UM ALGORITMO PARALELO PARA RESOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA

Raphael Carlos Cruz[1], Thaís Cotta Barbosa da Silva[1], Marcone Jamilson Freitas Souza[1], Alexandre Xavier Martins[1], Vitor Nazário Coelho[1]

[1]Departamento de Computação, Universidade Federal de Ouro Preto (UFOP)
Campus Universitário, Morro do Cruzeiro, CEP 35.400-000, Ouro Preto (MG), Brasil
raphaelcarlos25@yahoo.com.br, thais_cotta@yahoo.com.br
marcone@iceb.ufop.br, xmartins@decea.ufop.br, vncoelho@gmail.com

Abstract. *This work addresses the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). We propose the algorithm GENVNS-TS-CL-PR for solving it. This algorithm combines the heuristic procedures Cheapest Insertion, Cheapest Insertion with multiple routes, GENIUS, Variable Neighborhood Search (VNS), Variable Neighborhood Descent (VND), Tabu Search (TS) and Path Relinking (PR). The first three procedures aim to obtain a good initial solution, and the VND and TS are used as local search methods for VNS. TS is applied after some iterations without any improvement through of the VND. The PR procedure is performed for each VNS iteration and it connects a local optimum with an elite solution generated during the search. The algorithm also uses a strategy to reduce the number of solutions evaluated in the solution space, as well as the parallelization of the local searches. It was tested on benchmark instances taken from the literature and it was able to generate high quality solutions.*

KEYWORDS*: Vehicle Routing Problem with Simultaneous Pickup and Delivery, Variable Neighborhood Search, Parallelization.*

Resumo. *Este trabalho trata o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Para resolvê-lo, propõe-se o algoritmo PGENVNS-TS-CL-PR, que combina os procedimentos heurísticos Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas, GENIUS, Variable Neighborhood Search (VNS), Descida em Vizinhança Variável (VND), Busca Tabu (TS) e Reconexão por Caminhos (PR). Os três primeiros procedimentos visam a obtenção de uma boa solução inicial, enquanto os procedimentos VND e TS são usados como métodos de busca local para o VNS. A Busca Tabu somente é acionada após certo número de iterações sem sucesso do VND. O procedimento PR é acionado a cada iteração do VNS e conecta um ótimo local a uma solução elite gerada durante a busca. Utiliza-se, também, uma estratégia para reduzir o número de soluções avaliadas no espaço de soluções, assim como a paralelização das buscas locais. O algoritmo foi testado em instâncias da literatura e se mostrou capaz de gerar soluções de alta qualidade.*

PALAVRAS-CHAVE*: Roteamento de Veículos com Coleta e Entrega Simultânea, Variable Neighborhood Search, Paralelização.*

1 Introdução

O Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES) é uma variante do Problema de Roteamento de Veículos (PRV), no qual os clientes requerem os serviços de coleta e entrega de produtos. Este problema foi proposto por Min (1989) e é da classe NP-difícil, uma vez que ele pode ser reduzido ao PRV (Dantzig e Ramser, 1959) quando todas as demandas de coleta são nulas.

Dada a dificuldade de obter soluções ótimas para casos práticos do problema, o PRVCES é normalmente resolvido por meio de procedimentos heurísticos. Em Zachariadis et al. (2010) é proposto um algoritmo evolucionário que utiliza uma memória adaptativa para guardar informações das soluções de alta qualidade obtidas durante a busca. Essas informações são usadas para gerar novas soluções em regiões que possuem grande chance de trazer melhores resultados, sendo elas, posteriormente, melhoradas por um método de Busca Tabu.

Subramanian et al. (2010) apresentaram um algoritmo paralelo para a solução do PRVCES, chamado de PILS-RVND. O algoritmo utiliza uma heurística *multi-start*, em que, a cada iteração, uma solução inicial é gerada pelo procedimento Inserção mais Barata com Múltiplas Rotas. Essa solução é refinada pelo ILS, tendo como busca local o VND com ordem de vizinhança aleatória (RVND). O RVND explora o espaço de soluções por meio de seis movimentos, sendo a ordem das vizinhanças aleatória a cada chamada. Os experimentos foram realizados em dois *clusters* de computadores, sendo um com uma arquitetura composta por 128 núcleos e outro com 256. Os resultados obtidos em 72 problemas-teste consagrados da literatura provaram que ele é o algoritmo de melhor desempenho até então conhecido.

Mine et al. (2010) propuseram o algoritmo GENILS, que utiliza a melhor solução gerada pelos métodos de Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas e uma adaptação da heurística GENIUS (Gendreau et al., 1992), como solução inicial. Como refinamento é utilizado o método *Iterated Local Search* – ILS, tendo o *Variable Neighborhood Descent* – VND como método de busca local. Silva et al. (2011) aperfeiçoaram esse algoritmo incorporando um módulo de Busca Tabu como alternativa para a busca local do ILS. Nesse algoritmo, denominado GENILS-TS, a Busca Tabu é chamada somente após um número de iterações sem sucesso com o uso do VND.

Neste trabalho é expandido e aprimorado o algoritmo GENILS-TS-CL-PR de Silva et al. (2012). Ao invés de usar a metaheurística ILS, usa-se o método *Variable Neighborhood Search* – VNS, uma vez que a busca local é realizada com um conjunto de estruturas de vizinhança. O algoritmo proposto, denominado PGENVNS-TS-CL-PR, incorpora quatro estratégias de exploração do espaço de busca: 1) o VND e a Busca Tabu como alternativa de busca local do procedimento VNS, sendo a Busca Tabu acionada somente após certo número de iterações sem melhora; 2) uma Lista de Candidatos para evitar a avaliação de soluções não promissoras; 3) a Reconexão por Caminhos aplicada a cada ótimo local gerado pelo VNS e, finalmente, 4) a paralelização das buscas locais.

