UM ALGORITMO HÍBRIDO PARA A SOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA E JANELA DE TEMPO

Aline Aparecida de Carvalho Gonçalves, Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Av. Amazonas, 7675, CEP 30510-000, Belo Horizonte (MG), Brasil
acg.aline@gmail.com, sergio@dppg.cefetmg.br

Carlos Alexandre Silva
Departamento de Computação - Instituto Federal de Minas Gerais
Avenida Serra da Piedade, 299, Morada da Serra, Sabará (MG), Brasil
carlos.silva@ifmg.edu.br

RESUMO

O Problema de Roteamento de Veículos com Coleta e Entrega e Janela de Tempo consiste em estabelecer rotas para atendimento a clientes, em intervalos de tempo preestabelecidos, e satisfazer suas demandas de coleta ou entrega, restritas pelo emparelhamento dos clientes e ordem de precedência nas visitas. A solução do problema é encontrar a frota mínima e as rotas capazes de atender todos os clientes e percorrer a menor distância. Este trabalho propõe uma metodologia de solução em duas fases: construção e refinamento. Na fase de construção, a solução é gerada de duas formas: a primeira, por uma adaptação da *Push-Forward Insertion Heuristic*, e a segunda utilizando a fase de construção da metaheurística *Greedy Randomized Adaptative Search Procedure*. As soluções são refinadas através da metaheurística *Iterated Local Search*, combinada com a técnica de busca local *Variable Neighborhood Descent*. A metodologia foi testada utilizando instâncias clássicas da literatura e os resultados produzidos foram satisfatórios.

PALAVRAS CHAVE. Problema de Roteamento de Veículos com Coleta e Entrega e Janela de Tempo. *Iterated Local Search*. *Variable Neighborhood Descent*.

Área Principal: Metaheurísticas (MH) e Otimização Combinatória (OC)

ABSTRACT

The Pickup and Delivery Vehicle Routing Problem with Time Window (PDVRPTW) aims to create routes to serve customers, in fixed periods of time, and to satisfy their demands for pickup and delivery of goods, restricted by pair customers and order of precedence to visit. The solution is to find the minimum fleet and the routes that serves a set of customers and travel the shortest distance. This paper proposes a methodology in two phases to solve the PDVRPTW. In the construction phase, the solution is created in two different ways: first, by adapting the Push-Forward Insertion Heuristic (PFIH) and, second, using the construction phase of metaheuristic Greedy Randomized Adaptive Search Procedure (GRASP). The solutions are refined through metaheuristic Iterated Local Search (ILS) and the Variable Neighborhood Descent (VND) local search technique. The methodology was tested using classic instances of literature and the results were satisfying.

KEYWORDS. Pickup and Delivery Vehicle Routing Problem with Time Window. Iterated Local Search. Variable Neighborhood Descent.

Main Area: Metaheuristics (MH) and Combinatorial Optimization (OC)

1. Introdução

A logística é um elemento-chave na estratégia das empresas. A etapa de transporte é o processo dessa cadeia que absorve o maior custo operacional. A maioria das empresas de distribuição de mercadorias enfrenta o desafio de construir rotas para seus veículos, visando facilitar e agilizar o fluxo de produtos entre o depósito e o consumidor final, manter um nível de serviço adequado aos seus clientes e evitar atrasos nas entregas.

O Problema de Roteamento de Veículos (PRV), introduzido por Dantzig e Ramser (1959), envolve a distribuição de bens ou serviços de um depósito aos usuários finais. O objetivo do problema é encontrar rotas para uma frota de veículos que satisfaçam a um número de cidades ou clientes dispersos geograficamente, respeitando-se as restrições existentes. Variações do PRV surgiram ao longo do tempo, devido à incorporação de uma série de restrições, como capacidade dos veículos, janela de tempo para atendimento aos consumidores, serviço de coleta e/ou entrega, tipo de frota, quantidade de veículos, relação de precedência entre cidades e/ou consumidores, quantidade de depósitos, dentre várias situações do mundo real.

O Problema de Roteamento de Veículos com Coleta e Entrega e Janela de Tempo (PRVCEJT), variação do PRV clássico, pode ser descrito como um problema no qual uma frota de veículos, inicialmente situada em um depósito, deverá atender um conjunto de clientes que possuem diferentes demandas por coleta ou entrega de produtos. Cada cliente apresenta um instante de tempo mínimo e máximo para o atendimento ser iniciado, caracterizando a restrição de janela de tempo do problema. Neste problema aqui tratado, coleta-se produtos em um determinado cliente e estes produtos são entregues em um outro cliente já especificado. Assim, a coleta e a entrega de produtos têm restrição de precedência, ou seja, o cliente com demanda de coleta deve ser visitado antes do respectivo cliente com demanda de entrega. Também em consequência da descrição feita, há restrição de emparelhamento de clientes, ou seja, o par de clientes em que é realizada a coleta e é realizada a entrega, dados que são previamente especificados, devem ser atendidos na mesma rota. A solução consiste em encontrar o número mínimo de veículos, capaz de atender a todos os clientes, e um conjunto de rotas, que minimize a distância total percorrida pelos veículos.

