GENILS-TS: UM ALGORITMO HEURÍSTICO PARA RESOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA

Thaís Cotta Barbosa da Silva[1], Raphael Carlos Cruz[1], Marcio Tadayuki Mine[2], Marcone Jamilson Freitas Souza[1], Ernesto del Rosario Santibanez[1]

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

thais_cotta@yahoo.com.br, raphaelcarlos25@yahoo.com.br

marcone@iceb.ufop.br, santibanez.ernesto@gmail.com

[2]GAPSO
Rua Lauro Müller, 1160, sala 3402, 22.290-160, Rio de Janeiro (RJ), Brasil

marciomine@gmail.com

Abstract. *This work addresses the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). Due to its complexity, we propose a heuristic algorithm for solving it, so-called GENILS-TS. This algorithm combines six heuristic procedures Cheapest Insertion, Cheapest Insertion with multiple routes, GENIUS, Iterated Local Search (ILS), Variable Neighborhood Descent (VND) and Tabu Search (TS). The first three procedures aim to obtain an initial solution, and the VND and TS are used as local search methods for ILS. TS is called after some iterations without success by VND. The algorithm was tested on benchmark instances taken from the literature and it was able to generate good quality solutions.*

KEYWORDS*: Vehicle Routing Problem with Simultaneous Pickup and Delivery, Iterated Local Search, Variable Neighborhood Descent, GENIUS, Cheapest Insertion, Tabu Search.*

Resumo. *Este trabalho tem seu foco no Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Dada a dificuldade de solução deste problema, é proposto um algoritmo heurístico nomeado GENILS-TS, que combina os procedimentos heurísticos Inserção Mais Barata, Inserção Mais Barata Múltiplas Rotas, GENIUS,* Iterated Local Search *(ILS), Descida em Vizinhança Variável (VND) e Busca Tabu (TS). Os três primeiros procedimentos visam à obtenção de uma solução inicial, enquanto os procedimentos VND e Busca Tabu são usados como métodos de busca local para o ILS. A Busca Tabu somente é acionada após certo número de iterações sem sucesso do VND. O algoritmo proposto foi testado em problemas-teste disponíveis na literatura e se mostrou capaz de gerar soluções de qualidade.*

PALAVRAS-CHAVE*: Problema de Roteamento de Veículos com Coleta e Entrega Simultânea, Iterated Local Search, Descida em Vizinhança Variável, GENIUS, Inserção Mais Barata, Busca Tabu.*

1 Introdução

O Problema de Roteamento de Veículos (PRV), do inglês *Vehicle Routing Problem* (VRP), foi proposto por Dantzig and Ramser (1959), e pode ser definido como segue. Dado um conjunto *N* de clientes, cada qual associado a uma demanda d_i, e uma frota homogênea de veículos de capacidade *Q*, o objetivo é obter um conjunto de rotas a serem percorridas pelos veículos cujo custo seja mínimo, levando em consideração o atendimento completo da demanda dos clientes.

Uma importante variação do PRV, objeto de estudo deste trabalho, é o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES), ou VRPSPD, do inglês *Vehicle Routing Problem with Simultaneous Pickup and Delivery*. Proposto por Min (1989), este problema se diferencia do PRV clássico por ter associado aos clientes não só uma demanda d_i, mas também uma quantidade p_i de produtos a serem coletados, sendo que ambas as operações devem ser realizadas simultaneamente.

O PRVCES é da classe NP-difícil, uma vez que ele pode ser reduzido ao PRV clássico quando nenhum cliente necessita de serviço de coleta. Dada a dificuldade de obter a melhor solução, no caso geral, em tempos aceitáveis, as pesquisas por métodos de solução para o PRVCES têm se concentrado em heurísticas. Alguns dos principais algoritmos heurísticos são brevemente comentados a seguir.