O restante deste artigo está organizado como segue. Na seção 2 é descrito o problema de roteamento de veículos abordado. Na seção 3 é apresentado o algoritmo proposto e na seção 4, a estratégia de paralelização. Os resultados dos experimentos são relatados na seção 5. A seção 6 conclui o trabalho.

2 Descrição do Problema

O PRVCES envolve um conjunto de clientes, cada qual com uma demanda d_i e coleta p_i a serem completamente atendidas, e um depósito, que é o local onde ficam armazenados os produtos a serem entregues aos clientes ou coletados desses, e um conjunto de veículos de capacidade Q de carga, os quais são utilizados para fazer as operações de entrega e coleta. O objetivo é construir um conjunto de rotas, a custo mínimo, que iniciam e terminam no depósito, e atendam a todos os clientes sem ultrapassar a capacidade dos veículos. A Figura 1 ilustra um exemplo de uma solução para um PRVCES envolvendo 19 clientes e veículos de capacidade $Q = 150$ unidades.

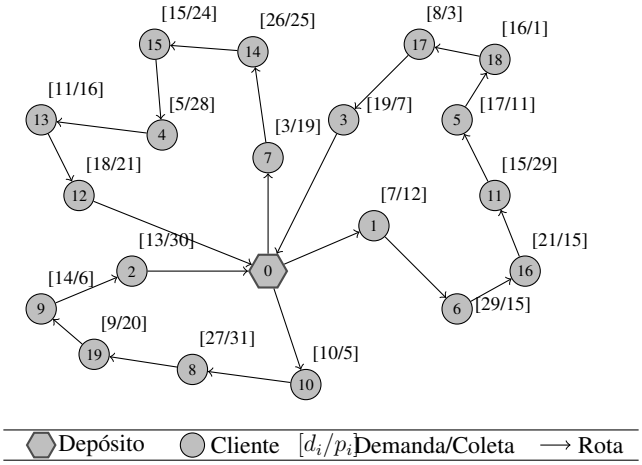

Figura 1. Exemplo do PRVCES, adaptado de Mine et al. (2010).

Considerando a rota do veículo do canto superior esquerdo da Figura 1, o veículo parte do depósito, visita o cliente 7 para entregar 3 unidades do produto e coletar 19 unidades. A seguir, se direciona para os clientes 14, 15, 4, 13, e 12, nesta ordem, entregando e coletando produtos, e retorna ao depósito.

3 Algoritmo Proposto

Nesta Seção é apresentado o algoritmo proposto para resolver o PRVCES, assim como os módulos que o compõe.

3.1 Algoritmo PGENVNS-TS-CL-PR

O algoritmo proposto, nomeado PGENVNS-TS-CL-PR, combina os procedimentos heurísticos Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas, GENIUS, VNS, VND, Busca Tabu (TS) e Reconexão por Caminhos (PR). Os três primeiros procedimentos são usados para gerar uma solução inicial, sendo o refinamento feito pelo VNS. O VNS, por sua vez, utiliza os procedimentos VND e TS como módulos de busca local. Além disso, ele utiliza uma estratégia de Lista de Candidatos para evitar a geração de movimentos não promissores, assim como aplica periodicamente a Reconexão por Caminhos entre um ótimo local gerado por VND ou TS e uma solução do conjunto elite formada durante a busca. Adicionalmente, o algoritmo explora paralelismo, ao distribuir as buscas locais aos diversos núcleos de processamento disponíveis na máquina. Seu pseudocódigo é apresentado pelo Algoritmo 1.

Algoritmo 1: *PGENVNS-TS-CL-PR*

1: $\gamma \leftarrow$ número real aleatório no intervalo $[0.0, 0.7]$;
2: $s^A \leftarrow$ *InserçãoMaisBarataRotaARota*(γ);
3: $s^A \leftarrow$ *VND*(s^A);
4: $nRotas \leftarrow$ número de rotas da solução s^A;
5: $s^B \leftarrow$ *InserçãoMaisBarataMRotas*$(\gamma, nRotas)$; $s^B \leftarrow$ *VND*(s^B);
6: $s^C \leftarrow$ *VRGENIUS*$(nRotas)$; $s^C \leftarrow$ *VND*(s^C);
7: $s \leftarrow argmin\{f(s^A), f(s^B), f(s^C)\}$;
8: $iter \leftarrow 1$;
9: **enquanto** ($iter \leq maxIter$) **faça**
10: **para** $i \leftarrow 0$ **até** p **faça**
11: $s' \leftarrow$ *Perturbação*(s);
12: **fim para**
13: **se** ($iter \leq iterMaxVND$) **então**
14: $s'' \leftarrow$ *VND*(s');
15: **senão**
16: $s'' \leftarrow$ *BuscaTabu*$(s', TamListaTabu, iterMaxTS)$;
17: **fim se**
18: $s \leftarrow ReconexaoPorCaminhos(solElite, s'')$;
19: **se** ($f(s'') < f(s)$) **então**
20: $s \leftarrow s''$; $iter \leftarrow 1$; $p \leftarrow 1$;
21: **senão**
22: $iter \leftarrow iter + 1$;
23: **fim se**
24: **se** ($iter \leq iterMaxPertubação$) **então**
25: $p \leftarrow p + 1$
26: **fim se**
27: *AtualizaConjElite(s")*;
28: **fim enquanto**
29: Retorne s;

Como pode ser observado no Algoritmo 1, os três métodos construtivos utilizados, descritos na Seção 3.3, produzem as soluções s^A, s^B e s^C, e cada uma delas é refinada por um procedimento VND. A melhor solução gerada se transforma na solução inicial s, sendo esta refinada pelo VNS. Para escapar das armadilhas de ótimos locais, e se dirigir para outras regiões do espaço de busca, o nível de perturbação é aumentado depois de *iterMaxPertubação* iterações sem melhora (linha 25). Os tipos de perturbações utilizados são descritos na Seção 3.7. Os métodos VND e Busca Tabu são utilizados como busca local, sendo a Busca Tabu acionada somente após certo número de iterações sem melhora, dado pelo parâmetro *iterMaxVND*. A cada iteração do VNS também é aplicada a técnica Reconexão por Caminhos (descrita na Seção 3.8) na solução gerada pela busca local s''.

