste para o PRVRUEOCS, há disponível na literatura apenas o conjunto utilizado por [1]. Tal conjunto consiste de 68 problemas-teste, de 16 a 100 clientes. O conjunto é dividido em 4 classes diferentes, com relação à quantidade de itens a serem coletados.

2.2 Formulações Matemáticas para o PRVRUEOCS

Existem de nosso conhecimento somente duas formulações matemáticas, propostas por [2] e [1], para resolver o problema em questão. Em [2], a formulação trabalha com clientes de entrega e coleta, separadamente.

2.2.1 Formulação de Sural *et al.* (2003)

A formulação matemática apresentada no trabalho de [2] é descrita a seguir.

Suponha que existam n clientes de entrega e m clientes de coleta, $m \leq n$ e $w = n+m$. Seja $G = (N,A)$ um grafo completo orientado. O conjunto de nós $N = \{0\} \cup N_d \cup N_b$, onde $\{0\}$ representa o depósito, $N_d = \{1,...,n\}$ os clientes de entrega, e $N_b = \{n+1,...,w\}$ os clientes de coleta. O conjunto de arcos A representa as ligações entre eles. Um cliente j demanda uma entrega $d_j > 0$ (do depósito), e uma demanda de coleta $b_j > 0$ (para o depósito), mas não ambos. Caso um cliente tenha uma demanda tanto de coleta quanto de entrega, então ele pode ser dividido em dois clientes. Há um custo c_{ij} de transporte entre os arcos $(i,j) \in A$. Satisfazendo uma demanda de coleta de um cliente j recebe-se um benefício r_j associado ao nó $j \in N_b$. A capacidade do veículo é denotada por $W \geq D = \sum_{j=1}^{n} d_j$. Todos os parâmetros são não negativos.

Seja $x_{ij} = 1$ caso o cliente j seja imediatamente precedido pelo cliente i ($i \neq j$) e 0 caso contrário. Quando um veículo saí de um cliente i, ele possui um peso total, denotado por y_i, que ainda deve ser entregue. Analogamente, quando um veículo chega no cliente i, ele possui um peso total, denotado por z_i, que já foi coletado. Assim, o modelo matemático é apresentado pelas Equações (2.1) a (2.7).

$$\textit{Minimizar} \quad \sum_i \sum_j c_{ij} x_{ij} - \sum_i \sum_{j=n+1}^{w} r_j x_{ij} \tag{2.1}$$

$$\textit{sa.} \quad \sum_i x_{ij} = 1 \qquad \forall j = 0,...,n \tag{2.2}$$

$$\sum_j x_{ij} - \sum_k x_{ki} = 0 \qquad \forall i \tag{2.3}$$

$$y_j - y_i + D x_{ij} \leq D - d_j \qquad \forall i, j(\neq 0) \tag{2.4}$$

$$z_i - z_j + W x_{ij} \leq W - b_i \qquad \forall i, j(\neq 0) \tag{2.5}$$

$$y_i + z_i \leq W - b_i - d_i \qquad \forall i(\neq 0) \tag{2.6}$$

$$y_i, z_i \geq 0, x_{ij} \in \{0,1\} \tag{2.7}$$

A Equação (2.2) força cada entrega ser realizada. As restrições (2.3) garantem o fluxo do veículo. As inequações (2.4) e (2.5) são adaptações das conhecidas restrições de eliminação de subrotas MTZ para o Problema do Caixeiro Viajante [20]. As restrições (2.6) garantem que a capacidade do veículo não é ultrapassada, e as Restrições (2.7) são de integralidade e não negatividade.

2.2.2 Formulação de Gribkovskaia *et al.* (2008)

No modelo proposto por [1], apresentado pelas Equações (2.8) a (2.21), cada cliente é representado por dois vértices, i e $i+n$, $i = 1,...,n$. Os vértices de 1 até n são associados com a primeira visita ao cliente. Como nunca é sub-ótimo realizar a entrega primeiro, cada um dos n primeiros vértices são visitados. Vértices $n+1,...,2n$ representam a segunda visita aos clientes em que a coleta é feita.

A formulação utiliza a seguinte notação:

$\bar{c}_{ij}$: custo da rota do vértice i para o vértice j ($i = 0,...,2n$; $j = 0,...,2n$)

$$\bar{c}_{ij} = \begin{cases} c_{ij} & \text{se } i \leq n \text{ e } j \leq n, \\ c_{i-n,j} & \text{se } i > n \text{ e } j \leq n, \\ c_{i,j-n} & \text{se } i \leq n \text{ e } j > n, \\ c_{i-n,j-n} & \text{se } i > n \text{ e } j > n, \end{cases}$$

r_i: é o bônus associado caso haja coleta no cliente i ($i = 1,...,n$)

d_i: demanda de entrega associada ao cliente i ($i = 1,...,n$)

p_i: demanda de coleta associada ao cliente i ($i = 1,...,n$)

Q: capacidade do veículo

$$x_{ij} = \begin{cases} 1 \text{ caso o veículo viaja diretamente de } i \text{ para } j \\ \qquad (i, j = 0, ..., 2n, i \neq j \text{ ; } j \neq i+n \text{ se } 1 \leq i \leq n \text{ ; } j \neq i-n \text{ se } n+1 \leq i \leq 2n) \\ 0 \qquad \text{caso contrário} \end{cases}$$

$$y_i = \begin{cases} 1 \text{ se é feita coleta e entrega simultaneamente no cliente } i \qquad (i = 1, ..., n) \\ 0 \text{ caso contrário} \end{cases}$$

$$z_i = \begin{cases} 1 \text{ se a coleta é feita na segunda visita ao cliente } i \qquad (i = 1, ..., n) \\ 0 \text{ caso contrário} \end{cases}$$

w_i: um limite superior (*upper bound*) da carga do veículo ao sair do vértice i

$$\textit{Minimizar} \quad \sum_{i=0}^{2n}\sum_{j=0}^{2n} c_{ij}x_{ij} - \sum_{i}^{n} r_i y_i - \sum_{j=n+1}^{2n} r_{j-n} - z_{j-n} \tag{2.8}$$

$$sa. \quad \sum_{j=0}^{2n} x_{ij} = 1 \qquad \forall i = 0,...,n \tag{2.9}$$

$$\sum_{i=0}^{2n} x_{ij} = 1 \qquad \forall j = 0,...,n \tag{2.10}$$

$$\sum_{j=0}^{2n} x_{ij} = z_{i-n} \qquad \forall i = n+1,...,2n \tag{2.11}$$

$$\sum_{i=0}^{2n} x_{ij} = z_{j-n} \qquad \forall j = n+1,...,2n \tag{2.12}$$

$$z_i + y_i \leq 1 \qquad i = 1,..n \tag{2.13}$$

$$w_0 = \sum_{i=1}^{n} d_i \tag{2.14}$$

$$0 \leq w_i \leq Q \qquad i = 1,...,2n \tag{2.15}$$

$$w_j \geq w_i - d_j + p_j y_j - (1 - x_{ij})Q \qquad i = 0,...,2n\ ;\ j = 1,...,n \tag{2.16}$$

$$w_j \geq w_i + p_{j-n} z_{j-n} - (1 - x_{ij})Q \qquad i = 0,...,2n \tag{2.17}$$

$$j = n+1,...,2n$$

$$\sum_{(i,j)\in S} x_{ij} \leq |S| - 1 \qquad S \subset \{0,...,2n\} \tag{2.18}$$

$$2 \leq |S| \leq 2n-1$$

$$x_{ij} \in \{0,1\} \qquad i,j = 0,...,2n, i \neq j \tag{2.19}$$

$$i \neq (j+n) \text{ se } 1 \leq i \leq n$$

$$j \neq (i-n) \text{ se } n+1 \leq i \leq 2n$$

$$y_i \in \{0,1\} \qquad i = 1,...,n \tag{2.20}$$

$$z_i \in \{0,1\} \qquad i = 1,...,n \tag{2.21}$$

As restrições (2.9) e (2.10) definem que o depósito e o primeiro vértice associado a cada cliente deve ser visitado somente uma vez. As restrições (2.11) e (2.12) garantem que, se uma coleta é feita em um cliente na segunda visita, então o veículo deve chegar ao cliente e deixá-lo pela segunda vez. Caso contrário, não é preciso visitar um cliente i pela segunda vez. As restrições (2.13) garantem que a coleta só pode ser feita em um cliente apenas uma vez. As restrições (2.14) define que o limite superior (*upper bound*) da carga do veículo ao sair do depósito é igual as demandas dos clientes. A capacidade do veículo é restrita pelas Equações (2.15). As restrições (2.16) e (2.17) definem w_j em termos de w_i sempre que o cliente j é visitado imediatamente após o cliente i. Nas restrições (2.16),

j é o primeiro vértice associado ao cliente j: $w_j \geq w_i - d_j + p_j$, se a coleta e entrega é feita simultaneamente no cliente j, e $w_j \geq w_i - d_j$, se somente a entrega é satisfeita. Do mesmo modo, nas restrições (2.17), $j > n$ é o segundo vértice associado ao cliente j: $w_j \geq w_i + p_{j-n}$ caso for feita uma coleta na segunda visita, e $w_j \geq w_i$ caso a coleta e entrega sejam feitas simultaneamente, ou a coleta é considerada inviável ou não rentável. As Restrições (2.19) apresentam uma clássica eliminação de sub-ciclos. E finalmente, as Restrições (2.19)-(2.21) impõem restrições binárias às variáveis.