Wassan et al. (2007) propuseram uma Busca Tabu Reativa para resolvê-lo. Zachariadis et al. (2009) usaram uma técnica híbrida, unindo as metaheurísticas Busca Tabu (Glover and Laguna, 1997) e *Guided Local Search* (Voudouris and Tsang, 1996). Posteriormente, os mesmos autores, em Zachariadis et al. (2010), propuseram um algoritmo evolucionário que utiliza uma memória adaptativa para guardar as 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, posteriormente, melhoradas pela Busca Tabu. Mine et al. (2010) propuseram um algoritmo, nomeado GENILS, que utiliza a melhor solução gerada pelos métodos de Inserção Mais Barata, 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 (Lourenço et al., 2003), tendo o *Variable Neighborhood Descent* – VND (Hansen and Mladenović, 2001) como método de busca local.

Subramanian et al. (2010) apresentaram um algoritmo paralelo para a solução do PRVCES. 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 agora conhecido.

Neste artigo focamos a solução do PRVCES por meio de algoritmos não-paralelos. Dada a simplicidade e competitividade do algoritmo GENILS frente aos demais algoritmos sequenciais da literatura, ele foi aperfeiçoado com a inserção de um módulo de Busca Tabu substituindo a fase de refinamento do ILS a partir de um determinado número de iterações sem melhora no processo de busca. A justificativa para isso foi o fato de que

alguns dos melhores resultados obtidos em problemas-teste consagrados da literatura foram obtidos por algoritmos que têm a Busca Tabu como algoritmo base. Outra justificativa, é que o algoritmo proposto, nomeado GENILS-TS, tem a vantagem de utilizar recursos computacionais mais acessíveis do que aqueles de Subramanian et al. (2010), o que permite seu uso em empresas de menor porte.

O restante deste artigo está organizado como segue. Na seção 2 descrevemos o problema de roteamento de veículos com coleta e entrega simultânea. Na seção 3 apresentamos a metodologia de solução do problema e na seção 4, os resultados dos experimentos computacionais. Na seção 5 o trabalho é concluído.

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, 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.

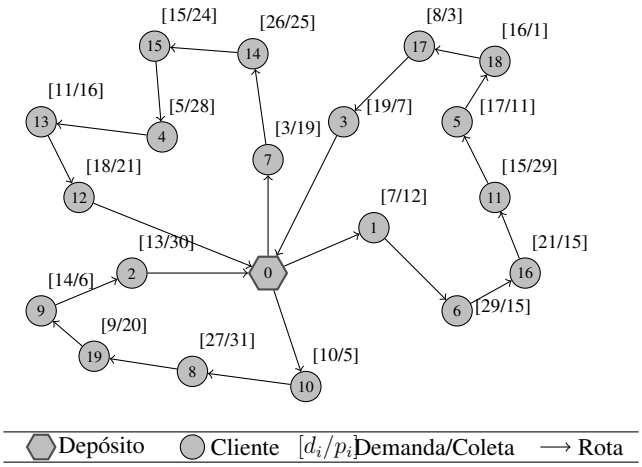

Figura 1. Exemplo do PRVCES.

Na rota do veículo do canto superior esquerdo da Figura 1, o veículo parte do depósito, passa pelo 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 Metodologia

Nesta Seção é apresentada a metodologia utilizada para resolver o PRVCES. Na Subseção 3.1 é apresentado o algoritmo proposto e nas seguintes é feito seu detalhamento.

3.1 Algoritmo GENILS-TS

O algoritmo proposto, nomeado GENILS-TS, é um aperfeiçoamento do algoritmo GENILS de Mine et al. (2010). Este algoritmo combina as heurísticas ILS, VND, Busca

Tabu (TS – do inglês *Tabu Search*), assim como os procedimentos heurísticos Inserção Mais Barata, Inserção Mais Barata Múltiplas Rotas e GENIUS. Seu pseudocódigo é apresentado pelo Algoritmo 1.