3.2 Função de Avaliação

Uma solução é avaliada pela função (1), a ser minimizada. A primeira parcela dessa função corresponde à distância total percorrida, enquanto a segunda parcela é uma penalidade aplicada ao excesso de carga no veículo, ou seja, toda vez que o limite de carga é ultrapassado, o valor excedido é multiplicado por um fator β, que corresponde a um valor suficientemente grande. A geração de soluções inviáveis, isto é, soluções em que a carga do veiculo não é respeitada, é permitida, porém não é incentivada, já que a função de avaliação deve ser minimizada.

$$f(s) = \sum_{(i,j)\in E} c_{ij}x_{ij} + \beta \times \sum_{l\in R} \max\{0, \sum_{j\in N_l}(-d_j + p_j) - Q\} \quad (1)$$

Na função de avaliação f, dada pela Equação (1), têm-se:
N: conjunto dos clientes, incluindo o depósito;
E: conjunto de arestas (i, j), com $i, j \in N$;
R: conjunto de rotas existentes na solução;
N_l: conjunto de clientes j pertencentes à rota l, com $j \in N$ e $l \in R$;
c_{ij}: custo de deslocamento ou distância de i a j, com $i, j \in N$;
x_{ij}: variável binária que assume valor 1 se na solução s a aresta $(i, j) \in E$ é utilizada ($x_{ij} = 1$) e valor zero ($x_{ij} = 0$), caso contrário.
d_j: quantidade a ser entregue ao cliente $j \in N$;
p_j: quantidade a ser coletada no cliente $j \in N$;
Q: capacidade de carga do veículo;

3.3 Geração da Solução Inicial

Para gerar a solução inicial, foram utilizadas as heurísticas construtivas de Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas e uma adaptação da heurística GENIUS proposta em Mine et al. (2010).

A Inserção Mais Barata Rota a Rota, também conhecida como IMB-1R, foi proposta por Dethloff (2001) e consiste, basicamente, em construir uma subrota inicial contendo um cliente aleatório, sendo os demais clientes inseridos a cada iteração respeitando-se as restrições do PRVCES. Para determinar qual cliente será incluído na rota, é utilizada a Equação (2), de forma que i e j correspondem a clientes já alocados a alguma rota e k representa um potencial cliente a ser inserido na rota. Já o fator $\gamma \in [0, 1]$ corresponde a uma bonificação dada a um cliente distante do depósito.

$$e_{ij}^{k} = (c_{ik} + c_{kj} - c_{ij}) - \gamma \times (c_{0k} + c_{k0}) \tag{2}$$

A Inserção Mais Barata com Múltiplas Rotas, também conhecida como IMB-MR, foi proposta por Subramanian (2008). Para inicializar o IMB-MR, é necessário informar o número inicial de rotas m. No algoritmo implementado este número é obtido pela aplicação do algoritmo IMB-1R. Ele funciona de forma similar ao procedimento construtivo anterior, com a diferença de que ele inicializa m rotas simultaneamente. Sendo assim, m rotas são iniciadas com um único cliente escolhido aleatoriamente e os demais são inseridos a partir da Equação (2), respeitando-se o limite de carga do veículo.

O último método construtivo, denominado VRGENIUS, foi proposto por Mine et al. (2010). Ele é composto pelo método construtivo VRGENI e pelo método de refinamento VRUS. Ambos utilizam como busca local procedimentos *3-opt* e *4-opt*. O funcionamento desses métodos pode ser encontrado no trabalho indicado.

3.4 Estruturas de vizinhança

São utilizados sete movimentos diferentes para explorar o espaço de soluções, todos bem conhecidos na literatura. O movimento *Shift* consiste em transferir um cliente de uma rota para outra, enquanto o *Shift*(2,0) consiste na realocação de dois clientes. O movimento *Swap* consiste na troca de um cliente de uma rota por um cliente pertencente a outra rota. O movimento *Swap*(2,1) consiste em trocar dois clientes consecutivos de uma rota com um cliente de outra rota, e o *Swap*(2,2) em trocar dois clientes consecutivos de uma rota com dois clientes consecutivos de outra. O *2-Opt* consiste em remover dois arcos e inserir dois

novos arcos de forma a gerar uma nova rota. Finalmente, o movimento *kOr-Opt* realoca uma sequência de k clientes consecutivos para outra posição na mesma rota, sendo k um parâmetro do método; neste trabalho, k foi fixado no valor 3.

Os métodos *G3-opt* e *G4-opt* utilizam técnicas de inserção e remoção de arcos combinados com a ideia do *3-opt* e *4-opt* para melhorar a solução. Detalhes desses métodos são apresentados em Mine et al. (2010).

O procedimento *reverse* consiste em inverter o sentido de uma rota, sendo aplicado somente se ocorrer um aumento na carga residual da rota. A carga residual de uma rota é o valor da capacidade do veículo subtraído da maior carga do veículo nessa rota.

3.5 Lista de Candidatos

Para diminuir a quantidade de soluções analisadas pelas estruturas de vizinhança descritas na Seção 3.4, foi utilizada uma Lista de Candidatos. Esta Lista permite que apenas movimentos promissores sejam analisados, descartando aqueles considerados desnecessários. A análise da vizinhança é restringida porque o movimento só é permitido se todas as arestas resultantes forem menor que o valor dado pela Equação (3).

$$dist = \left(\sum_{(i,j) \in E} c_{ij} \right) / (n-1)^2 \tag{3}$$

Na Equação (3), n indica o número de vértices do conjunto e E o conjunto de arestas ligando todos os vértices. Sendo assim, ela representa a média das distâncias entre todos os pares de pontos (depósito/clientes) existentes.

