Uma Estratégia Heurística para o Problema de Roteamento de Veículos com Coleta e Entrega Fracionadas Um-para-um

Matheus Nohra Haddad
Universidade Federal Fluminense
mathaddad@gmail.com

Thibaut Vidal
Pontifícia Universidade Católica do Rio de Janeiro
vidalt@mit.edu

Luiz Satoru Ochi
Universidade Federal Fluminense
satoru@ic.uff.br

Richard Hartl
Universidade de Viena
richard.hartl@univie.ac.at

Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto
marcone@iceb.ufop.br

RESUMO

O Problema de Roteamento de Veículos com Coleta e Entrega Fracionadas Um-para-um (PRVCEFU) tem como objetivo encontrar o roteamento mínimo para uma frota de veículos, que devem coletar e entregar produtos em um conjunto de clientes. No PRVCEFU, cada produto possui o seu cliente de coleta e o seu cliente de entrega correspondente, que podem ser visitados mais de uma vez. Este problema pode ser encontrado na prática, por exemplo em transporte a granel de produtos por navio, e ainda pertence à classe $\mathcal{NP}$-Difícil. Para resolver o PRVCEFU, é proposto um algoritmo heurístico chamado SVR, baseado na meta-heurística *Variable Neighborhood Search* (VNS) e incluindo *Random Variable Neighborhood Descent* (RVND) para buscas locais. O SVR foi aplicado em instâncias da literatura e seus resultados se comprovaram melhores do que os resultados de algoritmos prévios, superando-os em todas as instâncias testadas.

PALAVRAS CHAVE. PRVCEFU, VNS, RVND.

Área Principal: MH - Metaheuristicas

ABSTRACT

The Multi-vehicle One-to-one Pickup and Delivery Problem with Split Loads (MPDPSL) aims to find the minimum routing for a fleet of vehicles, which have to collect and deliver products to a set of customers. In the MPDPSL, each product has its pickup customer and the correspondent delivery customer, which can be visited more than once. This problem can be found in practice, for example in bulk product transportation by ship, and it also belongs to the $\mathcal{NP}$-Hard class. To solve the MPDPSL, it is proposed a heuristic algorithm called SVR, based on the Variable Neighborhood Search (VNS) and including Random Variable Neighborhood Descent (RVND) for local improvements. The SVR was applied to instances from the literature and its results were proved to be better than the results of previous algorithms, outperforming them on all tested instances.

KEYWORDS. MPDPSL, VNS, RVND.

Main Area: MH - Metaheuristics

1. Introdução

Um dos problemas mais desafiadores dos últimos 50 anos na área de otimização combinatória é o Problema de Roteamento de Veículos, em inglês *Vehicle Routing Problem* (PRV) (Dantzig and Ramser, 1959). A partir de um conjunto de clientes distribuídos geograficamente e de uma frota de veículos, o objetivo do PRV, basicamente, é encontrar uma rota de custo mínimo para cada um destes veículos de tal forma que todos os clientes sejam atendidos. Tal rota deve iniciar e finalizar no depósito, além disso ao designar os clientes a serem atendidos, a capacidade do veículo deve ser respeitada. A última imposição define que cada cliente tenha sua demanda suprida em apenas uma única visita.

Em Dror and Trudeau (1989), os autores retiraram esta última restrição, possibilitando que a demanda de um cliente possa ser atendida através de entregas fracionadas. Tal relaxação consolidou-se no Problema de Roteamento de Veículos com Entregas Fracionadas (PRVEF), em inglês *Split Delivery Vehicle Routing Problem*. A princípio, pode-se pensar que o fato de se permitir entregas fracionadas resultará em aumento dos custos, uma vez que mais viagens serão realizadas pelos veículos. De fato, a permissão de entregas fracionadas, por ser uma relaxação, faz com que o espaço de soluções fique muito maior, porém existe a possibilidade de se encontrar melhores soluções que reduzirão os custos totais de viagens, podendo até reduzir o número de veículos utilizados.

Após a definição do PRVEF, outras características foram incorporadas ao mesmo. Características tais como a abordagem de coleta e entrega um-para-um, criando-se assim o Problema de Roteamento de Veículos com Coleta e Entrega Fracionadas Um-para-um (PRVCEFU). Neste problema a frota de veículos deve atender a clientes de coleta e entrega, onde cada cliente de coleta possui seu cliente de entrega correspondente. Além disso, estas coletas e entregas também poderão ser fracionadas.

Considera-se que o estudo deste problema tem importância tanto prática quanto teórica. De fato, Nowak et al. (2008) trata o PRVCEFU ao abordar um problema prático de uma empresa de logística que presta serviços terceirizados. Além disso, Şahin et al. (2013) também apresenta outras aplicações práticas, como transporte a granel de produtos por navio, onde cada carga já está embalada em múltiplos contêineres, e serviços de correio que coletam e entregam múltiplos pacotes entre os mesmos pares de origem e destino. No âmbito teórico, mesmo sendo uma relaxação, o PRVCEFU é um problema de difícil solução e pertence à classe $\mathcal{NP}$-Difícil.

Dada a grande dificuldade de encontrar soluções ótimas para problemas da classe $\mathcal{NP}$-Difícil, frequentemente recorre-se à utilização de estratégias heurísticas. Desta forma, visa-se a obtenção de soluções de qualidade em tempos computacionais aceitáveis. Desta maneira, foi desenvolvido um algoritmo heurístico, nomeado SVR, que é baseado na meta-heurística *Variable Neighborhood Search* (VNS) (Mladenović and Hansen, 1997) e no método *Random Variable Neighborhood Descent* (RVND) (Souza et al., 2010; Subramanian et al., 2010) para resolver o PRVCEFU. A construção de uma solução inicial no SVR é realizada por uma heurística construtiva adaptativa gulosa que utiliza o critério do vizinho mais próximo. O método RVND é utilizado na fase de busca local do VNS e tem como característica o sorteio aleatório da ordem de aplicação das buscas locais a cada chamada do mesmo. É incorporado ao SVR a exploração de estruturas de vizinhanças clássicas e estruturas de vizinhanças específicas para tratar coleta e entrega fracionadas.