2.3 Estimativa de Limite Inferior

Um Limite Inferior, ou *Lower Bound*, pode ser estimado para o problema da seguinte maneira, de acordo com [1].

A função objetivo do PRVRUEOCS inclui dois termos: custo de roteamento e benefícios coletados. Soluções ótimas do Problema do Caixeiro Viajante fornecem *lower bounds* no custo de roteamento. Foi utilizado o resolvedor Concorde[1] para obter *lower bounds* para o custo de roteamento.

Da mesma maneira para a parte de benefícios da função objetivo, a solução ótima para o seguinte Problema da Mochila [21] fornece um limite superior (ou *upper bound*) em relação aos benefícios coletados, em um problema-teste particular. Seja b_i o benefício associado ao fazer uma coleta no cliente i ($i = 1,...,n$); c_i a demanda por coleta no cliente i ($i = 1,...,n$); Q é a capacidade do veículo; v_i é uma variável binária igual a 1 se, e somente se, uma coleta for feita no vértice i ($i = 1,...,n$).

O Problema da Mochila é então:

$$\textit{Maximizar} \qquad \sum_{i=1}^{n} b_i v_i \tag{2.22}$$

$$\textit{sujeito a.} \tag{2.23}$$

$$\sum_{i=1}^{n} c_i v_i \leq Q \tag{2.24}$$

$$v_i \in \{0,1\} \qquad \forall i = 1,...,n \tag{2.25}$$

[1] O resolvedor Concorde para o Problema do Caixeiro Viajante pode ser obtido em: *http://www.tsp.gatech.edu/concorde/ downloads/downloads.htm*

2.4 Heurísticas e Metaheurísticas

Heurísticas são técnicas que visam a obtenção de soluções de boa qualidade em um tempo computacional aceitável. Essas técnicas, no entanto, não garantem a obtenção da solução ótima para o problema nem são capazes de garantir o quão próximo a solução obtida está da ótima.

Metaheurísticas são ideias gerais para resolução de problemas genéricos de otimização. Através delas podem ser projetadas heurísticas específicas para cada problema, em geral tendo a capacidade de escapar das armadilhas dos ótimos locais. Elas podem ser de busca local ou populacional. Na primeira, a exploração do espaço de soluções geralmente é feita por meio de movimentos, os quais são aplicados a cada passo sobre a solução corrente, gerando outra solução promissora em sua vizinhança. Já na segunda, trabalha-se com um conjunto de soluções, recombinando-as com o intuito de aprimorá-las.

Neste trabalho, foi desenvolvido um método híbrido inspirado na metaheurística *General Variable Neighborhood Search* (GVNS), que faz uso do método de busca local *Variable Neighborhood Descent* (VND) [6], descritos nas Seções 2.4.1 e 2.4.2, respectivamente.

2.4.1 *General Variable Neighborhood Search*

A metaheurística *General Variable Neighborhood Search* (GVNS) é inspirada na ideia de buscar boas soluções por meio de estruturas de vizinhança diferentes. Uma das motivações desta abordagem é que um *ótimo global* é, consequentemente, um *ótimo local* para todas as possíveis vizinhanças de uma solução.

Um algoritmo inspirado em GVNS similar ao desenvolvido neste trabalho já existe na literatura com bons resultados [22]. Ele foi utilizado para resolver o Problema de Planejamento Operacional de Lavra em Minas a Céu Aberto, encontrando melhores resultados do que um modelo de programação matemática executado pelo resolvedor Cplex. Outras propostas de algoritmos baseados em *Variable Neighborhood Search* podem ser encontradas em [6], [9], [23], [24].

O Algoritmo 1, que faz uso do método Troca Vizinhança (Algoritmo 2), mostra o pseudocódigo do GVNS.

Algoritmo 1: *GVNS*

Entrada: Solução s, Função $f(.)$, $kMax$, $tMax$

Saída: Solução s^* de qualidade possivelmente superior a s de acordo com f

```
1  enquanto k < kMax faça
2      k ← 1
3      enquanto t < tMax faça
4          s' ← Perturbação(s, k)
5          s'' ← VND(s', f)
6          TrocaVizinhança(s, s'', f, k)
7          t ← TempoCPU
8      fim
9  fim
10 retorna s
```

Algoritmo 2: *TrocaVizinhança*

Entrada: Solução s, Solução s'', Função $f(.)$, k

```
1 se f(s'') < f(s) então
2     s ← s''
3     k ← 1
4 fim
5 senão
6     k ← k + 1
7 fim
```

2.4.2 *Variable Neighborhood Descent*

O método *Variable Neighborhood Descent* (VND) [24] consiste em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança.

Seja s a solução corrente e N a vizinhança de uma solução estruturada em r vizinhanças distintas, isto é, $N = N^{(1)} \cup N^{(2)} \cup \cdots \cup N^{(r)}$. O VND inicia-se analisando a primeira estrutura de vizinhança $N^{(1)}$. A cada iteração, o método gera o melhor vizinho s' da solução corrente s na vizinhança $N^{(k)}$. Caso s' seja melhor que s, então s' passa a ser a nova solução corrente e retorna-se à vizinhança $N^{(1)}$. Caso contrário, passa-se para a próxima estrutura de vizinhança $N^{(k+1)}$. O método termina quando não é possível encontrar uma solução $s' \in N^{(r)}$ melhor que a solução corrente.

O Algoritmo 3 mostra o pseudocódigo do VND.

Algoritmo 3: *Variable Neighborhood Descent*

Entrada: Solução s, Vizinhanças N, Função $f(.)$

```
1  Seja r o número de estruturas de vizinhança distintas
2  k ← 1
3  enquanto (k ≤ r) faça
4  |  Encontre o melhor vizinho s' ∈ N^(k)(s)
5  |  se (f(s') < f(s)) então
6  |  |  s ← s'
7  |  |  k ← 1
8  |  fim
9  |  senão
10 |  |  k ← k+1
11 |  fim
12 fim
13 Retorne s;
```

3 Metodologia Proposta

Neste capítulo é apresentada a metodologia proposta para resolver o PRVRUEOCS. Na Seção 3.1 mostra-se como uma solução do problema é representada, enquanto na Seção 3.2 são apresentados os movimentos utilizados para explorar o espaço de soluções do problema. Na Seção 3.3 mostra-se como uma solução é avaliada. Os métodos de geração de uma solução inicial são apresentados na Seção 3.4. Na Seção 3.5 é descrito o algoritmo proposto para resolver o PRVRUEOCS, o qual é inspirado na metaheurística *General Variable Neighborhood Search* (GVNS).

3.1 Representação de uma solução

Uma solução do PRVRUEOCS é representada como uma permutação de clientes de entrega e de coleta, onde os clientes de entrega são numerados de 1 a n e os de coleta numerados de $n+1$ a $2n$. Cada cliente i que inicialmente era de entrega **e** coleta é replicado como dois clientes i (apenas de entrega) e $i+n$ (apenas de coleta) com as mesmas coordenadas cartesianas. Existe ainda um elemento separador, que é indicado pelo valor zero (0), representando o depósito. Assim, clientes que ficarem após o segundo valor zero **não serão visitados**. Vale ainda ressaltar que apenas **clientes de coleta** poderão estar situados após o segundo zero em uma solução viável.