Conforme Ropke e Cordeau (2009), o PRVCEJT pertence à classe dos problemas de otimização combinatória que não podem ser resolvidos em tempo polinomial. Em outras palavras, pertence à classe de problemas NP-Difícil, e, na presença de janela de tempo, cada verificação de factibilidade é NP-Completo. Muitos trabalhos presentes na literatura para a resolução desse problema apresentam abordagens por heurísticas e metaheurísticas (Nanry e Barnes, 2000), (Li e Lim, 2001), (Braÿsy e Gendreau, 2005a), (Braÿsy e Gendreau, 2005b), (Lu e Dessouky, 2006), (Dondo et al., 2008), (Carabetti, 2010).

Este trabalho apresenta uma metodologia baseada na utilização de heurísticas e metaheurísticas para a solução do PRVCEJT. Dois procedimentos heurísticos foram empregados na construção da solução inicial: *Push-Forward Insertion Heuristic* (PFIH) e *Greedy Randomized Adaptive Search Procedure* (GRASP). A metaheurística *Iterated Local Search* (ILS) foi empregada para refinamento da solução, associada à técnica de busca local *Variable Neighborhood Descent* (VND).

A próxima Seção deste artigo apresenta uma visão global do PRVCEJT. A Seção 3 descreve a metodologia proposta para a solução do problema. A Seção 4 mostra os resultados computacionais obtidos pelo algoritmo, aplicado nas instâncias testes da literatura. Por fim, a Seção 5 descreve as conclusões obtidas neste trabalho.

2. Problema de Roteamento de Veículos com Coleta e Entrega e Janela de Tempo

O Problema de Roteamento de Veículos com Coleta e Entrega e Janela de Tempo (PRVCEJT), objeto de estudo deste trabalho, visa construir um conjunto de rotas para uma frota de veículos

que atenda um grupo de clientes com diferentes demandas de coleta ou entrega de mercadorias. A solução busca minimizar o número de veículos utilizados e a distância total percorrida por eles.

Considera-se uma frota homogênea de veículos, ou seja, todos os veículos são do mesmo modelo e possuem a mesma capacidade de carga. Para cada cliente, são especificados: o local de origem (coleta) e destino (entrega) da carga, a quantidade de carga a ser transportada, a janela de tempo de atendimento (instante de tempo inicial e final que o serviço pode iniciar) e o tempo de serviço para a realização do carregamento ou descarregamento.

As restrições operacionais abaixo devem ser respeitadas durante o processo de criação das rotas são consideradas essenciais para a factibilidade da solução:

(a) a sobreposição de rotas não é permitida, ou seja, um veículo não pode percorrer ao mesmo tempo mais de uma rota da solução;

(b) o depósito é o ponto inicial e final de todas as rotas da solução;

(c) o tempo de serviço associado ao depósito é nulo, pois não há atividades de carregamento ou descarregamento nesse ponto;

(d) a carga do veículo não pode exceder sua capacidade máxima em nenhum instante de tempo;

(e) a rota deve respeitar a restrição de emparelhamento: o cliente com demanda de coleta e o seu correspondente com demanda de entrega devem ser atendidos na mesma rota;

(f) a rota deve respeitar a restrição de precedência: o cliente com demanda de coleta de mercadoria deve ser visitado antes do correspondente cliente com demanda de entrega;

(g) um veículo deve partir e retornar vazio ao depósito;

(h) um veículo deve chegar ao cliente dentro da janela de tempo estipulada pelo cliente;

(i) o atendimento pode ser iniciado em qualquer instante da janela de tempo e pode ser concluído após o seu término; e

(j) um veículo não pode retornar ao depósito após o seu fechamento, ou seja, o tempo total de percurso da rota não pode ultrapassar o instante de tempo final do depósito.