Algoritmo 1: *GENILS-TS*

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: $s' \leftarrow$ *Perturbação*(s);
11: **se** ($iter \leq iterMaxSemMelhora$) **então**
12: $s'' \leftarrow$ *VND*(s');
13: **senão**
14: $s'' \leftarrow$ *BuscaTabu*$(s', TamListaTabu, iterMaxTS)$;
15: **fim se**
16: **se** ($f(s'') < f(s)$) **então**
17: $s \leftarrow s''$; $iter \leftarrow 1$;
18: **senão**
19: $iter \leftarrow iter + 1$;
20: **fim se**
21: **fim enquanto**
22: Retorne s;

Como pode ser observado no Algoritmo 1, os três métodos construtivos utilizados, os quais são descritos na Subseção 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 ILS. Para fugir de ótimos locais, se dirigindo para outras regiões do espaço de busca, foram utilizadas perturbações descritas na Subseção 3.8. Já os métodos VND e Busca Tabu (descritos nas Subseções 3.6 e 3.7, respectivamente) são utilizados como busca local, sendo a Busca Tabu acionada somente após certo número de iterações sem melhora, dado pelo parâmetro *iterMaxSemMelhora*.

3.2 Função de Avaliação

Uma solução é avaliada pela expressão (1). A primeira parcela dessa função corresponde à distância total percorrida. Já 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 β, que corresponde a um valor suficientemente grande. Vale ressaltar que a geração de soluções inviáveis, onde a carga do veiculo não é respeitada, é permitida, porém não é incentivada, já que a função de avaliação deve ser minimizada.

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

Na função de avaliação f, dada pela expressão (1), têm-se:
N: conjunto dos clientes, incluindo o depósito;
A: conjunto de arcos (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 o arco $(i, j) \in A$ é utilizado ($x_{ij} = 1$) 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 por último, uma adaptação da heurística GENIUS.

A Inserção Mais Barata Rota a Rota consiste basicamente na construção de 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 incluido 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.

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

O cliente k com o menor custo de inserção é adicionado à rota desde que a restrição de carga do veículo seja respeitada. O valor do parâmetro $\gamma \in [0, 1]$ é utilizado para favorecer clientes mais distantes do depósito, que normalmente entrariam tardiamente nas soluções, pelo seu alto custo de inserção.

A heurística Inserção Mais Barata com Múltiplas Rotas foi proposta por Subramanian (2008). Como parâmetro do método é necessário informar a *priori* o número de rotas, que neste caso é dada pelo algoritmo de Inserção Mais Barata Rota a Rota. Assim, cada uma das rotas é iniciada por um único cliente escolhido aleatoriamente. 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 os procedimentos *3-opt* e *4-opt*. O funcionamento desses métodos é descrito, em detalhes, no trabalho indicado.

3.4 Estruturas de vizinhança

Neste trabalho, foram aplicados sete movimentos diferentes para explorar o espaço de soluções, todos bem conhecidos na literatura. O *Shift* é o movimento de realocação, e consiste em transferir um cliente de uma rota para outra. Já o *Shift*(2,0) consiste na realocação de 2 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 outro lugar na mesma rota, sendo k um parâmetro do método; no nosso caso, k assume o valor 3, no máximo.

3.5 *G3-opt*, *G4-opt* e *reverse*

Os metódos *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.

3.6 VND

O VND explora o espaço de soluções por meio de uma mudança sistemática de vizinhanças. No VND clássico considera-se um conjunto de vizinhanças em uma ordem pré-estabelecida, sendo que quando uma solução melhor é encontrada, retorna-se à primeira vizinhança.

No VND implementado, a cada iteração a ordem das vizinhanças a serem exploradas pode mudar, pois há uma ordenação aleatória das sete vizinhanças descritas na Subseção 3.4. Logo depois é feita uma intensificação por buscas locais utilizando as sete vizinhanças, seguido dos procedimentos *G3-opt*, *G4-opt* e *reverse*, apresentados na Subseção 3.5. O pseudocódigo do VND é apresentado no Algoritmo 2.