Por exemplo, se existem cinco clientes de entrega e coleta a serem atendidos, então uma possível solução é:

$$s = [\ 0\ 2\ 5\ 10\ 3\ 1\ 6\ 4\ 7\ 9\ 0\ 8\ ]$$

em que:

- a rota é iniciada;
- o cliente de entrega 2 é visitado;

- o cliente de entrega 5 é visitado;
- o cliente de coleta 10 é visitado (logo, foi feita uma entrega e coleta no cliente 5 original);
- o cliente de entrega 3 é visitado;
- o cliente de entrega 1 é visitado e uma coleta é feita no mesmo cliente, indicada pelo 6 seguinte;
- o cliente de entrega 4 é visitado;
- o cliente de coleta 7 é visitado, porém a coleta é feita separadamente da entrega (que foi feita em 2);
- o cliente de coleta 9 é visitado, porém a coleta é feita separadamente da entrega (que foi feita em 4);
- a rota termina;
- o cliente de coleta 8 não foi visitado.

Esta representação foi adotada pela sua simplicidade, sendo capaz de representar todas possíveis combinações de entrega/coleta entre clientes.

A Figura 3.1 fornece uma interpretação visual da solução s.

3.2 Estruturas de vizinhança

Para explorar o espaço de soluções do problema, aplicam-se, neste trabalho, cinco tipos diferentes de movimentos, os quais são apresentados a seguir.

3.2.1 Movimento *Swap*

O movimento *Swap* consiste em trocar um cliente i com um outro cliente j.

A Figura 3.2 mostra a aplicação do movimento *Swap* para os clientes de coleta 7 e 9 da solução s.

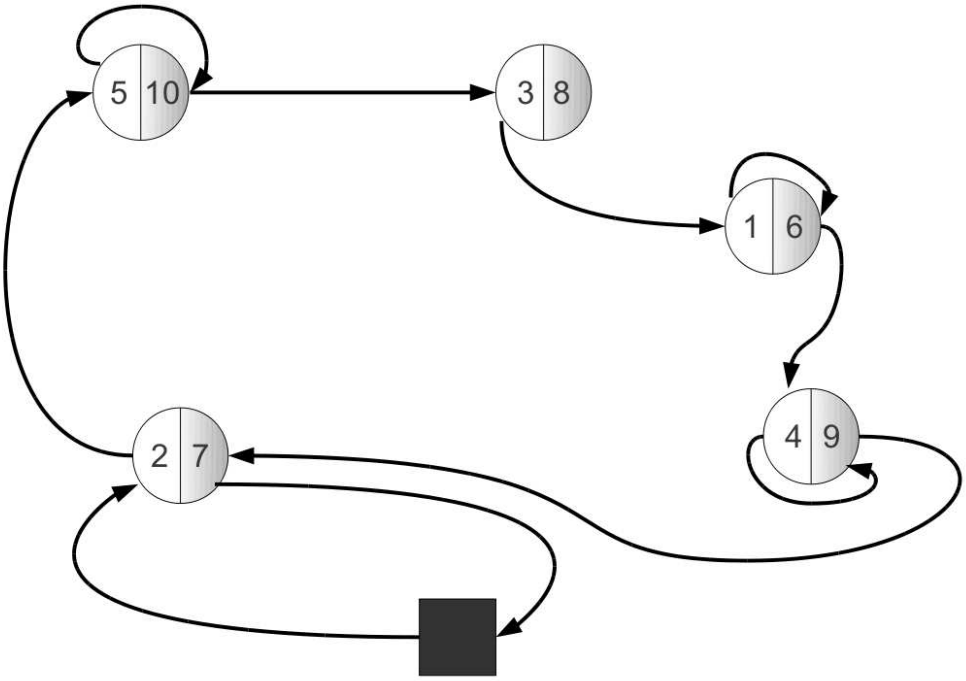

Figura 3.1: Exemplo de Solução para o PRVRUEOCS

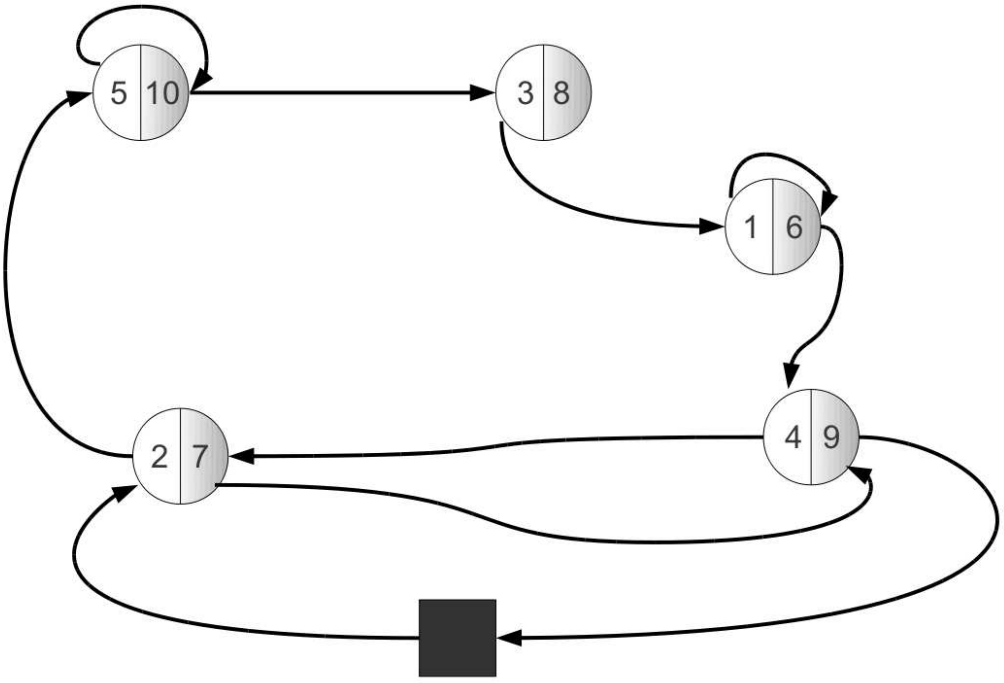

Figura 3.2: Aplicação do movimento *Swap* entre os clientes 7 e 9

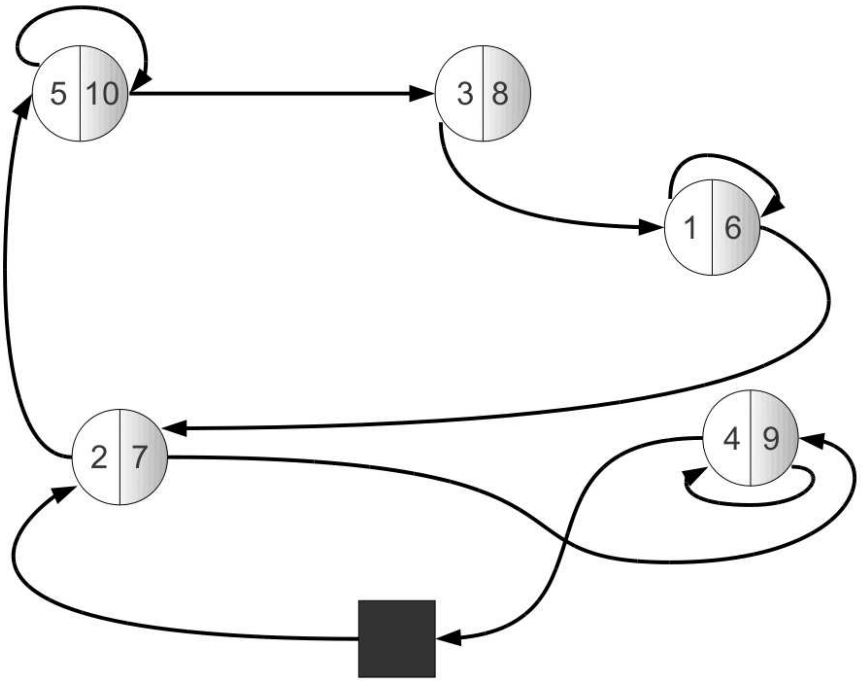

Figura 3.3: Aplicação do movimento *2-Opt* com a remoção dos arcos (6,4) e (7,0) e a inserção dos arcos (6,7) e (4,0)

3.2.2 Movimento *2-Opt*

O *2-Opt* consiste em remover dois arcos não adjacentes e reconstruir a solução de outra maneira.