Formalmente, o PRVCEJT pode ser modelado em um grafo $G = (V, A)$, em que o conjunto $V = \{v_0, v_1, ..., v_n\}$ representa os clientes. O depósito é representado pelo elemento v_0, ponto de partida e retorno de todos os veículos da frota. O problema tratado separa, de forma clara, nós de coleta e nós de entrega. Assim, define-se P como o conjunto de clientes com demanda de coleta de mercadorias e D como o conjunto de clientes com demanda de entrega, de modo que $P = \{1, ..., n/2\}$ e $D = \{n/2 + 1, ..., n\}$. Cada $v_i \in V$ possui uma demanda q_i associada, sendo $q_i > 0$ para $v_i \in P$ e $q_i < 0$ para $v_i \in D$; um tempo de serviço s_i; e uma janela de tempo $[e_i, l_i]$. Para o depósito v_0, a demanda $q_0 = 0$ e o tempo de serviço $s_0 = 0$. Entre um par de clientes $(v_i, v_j), (i \neq j, i, j = 1, 2, ...n)$ há uma distância d_{ij} e um tempo de viagem t_{ij}, sendo $d_{ij} > 0$ e $t_{ij} > 0$. Então, o conjunto A de arestas pode ser definido como $A = \{(v_i, v_j) \mid v_i, v_j \in V; v_i \neq v_j; t_{0i} + s_i + t_{ij} \leq l_j\}$. Se um veículo chegar ao cliente v_i antes de e_i, então deve aguardar até e_i para iniciar o serviço. O tempo total de percurso de uma rota não pode ser maior que l_0, que representa o momento de encerramento das atividades no depósito.

O PRVCEJT pode possuir vários objetivos e restrições, dependendo do contexto de aplicação. No caso de entrega de mercadorias, o objetivo é minimizar o número de veículos, custos e duração da viagem. Em problemas *dial-a-ride* é melhor minimizar os atrasos, devido aos inconvenientes causados pela realização do serviço antes ou depois do horário desejado. Nesse trabalho, consideram-se duas funções objetivo principais para o problema: a primeira é minimizar o número de veículos

utilizados e a segunda é minimizar a distância total percorrida, dada pela soma das distâncias entre as cidades adjacentes.

Uma análise completa do PRVCEJT é apresentada em Parragh et al. (2008a) e Parragh et al. (2008b). A importância da obtenção de boas soluções para o problema e, consequentemente, o emprego das frotas com o melhor custo-benefício se justifica pela modelagem de vários problemas logísticos e de tráfego urbano utilizando o PRVCEJT.

3. Metodologia

Este artigo apresenta o algoritmo ILS-VND, proposto para solução do problema objeto de estudo, em duas versões diferentes denominadas ILS-VND1 e ILS-VND2. Nesta seção estão descritas as principais características desse algoritmo.

3.1. Representação computacional da solução

A representação computacional adotada para uma solução do PRVCEJT é dada por um conjunto de rotas, pela soma das distâncias totais percorridas em cada rota e pelo número total de veículos empregados. Nesta representação, cada rota tem, em sua estrutura, uma lista de clientes, dispostos de acordo com a ordem de atendimento, a distância total e o tempo de percurso.

O cliente, elemento básico para a formação de uma rota, possui as seguintes informações:

(a) número do cliente, que é seu identificador único;

(b) coordenadas geográficas (coordenadas x e y), que definem a sua localização;

(c) demanda de carregamento (coleta) ou descarregamento (entrega);

(d) instante de tempo inicial e final da janela de tempo;

(e) duração do atendimento; e

(f) cliente com demanda emparelhada.

3.2. Construção da solução inicial

3.2.1. *Push-Forward Insertion Heuristic* (PFIH)

Solomon (1987) propôs a *Push-Forward Insertion Heuristic* (PFIH) para inserção de clientes nas rotas do Problema de Roteamento de Veículos com Janela de Tempo (PRVJT). Nessa heurística, os clientes são ordenados de forma crescente pelo custo de inserção e inseridos nas rotas de acordo com essa ordem. A função que calcula o custo de inserção c_i do cliente i é dada por:

$$c_i = -\alpha d_0 i + \beta l_i + \gamma((p_i/360)d_0 i) \quad (1)$$

Nesta expressão, d_{0i} é a distância do depósito ao cliente i; l_i é o instante de tempo final da janela de tempo; p_i é o ângulo da coordenada polar do cliente i referente ao depósito; e os pesos α, β, e γ recebem, respectivamente, os valores $\alpha = 0,7$; $\beta = 0,1$ e $\gamma = 0,2$, definidos empiricamente em Solomon (1987).

O processo de construção da solução apresentado em Solomon (1987) começa com o cálculo do custo de cada cliente e a criação de uma lista ordenada pelo custo de inserção. Definida essa ordem, é iniciada uma solução vazia (sem rotas). O primeiro cliente da lista cria a primeira rota da solução. A partir daí, os demais clientes da lista serão colocados nas melhores posições e rotas, sempre respeitando todas as restrições do problema tratado. Caso não exista nenhuma posição factível nas

rotas da solução atual, uma nova rota é criada na solução. Dentre todas as posições que respeitem as restrições do problema abordado, a melhor é aquela que tem a menor distância total percorrida.

Entretanto, no PRVCEJT, a inserção dos clientes nas rotas é feita aos pares e em ordem preestabelecida, em respeito às restrições de emparelhamento e precedência. O pseudocódigo do algoritmo utilizado neste trabalho para a construção da solução inicial é apresentado no Algoritmo 1.