Algoritmo 2: VND

Seja r o número de estruturas distintas de vizinhanças;
$\mathcal{RN} \leftarrow$ conjunto das vizinhanças $\mathcal{N}$, descritas na Seção 3.4, em ordem aleatória;
$k \leftarrow 1$;
enquanto ($k \leq r$) **faça**
  Encontre o melhor vizinho $s' \in \mathcal{RN}^{(k)}(s)$;
  se ($f(s') < f(s)$) **então**
    $s \leftarrow s'$; $k \leftarrow 1$;
    {Intensificação nas rotas alteradas}
    $s \leftarrow$ *BuscaLocalShift*(s);
    $s \leftarrow$ *BuscaLocalShift20*(s);
    $s \leftarrow$ *BuscaLocalSwap*(s);
    $s \leftarrow$ *BuscaLocal2-Opt*(s);
    $s \leftarrow$ *BuscaLocalSwap21*(s);
    $s \leftarrow$ *BuscaLocalSwap22*(s);
    $s \leftarrow$ *BuscaLocalG3-Opt*(s);
    $s \leftarrow$ *BuscaLocalG4-Opt*(s);
    $s \leftarrow$ *BuscaLocalkOr-Opt*(s) com $k = 3, 4, 5$;
    $s \leftarrow$ *Reverse*(s);
  senão
    $k \leftarrow k + 1$;
  fim se
fim enquanto
Retorne s;

3.7 Busca Tabu

Busca Tabu (TS - *Tabu Search*) é uma metaheurística que utiliza uma estrutura de memória para explorar o espaço de soluções de um problema. A TS caminha no espaço de soluções movendo-se de uma solução para outra que seja seu melhor vizinho, mesmo que este vizinho seja gerado por um movimento de piora. Como estratégia para não retornar a uma solução já gerada anteriormente e, portanto, ciclar, a TS guarda em uma lista, a

chamada lista tabu, informações sobre as soluções já geradas. O tamanho dessa lista é um parâmetro do método, que determina quanto tempo o movimento vai permanecer proibido (tabu). O Algoritmo 3 apresenta o pseudocódigo da Busca Tabu implementada.

Algoritmo 3: TS (*TamListaTabu*, *iterMaxTS*, *Solução* s)

1: $s^* \leftarrow s$; {Melhor solução obtida até então}
2: *TamOriginal* $\leftarrow$ *TamListaTabu*; {Tamanho original da Lista Tabu}
3: *iter* $\leftarrow 0$; {Número de iterações}
4: *MelhorIter* $\leftarrow 0$;{Número de iterações sem melhora na solução}
5: **enquanto** ($iter - MelhorIter \leq iterMaxTS$) **faça**
6: **se** ($iter - MelhorIter \leq IterAux$) **então**
7: *TamListaTabu* $\leftarrow$ *TamListaTabu* $+ 1$;
8: **fim se**
9: $s' \leftarrow Shift(s)$;
10: $s'' \leftarrow Swap(s)$;
11: $s \leftarrow \text{argmin}\{f(s'), f(s'')\}$;
12: *AtualizaListaTabu*(*TamListaTabu*, *Lista*, *Ant*, *Prox*, *iter*, Δ) {Algoritmo 4 }
13: **se** ($f(s) < f(s^*)$) **então**
14: $s^* \leftarrow s$;
15: *MelhorIter* $\leftarrow$ *iter*;
16: *TamListaTabu* $\leftarrow$ *TamOriginal*;
17: **fim se**
18: *iter* $\leftarrow$ *iter* $+ 1$;
19: **fim enquanto**
20: Retorne s^*;

No Algoritmo 3, duas soluções s' e s'' são geradas a cada iteração utilizando os movimentos *Shift* e *Swap* descritos na Subseção 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, como descrita na Subseção 3.7.1. O tamanho da Lista Tabu (*TamListaTabu*) é incrementado em uma unidade à medida que o número de iterações sem melhora aumenta; porém, para o algoritmo não ficar extremamente restritivo, essa atualização para de ser efetuada quando certo número de iterações sem melhora é atingido (linhas 6 e 7 do Algoritmo 3). O procedimento é interrompido quando o número máximo de iterações sem melhora na solução é alcançado.