A Figura 3.3 mostra a aplicação do movimento *2-Opt* entre o cliente de coleta 6 e o depósito, ou seja, a remoção dos arcos (6,4) e (7,0) e a inserção dos arcos (6,7) e (4,0).

3.2.3 Movimento *kOr-Opt*

O movimento *kOr-Opt* consiste em remover k clientes consecutivos de uma rota r e, em seguida, reinserí-los em uma outra posição nessa mesma rota. O valor de k é um parâmetro do movimento. Neste trabalho são utilizados valores de $k = 1,2,3$.

A Figura 3.4 mostra a aplicação do movimento *kOr-Opt* com $k = 1$, ocorrendo realocação do cliente 3 para entre os clientes 2 e 5.

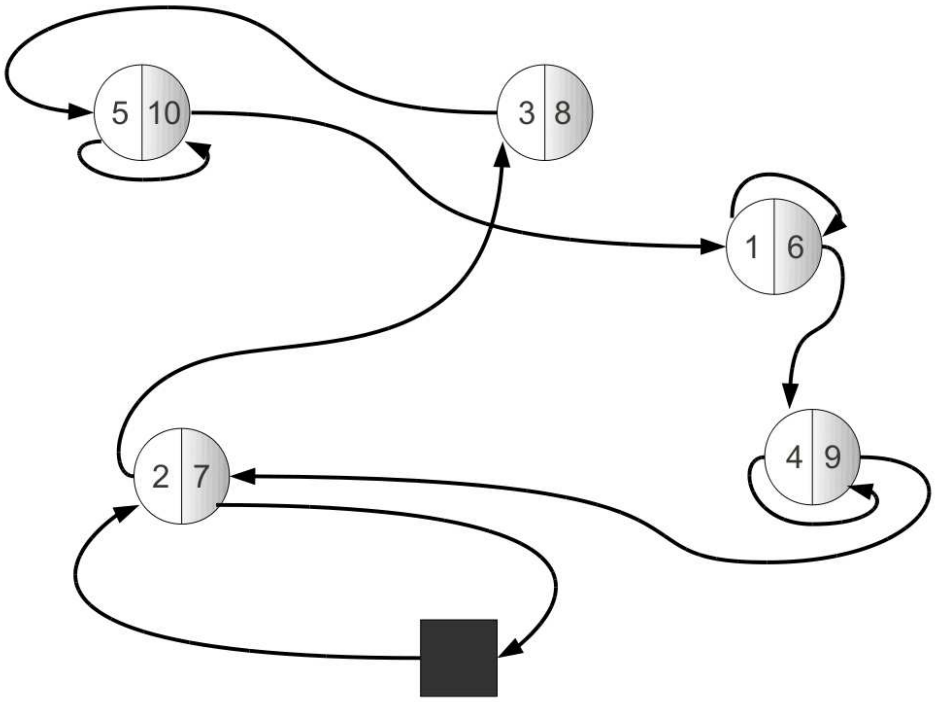

Figura 3.4: Aplicação do movimento *kOr-Opt* com $k = 1$, ocorrendo realocação do cliente 3 para entre os clientes 2 e 5

3.3 Função de avaliação

Uma solução s é avaliada pela função f apresentada pela Equação 3.1.

$$f(s) = \sum_{(i,j)\in A} c_{ij}x_{ij} - \sum_{i} \sum_{j=n+1}^{2n} b_j x_{ij} + f_{inv}(s) \tag{3.1}$$

em que:

- n : número de clientes de entrega e coleta;
- N : conjunto dos clientes (duplicados), incluindo o depósito ($N = \{0\} \cup \{1 \cdots n\} \cup \{(n+1) \cdots 2n\}$;
- A : conjunto dos arcos (i, j), com $i, j \in N$;
- c_{ij} : custo de deslocamento ou distância de um cliente i a um cliente j $(i, j \in N)$;
- b_j : benefício por visitar um cliente de coleta $(j \in \{(n+1) \cdots 2n\})$;

- x_{ij} : indica se o arco $(i, j) \in A$ é utilizado ($x_{ij} = 1$) ou não ($x_{ij} = 0$) na solução.

Note que existe uma parcela de penalização de inviabilidades na função de avaliação f_{inv}. Neste caso, cada unidade coletada que exceder a capacidade do caminhão é penalizada por um fator α e cada cliente de entrega que não for visitado (estiver após o zero) será penalizado por um fator β. De forma a permitir que se caminhe no espaço de soluções inviáveis, mas que elas sejam evitadas, neste trabalho foram adotados os fatores $\alpha = \beta = 10000$.

3.4 Geração de uma solução inicial

Nesta seção são apresentados os procedimentos heurísticos para a geração de uma solução inicial para o PRVRUEOCS. Nas Subseções 3.4.1 e 3.4.2 são descritas, respectivamente, as heurísticas de construção sequencial e global para o problema, utilizando informações do limite inferior estimado no capítulo anterior (Subseção 2.3).

3.4.1 Construção Sequencial

Seja S_{CV} um vetor de números inteiros, representando uma solução ótima do Problema do Caixeiro Viajante associado ao problema abordado. Seja S_M um conjunto de números inteiros, representando os elementos pertencentes a uma solução ótima do Problema da Mochila associado ao problema abordado.[1]

Este método consiste em criar uma solução inicial S para o problema abordado. Com S inicialmente vazia, adiciona-se o primeiro cliente de entrega da solução S_{CV} e depois, a cada passo, inclui-se um cliente de coleta da solução S_M e novamente mais um cliente de entrega, até que não haja mais clientes de entrega a serem incluídos.

Os clientes de entrega são escolhidos sequencialmente, de acordo com a ordem no vetor S_{CV}, mas os clientes de coleta são escolhidos de acordo com um dos sete critérios a seguir (apenas clientes que pertencem à mochila ótima S_M):

- Menor distância em relação ao último cliente de entrega;
- Maior benefício;

[1] Vale ressaltar que, embora os métodos de geração de solução inicial partam de soluções ótimas para os problemas do Caixeiro Viajante e Mochila, é possível desenvolver uma versão puramente heurística destes métodos, partindo de soluções obtidas por meio de heurísticas específicas para cada problema, o que os tornaria mais escaláveis em relação ao crescimento do problema.

- Menor diferença: *distância - benefício*;
- Menor quantidade de itens a serem coletados;
- Maior quantidade de itens a serem coletados;
- Balanço entre distância e coleta (menor distância e menores coletas):

$$\frac{DISTANCIA - BENEFICIO}{(C\_TOTAL - CAR\_ATUAL) - COLETA} \tag{3.2}$$

- Balanço entre distância e coleta (menor distância e maiores coletas):

$$\frac{DISTANCIA - BENEFICIO}{COLETA} \tag{3.3}$$

  onde: DISTANCIA é a distância entre o cliente anterior; BENEFICIO é o valor de benefício do cliente a ser inserido; C_TOTAL é a capacidade total do veículo; CAR_ATUAL é a carga atual do veículo; COLETA é o número de unidades a serem coletadas no cliente.

Desta maneira, de acordo com os critérios anteriores, respectivamente foram criados sete estratégias de construção sequencial de soluções, denominadas *SEQ-0*, *SEQ-1*, *SEQ-2*, *SEQ-3*, *SEQ-4*, *SEQ-5* e *SEQ-6*.

3.4.2 Construção Global

Seja S_{CV} um vetor de números inteiros, representando uma solução ótima do Problema do Caixeiro Viajante associado ao problema abordado. Seja S_M um conjunto de números inteiros, representando os elementos pertencentes a uma solução ótima do Problema da Mochila associado ao problema abordado.

Este método consiste em criar uma solução inicial S para o problema abordado. Neste método, S é inicializada com a solução ótima S_{CV}. A cada passo, inclui-se um cliente de coleta da solução S_M na *melhor* posição possível da solução S, de acordo com um dos sete critérios a seguir:

- Menor distância em relação ao último cliente de entrega;
- Maior benefício;
- Menor diferença: *distância - benefício*;

- Menor quantidade de itens a serem coletados;
- Maior quantidade de itens a serem coletados;
- Balanço entre distância e coleta (menor distância e menores coletas):

$$\frac{DISTANCIA - BENEFICIO}{(C\_TOTAL - CAR\_ATUAL) - COLETA} \tag{3.4}$$

- Balanço entre distância e coleta (menor distância e maiores coletas):

$$\frac{DISTANCIA - BENEFICIO}{COLETA} \tag{3.5}$$

  onde: DISTANCIA é a distância entre o cliente anterior e posterior; BENEFICIO é o valor de benefício do cliente a ser inserido; C_TOTAL é a capacidade total do veículo; CAR_ATUAL é a carga atual do veículo; COLETA é o número de unidades a serem coletadas no cliente.