Algoritmo 1: *Solucao PFIH*$(grafo, listadeclientes)$

```
1  Calcule o custo de inserção de cada cliente
2  Crie lista de clientes ordenados crescentemente pelo custo
3  enquanto existir algum cliente na lista faça
4      Selecione o primeiro cliente da lista
5      Selecione o cliente com demanda emparelhada ao cliente selecionado
6      Remova da lista os clientes selecionados
7      enquanto existir rotas não vazias na solução faça
8          Percorra cada rota da solução
9          Insira o cliente com demanda de coleta em uma posição da rota
10         se inserção é factível então
11             Insira o cliente emparelhado em uma posição posterior
12             se inserção é factível então
13                 Calcule a distância total percorrida na rota
14     se não existir uma inserção factível para os dois clientes então
15         Crie uma nova rota com o par de clientes selecionados
```

Nessa adaptação da PFIH para o PRVCEJT, o primeiro passo é calcular o custo de inserção dos clientes (linha 1) e ordená-los de maneira crescente (linha 2). Em seguida, inicia-se o processo iterativo de construção da solução inicial, que encerra-se somente quando a lista ordenada de clientes estiver vazia. O primeiro cliente da lista ordenada e o respectivo cliente com demanda emparelhada são selecionados e removidos da lista (linhas 4, 5 e 6). O cliente com demanda de coleta é inserido em todas as posições de todas as rotas da solução, até que seja obtida uma posição que garanta a factibilidade da solução (linhas 7, 8 e 9). Em seguida, o cliente com demanda de entrega é inserido em todas as posições posteriores à posição do cliente com demanda de coleta na mesma rota (linhas 10 e 11). Se a posição de inserção do cliente com demanda de entrega também garantir a factibilidade da solução, é calculada a distância total percorrida nessa rota (linhas 12 e 13). Se não existir uma posição factível para inserção do par de clientes em alguma rota, uma nova rota é criada (linhas 14 e 15).

3.2.2. Metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP)

A metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP), proposta por Feo e Resende (1995), é um algoritmo iterativo, composto por uma fase de construção da solução e uma fase de refinamento, em que uma busca local encontra o ótimo local da solução inicial gerada. As iterações do algoritmo são independentes, ou seja, na iteração atual não se leva em conta nenhuma informação das iterações anteriores. O critério de parada mais utilizado é o número máximo de iterações.

Neste trabalho é proposta uma adaptação da fase de construção da metaheurística GRASP para a geração de uma solução para o PVCEJT. As características gulosas e aleatórias da metaheurística são atendidas pelo cálculo do custo de inserção do cliente, conforme função proposta pela PFIH e mostrada pela expressão 1, e através de sorteio na Lista Restrita de Candidatos (LRC).

Inicialmente, é feito o cálculo do custo de inserção de cada cliente na rota e a ordenação crescente desses clientes. Em seguida, inicia-se a construção da solução, elemento a elemento, através de iterações do algoritmo. A LRC possui tamanho fixo de 03 (três) candidatos e é atualizada a cada iteração com os clientes que possuem o menor custo de inserção. Um cliente da LRC é selecionado aleatoriamente e, junto com seu respectivo par (restrição de emparelhamento do PRVCEJT), são inseridos nas posições que correspondem à menor distância percorrida e respeitem a factibilidade da solução. Caso a inserção nas rotas da solução atual não seja viável, uma nova rota é criada. O pseudocódigo desse procedimento utilizado para a construção da solução é apresentado no Algoritmo 2.

Algoritmo 2: *Solucao GRASP*$(grafo, listadeclientes)$

```
1  Calcule o custo de inserção de cada cliente
2  Crie lista de clientes ordenados crescentemente pelo custo
3  Crie a LRC
4  enquanto existir algum cliente na lista faça
5      Selecione aleatoriamente um cliente da lista LRC
6      Selecione o cliente com demanda emparelhada ao cliente selecionado
7      Atualize a lista ordenada e a LRC
8      enquanto existir rotas não vazias na solução faça
9          Percorra cada rota da solução
10         Insira o cliente com demanda de coleta em uma posição da rota
11         se inserção é factível então
12             Insira o cliente emparelhado em uma posição posterior
13             se inserção é factível então
14                 Calcule a distância total percorrida na rota
15     se não existir uma inserção factível para os dois clientes então
16         Crie uma nova rota com o par de clientes selecionados
```

3.3. Estrutura de vizinhança

A estrutura de vizinhança na busca local é obtida através dos seguintes movimentos:

(a) trocar um cliente com demanda de coleta com todos os outros clientes da mesma rota que estão nas posições anteriores ao cliente com demanda de entrega emparelhada ao selecionado inicialmente;

(b) trocar um cliente com demanda de entrega com todos os outros clientes da mesma rota que estão nas posições posteriores ao cliente com demanda de coleta emparelhada ao selecionado inicialmente;

(c) trocar um par de clientes de uma rota com outro de rota diferente, respeitando-se a restrição de emparelhamento e de precedência;

(d) realocar um par de clientes emparelhados de uma rota para outra; e

(e) eliminar rota.