3.7.1 Lista Tabu

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 (*Prox*) do cliente adjacente (*Ant*) a ele antes do movimento.

Para se atualizar a Lista Tabu é utilizada a função descrita pelo Algoritmo 4.

Algoritmo 4: *AtualizaListaTabu(TamListaTabu, Lista, Ant, Prox, iter,* Δ*)*

1: min $\leftarrow$ *TamListaTabu* $+ iter - \Delta$;
2: max $\leftarrow$ *TamListaTabu* $+ iter + \Delta$;
3: *DuracaoTabu* $\leftarrow$ *Valor aleatório no intervalo* $[\min, \max]$;
4: *Lista*(*Ant*, *Prox*) $\leftarrow$ *DuracaoTabu*;
5: Retorne *Lista*;

Neste trabalho foi utilizada uma Lista Tabu de tamanho 10, e um valor Δ igual a 3. Desta forma, a lista assume tamanhos diferentes, devido à aleatoriedade presente no

procedimento. Quando o tamanho da lista é menor, ou seja, menos proibitiva, se incentiva a intensificação da busca naquele espaço de soluções, contrastando com a diversificação, que ocorre quando a lista é maior. Sendo assim, além de explorar melhor o espaço de soluções, essa estratégia procura evitar a ocorrência de ciclagem.

Para representar computacionalmente a Lista Tabu, utilizou-se uma matriz quadrada, em que as linhas e colunas correspondem aos clientes. Quando o cliente i fica proibido de ser sucessor do cliente j, é atualizada a posição (i, j) dessa matriz. As Figuras 2 e 3 ilustram essa operação com o valor (variável *DuracaoTabu*) calculado pelo Algoritmo 4. A principal vantagem dessa representação está na ordem de complexidade da consulta para verificar se o movimento é tabu, a qual é $O(1)$. Caso fosse utilizada uma lista encadeada, a consulta seria $O(m)$ no pior caso, considerando m o tamanho da lista.

	0	1	2	3	4	5	6	...	19
0	0	0	0	0	0	0	0	...	0
1	0	0	0	0	0	0	0	...	0
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
19	0	0	0	0	0	0	0	...	0

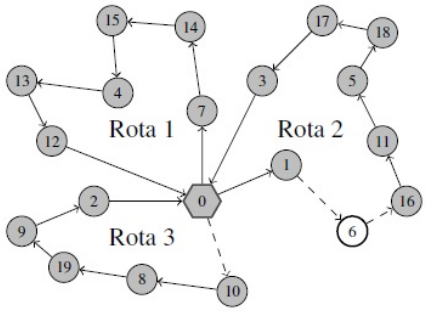

Figura 2. Configuração da Matriz antes do Movimento

	0	1	2	3	4	5	6	...	19
0	0	0	0	0	0	0	0	...	0
1	0	0	0	0	0	0	10	...	0
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
19	0	0	0	0	0	0	0	...	0