Os critérios utilizados foram os mesmos da construção sequencial, porém este algoritmo pode ter uma visão mais *global* da solução, visto que faz inserções entre cada cliente da solução corrente. De acordo com os critérios anteriores, respectivamente foram criados sete estratégias de construção sequencial de soluções, denominadas *GLB-0*, *GLB-1*, *GLB-2*, *GLB-3*, *GLB-4*, *GLB-5* e *GLB-6*.

3.5 Algoritmo proposto HGVNS

O algoritmo proposto, denominado *Hybrid General Variable Neighborhood Search* (HGVNS), consiste na combinação de uma geração de uma solução inicial utilizando informações de um limite inferior para o problema e um algoritmo heurístico baseado em na metaheurística *General Variable Neighborhood Search* (GVNS).

O pseudocódigo do *HGVNS* está esquematizado no Algoritmo 4. Nele, *IterMax* indica o número máximo de iterações realizadas em um dado nível de perturbação.

Algoritmo 4: *HGVNS*

Entrada: r vizinhanças: N_{SWAP}, N_{2_OPT}, $N_{OR\_OPT\_1}$, $N_{OR\_OPT\_2}$ e $N_{OR\_OPT\_3}$

Entrada: Função $f(.)$, *IterMax*, Nível p

```
1  s_CV ← Solução ótima do Problema do Caixeiro Viajante (resolvedor exato concorde)
2  s_M ← Solução ótima do Problema da Mochila (resolvedor exato cplex)
3  s_0 ← Solução construída pelo método de solução inicial utilizando s_CV e s_M
4  s* ← VND(s_0, f)
5  p ← 0
6  enquanto critério de parada não satisfeito faça
7      iter ← 0
8      enquanto iter < IterMax e critério de parada não satisfeito faça
9          s' ← s*
10         para i = 1 to p + 2 faça
11             k ← SelectNeighborhood()
12             s' ← Shake(s', k)
13         fim
14         s'' ← VND(s')
15         se s'' for melhor que s* de acordo com a função f então
16             s* ← s''; p ← 0; iter ← 0
17         fim
18         senão
19             iter ← iter + 1
20         fim
21     fim
22     p ← p + 1
23 fim
24 retorna s
```

A construção da solução inicial s_0 (linha 3 do Algoritmo 4) é feita com base em soluções ótimas obtidas por resolvedores exatos para os problemas associados da Mochila e do Caixeiro Viajante. A busca local (linhas 4 e 14 do Algoritmo 4) usam o procedimento VND (ver pseudo-código no Algoritmo 3) com os movimentos descritos na Subseção 3.2. A cada etapa do VND é encontrado o melhor vizinho em relação à vizinhança corrente, de acordo com a seguinte sequência de vizinhanças: N_{SWAP}, N_{2_OPT}, $N_{OR\_OPT\_1}$, $N_{OR\_OPT\_2}$

e $N_{OR\_OPT\_3}$.

Quando um certo número de iterações sem melhora é alcançado, o `HGVNS` aplica $p+2$ vezes o procedimento *Shake* utilizando uma vizinhança escolhida previamente. O procedimento *SelectNeighborhood* (linha 11 do Algoritmo 4) consiste em escolher aleatoriamente uma vizinhança k entre N_{SWAP}, N_{2_OPT}, $N_{OR\_OPT\_1}$, $N_{OR\_OPT\_2}$ e $N_{OR\_OPT\_3}$.

Cada chamada de $Shake(s',k)$ (linha 12 do Algoritmo 4) executa um movimento aleatório da vizinhança k da solução perturbada s'. Após *IterMax* iterações sem melhora, p é incrementada de forma a gerar soluções cada vez mais distantes da solução corrente no espaço de busca.

Se `VND` encontra uma solução melhor, a variável p retorna ao menor valor, ou seja, $p = 0$.

4 Resultados Computacionais

Para o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas (PRVRUEOCS) existe na literatura um conjunto de testes contendo 68 problemas-teste, proposto por [1]. Os problemas são divididos em quatro grupos de 17 problemas, denominados: p_two, $half$, one e two.

Limites inferiores para os problemas-teste foram computados de acordo com a Subseção 2.3 e podem ser vistos nas Tabelas 4.1 e 4.2.

Podemos observar pelas Tabelas 4.1 e 4.2 que os valores calculados para o Problema do Caixeiro Viajante se repetem sempre quatro vezes. Esse fato é decorrente da forma como os problemas-teste foram gerados, visto que foram tomados inicialmente 17 problemas-teste do Problema de Roteamento de Veículos clássico e para cada um deles foram gerados quatro problemas com diferentes valores de coleta.

Tabela 4.1: Limites Inferiores (dos problemas-teste 016_*B_half* a 031_*B_two*)

Problema-teste	Caixeiro Viajante (CV)	Mochila (M)	Limite Inferior (CV - M)
016_B_half	219,45	182,85	36,60
016_B_one	219,45	374,86	-155,41
016_B_p_two	219,45	88,46	130,99
016_B_two	219,45	760,26	-540,81
021_B_half	255,69	275,85	-20,16
021_B_one	255,69	571,78	-316,09
021_B_p_two	255,69	123,48	132,21
021_B_two	255,69	1165,26	-909,57
022_B_half	278,43	343,40	-64,97
022_B_one	278,43	707,58	-429,15
022_B_p_two	278,43	162,42	116,01
022_B_two	278,43	1435,92	-1157,49
023_B_half	470,04	564,10	-94,06
023_B_one	470,04	1181,50	-711,46
023_B_p_two	470,04	209,16	260,88
023_B_two	470,04	2416,25	-1946,21
026_B_half	306,93	399,40	-92,47
026_B_one	306,93	811,33	-504,40
026_B_p_two	306,93	167,26	139,67
026_B_two	306,93	1648,42	-1341,49
030_B_half	382,74	765,54	-382,80
030_B_one	382,74	1538,97	-1156,23
030_B_p_two	382,74	301,41	81,33
030_B_two	382,74	3085,90	-2703,16
031_B_half	317,44	408,46	-91,02
031_B_one	317,44	830,72	-513,28
031_B_p_two	317,44	201,15	116,29
031_B_two	317,44	1675,24	-1357,80

Tabela 4.2: Limites Inferiores (dos problemas-teste 033_*B_half* a 101*b_B_two*)

Problema-teste	Caixeiro Viajante (CV)	Mochila (M)	Limite Inferior (CV - M)
033_B_half	442,88	599,74	-156,86
033_B_one	442,88	1220,86	-777,98
033_B_p_two	442,88	254,21	188,67
033_B_two	442,88	2463,05	-2020,17
036_B_half	352,15	479,59	-127,44
036_B_one	352,15	975,73	-623,58
036_B_p_two	352,15	229,12	123,03
036_B_two	352,15	1967,96	-1615,81
041_B_half	364,94	550,78	-185,84
041_B_one	364,94	1132,40	-767,46
041_B_p_two	364,94	263,54	101,40
041_B_two	364,94	2295,74	-1930,80
045_B_half	619,06	1110,21	-491,15
045_B_one	619,06	2267,57	-1648,51
045_B_p_two	619,06	421,02	198,04
045_B_two	619,06	4582,38	-3963,32
048_B_half	33523,67	71276,63	-37752,96
048_B_one	33523,67	140582,45	-107058,78
048_B_p_two	33523,67	38320,70	-4797,03
048_B_two	33523,67	281821,97	-248298,30
051_B_half	428,86	749,47	-320,61
051_B_one	428,86	1527,72	-1098,86
051_B_p_two	428,86	312,28	116,58
051_B_two	428,86	3084,16	-2655,30
072_B_half	195,16	603,71	-408,55
072_B_one	195,16	1221,17	-1026,01
072_B_p_two	195,16	233,17	-38,01
072_B_two	195,16	2456,14	-2260,98
076_B_half	544,35	1123,85	-579,50
076_B_one	544,35	2303,52	-1759,17
076_B_p_two	544,35	530,00	14,35
076_B_two	544,35	4662,83	-4118,48
101a_B_half	641,61	1562,84	-921,23
101a_B_one	641,61	3192,51	-2550,90
101a_B_p_two	641,61	706,72	-65,11
101a_B_two	641,61	6451,82	-5810,21
101b_B_half	503,18	1889,41	-1386,23
101b_B_one	503,18	3823,57	-3320,39
101b_B_p_two	503,18	759,92	-256,74
101b_B_two	503,18	7691,66	-7188,48

4.1 Considerações sobre a medida *gap* utilizada

A medida *gap* é comumente utilizada para avaliar a qualidade de um limite superior ou inferior para um problema.