Este último movimento auxilia no objetivo de redução do número de veículos do PRVCEJT e consiste na tentativa de eliminação da rota com o menor número de clientes. Primeiramente, seleciona-se a rota com o menor número de clientes da solução e os pares de clientes com demandas emparelhadas são removidos e re-inseridos em posições factíveis nas demais rotas. Se todos os clientes da rota selecionada forem realocados, a solução atual assume a nova composição de rotas e o procedimento termina. Se não for possível eliminar totalmente a rota, a solução permanece inalterada.

Esse processo se repete para as demais rotas até que o número de veículos seja minimizado ou até que todas as rotas da solução tenham sido analisadas.

3.4. Estratégias de perturbação

As perturbações aplicadas a uma solução ótima local permitem que a busca local explore diferentes regiões do espaço de busca. Neste trabalho, a estratégia de perturbação adotada consiste na aplicação sucessiva, em dois níveis, dos movimentos citados na subseção 3.3, conforme sequência a seguir:

(a) aplicação de dois movimentos de troca em uma mesma rota e dois movimentos de troca de clientes de rotas diferentes;

(b) aplicação de dois movimentos de realocação.

3.5. Função de avaliação da solução

Uma solução s para o PRVCEJT é avaliada pelo número total de veículos empregados e por uma função $f(s)$, dada por:

$$f(s) = \sum_{i,j} d_{ij} \tag{2}$$

A expressão (2) calcula a distância total percorrida por um veículo em uma determinada rota, sendo que o custo de deslocamento de um cliente i para o cliente j é definido por d_{ij}. A distância total da solução é dada pela soma das distâncias percorridas em cada rota.

3.6. Algoritmo proposto

O algoritmo híbrido ILS-VND é proposto, neste trabalho, para resolver o PRVCEJT. Ele combina os procedimentos metaheurísticos do *Iterated Local Search* - ILS (Lourenço et al., 2003) com a técnica de busca local *Variable Neighborhood Descent* - VND (Mladenović e Hansen, 1997); (Hansen e Mladenovic, 2001).

O ILS consiste, basicamente, em explorar o espaço de soluções por meio da aplicação de perturbações em ótimos locais encontrados por um procedimento de busca local. Quatro componentes devem ser especificados para a aplicação do algoritmo ILS, conforme Algoritmo 3:

(a) geração de solução inicial;

(b) técnica de busca local;

(c) metodologia de perturbação;

(d) critério de aceitação.

A definição desses componentes mostra como será estruturado o algoritmo para a solução do problema em análise.

O procedimento de busca local VND realiza a exploração do espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança. O Algoritmo 4 apresenta o funcionamento do VND. Seja r as estruturas de vizinhança, k a estrutura de vizinhança corrente e $N^{(k)}(s)$ o conjunto de soluções na k-ésima vizinhança da solução s. O laço nas linhas 3-10 é repetido até que todas as vizinhanças sejam analisadas sem sucesso. Na linha 4 uma busca local na vizinhança $N^{(k)}$ é aplicada a s, gerando a solução s'. Se s' for melhor que s (linha 5), então s' é atribuída a s e o processo é reiniciado com a vizinhança $N^{(k)}$, como mostrado nas linhas 6 e 7. Caso contrário, o processo continua com a vizinhança $N^{(k+1)}$ (linha 9).

Algoritmo 3: Algoritmo ILS

Entrada: Número de Veículos, Capacidade dos Veículos, Lista de Clientes com suas Coordenadas Geográficas, Demandas e Janelas de Tempo
Saída: Melhor indivíduo gerado

```
1 s_0 ← GeraSolucaoInicial(·)
2 s ← BuscaLoca(s_0)
3 enquanto Criterio de Parada nao satisfeito faça
4 |   s' ← Perturbacao(historico, s)
5 |   s'' ← BuscaLocal(s')
6 |   s ← CriteriodeAceitacao(s, s'', historico)
```

Algoritmo 4: Algoritmo *VND*

Entrada: Número de Veículos, Capacidade dos Veículos, Lista de Clientes com suas Coordenadas Geográficas, Demandas e Janelas de Tempo
Saída: Melhor indivíduo gerado

```
1  Seja r o número de estruturas diferentes de vizinhança
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  |   senão
9  |   |   k ← k + 1
10 Retorne s
```

A diferença entre as versões ILS-VND1 e ILS-VND2 está no procedimento de construção da solução inicial. Na versão ILS-VND1, a solução inicial é gerada através da PFIH. Na versão ILS-VND2, a construção é feita com o emprego do GRASP. Essa adaptação foi proposta para aumentar a diversidade da solução inicial.