3.6 Busca Tabu

No Algoritmo implementado cinco soluções são geradas a cada iteração utilizando os movimentos *Shift, Swap, Shift (2,0), Swap (2,1) e Swap (2,2)* descritos na Seção 3.4, sendo que a melhor delas passa a ser a solução corrente s. Cada vez que se move para uma nova solução, a Lista Tabu é atualizada. O tamanho da Lista Tabu (`TamListaTabu`) é incrementado em uma unidade à medida que o número de iterações sem melhora aumenta; porém, para a busca não ficar extremamente restritiva, essa atualização para de ser efetuada quando certo número de iterações sem melhora é atingido. O procedimento é interrompido quando o número máximo de iterações sem melhora na solução é alcançado.

O movimento tabu utilizado pelo procedimento TS para evitar o retorno a uma solução gerada anteriormente consiste em proibir que o cliente afetado pelo movimento gerado seja sucessor do cliente adjacente a ele antes do movimento. Porém, caso esse movimento tabu produza uma solução vizinha melhor que a melhor solução encontrada até então, o movimento é aceito, ou seja, é feita a aspiração por objetivo.

3.7 Perturbações

As perturbações são realizadas por um dos três procedimentos, escolhidos aleatoriamente a cada chamada: Múltiplos *Shifts*, Múltiplos *Swaps* e *Ejection Chain*.

Os procedimentos Múltiplos *Shifts* e Múltiplos *Swaps* consistem em k aplicações sucessivas dos movimentos *shift* e *swap*, sendo k um valor inteiro aleatório entre 1 e 3.

A perturbação *Ejection chain* (Rego e Roucairol, 1996) consiste em selecionar um subconjunto de m rotas $R = r_1, r_2, \cdots, r_m$ de forma arbitrária. Em seguida, transfere-se um cliente da rota r_1 para a rota r_2, um cliente de r_2 para r_3 e assim sucessivamente até que um cliente seja transferido da rota r_m para a primeira rota r_1. Nessa perturbação os clientes de cada rota são escolhidos de forma aleatória.

3.8 Reconexão por Caminhos

Como forma de fazer um balanço entre intensificação e diversificação, foi utilizada a técnica de Reconexão por Caminhos (PR, do inglês *Path Relinking*), que é aplicada após cada busca local do VNS.

O conjunto elite é composto por cinco soluções. Assim, quando uma nova solução é adicionada ao conjunto e sua capacidade é extrapolada, a solução de pior custo sai, dando lugar à nova solução. Para determinar quais soluções farão parte do conjunto elite foram utilizados os seguintes critérios:

- Se a solução corrente tiver a função de avaliação menor que a melhor solução encontrada até então;
- Se a solução corrente for melhor que a pior do conjunto elite, e for pelo menos 10% diferente das demais soluções do conjunto.

Para que a reconexão de caminhos comece a ser executada, se faz necessário que o conjunto elite esteja completo. Assim, a estratégia adotada consiste em utilizar cada uma das soluções do conjunto elite como solução base, e como guia o ótimo local gerado após a aplicação da busca local do procedimento VNS. Ou seja, são realizadas cinco aplicações de Reconexão por Caminhos. Caso durante a aplicação desta estratégia seja encontrada uma solução melhor que a melhor já obtida, o procedimento de Reconexão é abortado.

Cada iteração da Reconexão por Caminhos consiste em incluir na solução inicial (solução base), um atributo da solução guia. O atributo escolhido para ser inserido é aquele que produz na solução base o melhor valor para a função de avaliação. A seguir, à solução com o atributo inserido é aplicada uma busca local que não altere os atributos herdados, no caso, uma descida completa utilizando o movimento *Shift*. Foi considerado como atributo a ser herdado pela solução base, a posição de determinado cliente no vetor referente à solução guia. Este procedimento é repetido até que as soluções tenham as mesmas configurações.

4 Paralelização das Estruturas de Vizinhança

A introdução de arquiteturas de processamento paralelo permite que o tempo de processamento de um algoritmo possa ser reduzido dividindo o esforço computacional para mais de um processador. A proposta de paralelização do algoritmo foi baseada no trabalho de Gonçalves (2010), em que retirou-se informações importantes inerentes a diversos métodos envolvendo multiprocessamento de tarefas, que do termo em inglês é designado como *thread*. Após realizadas diversas estratégias de paralelização, decidiu-se implementar no algoritmo a paralelização das estruturas de vizinhança.

Internamente em um processador, diversas unidades funcionais executam tarefas em paralelo que estão escondidas no conjunto de instruções da arquitetura do processador. Em algumas aplicações específicas ou determinados programas, os compiladores podem automaticamente detectar e explorar o paralelismo entre vários computadores de

uma maneira eficiente. No entanto, na maioria dos casos, os programadores necessitam instruir aos compiladores, quando e onde utilizar ações paralelas. Não é uma tarefa trivial, pois inclui técnicas de replicação de dados, movimentação de dados, balanceamento entre a carga de execução e comunicação.

Para ajudar nesta tarefa, utilizou-se a interface de programação chamada *OpenMP*, baseada no modelo de programação paralela de memória compartilhada para arquiteturas de múltiplos processadores. A biblioteca utilizada possui diversos recursos, que por meio de chamada de funções já implementadas, consegue-se paralelizar uma estrutura do algoritmo.