Porém, deve-se tomar certo cuidado com a forma de se calcular esse valor percentual quando o limite inferior do problema (de minimização) é um valor negativo (como no problema em questão).

Assim como no trabalho de [1] utilizaremos o seguinte cálculo do *gap*:

$$gap = \left| \frac{ls - li}{li} \right| \tag{4.1}$$

onde: ls é um limite superior, li é um limite inferior e $|x|$ é o valor absoluto de x.

4.2 Métodos construtivos

Os métodos construtivos foram comparados pelas métricas *score* e *gap*. A métrica *score* representa o número de vezes que o algoritmo perde para os concorrentes. A métrica *gap* é a distância entre o valor final gerado pelo método (limite superior) e o limite inferior para o problema-teste, dividido pelo limite inferior.

Foi utilizado um conjunto de testes reduzido de 40 problemas-teste, a saber: 016*_B_half*, 016*_B_one*, 016*_B_p_two*, 016*_B_two*, 022*_B_half*, 022*_B_one*, 022*_B_p_two*, 022*_B_two*, 026*_B_half*, 026*_B_one*, 026*_B_p_two*, 026*_B_two*, 031*_B_half*, 031*_B_one*, 031*_B_p_two*, 031*_B_two*, 036*_B_half*, 036*_B_one*, 036*_B_p_two*, 036*_B_two*, 045*_B_half*, 045*_B_one*, 045*_B_p_two*, 045*_B_two*, 051*_B_half*, 051*_B_one*, 051*_B_p_two*, 051*_B_two*, 072*_B_half*, 072*_B_one*, 072*_B_p_two*, 072*_B_two*, 076*_B_half*, 076*_B_one*, 076*_B_p_two*, 076*_B_two*, 101*a_B_half*, 101*a_B_one*, 101*a_B_p_two*, 101*a_B_two*.

Segue um resumo dos métodos construtivos previamente descritos na Seção 3.4:

- SEQ-0: construção sequencial com critério de *menor distância*;
- SEQ-1: construção sequencial com critério de *maior benefício*;
- SEQ-2: construção sequencial com critério de *menor diferença distância - benefício*;
- SEQ-3: construção sequencial com critério de *menor coleta*;

- SEQ-4: construção sequencial com critério de *maior coleta*;
- SEQ-5: construção sequencial com critério de *balanço entre distância e coleta (menor distância e menores coletas)*;
- SEQ-6: construção sequencial com critério de *balanço entre distância e coleta (menor distância e maiores coletas)*;
- GLB-0: construção global com critério de *menor distância*;
- GLB-1: construção global com critério de *maior benefício*;
- GLB-2: construção global com critério de *menor diferença distância - benefício*;
- GLB-3: construção global com critério de *menor coleta*;
- GLB-4: construção global com critério de *maior coleta*;
- GLB-5: construção global com critério de *balanço entre distância e coleta (menor distância e menores coletas)*;
- GLB-6: construção global com critério de *balanço entre distância e coleta (menor distância e maiores coletas)*.

Os resultados dos testes entre métodos construtivos pode ser visto na Tabela 4.3.

Tabela 4.3: Métodos Construtivos

Método	SEQ-0	SEQ-1	SEQ-2	SEQ-3	SEQ-4	SEQ-5	SEQ-6
score total	131	443	174	456	456	252	237
gap médio	466%	1353%	451%	1380%	1380%	492%	507%
Método	GLB-0	GLB-1	GLB-2	GLB-3	GLB-4	GLB-5	GLB-6
score total	47	356	41	362	353	175	59
gap médio	34%	1061%	44%	1032%	1050%	152%	51%

Pela Tabela 4.3, percebe-se que os dois melhores métodos em relação à métrica *score* são GLB-0 e GLB-2 (visto que são os que perderam menos). Já em relação à métrica *gap*, os dois melhores são os mesmos, mas na sequência inversa: GLB-2 e GLB-0.

Como previsto, a distância é um critério importante a ser considerado na construção de uma boa solução inicial. Os critérios baseados unicamente em quantidade coletada

ou valor em benefícios geraram as soluções de pior qualidade. Em geral, os métodos de construção global obtiveram melhor desempenho.

Como os métodos GLB-0 e GLB-2 ficaram empatados e ambos obtiveram soluções de boa qualidade, adotou-se o GLB-2 como método de geração de solução inicial, visto que ele considera mais características do problema que o método GLB-0.

4.3 Algoritmo HGVNS

O algoritmo proposto HGVNS foi implementado na linguagem C++ no *framework* computacional OptFrame[1]. O compilador utilizado foi *gcc 4.4* e a máquina utilizada foi um Intel Core i7 2,93GHz, 8GB de memória, sistema operacional Ubuntu 10.04 kernel 2.6.32-28-generic. Foi utilizado o conjunto de testes de [1], contendo 68 problemas-teste. O algoritmo proposto foi executado 30 vezes para cada problema-teste de forma a comprovar sua eficiência, mesmo que de forma empírica.

A calibração dos parâmetros do HGVNS foi empírica e resultou nos seguintes valores: número de iterações $IterMax = \lceil N/2 \rceil$; nível máximo de perturbação $p = \frac{\lceil 1{,}53 \times N + 95 \rceil}{\lceil N/2 \rceil}$, sendo N o número de clientes (com duplicação) do problema-teste em questão. Tal calibração busca oferecer ao algoritmo um número de iterações suficiente para encontrar bons resultados nos problemas menores e também controlar o crescimento do tempo computacional do algoritmo nos problemas de dimensões mais elevadas.

Os resultados para 30 execuções do método HGVNS com o método construtivo GLB2 são apresentados nas Tabelas 4.4 e 4.5. As colunas **Problema-teste**, **Limite Inferior**, **Melhor**, **Média**, ***gap*(%)**, **Var. (%)** e **Tempo Médio (s)** representam, respectivamente, o problema-teste utilizado, o limite inferior estimado para o problema-teste, a melhor solução encontrada em 30 execuções do método HGVNS, a média da soluções encontradas em 30 execuções do método HGVNS, o *gap* percentual entre a melhor solução encontrada pelo método HGVNS e o limite inferior estimado, a variabilidade percentual do método calculada de acordo com a Equação 4.2 e o tempo médio, em segundos, de 30 execuções do método.

$$Var.(\%) = \frac{Media - Melhor}{Melhor} \times 100 \tag{4.2}$$

[1]OptFrame está disponível na página *http://sourceforge.net/projects/optframe/* sob licença LGPLv3.

Tabela 4.4: Resultados HGVNS (Problemas de sufixo *p_two* e *half*)