O restante deste artigo está estruturado como segue. Na seção 2 o problema tratado, PRVCEFU, é caracterizado. A revisão bibliográfica é realizada na Seção 3, destacando as principais fontes que serviram de inspiração para o desenvolvimento deste artigo. A Seção 4 apresenta a metodologia que foi utilizada durante o desenvolvimento do artigo. Resultados computacionais são apresentados e discutidos na Seção 5. Finalmente, na Seção 6, são apresentadas as conclusões obtidas a partir do desenvolvimento deste trabalho, bem como as etapas futuras necessárias para aperfeiçoá-lo.

2. Problema de Roteamento de Veículos com Coleta e Entrega Fracionadas Um-para-um

O Problema de Roteamento de Veículos com Coleta e Entrega Fracionadas Um-para-um (PRVCEFU), em inglês *Multi-Vehicle One-to-one Pickup and Delivery Problem with Split Loads*, foi definido por Şahin et al. (2013) ao generalizar sua versão com apenas um veículo, apresentada em Nowak et al. (2008).

Seja um grafo $G = (V, E)$, onde $V = \{P \cup D \cup \{0, 2n + 1\}\}$ é o conjunto de vértices que possui n pares *p-d* de clientes de coleta e clientes de entrega, além dos vértices $\{0, 2n + 1\}$, que representam localizações de dois depósitos. O conjunto $P = \{1, 2, ..., n\}$ define os clientes de coleta, enquanto o conjunto $D = \{n+1, ..., 2n\}$ representa os clientes de entrega correspondentes. Desta forma, cada carga i possui um local para coleta $i \in P$ e um local correspondente para entrega $(n + i) \in D$. O conjunto de arestas é definido como $E = \{(i, j)|i, j \in V^2\}$.

Existe uma frota homogênea que possui m veículos disponíveis, representada pelo conjunto $K = \{1, 2, ..., m\}$, com cada veículo $k \in K$ possuindo uma mesma capacidade Q. Para cada vértice $i \in V$ está associada uma quantidade de carga q_i, que será $q_i > 0$ para o cliente de coleta $i \in P$ e $q_{n+i} = -q_i$ para o cliente de entrega $(n + i) \in D$. Denota-se ainda que $q_0 = q_{2n+1} = 0$. c_{ij} é o custo associado a uma aresta $(i, j) \in E$ e L é a máxima distância que um veículo poderá percorrer. Quando um veículo se encontra em um cliente de coleta, existe a opção de coletar toda a carga ou apenas uma parte dela. Além disso, quando um veículo se encontra em um cliente de entrega, toda a carga (ou parte dela) destinada à este cliente deve ser entregue ao mesmo.

No PRVCEFU deve-se encontrar um roteamento utilizando no máximo m veículos, com cada roteamento contendo pares *p-d* correspondentes de coleta e entrega, que atenda todas as requisições de coleta e entrega, respeitando sempre a capacidade dos veículos, a máxima distância permitida e a precedência da coleta sobre a entrega de uma mesma carga. Cada veículo deve partir do depósito inicial e terminar sua rota no depósito final, que poderá estar na mesma localização do depósito inicial.

O problema de otimização relacionado ao PRVCEFU é o de encontrar o roteamento de custo mínimo que satisfaça tais restrições. Segundo provado em Nowak et al. (2008) e Şahin et al. (2013), o PRVCEFU também pertence à classe $\mathcal{NP}$-Difícil.

Um possível roteamento para sete pares *p-d* de clientes de coleta e entrega que são atendidos por dois veículos $\{r_1, r_2\}$ está exemplificado pela Figura 1. No exemplo, o conjunto de clientes de coleta é $P = \{1, 2, 3, 4, 5, 6, 7\}$ e o conjunto dos respectivos clientes de entrega é $D = \{8, 9, 10, 11, 12, 13, 14\}$. Para facilitar o entendimento, cada cliente de coleta e entrega correspondente está representado pela mesma cor. Pode-se perceber também que o depósito inicial e o depósito final é o mesmo, representado pelo 0.

Ao analisar o par *6-13* pode-se verificar que no cliente 6 devem ser coletados $q_6 = +8$ e esta carga deverá ser entregue no cliente 13, ou seja, $q_13 = -8$. Tal demanda é suprida tanto pelo veículo r_1 quanto pelo veículo r_2. Seja $z_{ij} > 0$ a quantidade de carga coletada pelo veículo i na j-ésima visita e $z_{ij} < 0$ a quantidade de carga entregue pelo veículo i na j-ésima visita. No exemplo, é visto que $z_{12} = +3$, $z_{13} = -3$, $z_{25} = +5$ e $z_{26} = -5$, isto significa que o veículo 1 irá coletar uma quantia de 3 unidades no cliente 6 na segunda viagem e entregar esta quantia no cliente 13 na terceira viagem e o veículo 2, por sua vez, irá coletar 5 unidades no cliente 6 em sua quinta visita e entregá-las no cliente 13 na sexta visita. Suprindo, portanto, as demandas do par *6-13*.