A paralelização das estruturas de vizinhanças foi realizada de forma a ordenar todos os movimentos descritos na Seção 3.4 e executá-los simultaneamente em cada um dos núcleos disponíveis na máquina. Terminada a busca local com os movimentos alocados nos núcleos do processador, uma nova chamada é realizada para novos movimentos entrarem na lista de execução, conforme ilustração 2. Esse procedimento é realizado até que todos os movimentos sejam concluídos. Desta forma, ganha-se tempo no processo de busca local, uma vez que não é necessário esperar que apenas um movimento seja executado até o fim para que em seguida um outro seja processado.

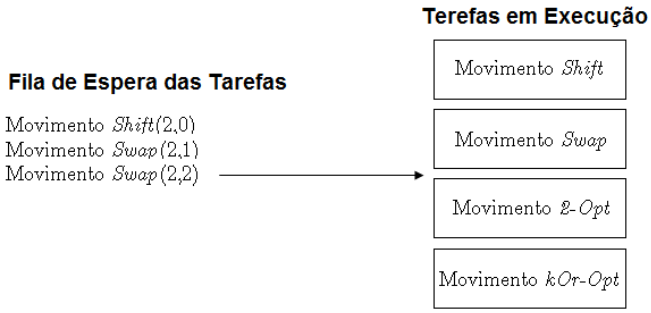

Figura 2. Paralelização das Estruturas de Vizinhança

5 Resultados

O algoritmo PGENVNS-TS-CL-PR foi codificado em C++ usando o compilador Visual C++ 2005. Para testá-lo, foi usado um microcomputador com processador Intel *Core 2 Quad*, 1,66 GHz e 4 GB de memória RAM e sistema operacional Windows Vista.

Para validá-lo, foram usados 40 problemas-teste de Dethloff (2001); 14 de Salhi e Nagy (1999) e 18 de Montané e Galvão (2006). Os parâmetros adotados, obtidos experimentalmente em uma bateria preliminar de testes, foram os seguintes: *TamListaTabu* = 10, *iterMaxTS* = 300, *iterMaxVND*= 500 e *maxIter* = 10000.

Na Subseção 5.1, o algoritmo proposto é comparado com o algoritmo sequencial GENILS-TS-CL-PR de Silva et al. (2012), o qual não usa a estratégia de paralelização das estruturas de vizinhança. Posteriormente, na Subseção 5.2, ele é comparado com os principais algoritmos da literatura.

5.1 GENILS-TS-CL-PR Sequencial × PGENVNS-TS-CL-PR

Nesta Seção comparam-se os resultados dos algoritmos GENILS-TS-CL-PR Sequencial e PGENVNS-TS-CL-PR. O segundo é um aprimoramento do primeiro, e incorpora a Paralelização das Estruturas de Vizinhança (Seção 4) com o objetivo de tentar

diminuir o tempo de execução da versão sequencial do GENILS-TS-CL-PR, mantendo-se a qualidade das soluções geradas.

As Tabelas 1, 2 e 3 comparam os resultados dos algoritmos analisados nos três conjuntos de instâncias utilizadas. Nestas tabelas, a primeira coluna representa o problema-teste e a segunda o melhor resultado da literatura. As colunas *Melhor* e *Média* apresentam, respectivamente, o melhor resultado de cada algoritmo e a média referente ao resultado encontrado pelo algoritmo nas 30 execuções. Já a coluna $Desv^{Avg}$ apresenta o desvio percentual do valor das soluções médias em relação ao melhor resultado da literatura. A coluna $Desv^{Best}$ apresenta o desvio percentual do melhor resultado encontrado pelo algoritmo em relação ao melhor conhecido na literatura. A coluna *Tempo* apresenta o tempo médio das 30 execuções em segundos.

Tabela 1. GENILS-TS-CL-PR × PGENVNS-TS-CL-PR nas instâncias: Dethloff (2001)