Problema-teste	**Limite Inferior**	**Melhor**	**Média**	***gap*(%)**	**Var. (%)**	**Tempo Médio (s)**
016_B_p_two	130,99	132,28	133,426	0,98	0,87	6,92
021_B_p_two	132,21	137,59	141,168	4,07	2,60	19,28
022_B_p_two	116,01	123,59	124,859	6,53	1,03	16,12
023_B_p_two	260,88	269,86	273,234	3,44	1,25	16,57
026_B_p_two	139,67	146,9	147,179	5,18	0,19	27,65
030_B_p_two	81,33	82,29	85,694	1,18	4,14	55,26
031_B_p_two	116,29	123,15	123,392	5,90	0,20	57,99
033_B_p_two	188,67	199,17	202,084	5,57	1,46	93,88
036_B_p_two	123,03	130,42	136,406	6,01	4,59	103,91
041_B_p_two	101,4	109,48	115,164	7,97	5,19	157,81
045_B_p_two	198,04	199,82	201,85	0,90	1,02	149,22
048_B_p_two	-4797,03	-4244,69	-4122,465	11,51	2,88	101,44
051_B_p_two	116,58	126,7	127,992	8,68	1,02	228,01
072_B_p_two	-38,01	-36,03	-35,848	5,21	0,51	399,43
076_B_p_two	14,35	23,62	26,946	64,60	14,08	818,56
101a_B_p_two	-65,11	-53,89	-47,439	17,23	11,97	3176,26
101b_B_p_two	-256,74	-252,84	-251,347	1,52	0,59	2011,15
Média	**-**	**-**	**-**	**9,20**	**3,15**	**437,62**
016_B_half	36,6	39,86	40,61	8,91	1,88	2,77
021_B_half	-20,16	-18,78	-13,964	6,85	25,64	11,40
022_B_half	-64,97	-62,63	-58,591	3,60	6,45	11,14
023_B_half	-94,06	-80,95	-80,762	13,94	0,23	10,61
026_B_half	-92,47	-87,87	-85,526	4,97	2,67	28,24
030_B_half	-382,8	-378,64	-370,694	1,09	2,10	50,46
031_B_half	-91,02	-87,06	-84,896	4,35	2,49	32,54
033_B_half	-156,86	-150,73	-146,008	3,91	3,13	86,76
036_B_half	-127,44	-127,06	-124,648	0,30	1,90	92,09
041_B_half	-185,84	-183,86	-179,359	1,07	2,45	111,94
045_B_half	-491,15	-491,15	-490,524	0,00	0,13	14,70
048_B_half	-37752,96	-37200,62	-37200,62	1,46	0,00	119,14
051_B_half	-320,61	-310,24	-307,354	3,23	0,93	221,55
072_B_half	-408,55	-406,57	-404,964	0,48	0,40	395,60
076_B_half	-579,5	-579,25	-576,189	0,04	0,53	402,21
101a_B_half	-921,23	-906,79	-900,822	1,57	0,66	2248,46
101b_B_half	-1386,23	-1384,06	-1380,258	0,16	0,27	2291,52
Média	**-**	**-**	**-**	**3,29**	**3,05**	**360,65**

Tabela 4.5: Resultados HGVNS (Problemas de sufixo *one* e *two*)

Problema-teste	**Limite Inferior**	**Melhor**	**Média**	***gap*(%)**	**Var. (%)**	**Tempo Médio (s)**
016_B_one	-155,41	-150,73	-150,73	3,01	0,00	3,89
021_B_one	-316,09	-307,8	-307,213	2,62	0,19	9,62
022_B_one	-429,15	-421,04	-421,04	1,89	0,00	8,75
023_B_one	-711,46	-698,35	-698,256	1,84	0,01	11,50
026_B_one	-504,4	-497,17	-497,17	1,43	0,00	12,14
030_B_one	-1156,23	-1152,07	-1146,639	0,36	0,47	56,60
031_B_one	-513,28	-511,07	-508,038	0,43	0,59	35,24
033_B_one	-777,98	-765,79	-761,796	1,57	0,52	43,38
036_B_one	-623,58	-616,12	-610,631	1,20	0,89	56,98
041_B_one	-767,46	-760,52	-758,597	0,90	0,25	89,82
045_B_one	-1648,51	-1648,51	-1647,884	0,00	0,04	14,71
048_B_one	-107058,78	-107058,78	-106682,877	0,00	0,35	176,52
051_B_one	-1098,86	-1086,52	-1085,089	1,12	0,13	208,93
072_B_one	-1026,01	-1024,04	-1023,069	0,19	0,09	173,08
076_B_one	-1759,17	-1758,92	-1753,271	0,01	0,32	340,06
101a_B_one	-2550,9	-2541,35	-2536,941	0,37	0,17	1651,86
101b_B_one	-3320,39	-3315,78	-3312,665	0,14	0,09	1017,48
Média	**-**	**-**	**-**	**1,01**	**0,24**	**230,03**
016_B_two	-540,81	-536,13	-536,13	0,87	0,00	3,10
021_B_two	-909,57	-901,28	-900,693	0,91	0,07	10,50
022_B_two	-1157,49	-1149,38	-1149,38	0,70	0,00	9,19
023_B_two	-1946,21	-1933,1	-1932,912	0,67	0,01	11,17
026_B_two	-1341,49	-1334,26	-1334,26	0,54	0,00	12,19
030_B_two	-2703,16	-2699	-2697,256	0,15	0,06	50,94
031_B_two	-1357,8	-1354,49	-1352,73	0,24	0,13	42,33
033_B_two	-2020,17	-2007,98	-2003,986	0,60	0,20	42,64
036_B_two	-1615,81	-1608,35	-1604,649	0,46	0,23	78,17
041_B_two	-1930,8	-1923,86	-1921,497	0,36	0,12	100,39
045_B_two	-3963,32	-3963,32	-3962,694	0,00	0,02	14,72
048_B_two	-248298,3	-247946,25	-247871,39	0,14	0,03	90,46
051_B_two	-2655,3	-2645,35	-2640,952	0,37	0,17	218,32
072_B_two	-2260,98	-2259,01	-2259,01	0,09	0,00	3,64
076_B_two	-4118,48	-4118,23	-4113,117	0,01	0,12	371,45
101a_B_two	-5810,21	-5800,66	-5793,329	0,16	0,13	1478,83
101b_B_two	-7188,48	-7186,31	-7181,598	0,03	0,07	1723,51
Média	**-**	**-**	**-**	**0,37**	**0,08**	**250,68**
Média Geral	**-**	**-**	**-**	**3,47**	**1,63**	**319,75**

Pelas Tabelas 4.4 e 4.5, podemos perceber que os *gaps* médios são baixos, indicando que as soluções encontradas pelo método proposto HGVNS estão geralmente próximas ao limite inferior estimado. Um resultado bem interessante pode ser visto nos problemas-teste 045_*B_half*, 045_*B_one* e 045_*B_two*, onde o método proposto conseguiu alcançar uma solução de valor igual ao do limite inferior, indicando então que estas soluções são ambas ótimas. A variabilidade média do método foi de $1,63\%$, indicando a robustez do método. Porém, este valor chega no pior caso até $25,64\%$ no problema-teste 021_*B_half*, um valor alto, mas isto possivelmente foi causado pelo limite inferior ser muito pequeno (próximo de zero) fornecendo um limite muito pobre neste problema-teste.

Na Tabela 4.6 é feita uma comparação de resultados em termos de *gap* com os valores reportados por [1], método chamado **CI5+IMP**. Também são reportados os resultados obtidos em cada grupo de problemas-teste (*p_two*, *half*, *one*, *two*) separadamente.

Tabela 4.6: Comparação em termo de *gaps* com o método **CI5+IMP** de [1]

	p_two	*half*	*one*	*two*	**Média**
Gribkovskaia *et al.* (2008) [1]	86,78	28,12	5,38	2,19	**30,62**
Presente Trabalho	9,20	3,29	1,01	0,37	**3,47**

Como pode ser visto na Tabela 4.6, o método HGVNS proposto supera a literatura em todos os casos, em termos de *gap*. Dados os altos *gaps* da literatura, os autores dividiram o *gap* em duas medidas: $cgap = (\bar{c} - c)/c$, onde $\bar{c}$ é o custo de roteamento obtido pela heurística e c é o limite inferior para o custo de roteamento; e $pgap = (\bar{p} - c)/\bar{p}$, onde $\bar{p}$ é o limite superior do Problema da Mochila associado e p é o valor em benefícios obtido pela heurística.

A Tabela 4.7 mostra os resultados utilizando as medidas *pgap* e *cgap* com os valores reportados por [1]. Porém, para esta medida a média geral dos métodos baseados na metaheurística Busca Tabu foi utilizado pelos autores.

Tabela 4.7: Comparação em termo de *pgap* e *cgap* com [1]

	p_two		*half*		*one*		*two*		**Média**	
	cgap	*pgap*	*cgap*	*pgap*	*cgap*	*pgap*	*cgap*	*pgap*	***cgap***	***pgap***
Gribkovskaia *et al.* (2008) [1]	2,94	1,06	2,83	0,27	4,42	0,09	5,92	0,03	**4,03**	**0,36**
Presente Trabalho	1,04	1,01	0,64	0,36	1,45	0,07	1,40	0,04	**1,13**	**0,37**

Na Tabela 4.7, podemos observar que o método proposto neste trabalho supera a literatura na maioria dos valores. Pela média geral percebemos, também, a superioridade do método pela métrica *cgap*, e que *pgap* está pior em apenas $0,01\%$. É necessário ressaltar que as medidas *cgap* e *pgap* não correspondem à função objetivo original do problema; portanto, não devem ser utilizadas como base de comparação. Porém, estas medidas foram também feitas no presente trabalho tendo em vista que a literatura utilizou-se desta avaliação.