2.1. Coleta e Entrega Fracionadas

Conforme dito anteriormente, o PRVCEFU possui a característica de permitir que um veículo colete apenas uma parte da carga em um cliente de coleta, desta maneira, múltiplas viagens deverão ser realizadas para poder suprir a demanda total do par *p-d* em questão. Entretanto, o fato da demanda de um par *p-d* ser suprida por mais de uma viagem não é necessariamente definido como coleta e entrega fracionada. Por exemplo, se uma demanda qualquer for maior do que a capacidade do veículo, serão necessárias pelo menos duas viagens para poder suprir esta demanda.

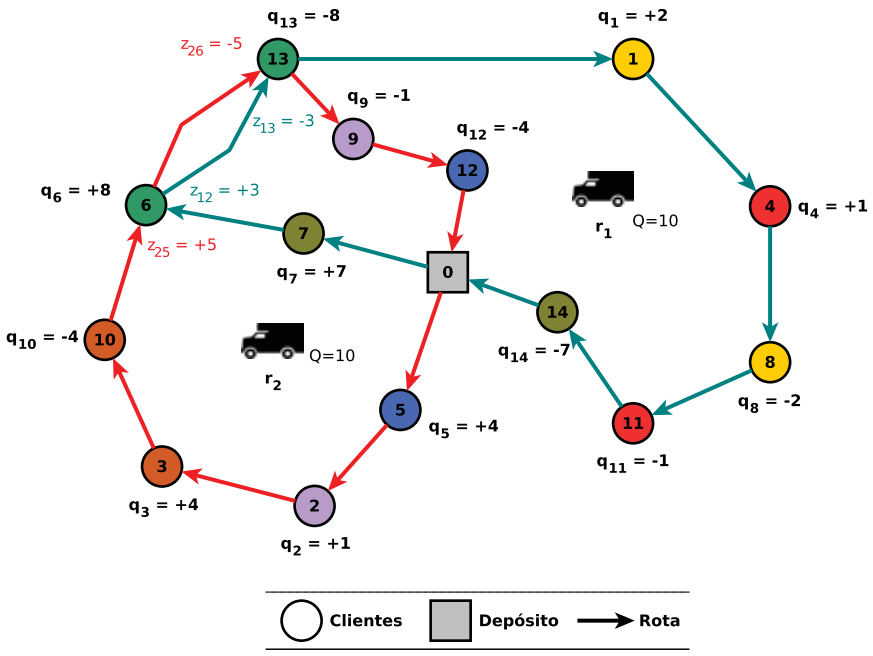

Figura 1: Exemplo de um possível roteamento para o PRVCEFU

A definição de coleta e entrega fracionada adotada no PRVCEFU é a mesma que pode ser encontrada na literatura sobre PRVEF (Archetti et al., 2006). Adota-se a definição de coleta e entrega fracionada quando uma carga é particionada em mais do que o número mínimo de divisões necessário para se atender a demanda em sua totalidade. Sendo assim, caso uma demanda seja $2,4$ e a capacidade do veículo seja 1, o número mínimo de viagens para atender tal demanda é 3, se esta demanda for suprida com 4 ou mais viagens, pode-se dizer que foi aplicada a ideia de coleta e entrega fracionada.

3. Revisão Bibliográfica

O problema clássico de Roteamento de Veículos com Coleta e Entrega é amplamente encontrado na literatura, porém o primeiro trabalho a tratar o PRV com Coleta e Entrega Fracionadas foi Mitra (2005). Neste trabalho o autor considera a existência de uma frota homogênea de veículos que irão realizar coletas e entregas simultâneas, além disso não existe nenhuma restrição quanto à demanda (de entrega ou coleta) de um cliente. Outra característica do problema abordado é que um cliente pode ser visitado por mais de um veículo e mais de uma vez pelo mesmo veículo. Os objetivos são determinar o número mínimo de veículos necessários para satisfazer as demandas de coleta e entrega, além de definir as rotas dos veículos de forma que o custo total de todas as rotas seja o mínimo. O trabalho propõe uma formulação de Programação Inteira Mista (PIM) para este problema e também uma heurística de construção de rotas. A heurística primeiramente determina o número mínimo de veículos necessários e em seguida constrói rotas com base no critério da inserção mais barata. A heurística conseguiu atingir o custo ótimo ou no máximo igual ao limite superior em 78 de 110 problemas testes.

Mais adiante em Mitra (2008), o mesmo problema foi estudado e foi proposta além de uma nova formulação de PIM, uma heurística de construção de rotas que utiliza uma técnica de clusterização paralela. Os testes foram realizados nas mesmas instâncias utilizadas em Mitra (2005) e análises estatísticas mostraram a superioridade da nova proposta.

Em Nowak et al. (2008), os autores quantificaram o benefício existente ao considerar coleta e entrega fracionadas no Problema de Coleta e Entrega Fracionadas Um-para-um (PCEFU).

Este problema, até então ainda não estudado, tem como objetivo encontrar uma rota de custo mínimo que apenas um veículo deverá percorrer de modo que o serviço requerido seja atendido. Tal serviço deve atender um cliente de coleta e o cliente de entrega correspondente, nesta ordem. Ainda, este veículo poderá ou não realizar coletas de parte ou da totalidade da carga. Quando um veículo chega em um destino, toda a carga endereçada a este destino é retirada do mesmo. Foi desenvolvida uma heurística para resolver este problema e instâncias aleatórias de grande escala foram criadas. Constatou-se que o benefício do particionamento de cargas está estreitamente ligado a três características: tamanho da carga, custo associado à coleta ou entrega e a frequência com que as cargas possuem origens e destinos em comum. Foi mostrado ainda que para um dado conjunto de origens e destinos, o maior benefício ocorre quando o tamanho da carga é maior que a metade da capacidade do veículo.