Problema	MelhorLit	GENILS-TS-CL-PR (sequencial)					PGENVNS-TS-CL-PR				
-	-	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo(s)	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo(s)
SCA3-0	**635,62**	**635,62**	**635,62**	0,00	0,00	0,74	**635,62**	635,62	0,00	0,00	0,74
SCA3-1	**697,84**	**697,84**	**697,84**	0,00	0,00	6,80	**697,84**	697,84	0,00	0,00	0,55
SCA3-2	**659,34**	**659,34**	**659,34**	0,00	0,00	5,06	**659,34**	659,34	0,00	0,00	0,27
SCA3-3	**680,04**	**680,04**	**680,04**	0,00	0,00	0,29	**680,04**	680,04	0,00	0,00	2,87
SCA3-4	**690,50**	**690,50**	**690,50**	0,00	0,00	12,02	**690,50**	690,50	0,00	0,00	0,98
SCA3-5	**659,90**	**659,90**	**659,90**	0,00	0,00	1,11	**659,90**	659,90	0,00	0,00	0,85
SCA3-6	**651,09**	**651,09**	**651,09**	0,00	0,00	1,12	**651,09**	651,09	0,00	0,00	0,95
SCA3-7	**659,17**	**659,17**	**659,17**	0,00	0,00	1,47	**659,17**	659,17	0,00	0,00	1,47
SCA3-8	**719,48**	**719,48**	**719,48**	0,00	0,00	1,17	**719,48**	719,48	0,00	0,00	1,15
SCA3-9	**681,00**	**681,00**	**681,00**	0,00	0,00	5,18	**681,00**	681,00	0,00	0,00	1,13
SCA8-0	**961,50**	**961,50**	**961,50**	0,00	0,00	3,65	**961,50**	961,50	0,00	0,00	1,34
SCA8-1	**1049,65**	**1049,65**	**1049,65**	0,00	0,00	3,81	**1049,65**	1049,65	0,00	0,00	1,65
SCA8-2	**1039,64**	**1039,64**	**1039,64**	0,00	0,00	1,27	**1039,64**	1039,64	0,00	0,00	1,16
SCA8-3	**983,34**	**983,34**	**983,34**	0,00	0,00	2,20	**983,34**	983,34	0,00	0,00	1,17
SCA8-4	**1065,49**	**1065,49**	**1065,49**	0,00	0,00	3,26	**1065,49**	1065,49	0,00	0,00	1,63
SCA8-5	**1027,08**	**1027,08**	**1027,08**	0,00	0,00	0,21	**1027,08**	1027,08	0,00	0,00	0,20
SCA8-6	**971,82**	**971,82**	**971,82**	0,00	0,00	4,13	**971,82**	971,82	0,00	0,00	1,97
SCA8-7	**1051,28**	**1051,28**	**1051,28**	0,00	0,00	3,26	**1051,28**	1051,28	0,00	0,00	1,15
SCA8-8	**1071,18**	**1071,18**	**1071,18**	0,00	0,00	0,75	**1071,18**	1071,18	0,00	0,00	0,68
SCA8-9	**1060,50**	**1060,50**	**1060,50**	0,00	0,00	0,55	**1060,50**	1060,50	0,00	0,00	0,26
CON3-0	**616,52**	**616,52**	**616,52**	0,00	0,00	0,49	**616,52**	616,52	0,00	0,00	0,39
CON3-1	**554,47**	**554,47**	**554,47**	0,00	0,00	0,84	**554,47**	554,47	0,00	0,00	0,35
CON3-2	**518,00**	**518,00**	**518,00**	0,00	0,00	2,11	**518,00**	518,00	0,00	0,00	1,18
CON3-3	**591,19**	**591,19**	**591,19**	0,00	0,00	0,30	**591,19**	591,19	0,00	0,00	0,19
CON3-4	**588,79**	**588,79**	**588,79**	0,00	0,00	0,62	**588,79**	588,79	0,00	0,00	0,41
CON3-5	**563,70**	**563,70**	**563,70**	0,00	0,00	0,70	**563,70**	563,70	0,00	0,00	0,54
CON3-6	**499,05**	**499,05**	**499,05**	0,00	0,00	0,39	**499,05**	499,05	0,00	0,00	0,18
CON3-7	**576,48**	**576,48**	**576,48**	0,00	0,00	1,51	**576,48**	576,48	0,00	0,00	0,65
CON3-8	**523,05**	**523,05**	**523,05**	0,00	0,00	0,14	**523,05**	523,05	0,00	0,00	0,11
CON3-9	**578,25**	**578,25**	**578,25**	0,00	0,00	1,25	**578,25**	578,25	0,00	0,00	1,17
CON8-0	**857,17**	**857,17**	**857,17**	0,00	0,00	0,55	**857,17**	857,17	0,00	0,00	0,27
CON8-1	**740,85**	**740,85**	**740,85**	0,00	0,00	1,60	**740,85**	740,85	0,00	0,00	1,22
CON8-2	**712,89**	**712,89**	**712,89**	0,00	0,00	0,52	**712,89**	712,89	0,00	0,00	0,43
CON8-3	**811,07**	**811,07**	**811,07**	0,00	0,00	1,24	**811,07**	811,07	0,00	0,00	0,44
CON8-4	**772,25**	**772,25**	**772,25**	0,00	0,00	0,31	**772,25**	772,25	0,00	0,00	0,13
CON8-5	**754,88**	**754,88**	**754,88**	0,00	0,00	0,49	**754,88**	754,88	0,00	0,00	0,36
CON8-6	**678,92**	**678,92**	**678,92**	0,00	0,00	1,53	**678,92**	678,92	0,00	0,00	1,53
CON8-7	**811,96**	**811,96**	**811,96**	0,00	0,00	1,62	**811,96**	811,96	0,00	0,00	1,12
CON8-8	**767,53**	**767,53**	**767,53**	0,00	0,00	3,36	**767,53**	767,53	0,00	0,00	1,26
CON8-9	**809,00**	**809,00**	**809,00**	0,00	0,00	2,47	**809,00**	809,00	0,00	0,00	1,08
Média	-	-	-	**0,00**	**0,00**	**2,00**	-	-	**0,00**	**0,00**	**0,88**

Avaliando as Tabelas 1, 2 e 3, em que são apresentados os resultados dos algoritmos nos problemas-teste de Dethloff (2001), Salhi e Nagy (1999) e Montané e Galvão (2006), respectivamente, observa-se que o PGENVNS-TS-CL-PR requer, como esperado, um menor tempo de processamento quando comparado com o do algoritmo GENILS-TS-

Tabela 2. GENILS-TS-CL-PR × PGENVNS-TS-CL-PR nas instâncias: Salhi e Nagy (1999)

Problema	MelhorLit	GENILS-TS-CL-PR (sequencial)					PGENVNS-TS-CL-PR				
-	-	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo(s)	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo(s)
CMT1X	**466,77**	**466,77**	472,23	1,17	0,00	7,28	**466,77**	471,08	0,92	0,00	2,25
CMT1Y	**466,77**	**466,77**	472,23	1,17	0,00	41,33	**466,77**	472,01	1,12	0,00	32,43
CMT2X	**668,77**	684,11	693,40	3,68	2,29	112,03	684,11	688,40	0,63	2,29	98,65
CMT2Y	**663,25**	684,11	693,40	4,55	3,15	92,15	684,11	695,11	1,61	3,15	78,45
CMT3X	**721,27**	**721,27**	726,34	0,70	0,00	213,87	**721,27**	727,06	0,80	0,00	160,65
CMT3Y	**721,27**	**721,27**	730,56	1,29	0,00	207,21	**721,27**	731,32	1,39	0,00	165,54
CMT12X	**644,70**	662,22	666,77	3,42	2,72	191,56	662,22	666,79	0,69	2,72	123,60
CMT12Y	**659,52**	662,22	673,91	2,18	0,41	90,19	662,22	673,91	1,76	0,41	78,41
CMT11X	**833,92**	**833,92**	867,96	4,08	0,00	318,81	**833,92**	867,95	4,08	0,00	300,52
CMT11Y	**830,39**	833,92	856,32	3,12	0,43	618,36	833,92	856,41	2,70	0,43	541,76
CMT4X	**852,46**	**852,46**	865,03	1,47	0,00	698,12	**852,46**	865,28	1,50	0,00	256,76
CMT4Y	**852,35**	855,52	866,06	1,61	0,37	376,54	855,52	866,11	1,24	0,37	137,98
CMT5X	**1029,25**	1030,50	1052,67	2,28	0,12	113,65	1030,50	1052,71	2,16	0,12	689,41
CMT5Y	**1029,25**	1030,50	1053,73	2,38	0,12	708,21	1030,50	1053,98	2,28	0,12	489,52
Média	**-**	**-**	**-**	**2,36**	**0,69**	**343,74**	**-**	**-**	**1,63**	**0,69**	**233,31**