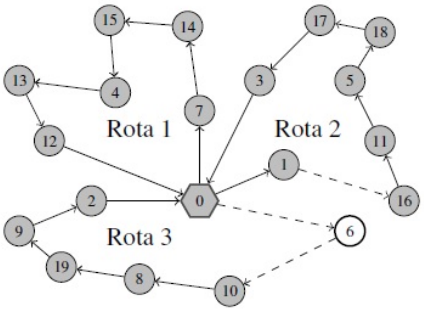

Figura 3. Atualização da Matriz após o movimento

Na Figura 2, o lado esquerdo representa a Lista Tabu em forma de matriz. Considerando que é a primeira iteração, ainda não existe nenhum movimento tabu. Já do lado direito temos a configuração da solução, onde pode ser observado que na rota 2 o veículo sai do depósito e atende aos clientes 1, 6, 16, 11, 5, 18, 17 e 3, nesta ordem, e retorna ao depósito. Na Figura 3 mostra-se o cliente 6 realocado da rota 2 para rota 3 pelo movimento *Shift*. A matriz tabu tem, então, a sua linha 1 e coluna 6 atualizada, já que na configuração anterior o cliente 6 é o sucessor do cliente 1. Neste exemplo, qualquer movimento em que o cliente 1 seja sucedido pelo cliente 6 estará proibido até a iteração 10.

3.8 Perturbações

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

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.

O *Ejection chain* foi proposto por Rego and Roucairol (1996). Inicialmente, seleciona-se 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. Nesse movimento os clientes são escolhidos de forma aleatória.

4 Resultados Parciais

O algoritmo GENILS-TS foi codificado em C++ usando o compilador Visual C++ 2005 e executado em um microcomputador *Quad core*, 1,66 GHz e 4 GB de memória RAM e sistema operacional Windows Vista. Apesar de o microcomputador possuir quatro núcleos, o algoritmo proposto não explora multiprocessamento.

Para testar o algoritmo foram usados 40 problemas-teste de Dethloff (2001); 14 de Salhi and Nagy (1999) e 18 de Montané and Galvão (2006). Os parâmetros adotados foram os seguintes: *TamListaTabu* = 10, *iterMaxTS* =300, *maxIter* = 10000 e $\Delta = 3$.

Dado seu caráter estocástico, o algoritmo foi executado 50 vezes em cada problema-teste. As Tabelas 1, 2 e 3 apresentam os melhores resultados encontrados pelo GENILS-TS e os compara com os melhores resultados de Zachariadis et al. (2010), Subramanian et al. (2010) (256 núcleos) e Mine et al. (2010). Nesta tabela, a primeira coluna representa o problema-teste e a segunda o melhor resultado da literatura. As colunas *Melhor* e $Desv^{Best}$ apresentam, respectivamente, o melhor resultado de cada algoritmo e o desvio percentual em relação ao melhor resultado conhecido na literatura. A coluna *Tempo* apresenta o tempo médio das 50 execuções do GENILS-TS, em segundos.

Como pode ser observado na Tabela 1, o algoritmo proposto foi capaz de alcançar as melhores soluções conhecidas em todos os problemas-teste de Dethloff (2001).

Analisando a Tabela 2, na qual são apresentados os resultados nos problemas-teste de Salhi and Nagy (1999), verifica-se que o algoritmo de Subramanian et al. (2010) é o de melhor desempenho. De fato, ele alcança 9 das melhores soluções e um desvio médio de 0,65%, GENILS-TS obtém 6 melhores resultados da literatura e um desvio médio de 0,70%, o algoritmo de Zachariadis et al. (2010), por sua vez, encontrou 4 e obteve um desvio de 0,76%, enquanto o de Mine et al. (2010) alcançou 5 das melhores soluções da literatura, mas com um desvio mais elevado, de 0,94%.

Com relação à Tabela 3, que apresenta os resultados nos problemas-teste de Montané and Galvão (2006), observa-se a superioridade absoluta do algoritmo de Subramanian et al. (2010). Este algoritmo é o que encontra a maior quantidade de melhores soluções (15) e tem o menor desvio (0,01%), seguido do de Mine et al. (2010), com 11 melhores soluções e um desvio de 0,19%. O algoritmo proposto é o de pior desempenho, em termos da média do desvio, com 0,36%. No entanto, o GENILS-TS supera o de Zachariadis et al. (2010) no número de melhores soluções encontradas, no caso, 9, contra 8 e, além disso, encontra duas soluções melhores que a de Subramanian et al. (2010).