A seguir, em Nowak et al. (2009), foi realizada uma análise empírica da heurística apresentada em Nowak et al. (2008). Os autores observaram que quando as demandas estão entre 51% e 60% da capacidade do veículo, pode-se economizar até 30% nos custos de transporte. Outros fatores que contribuem para o aumento dos benefícios do uso de coleta e entrega fracionadas são: o número de cargas que serão coletadas ou entregues em uma localização comum e a distância média de uma origem para um destino com relação à distância de uma origem para um origem e de um destino para um destino.

Uma variante do problema abordado em Nowak et al. (2008) pode ser encontrada em Thangiah et al. (2007), onde há a inclusão de janelas de tempo. Neste trabalho há a apresentação e aplicação de uma heurística em conjuntos estáticos e de tempo real.

Em Şahin et al. (2013) os autores generalizaram o PCEFU (Nowak et al., 2008) para considerar a utilização de mais de um veículo e foi definido o Problema de Roteamento de Veículos com Coleta e Entrega Fracionadas Um-para-um (PRVCEFU). Os autores desenvolveram uma heurística que se baseia na Busca Tabu e na meta-heurística *Simulated Annealing* para resolver tal problema. A solução inicial é construída pelo algoritmo de Clarke and Wright (1964) e em seguida melhorada utilizando buscas locais que utilizam movimentos de troca de pares de clientes de coleta e entrega, movimentos de realocação dos clientes de coleta/entrega e movimentos que se baseiam na criação de pares de coleta e entrega fracionadas. A meta-heurística *Simulated Annealing* é utilizada para escolher qual movimento deve ser aplicado, levando em consideração a lista tabu construída ao longo da execução do algoritmo. Apesar de tratar o PRVCEFU, os autores testaram o algoritmo nas instâncias de Nowak et al. (2008), logo, os resultados apresentados são de soluções que utilizam apenas um veículo. Mesmo assim, o algoritmo foi capaz de melhorar a maioria das soluções encontradas por Nowak et al. (2008).

4. Metodologia

Conforme constatado, existem apenas dois trabalhos na literatura que abordam o PRVCEFU, entretanto, os mesmos apenas disponibilizaram resultados para a versão do PRVCEFU utilizando somente um veículo. Assim sendo, o presente trabalho visa a abordagem desta mesma versão do problema, para que os resultados possam ser comparados a fim de validar a metodologia utilizada.

4.1. Representação e Avaliação de uma Solução

Uma solução do PRVCEFU é representada explicitamente como uma permutação de clientes de coleta e entrega com suas respectivas quantidades de carga associadas.

A avaliação de uma solução do PRVCEFU é realizada através do cálculo da soma dos custos das viagens.

4.2. Estruturas de Vizinhança

Para entender o funcionamento das estruturas de vizinhanças abordadas, primeiramente deve-se compreender a definição de Bloco. Um bloco B_i é um caminho que começa em um cliente de coleta i e termina no cliente de entrega correspondente $n+i$. É chamado de bloco simples quando

não existe nenhum vértice no caminho entre i e $n+i$. Um bloco B_i é definido como bloco composto quando se tem pelo menos um bloco B_j em B_i tal que a $pos(i) < pos(j) < pos(n+j) < pos(n+i)$, onde $pos(i)$ é a posição do cliente i no roteamento. É importante ressaltar que um bloco composto não pode conter um cliente de coleta sem o seu cliente de entrega correspondente e vice-versa.

Em seguida, é descrita a aplicação de cada movimento em cada uma das estruturas de vizinhança abordadas.

$N'^{(1)}$**:** ***PairSwap*** seleciona dois pares de clientes *i-(n+i)* e *j-(n+j)* e troca a posição de i com j e a posição de $n+i$ com $n+j$.

$N'^{(2)}$**:** ***PairShift*** toma um par de clientes *i-(n+i)* e o realoca em outra posição da rota. Primeiramente é realocado o cliente de coleta i e, em seguida, busca-se a melhor posição para inserir o cliente de entrega correspondente que respeita as restrições de precedência e de capacidade do veículo.

$N'^{(3)}$**:** ***PickShift*** realiza a realocação de um cliente de coleta i em outra posição, anterior ao cliente de entrega, da rota.

$N'^{(4)}$**:** ***DelShift*** realiza a realocação de um cliente de entrega i em outra posição, posterior ao cliente de coleta, da rota.

$N'^{(5)}$**:** ***BlockSwap*** seleciona um bloco B_i e outro bloco B_j e os troca de lugar.

$N'^{(6)}$**:** ***BlockShift*** realoca um bloco B_i em outra posição da rota.

$N'^{(7)}$**:** ***SplitPert*** seleciona um par de clientes *i-(n+i)* e procura por uma posição j da rota, cuja a realocação do cliente de coleta i viole a restrição de capacidade do veículo. Isto significa dizer que $z_j + q_i > Q$, onde z_j é a quantidade de carga coletada pelo veículo na posição j, q_i é a demanda (parcial) do cliente de coleta e Q é a capacidade do veículo. Desta maneira, será inserida uma cópia do cliente de coleta i com a demanda igual à capacidade residual $q'_i = Q - z_j$ na posição j. O cliente de entrega *(n+i)*, por sua vez, será inserido em sua melhor posição, posteriormente ao cliente i.

4.3. Algoritmo Proposto

O algoritmo proposto para resolver o PRVCEFU combina os procedimentos heurísticos *Variable Neighborhood Search* (VNS) e *Random Variable Neighborhood Descent* (RVND). A ideia consiste em realizar as buscas locais do VNS através do método RVND. O pseudocódigo deste algoritmo, nomeado SVR, é mostrado no Algoritmo 1.