No algoritmo proposto, a solução inicial passa por uma busca local para refinamento. Em seguida, a solução melhorada sofre perturbações, conforme detalhado na subseção 3.4. A solução perturbada é, então, refinada através do procedimento VND, obtendo-se um novo ótimo local. O VND implementado utiliza as estruturas de vizinhança detalhadas na subseção 3.3. A solução ótima local retornada pelo VND torna-se a solução corrente caso seja melhor que a anterior, em relação à função objetivo; caso contrário, ela é descartada e nova perturbação é feita a partir da solução melhorada. Se a perturbação não contribuiu para a melhoria da solução, o seu nível é aumentado na próxima iteração, senão o mesmo nível é mantido. O critério de parada da execução do algoritmo ILS-VND é o número de iterações sem melhora.

4. Resultados computacionais

Nesta seção são apresentados os resultados computacionais obtidos pelas versões ILS-VND1 e ILS-VND2 do algoritmo proposto. Esse algoritmo foi desenvolvido na linguagem C, utilizando o ambiente C++ Builder, versão 2006. Os testes foram executados em um computador com processador Intel Core i3 2,4 GHz, 4GB de memória RAM, sistema operacional Windows 7 Home Premium de 32 bits.

Para testar o algoritmo, foram utilizadas as instâncias de Li e Lim (2001), usadas como referência de desempenho de algoritmos para solução do PRVCEJT. Essas instâncias são divididas em seis

classes distintas: LC1, LC2, LR1, LR2, LRC1 e LRC2. Todos os problemas possuem 100 clientes com depósito, capacidade do veículo e janela de tempo. Clientes das instâncias LC são distribuídos na forma de *cluster*. Nos problemas LR, os clientes são postos de forma aleatória. Nas instâncias LRC, os clientes são parcialmente colocados na forma de *cluster* e parcialmente aleatórios. De acordo com Li e Lim (2001), as instâncias LC1, LR1 e LRC1 são problemas de curto horizonte de programação, enquanto as instâncias LC2, LR2 e LRC2 são de longo horizonte de programação.

Os resultados obtidos estão apresentados nas Tabelas 1 a 6. Nestas tabelas, NV significa o número de veículos utilizados e DT a distância total percorrida. Esses valores estão comparados com os melhores resultados da literatura, expostos na coluna "Literatura", determinados para cada instância e encontrados em `http://www.sintef.no/Projectweb/TOP/PDPTW/Li--Lim-benchmark/100-customers/`.

Tabela 1: Resultados das instâncias da classe LC1

	Literatura			ILS-VND1		ILS-VND2	
Instância	NV	DT	Autor	NV	DT	NV	DT
LC101	10	828,94	Li e Lim (2001)	11	1370,14	10	828,94
LC102	10	828,94	Li e Lim (2001)	10	1969,97	10	838,50
LC103	09	1035,35	Bent e Hentenryck (2003)	10	2165,30	09	1292,63
LC104	09	860,01	Hasle e Kloster (2007)	10	1750,01	10	880,38
LC105	10	828,94	Li e Lim (2001)	10	2072,60	10	828,94
LC106	10	828,94	Li e Lim (2001)	12	2667,73	10	828,94
LC107	10	828,94	Li e Lim (2001)	12	2664,06	10	828,94
LC108	10	826,44	Li e Lim (2001)	10	2523,03	11	866,64
LC109	09	827,82	Bent e Hentenryck (2003)	10	2155,94	10	827,82
Total	87	7694,32		95	8562,91	90	8021,72

Tabela 2: Resultados das instâncias da classe LC2

	Literatura			ILS-VND1		ILS-VND2	
Instância	NV	DT	Autor	NV	DT	NV	DT
LC201	03	591,56	Li e Lim (2001)	03	591,56	03	591,56
LC202	03	591,56	Li e Lim (2001)	05	776,87	04	673,14
LC203	03	585,56	Li e Lim (2001)	04	1159,7	03	645,40
LC204	03	590,6	Hasle e Kloster (2007)	04	902,68	04	833,72
LC205	03	588,88	Li e Lim (2001)	05	938,02	04	647,14
LC206	03	588,49	Li e Lim (2001)	05	1357,03	03	588,49
LC207	03	588,29	Li e Lim (2001)	03	591,62	03	589,20
LC208	03	588,32	Li e Lim (2001)	04	840,83	03	592,74
Total	24	4713,26		33	7158,33	27	5161,39

Um estudo comparativo global dos resultados encontrados nas duas versões do algoritmo ILS-VND com os melhores da literatura é apresentado nas Tabelas 7 e 8. Esse estudo comprova a viabilidade da metodologia proposta, porém indica que é necessário um esforço na tentativa de melhorar seus resultados. O resultado final do algoritmo na versão ILS-VND1 está 26,12% acima dos melhores resultados conhecidos para o número total de veículos e 21,38% acima em relação à distância total percorrida. Já na versão ILS-VND2, o resultado final está 15,17% acima para o número de veículos e 12,02% para a distância total.