O modelo de programação matemática proposto por [2] foi implementado para validar as soluções encontradas pela heurística proposta. Nenhum ótimo foi provado por este modelo e o modelo proposto por [1] não foi testado visto que não havia perspectivas de que algum ótimo poderia ser provado com uma implementação pura deste modelo, uma vez que esta alternativa de solução já havia sido experimentado anteriormente pelos autores do modelo.

5 Conclusões e Trabalhos Futuros

Este trabalho abordou o Problema de Roteamento de Veículos de Rota Única com Entregas Obrigatórias e Coletas Seletivas (PRVRUEOCS). Dada a sua dificuldade de resolução na otimalidade, em virtude de o mesmo ser da classe NP-difícil, foi proposto um algoritmo heurístico híbrido, denominado HGVNS. Este algoritmo consiste em duas etapas, onde na primeira uma solução inicial é gerada a partir de uma heurística construtiva tomando como base soluções ótimas para os problemas do Caixeiro Viajante e Mochila associados ao problema atual. Na segunda etapa um refinamento é feito por uma heurística inspirada na metaheurística *General Variable Neighborhood Search*, tendo assim, uma busca local baseada no método VND com 5 estruturas de vizinhança diferentes.

Para validar o algoritmo proposto, foi utilizado um conjunto de problemas-teste da literatura. Esse conjunto foi gerado a partir de problemas clássicos de roteamento de veículos, contendo 68 problemas-teste.

De acordo com os resultados empíricos obtidos, verifica-se que o HGVNS é um algoritmo robusto, com baixa variabilidade média, e com soluções muito próximas ao *lower bound*. O *gap* médio final foi de $3,47\%$. Segundo as métricas *pgap* e *cgap* da literatura, o algoritmo obteve um *pgap* de $0,37\%$, bem próximo ao valor de $0,36\%$ da literatura, e um *cgap* de $1,13\%$, inferior ao valor de $4,03\%$ da literatura, comprovando assim que as soluções finais geradas pelo algoritmo proposto estão, em média, mais próximas dos limites inferiores estimados.

Como trabalho futuro, pretende-se incorporar suporte a GPU's para diminuir o tempo médio do algoritmo, ou mesmo explorar maiores vizinhanças, tentando melhorar a qualidade das soluções finais geradas. Em [25] uma estratégia similar é implementada em GPU para o Problema do Caixeiro Viajante, utilizando a busca local *2-Opt*. Os autores reportam um ganho de velocidade de aproximadamente 24 vezes em relação à implementação correspondente em CPU.

Também, como sugestão para trabalhos futuros, podemos incluir o desenvolvimento de novas heurísticas utilizando conceitos de outras metaheurísticas, tais como os Algoritmos Evolutivos e o *Iterated Local Search* (ILS) que tem tido bom desempenho em problemas similares ao PRVRUEOCS.

Referências Bibliográficas

[1] GRIBKOVSKAIA, I.; LAPORTE, G.; SHYSHOU, A. The single vehicle routing problem with deliveries and selective pickups. *Computers and Operations Research*, Elsevier Science Ltd., Oxford, UK, UK, v. 35, p. 2908–2924, Setembro 2008. ISSN 0305-0548. Disponível em: <http://portal.acm.org/citation.cfm?id=1343112.1343206>.

[2] SÜRAL, H.; BOOKBINDER, J. H. The single-vehicle routing problem with unrestricted backhauls. *Networks*, v. 41, p. 127–136, 2003.

[3] DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. *Management Science*, v. 6, p. 80–91, 1959.

[4] APPLEGATE, D. et al. *The Traveling Salesman Problem: A Computational Study*. Disponível em: $< http://www.it-weise.de/ >$. Acesso em: 01 Dezembro 2008, 2006. Hardcover. ISBN 0691129932. Disponível em: <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0691129932>.

[5] LOURENÇO, H. R.; MARTIN, O.; STÜTZLE, T. Iterated local search. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of Metaheuristics*. [S.l.]: Kluwer Academic Publishers, Norwell, MA, 2003, (International Series in Operations Research & Management Science, v. 57). p. 321–353.

[6] MLADENOVIĆ, N.; HANSEN, P. Variable neighboorhood search. *Computers and Operations Research*, v. 24, p. 1097–1100, 1997.

[7] FEO, T.; RESENDE, M. Greedy randomized search procedures. *Journal of Global Optimization*, v. 6, p. 109–133, 1995.

[8] GLOVER, F.; LAGUNA, M. *Tabu Search*. Boston: Kluwer Academic Publisher, 1997.

[9] HANSEN, P.; MLADENOVIĆ, N.; PÉREZ, J. A. M. Variable neighborhood search: methods and applications. *4OR: A Quarterly Journal of Operations Research*, v. 6, p. 319–360, 2008.

[10] GOESCHALEKX, M.; JACOBS, B. The vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 42, p. 39–51, 1989.

[11] TOTH, P.; VIGO, D. A heuristic algorithm for the symmetric and asymmetric vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 113, p. 528–543, 1999.

[12] BRANDÃO, J. A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 173, p. 540–555, 2006.

[13] TAVAKKOLI-MOGHADDAM, R.; SAREMI, A.; ZIAEE, M. A memetic algorithm for a vehicle routing problem with backhauls. *Applied Mathematics and Computation*, v. 181, p. 1049–1060, 2006.

[14] GAJPAL, Y.; ABAD, P. Multi-ant colony system (macs) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 196, p. 102 – 117, 2009. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/B6VCT-4RY6WP4-2/2/56da7aec7d9797000d6b36125be3880c>.

[15] GOMES, L.; DINIZ, V.; MARTINHON, C. A. An hybrid grasp+vns metaheuristic for the prize-collecting traveling salesman problem. In: *Anais do XXXII Simpósio Brasileiro de Pesquisa Operacional*. [S.l.: s.n.], 2000. p. 1657–1665.

[16] CHAVES, A. A. et al. Metaheurísticas híbridas para resolução do problema do caixeiro viajante com coleta de prêmios. *Produção*, v. 17, p. 263 – 272, 2007. ISSN 0103-6513.

[17] CHAVES, A. A.; LORENA, L. A. N. Hybrid metaheuristic for the prize collecting travelling salesman problem. In: *Proceedings of the 8th European conference on Evolutionary computation in combinatorial optimization*. Berlin, Heidelberg: Springer-Verlag, 2008. (EvoCOP'08), p. 123–134. ISBN 3-540-78603-1, 978-3-540-78603-0. Disponível em: <http://portal.acm.org/citation.cfm?id=1792634.1792645>.

[18] GAJPAL, Y.; ABAD, P. An ant colony system (acs) for vehicle routing problem with simultaneous delivery and pickup. *Computers and Operations Research*, Elsevier Science Ltd., Oxford, UK, UK, v. 36, p. 3215–3223, Dezembro 2009. ISSN 0305-0548. Disponível em: <http://portal.acm.org/citation.cfm?id=1550960.1551160>.

[19] SUBRAMANIAN, A. et al. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers and Operations Research*, v. 37, p. 1899–1911, 2010.

[20] MILLER, C.; TUCKER, A.; ZEMLIN, R. Integer programming formulations and travelling salesman problems. *Journal of Association for Computing Machinery*, v. 7, p. 326–329, 1960.

[21] MARTELLO, S.; TOTH, P. *Knapsack Problems: Algorithms and Computer Implementations*. [S.l.]: Wiley, 1989.

[22] SOUZA, M. et al. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, v. 207, n. 2, p. 1041–1051, 2010. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/B6VCT-507CRR3-1/2/7e261f08e1d612b9af0eddb2deb7fee0>.

[23] HANSEN, P.; MLADENOVIĆ, N.; PÉREZ, J. A. M. Variable neighborhood search. *European Journal of Operational Research*, v. 191, p. 593–595, 2008.

[24] HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, v. 130, n. 3, p. 449–467, May 2001.

[25] FUJIMOTO, N.; TSUTSUI, S. A highly-parallel tsp solver for a gpu computing platform. In: *Proceedings of the 7th international conference on Numerical methods and applications*. Berlin, Heidelberg: Springer-Verlag, 2011. (NMA'10), p. 264–271. ISBN 978-3-642-18465-9. Disponível em: <http://portal.acm.org/citation.cfm?id=1945690.1945726>.