O Algoritmo 1 recebe como entrada a função de avaliação $f(.)$, o conjunto de vizinhanças $N(.)$ e o tempo limite de execução do algoritmo, $tempoExec$. No início da execução do SVR, três soluções (linha 1) vazias são criadas. Em seguida, na linha 2, é gerada uma solução através de uma heurística construtiva adaptativa gulosa. Uma vez obtida uma solução, a variável que armazena a melhor solução conhecida até então, $melhorSol$, é atualizada (linha 3).

As linhas 4-18 controlam o funcionamento do algoritmo, terminando quando o tempo limite acabar, ou seja, quando $tempoAtual > tempoExec$. É inicializada a primeira estrutura de vizinhança na linha 5. O próximo laço (linha 6 a 17) é executado até que todas as estruturas de vizinhanças sejam exploradas. Um vizinho s' é gerado de forma aleatória a partir de $N^{(k)}(s)$, com k sendo a vizinhança atual e s a solução corrente (linha 7). A busca local é realizada através do método RVND neste novo vizinho, linha 8. Nas linhas 9 a 13 é averiguado se as alterações feitas na solução corrente s, gerando a solução s'', contribuíram para uma melhor qualidade da mesma. Caso isto se verifique, a nova solução corrente passa a ser s'' (linha 10), a melhor solução conhecida é atualizada (linha 11) e a busca é reiniciada a partir da primeira estrutura de vizinhança (linha 12). Ao final do processo, a melhor solução obtida $melhorSol$ é retornada (linha 19).

Algoritmo 1: SVR

```
Entrada: f(.), N(.), tempoExec
Saída: melhorSol
Solucao s, s', melhorSol;
s ← geraSolucaoInicial();
melhorSol ← s;
enquanto tempoAtual ≤ tempoExec faça
    k ← 1 ;                          /* Estrutura de vizinhança corrente */
    enquanto (k ≤ |N|) faça
        Gere um vizinho qualquer s' ∈ N^(k)(s);
        s'' ← RVND(s');
        se (f(s'') < f(s)) então
            s ← s'' ;
            atualizaMelhor(s, melhorSol) ;
            k ← 1 ;
        fim
        senão
            k ← k + 1;
        fim
    fim
fim
Retorne melhorSol ;
```

4.4. Heurística Construtiva Adaptativa Gulosa

Para se construir uma solução do PRVCEFU foi utilizada uma heurística construtiva adaptativa gulosa baseada no critério do vizinho mais próximo ao cliente de coleta.

Primeiramente, cria-se uma lista LC de pares candidatos $LC = \{(1, n+1), (2, n+2), .., (n, 2n+1)\}$ contendo todos os pares *p-d* de clientes de coleta e clientes de entrega. A partir desta lista LC, classificam-se os pares *p-d* em ordem crescente de acordo com a função $g(p, i)$. Esta função calcula para um cliente de coleta p do par *p-d* respectivo, qual a distância a ser incrementada no roteamento se p for inserido na posição i. É avaliada a inserção de cada cliente de coleta p em todas as posições i do roteamento parcial por meio da função $g(p, i)$. Procura-se, desta forma, obter em qual posição o cliente p acarretará a menor distância a ser percorrida, o chamado $g_{\min}$.

O $g_{\min}$ é definido como $\min\{g(p, i), \forall p \in LC, \forall i\}$; sendo $g(p, i) = c_{i,p} + C$, p o cliente de coleta candidato, i a posição a ser inserido o cliente p na rota e C o custo de roteamento parcial da rota. Em seguida, ao se obter o $g_{\min}$, o cliente p será inserido em sua posição respectiva no roteamento parcial.

Após a inserção de um cliente p, o procedimento verifica qual a melhor posição para se inserir o seu respectivo cliente de entrega d, respeitando as restrições de capacidade do veículo e de precedência. Ao final, retira-se o par *p-d*, que foi inserido na solução parcial, de LC.

A heurística termina quando todos os pares de clientes forem designados à alguma posição do roteamento, produzindo então uma solução, s, viável. Esta solução é retornada pela heurística.

O algoritmo é dito adaptativo porque a escolha de um cliente de coleta a ser inserido em cada iteração depende do roteamento pré-existente.

4.5. *Variable Neighborhood Search*

A meta-heurística *Variable Neighborhood Search* (VNS), implementada no SVR, realiza as trocas de estruturas de vizinhanças baseando-se nas seguintes estruturas, ordenadas pela sequência de exploração: [*PickShift*, *DelShift*, *PairSwap*, *PairShift*, *BlockSwap*, *BlockShift*, *SplitPert*].

Conforme uma solução é gerada aleatoriamente por uma destas estruturas de vizinhanças, o módulo RVND (Seção 4.6) é acionado para que buscas locais sejam realizadas com a intenção de melhorar esta nova solução. Assim que as buscas locais são realizadas, aceita-se a solução

resultante se a mesma gerou uma diminuição da distância total percorrida pelo veículo. Se isto for verdade, então a solução corrente do VNS passa a ser esta nova solução e a busca é reiniciada a partir da primeira estrutura de vizinhança. Do contrário, a busca continua na próxima estrutura de vizinhança e encerra quando o tempo limite de processamento for atingido.

4.6. *Variable Neighborhood Descent* com Ordem Aleatória

No método *Variable Neighborhood Descent* com ordem aleatória (*Random Neighborhood Variable Descent* – RVND) não existe uma pré-definição da ordem de aplicação das buscas locais. As buscas locais no RVND, utilizadas no SVR, são realizadas pelo Método de Descida Completa e se baseiam nas seguintes estruturas de vizinhanças: [*PickShift*, *DelShift*, *PairSwap*, *BlockSwap*, *BlockShift*].