Tabela 3: Resultados das instâncias da classe LR1

	Literatura			ILS-VND1		ILS-VND2	
Instância	NV	DT	Autor	NV	DT	NV	DT
LR101	19	1650,8	Li e Lim (2001)	20	1659,98	19	1650,8
LR102	17	1487,57	Li e Lim (2001)	19	1579,72	17	1602,14
LR103	13	1292,68	Li e Lim (2001)	15	1491,43	14	1341,47
LR104	09	1013,39	Li e Lim (2001)	13	1227,74	12	1152,92
LR105	14	1377,11	Li e Lim (2001)	16	1456,12	14	1377,11
LR106	12	1252,52	Li e Lim (2001)	14	1371,3	12	1276,48
LR107	10	1111,31	Li e Lim (2001)	12	1177,14	11	1143,19
LR108	09	968,97	Li e Lim (2001)	12	1168,24	09	972,36
LR109	11	1208,96	Hasle e Kloster (2007)	13	1331,38	12	1241,25
LR110	10	1159,35	Li e Lim (2001)	13	1268,87	13	1256,53
LR111	10	1108,9	Li e Lim (2001)	13	1253,1	11	1114,95
LR112	09	1003,77	Li e Lim (2001)	13	1169,71	10	1045,52
Total	143	14635,33		173	16154,73	154	15174,72

Tabela 4: Resultados das instâncias da classe LR2

	Literatura			ILS-VND1		ILS-VND2	
Instância	NV	DT	Autor	NV	DT	NV	DT
LR201	04	1253,23	Hasle e Kloster (2007)	07	1339,33	06	1427,35
LR202	03	1197,67	Li e Lim (2001)	06	1494,27	05	1328,83
LR203	03	949,4	Li e Lim (2001)	04	1228,28	04	1278,65
LR204	02	849,05	Li e Lim (2001)	04	1253,53	04	1113,2
LR205	03	1054,02	Li e Lim (2001)	05	1409,96	04	1438,1
LR206	03	931,63	Li e Lim (2001)	04	1168,83	03	1115,84
LR207	02	903,06	Li e Lim (2001)	04	1159,51	04	1157,73
LR208	02	734,85	Li e Lim (2001)	04	1117,85	03	969,02
LR209	03	930,59	Hasle e Kloster (2007)	05	1342,15	04	1290,38
LR210	03	964,22	Li e Lim (2001)	04	1371,26	05	1221,65
LR211	02	911,52	Hasle e Kloster (2007)	04	1157,45	04	1027,77
Total	30	10679,24		51	14042,43	46	13368,53

Tabela 5: Resultados das instâncias da classe LRC1

	Literatura			ILS-VND1		ILS-VND2	
Instância	NV	DT	Autor	NV	DT	NV	DT
LRC101	14	1708,8	Li e Lim (2001)	17	1904,52	16	1759,79
LRC102	12	1558,07	Hasle e Kloster (2007)	17	1744,34	15	1744,95
LRC103	11	1258,74	Li e Lim (2001)	12	1354,44	11	1285,4
LRC104	10	1128,74	Li e Lim (2001)	12	1305,57	11	1184,14
LRC105	13	1637,62	Li e Lim (2001)	15	1762,35	13	1637,69
LRC106	11	1424,73	Hasle e Kloster (2007)	15	1691,76	14	1590,03
LRC107	11	1230,15	Li e Lim (2001)	14	1554,89	14	1432,35
LRC108	10	1147,43	Hasle e Kloster (2007)	11	1223,42	11	1238,18
Total	92	11094,28		113	12541,29	105	11872,53

5. Conclusão

Este artigo apresentou duas versões distintas, denominadas ILS-VND1 e ILS-VND2, do algoritmo híbrido ILS-VND, proposto para a solução do PRVCEJT. Estas duas versões desse algoritmo foram

Tabela 6: Resultados das instâncias da classe LRC2

	Literatura			ILS-VND1		ILS-VND2	
Instância	NV	DT	Autor	NV	DT	NV	DT
LRC201	04	1406,94	Hasle e Kloster (2007)	06	1625,39	06	1583,2
LRC202	03	1374,27	Li e Lim (2001)	05	1590,56	05	1597,81
LRC203	03	1089,07	Li e Lim (2001)	05	1337,27	05	1278,07
LRC204	03	818,66	Hasle e Kloster (2007)	05	1345,79	04	1010,79
LRC205	04	1302,2	Li e Lim (2001)	06	1671,69	05	1474,58
LRC206	03	1159,03	Hasle e Kloster (2007)	05	1563,14	06	1552,96
LRC207	03	1062,05	Hasle e Kloster (2007)	05	1436,42	05	1524,52
LRC208	03	852,76	Li e Lim (2001)	05	1223,75	05	1223,5
Total	26	9064,98		42	11794,01	41	11245,43