Tabela 3. GENILS-TS-CL-PR × PGENVNS-TS-CL-PR nas instâncias: Montané e Galvão (2006)

Problema	MelhorLit	GENILS-TS-CL-PR (sequencial)					PGENVNS-TS-CL-PR				
-	-	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo(s)	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo(s)
r101	**1009,95**	**1009,95**	1013,71	0,37	0,00	199,98	**1009,95**	1012,71	0,27	0,00	108,48
r201	**666,20**	**666,20**	666,81	0,09	0,00	182,67	**666,20**	666,81	0,09	0,00	127,92
c101	**1220,18**	**1220,18**	1222,81	0,22	0,00	234,43	**1220,18**	1222,81	0,22	0,00	139,49
c201	**662,07**	**662,07**	664,31	0,34	0,00	96,31	**662,07**	664,31	0,34	0,00	47,53
rc101	**1059,32**	**1059,32**	1064,49	0,49	0,00	88,12	**1059,32**	1062,97	0,34	0,00	32,85
rc201	**672,92**	**672,92**	677,49	0,68	0,00	213,67	**672,92**	677,49	0,68	0,00	167,43
r1_2_1	**3357,64**	**3357,64**	3450,31	2,76	0,00	610,83	**3357,64**	3459,12	3,02	0,00	420,67
r2_2_1	**1665,58**	**1665,58**	1670,35	0,29	0,00	1087,32	**1665,58**	1667,44	0,11	0,00	795,39
c1_2_1	**3629,89**	3634,65	3657,16	0,75	0,13	789,04	3631,62	3637,19	0,20	0,13	642,99
c2_2_1	**1726,59**	**1726,58**	1745,10	1,07	0,00	854,12	**1726,58**	1745,10	1,07	0,00	461,37
rc1_2_1	**3306,00**	3312,92	3327,22	0,64	0,21	872,98	3312,92	3327,22	0,64	0,21	692,71
rc2_2_1	**1560,00**	**1560,00**	1584,84	1,59	0,00	109,54	**1560,00**	1584,84	1,59	0,00	74,21
r1_4_1	**9605,75**	9627,88	9706,91	1,05	0,23	2897,99	9627,88	9643,22	0,39	0,23	1568,23
r2_4_1	**3551,38**	3582,08	3612,11	1,71	0,86	2160,21	3582,08	3586,15	0,98	0,86	1967,34
c1_4_1	**11098,21**	**11098,21**	11133,59	0,32	0,00	3876,82	**11098,21**	11120,38	0,20	0,00	3198,49
c2_4_1	**3546,10**	3592,71	3617,86	2,02	1,31	2790,21	3592,71	3595,38	1,39	1,31	2071,48
rc1_4_1	**9535,46**	**9535,46**	9638,30	1,08	0,00	5789,21	**9535,46**	9538,12	0,03	0,00	4820,13
rc2_4_1	**3403,70**	3419,76	3502,01	2,89	0,47	3976,76	3419,76	3424,67	0,62	0,47	3281,65
Média	**-**	**-**	**-**	**1,02**	**0,18**	**1490,57**	**-**	**-**	**0,68**	**0,17**	**1145,47**

CL-PR, sem com isso deixar de alcançar as melhores soluções. Além disso, o PGENVNS-TS-CL-PR gera soluções finais com menor variabilidade sobre os valores médios, em vista de o $Desv^{Avg}$ ser menor.

5.2 PGENVNS-TS-CL-PR × algoritmos da literatura

Nesta Seção são mostrados os resultados da comparação entre o algoritmo GENVNS-TS-CL-PR e outros três algoritmos da literatura: o algoritmo evolutivo de Zachariadis et al. (2010), o algoritmo PILS-RVND paralelo de Subramanian et al. (2010) (que usa 256 núcleos de processamento) e o GENILS de Mine et al. (2010). Os algoritmos de Zachariadis et al. (2010) e Mine et al. (2010) são, de nosso conhecimento, os melhores algoritmos sequenciais conhecidos, enquanto o de Subramanian et al. (2010) é o melhor algoritmo encontrado para o PRVCES.

A Tabela 4 mostra os resultados alcançados pelo algoritmo proposto e os três algoritmos da literatura acima citados nos três conjuntos de problemas-teste. Nesta Tabela, a coluna "# Prob." indica a quantidade de problemas-teste que fazem parte de cada um dos

conjuntos de Dethloff (2001), Salhi e Nagy (1999) e Montané e Galvão (2006). As colunas "Média $Desv^{Best}$" e "# sol." correspondem, respectivamente, à média dos $Desv^{Best}$ de cada conjunto de problemas-teste analisado e a quantidade de soluções encontradas por cada algoritmo iguais aos melhores resultados conhecidos na literatura. O detalhamento dos dados analisados podem ser observados pelas Tabelas 1, 2 e 3.