Não é estabelecida uma ordem de processamento fixa destas buscas locais, ou seja, a cada chamada do método RVND, uma ordem de processamento dessas buscas é definida de forma aleatória. O critério de aceitação da solução gerada é relacionado à diminuição da distância total percorrida pelo veículo. Caso isso se concretize, então a solução corrente do RVND passa a ser esta nova solução e a busca é reiniciada com a primeira estrutura de vizinhança. Caso contrário, a busca continua na próxima estrutura de vizinhança e encerra quando todas as vizinhanças forem analisadas.

5. Resultados Preliminares

De maneira a testar o algoritmo SVR, tomou-se 10 instâncias geradas em Nowak et al. (2008). Destas 10 instâncias, 5 possuem 75 pares de clientes de coleta e entrega e 5 possuem 100 pares de clientes de coleta e entrega. Além disso, todas as instâncias foram geradas de forma que cada carga ocupe de 51% a 60% da capacidade do veículo, tal fato induz a utilização de coleta e entrega fracionadas para que se possa obter diminuição dos custos de distribuição.

O algoritmo SVR foi desenvolvido na linguagem C++ e cada experimento foi executado utilizando somente um núcleo de um computador com processador *Intel Core 2 Quad 2,4 GHz* com 4 GB de memória RAM e sistema operacional *Linux* (*Ubuntu 14.04*). Nota-se que o único parâmetro de entrada do SVR é o critério de parada, sendo o tempo máximo de duração do processamento $tempoExec$, em minutos. É importante ressaltar ainda que os tempos limites dados ao SVR são iguais aos tempos encontrados nos trabalhos de Nowak et al. (2008) e Şahin et al. (2013). Sendo 25,50 minutos para cada instância de 75 pares de clientes e 56,20 minutos para cada instância com 100 pares de clientes. Outro ponto importante é que a configuração do computador utilizado também foi a mesma utilizada nos testes de Şahin et al. (2013).

A métrica adotada para realizar a comparação dos algoritmos, dada pela Eq. (1), tem como objetivo verificar a variabilidade das soluções finais produzidas pelos algoritmos. Nesta medida, calcula-se, para cada algoritmo *Alg* aplicado a um problema-teste i, o desvio percentual GAP_i da solução encontrada $\bar{f}_i^{Alg}$ em relação à melhor solução f_i^* conhecida até então. De maneira a verificar a confiabilidade do algoritmo SVR, cada instância foi executada 30 vezes.

$$GAP_i = \frac{\bar{f}_i^{Alg} - f_i^*}{f_i^*} \tag{1}$$

Os trabalhos encontrados na literatura somente disponibilizam um valor de solução para cada instância. Além disso, não se tem a informação de quantas execuções foram realizadas para que estes valores fossem alcançados. Desta maneira, estes resultados serão comparados com o GAP_avg, que é o desvio percentual do valor médio das soluções geradas pelo SVR. O GAP_avg é calculado ao final das 30 execuções utilizando-se o valor médio das soluções encontradas.

Na Tabela 1, nas colunas Nowak e Sahin, para cada instância, são encontrados o valor da solução (Valor) e seu desvio percentual relativo (GAP) obtidos pelos algoritmos de Nowak et al. (2008) e Şahin et al. (2013). Ainda, na coluna SVR estão relacionados, para cada instância, o valor médio das soluções encontradas pelo SVR (Valor Médio) e o desvio percentual relativo (GAP_avg).

Os menores valores para cada medida e para cada instância estão em negrito. Destaca-se ainda que quando se tem GAP igual a 0,00 significa que esta é a melhor solução conhecida.

Tabela 1: Soluções obtidas e GAP dos algoritmos de Nowak et al. (2008), Şahin et al. (2013) e SVR

	Nowak[1]		Sahin[2]		SVR[2]	
Instância	**Valor**	**GAP(%)**	**Valor**	**GAP**	**Valor Médio**	**GAP_avg**
75_1A	**3830,12**	**0,00**	3894,34	1,68	3914,38	2,20
75_1B	3857,11	0,37	**3842,88**	**0,00**	3883,91	1,07
75_1C	3810,50	0,54	**3790,03**	**0,00**	3890,32	2,65
75_1D	**3799,32**	**0,00**	3862,22	1,66	3888,07	2,34
75_1E	3868,95	1,26	**3820,87**	**0,00**	3911,93	2,38
100_1A	5073,39	1,62	**4992,59**	**0,00**	5030,80	0,77
100_1B	**5036,54**	**0,00**	5042,29	0,11	5050,59	0,28
100_1C	5029,38	0,29	**5015,08**	**0,00**	5036,03	0,42
100_1D	5012,97	0,34	**4996,08**	**0,00**	5049,68	1,07
100_1E	5130,15	2,29	**5015,26**	**0,00**	5015,35	0,00
Média		0,67		**0,34**		1,32

[1]Testes realizados em um *Xeon* com 2,4 GHz e 2 GB de RAM
[2]Testes realizados em um *Intel Core 2 Quad 2,4 GHz* e 4 GB de RAM

Nota-se que o algoritmo de Sahin é o algoritmo capaz de obter as melhores soluções, pois além de ter uma variabilidade baixa de GAP, somente em três instâncias ele não consegue alcançar a melhor solução conhecida. O algoritmo de Nowak, por sua vez, é capaz de alcançar três melhores soluções conhecidas e também possui variabilidade baixa de GAP. Levando-se em consideração que a comparação é realizada sobre a média das soluções produzidas pelo SVR após 30 execuções, pode-se dizer que o SVR também possui variabilidade baixa de GAP, sendo no pior caso 2,38 e no melhor caso ele foi capaz de chegar encontrar a melhor solução conhecida.

A visualização destes dados é melhor realizada a partir de um *box plot* dos valores de GAP para cada algoritmo (Figura 2). Neste *box plot*, percebe-se uma leve semelhança entre os valores encontrados em Nowak et al. (2008) e os valores encontrados no SVR. Além do mais pode-se perceber que o algoritmo de Sahin se destaca dos outros dois.