Tabela 7: Resultados geral das classes de instâncias

	Literatura		ILS-VND1		ILS-VND2	
Classe	NV	DT	NV	DT	NV	DT
LC1	87	7694,32	95	8562,91	90	8021,72
LC2	24	4713,26	33	7158,33	27	5161,39
LR1	143	14635,33	173	16154,73	154	15174,72
LR2	30	10679,24	51	14042,43	46	13368,53
LRC1	92	11094,28	113	12541,29	105	11872,53
LRC2	26	9064,98	42	11794,01	41	11245,43

Tabela 8: Estudo comparativo final

		ILS-VND1		ILS-VND2	
	Literatura	Resultado	%	Resultado	%
Número total de veículos	402	507	26,12	463	15,17
Distância total	57881,41	70253,70	21,38	64844,32	12,02

avaliadas utilizando as 56 instâncias de 100 clientes propostas em Li e Lim (2001) para o PRVCEJT.

A adaptação da PFIH na construção da solução inicial do ILS-VND1 se mostrou bastante eficiente, pois gerou soluções com um número de rotas muito próximo dos melhores da literatura. No entanto, observou-se que os movimentos e as estratégias de perturbação geraram uma enorme quantidade de soluções inviáveis em função da boa qualidade da solução inicial.

O ILS-VND2 foi desenvolvido com o objetivo de diversificar a solução inicial. Utilizando uma solução inicial gerada através da fase de construção do GRASP, foi possível atingir um número maior de regiões do espaço de busca e, consequentemente, melhorar os resultados finais do algoritmo.

Em trabalhos futuros, pretende-se investir em técnicas que auxiliem na eliminação de rotas da solução e na implementação de novos movimentos, tentando aumentar a exploração do espaço de busca e reduzir a diferença entre o resultado alcançado e o ótimo conhecido na literatura.

Referências

Bent, R. e Hentenryck, P. V. (2003). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, v. 33, p. 875–893.

Braÿsy, O. e Gendreau, M. (2005)a. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, v. 39, n. 1, p. 104–118.

Braÿsy, O. e Gendreau, M. (2005)b. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, v. 39, n. 1, p. 119–139.

Carabetti, E. G. (2010). Metaheurística colônia de formigas aplicada ao problema de roteamento de veículos com coleta e entrega e janela de tempo. Dissertação de mestrado, Programa de Pós-Graduação em Modelagem Matemática e Computacional (CEFET-MG), Belo Horizonte.

Dantzig, G. B. e Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, v. 6, n. 1, p. 80–91.

Dondo, R.; Mendez, C. e Cerda, J. (2008). Optimal management of logistic activities in multi-site environments. *Computers & Chemical Engineering*, v. 32, p. 2547–2569.

Feo, T. A. e Resende, M. G. C. (1995). Greedy randomized adaptive search. *Journal of Global Optimization*, v. 3, p. 109–133.

Hansen, P. e Mladenovic, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operations Research*, v. 130, p. 449–467.

Hasle, G. e Kloster, O. (2007). Industrial vehicle routing problem. *Geometric Modelling, Numerical Simulation, and Optimization*, v. 3, p. 397–435.

Li, H. e Lim, A. (2001). A metaheuristic for the pickup and delivery problem with time windows. *Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence – ICTAI-2001*, p. 160–167, Washington, DC, USA.

Lourenço, H. R.; Martin, O. C. e Stützle, T. (2003). Iterated local search. Glover, F. e Kochenberger, G. A., editors, *Handbook of Metaheuristics*, p. 321–353. Kluwer Academic Publishers.

Lu, Q. e Dessouky, M. M. (2006). A new insertion-based construction heuristic for solving the pickup and delivery problem with time-windows. *European Journal of Operational Research*, v. 175, p. 672–687.

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

Nanry, W. P. e Barnes, J.W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research, Part B: Methodological*, v. 34, n. 2, p. 107–121.

Parragh, S.; Doerner, K. e Hartl, R. (2008)a. A survey on pickup and delivery problems part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, v. 58, n. 2, p. 21–51.

Parragh, S.; Doerner, K. e Hartl, R. (2008)b. A survey on pickup and delivery problems part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, v. 58, n. 2, p. 81–117.

Ropke, S. e Cordeau, J. F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, v. 43, n. 3, p. 267–286.

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