Tabela 4. PGENVNS-TS-CL-PR $\times$ melhores algoritmos da literatura

Conjunto de problemas-teste			Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		PGENVNS-TS-CL-PR	
	# Prob.	# sol.	Média $Desv^{Best}$	# sol.	Média $Desv^{Best}$	# sol.	Média $Desv^{Best}$	# sol.	Média $Desv^{Best}$
Dethloff (2001)	40	40	0	40	0	40	0	40	0
Salhi e Nagy (1999)	14	4	0,76	8	0,65	4	0,94	6	0,69
Montané e Galvão (2006)	18	8	0,31	15	0,01	12	0,19	12	0,17

Na Tabela 4 observa-se que nos problemas-teste de Dethloff (2001) todos os algoritmos obtiveram desempenho semelhante, encontrando sempre o melhor resultado conhecido. Já nos conjuntos de problemas-teste de Salhi e Nagy (1999) e Montané e Galvão (2006), observou-se a superioridade do PGENVNS-TS-CL-PR frente aos algoritmos de Zachariadis et al. (2010) e Mine et al. (2010). Essa superioridade se dá em termos de qualidade da solução, já que em todos eles o algoritmo proposto sempre encontra um maior número de melhores soluções, além de ter uma menor variabilidade nos melhores valores, uma vez que a média dos desvios é menor. Por outro lado, o algoritmo PILS-RVND de Subramanian et al. (2010) superou todos os demais algoritmos em todos os conjuntos de problemas-teste analisados, pois encontrou soluções melhores com menor variabilidade.

6 Conclusões

Este trabalho teve seu foco no Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Em vista de sua dificuldade de resolução na otimalidade, foi desenvolvido um algoritmo heurístico que combina os procedimentos Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas, GENIUS, *Variable Neighborhood Search* (VNS), *Variable Neighborhood Descent*, Busca Tabu e Reconexão por Caminhos. Os três primeiros procedimentos são usados para gerar uma solução inicial, que é refinada a seguir pelo VNS. O VNS, por sua vez, tem suas buscas locais feitas pelo VND e pela Busca Tabu. Além disso, o algoritmo proposto utiliza uma Lista de Candidatos na fase de refinamento com o objetivo de reduzir o número de soluções não promissoras analisadas e recebeu um módulo responsável pela paralelização das estruturas de vizinhança.

Uma bateria de testes foi feita para verificar a influência da inserção do recurso de paralelização das estruturas de vizinhança no algoritmo GENILS-TS-CL-PR. Por esses experimentos pode-se observar que o algoritmo PGENVNS-TS-CL-PR requereu um menor tempo de processamento, manteve a capacidade de encontrar as melhores soluções e, além disso, diminuiu a variabilidade das soluções finais geradas, tanto para os melhores valores quanto para os valores médios.

O objetivo de diminuir o tempo computacional foi alcançado e, levando em consideração o recurso computacional utilizado, pode-se afirmar que algoritmo proposto ainda é uma ótima solução para as empresas de transporte, pois atualmente os computadores multiprocessados são largamente encontrados no mercado e possuem ótimo custo benefício, diferentemente do recurso de *cluster* com 256 núcleos utilizado por Subramanian et al. (2010), que é consideravelmente mais caro e sofisticado. Logo, o algoritmo proposto se mostra uma ótima alternativa.

Agradecimentos

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

Referências

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

Dethloff, Jan. (2001). Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum*, v. 23, p. 79–96.

Gendreau, M.; Hertz, A. e Laporte, G. (1992). New insertion and post optimization procedures or the traveling salesman problem. *Operations Research*, v. 40, p. 1086–1094.

Gonçalves, F. A. C. A. (2010). Sequenciamento em uma máquina: otimização heurística via multiprocessamento paralelo. Dissertação de mestrado, Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, Belo Horizonte.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research A*, v. 23, n. 5, p. 377–386.

Mine, M. T.; Silva, M. S. A.; Ochi, L. S. e Souza, M. J. F. (2010). O problema de roteamento de veículos com coleta e entrega simultânea: uma abordagem via iterated local search e genius. *Transporte em transformação XIV: trabalhos vencedores do prêmio CNT de Produção Acadêmica 2009*, p. 59–78. Editora Positiva, Brasília.

Montané, Fermín Alfredo Tang e Galvão, Roberto Diéguez. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers and Operations Research*, v. 33, n. 3, p. 595–619. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2004.07.009.

Rego, C. e Roucairol, C. (1996). *Meta-Heuristics Theory and Applications*, Capítulo A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem, p. 661–675. Kluwer Academic Publisher, Boston.

Salhi, S. e Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, v. 50, p. 1034–1042.

Silva, T. C. B.; Cruz, R. C.; Mine, M. T.; Souza, M. J. F. e Santibanez, E. R. (2011). GENILS-TS: um algoritmo heurístico híbrido para resolução do problema de roteamento de veículos com coleta e entrega simultânea. *Anais do XLIII Simpósio Brasileiro de Pesquisa Operacional - XLIII SBPO*, p. 1883–1894, Bento Gonçalves, RS.

Silva, T. C. B.; Cruz, R. C.; Souza, M. J.; Martins, A. X.; Coelho, V. N. e Mine, M. T. (2012). GENILS-TS-CL-PR: Um algoritmo heurístico para resolução do problema de roteamento de veículos com coleta e entrega simultânea. *Anais do XVI CLAIO / XLIV SBPO*, Rio de Janeiro, RJ. SOBRAPO.

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

Subramanian, Anand. (2008). Metaheurística Iterated Local Search aplicada ao problema de roteamento de veículos com coleta e entrega simultânea. Dissertação de mestrado, Programa de Pós-Graduação em Ciência da Computação, UFPB, João Pessoa.

Zachariadis, Emmanouil E.; Tarantilis, Christos D. e Kiranoudis, Chris T. (2010). An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research*, v. 202, p. 401–411.