A partir das análises feitas, tem-se indícios de que o algoritmo de Sahin é capaz de encontrar melhores soluções do que os outros dois algoritmos. Com o intuito de validar esses indícios, foi realizado um teste não paramétrico de Friedman (Friedman, 1937), com nível de significância de 5% e baseando-se nos valores de GAP encontrados pelos três algoritmos. O resultado do teste foi $\chi^2(2) = 10,20$ e $p = 0,006081$, isso mostra que existe diferença estatística entre os 3 algoritmos, ao analisar os valores de GAP.

Para verificar onde estão estas diferenças, foi utilizado um teste de Wilcoxon (Wilcoxon, 1945), também com nível de significância de 5%. A Tabela 2 contém os valores p obtidos em cada teste realizado para cada par de algoritmos. Pode ser visto pelos valores p que o algoritmo SVR se difere estatisticamente do algoritmo de Sahin, pois o valor p é menor do que o nível de significância. Em outro teste, foi verificado que o algoritmo SVR não se difere do algoritmo de Nowak, sendo o valor p maior do que o nível de significância. Também se verifica que o algoritmo de Sahin e de Nowak, não são diferentes estatisticamente, pois o valor p é maior do que o nível de significância.

Tabela 2: Resultados do teste de Wilcoxon

Algoritmos	Valor p
Sahin-Nowak	0,375
SVR-Nowak	0,1934
SVR-Sahin	0,009091

Desta forma, para estas instâncias e com base em uma análise estatística, o único algo-

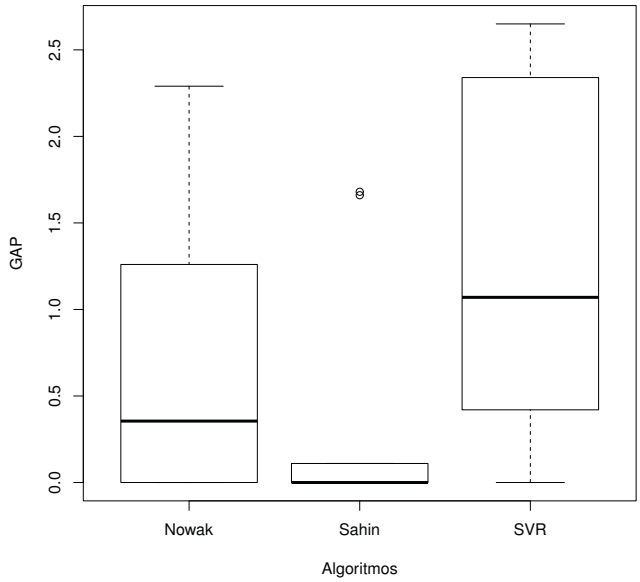

Figura 2: *Box plot* mostrando os GAPs dos algoritmos de Nowak et al. (2008), Şahin et al. (2013) e SVR

ritmo que parece se diferenciar de algum outro é o algoritmo de Sahin, que se mostrou diferente do que o SVR. Porém, necessita-se de uma amostra maior para poder garantir estatisticamente tal fato, visto que o teste não comprovou diferença do mesmo sobre o algoritmo de Nowak.

Na Tabela 3 estão disponibilizados os resultados completos alcançados pelo SVR com 30 execuções em cada instância. A tabela contém para cada instância, a pior solução (Pior), a melhor solução (Melhor), o valor médio (Média), a mediana (Mediana), o desvio percentual relativo médio (GAP_avg), o desvio percentual em relação à melhor solução conhecida (GAP_m) e o desvio padrão (Desv Pad). Destaca-se ainda que os valores negativos de GAP indicam que as encontradas são melhores do que as melhores conhecidas até então (obtidas de Nowak et al. (2008) ou Şahin et al. (2013)).

Tabela 3: Resultados completos do SVR com 30 execuções em cada instância

Instância	Pior	Melhor	Média	Mediana	GAP_avg	GAP_m	Desv Pad
75_1A	4077,50	3806,85	3914,38	3918,24	2,20	-0,61	63,32
75_1B	4101,92	3771,68	3883,91	3878,43	1,07	-1,85	67,64
75_1C	4012,20	3776,79	3890,32	3897,06	2,65	-0,35	54,96
75_1D	4012,89	3767,58	3888,07	3880,57	2,34	-0,84	62,00
75_1E	4040,67	3791,31	3911,93	3908,69	2,38	-0,77	53,94
100_1A	5212,18	4919,63	5030,80	5035,01	0,77	-1,46	63,52
100_1B	5138,15	4957,67	5050,59	5063,83	0,28	-1,57	56,61
100_1C	5151,68	4930,37	5036,03	5031,98	0,42	-1,69	54,85
100_1D	5144,08	4961,07	5049,68	5045,02	1,07	-0,70	51,30
100_1E	5121,94	4915,11	5015,35	5016,37	0,00	-2,00	58,79
				Média	**1,32**	**-1,18**	**58,69**

É notável que o SVR foi capaz de superar as melhores soluções conhecidas em todas as instâncias testadas, com o GAP_m médio em relação às melhores soluções sendo de -1,18%. Além disso, o desvio padrão médio também pode ser considerado baixo, sendo 58,69.

Percebe-se que o algoritmo SVR é capaz de encontrar soluções de melhor qualidade do

que os outros dois algoritmos. Para verificar isto, novamente foi realizado um teste não paramétrico de Friedman (Friedman, 1937), com nível de significância de 5% e se baseando nos valores das melhores soluções encontradas pelo SVR – Tabela 3, Nowak – Tabela 1 e Sahin – *100 restarts* (Şahin et al., 2013). O resultado do teste foi $\chi^2(2) = 12,8$ e $p = 0,001662$, isso significa que existe diferença estatística ao comparar os valores de GAP dos outros algoritmos e o valor de GAP_m do SVR.

Mais uma vez utilizou-se um teste de Wilcoxon (Wilcoxon, 1945), também com nível de significância de 5% para verificar onde estão tais diferenças. A Tabela 4 contém os valores p obtidos nos testes realizados. Pode ser visto pelos valores p que o SVR se difere estatisticamente tanto do algoritmo de Şahin et al. (2013) quanto do algoritmo Nowak et al. (2008), por conta desses valores serem menores que o nível de significância. Desta forma, pode-se concluir que para estas instâncias e com base em uma análise estatística, o SVR é o algoritmo capaz de obter as soluções de melhor qualidade para o PRVCEFU.

Tabela 4: Resultados do teste de Wilcoxon

Algoritmos	Valor p
SVR-Nowak	0,009766
SVR-Sahin	0,001953

6. Conclusões

Este trabalho propôs o estudo do Problema de Roteamento de Veículos com Coleta e Entrega Fracionadas Um-para-um (PRVCEFU). Este problema é de grande importância, pois além de ser encontrado na prática, ele também pertence à classe $\mathcal{NP}$-Difícil. Devido a este fato, instâncias de maior porte podem requerer tempos de processamento inviáveis na busca por uma resolução exata.

Desta maneira, o desenvolvimento de um algoritmo heurístico é incentivado, visando a busca por soluções de qualidade em tempos aceitáveis. A partir de instâncias obtidas da literatura, o algoritmo nomeado SVR foi implementado e aplicado para resolver o Problema de Roteamento de Veículos com Coleta e Entrega Fracionadas Um-para-um (PRVCEFU) com apenas um veículo. O SVR se baseia na meta-heurística *Variable Neighborhood Search* (VNS) e no método *Random Variable Neighborhood Descent* (RVND). A construção da solução inicial é realizada por uma heurística construtiva adaptativa gulosa que se baseia no critério do vizinho mais próximo. O método RVND é utilizado para realizar as buscas locais do VNS e caracterizado por sortear a ordem de aplicação das buscas locais em cada chamada do método. O SVR explora estruturas de vizinhanças clássicas e estruturas de vizinhanças específicas para tratar entregas fracionadas.

As soluções obtidas foram comparadas a dois algoritmos da literatura e, apesar de não se comprovar uma superioridade de algum algoritmo sobre todos os outros, pode se comprovar estatisticamente a capacidade do SVR de gerar soluções de melhor qualidade do que os outros dois algoritmos, superando-os em todas as instâncias testadas. Além disso, os resultados de todos os experimentos com o SVR também foram apresentados. Da mesma forma, a robustez do algoritmo foi confirmada, produzindo GAPs baixos e baixa variabilidade das soluções geradas.

Neste sentido, pode-se dizer que o SVR é um ótimo algoritmo para resolver o PRVCEFU com um veículo e existem bons indícios de que a adaptação do SVR para resolver o PRVCEFU com mais de um veículo poderá obter bons resultados. Ainda, a busca pelas melhores posições de um par de clientes *p-d* pode ser vista como um Problema de Caminho Mínimo com Recursos Limitados (em inglês, *Resource Constrained Shortest Path Problem*). A transformação para este problema e a resolução por programação dinâmica poderá tornar o SVR ainda mais poderoso e capaz de melhorar os resultados atuais. Uma vez que este algoritmo estiver implementado e seu funcionamento validado, a ideia é adaptá-lo para que ele seja capaz de tratar outras variantes como a inclusão de janelas de tempo por exemplo.

Agradecimentos

Os autores agradecem à UFF, ao CNPq e à CAPES pelo apoio ao desenvolvimento deste trabalho.

Referências

Archetti, C., Savelsbergh, M. W. P., and Speranza, M. G. (2006). Worst-Case Analysis for Split Delivery Vehicle Routing Problems. *Transportation Science*, 40(2):226–234.

Clarke, G. and Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581.

Şahin, M., Çavuşlar, G., Öncan, T., Şahin, G., and Tüzün Aksu, D. (2013). An efficient heuristic for the Multi-vehicle One-to-one Pickup and Delivery Problem with Split Loads. *Transportation Research Part C: Emerging Technologies*, 27:169–188.

Dantzig, G. B. and Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1):80–91.

Dror, M. and Trudeau, P. (1989). Savings by Split Delivery Routing. *Transportation Science*, 23(2):141–145.

Friedman, M. (1937). The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701.

Mitra, S. (2005). An Algorithm for the generalized Vehicle Routing Problem with Backhauling. *APJOR*, 22(2):153–170.

Mitra, S. (2008). A Parallel Clustering Technique for the Vehicle Routing Problem with Split Deliveries and Pickups. *The Journal of the Operational Research Society*, 59(11):1532–1546.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Nowak, M., Ergun, Ö., and III, C. C. W. (2008). Pickup and Delivery with Split Loads. *Transportation Science*, 42(1):32–43.

Nowak, M., Ergun, Ö., and III, C. C. W. (2009). An empirical study on the benefit of split loads with the pickup and delivery problem. *European Journal of Operational Research*, 198(3):734–740.

Souza, M., Coelho, I., Ribas, S., Santos, H., and Merschmann, L. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, 207(2):1041–1051.

Subramanian, A., Drummond, L., Bentes, C., Ochi, L., and Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.

Thangiah, S., Fergany, A., and Awan, S. (2007). Real-time split-delivery pickup and delivery time window problems with transfers. *Central European Journal of Operations Research*, 15(4):329–349.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83.