Considerando o conjunto de todos os 72 problemas-teste, em termos de número de melhores resultados, tem-se: Subramanian et al. (2010): 64; Mine et al. (2010): 57, GENILS-TS: 55 e Zachariadis et al. (2010): 52. Já em termos de desvio médio, tem-se:

Subramanian et al. (2010): 0,22%, GENILS-TS: 0,35%, Zachariadis et al. (2010): 0,36% e Mine et al. (2010): 0,38%.

Tabela 1. Resultados para os problemas-teste de Dethloff (2001)

Problema	MelhorLit	Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		GENILS-TS		
-	-	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Tempo(s)**
SCA3-0	**635,62**	**635,62**	0,00	**635,62**	0,00	**635,62**	0,00	**635,62**	0,00	200,31
SCA3-1	**697,84**	**697,84**	0,00	**697,84**	0,00	**697,84**	0,00	**697,84**	0,00	200,31
SCA3-2	**659,34**	**659,34**	0,00	**659,34**	0,00	**659,34**	0,00	**659,34**	0,00	200,31
SCA3-3	**680,04**	**680,04**	0,00	**680,04**	0,00	**680,04**	0,00	**680,04**	0,00	200,30
SCA3-4	**690,50**	**690,50**	0,00	**690,50**	0,00	**690,50**	0,00	**690,50**	0,00	200,31
SCA3-5	**659,90**	**659,90**	0,00	**659,90**	0,00	**659,90**	0,00	**659,90**	0,00	200,30
SCA3-6	**651,09**	**651,09**	0,00	**651,09**	0,00	**651,09**	0,00	**651,09**	0,00	200,31
SCA3-7	**659,17**	**659,17**	0,00	**659,17**	0,00	**659,17**	0,00	**659,17**	0,00	200,30
SCA3-8	**719,48**	**719,48**	0,00	**719,48**	0,00	**719,48**	0,00	**719,48**	0,00	200,31
SCA3-9	**681,00**	**681,00**	0,00	**681,00**	0,00	**681,00**	0,00	**681,00**	0,00	200,30
SCA8-0	**961,50**	**961,50**	0,00	**961,50**	0,00	**961,50**	0,00	**961,50**	0,00	200,17
SCA8-1	**1049,65**	**1049,65**	0,00	**1049,65**	0,00	**1049,65**	0,00	**1049,65**	0,00	200,17
SCA8-2	**1039,64**	**1039,64**	0,00	**1039,64**	0,00	**1039,64**	0,00	**1039,64**	0,00	200,17
SCA8-3	**983,34**	**983,34**	0,00	**983,34**	0,00	**983,34**	0,00	**983,34**	0,00	200,15
SCA8-4	**1065,49**	**1065,49**	0,00	**1065,49**	0,00	**1065,49**	0,00	**1065,49**	0,00	200,17
SCA8-5	**1027,08**	**1027,08**	0,00	**1027,08**	0,00	**1027,08**	0,00	**1027,08**	0,00	200,17
SCA8-6	**971,82**	**971,82**	0,00	**971,82**	0,00	**971,82**	0,00	**971,82**	0,00	200,17
SCA8-7	**1051,28**	**1051,28**	0,00	**1051,28**	0,00	**1051,28**	0,00	**1051,28**	0,00	200,17
SCA8-8	**1071,18**	**1071,18**	0,00	**1071,18**	0,00	**1071,18**	0,00	**1071,18**	0,00	200,17
SCA8-9	**1060,50**	**1060,50**	0,00	**1060,50**	0,00	**1060,50**	0,00	**1060,50**	0,00	200,17
CON3-0	**616,52**	**616,52**	0,00	**616,52**	0,00	**616,52**	0,00	**616,52**	0,00	200,15
CON3-1	**554,47**	**554,47**	0,00	**554,47**	0,00	**554,47**	0,00	**554,47**	0,00	200,30
CON3-2	**518,00**	**518,00**	0,00	**518,00**	0,00	**518,00**	0,00	**518,00**	0,00	200,31
CON3-3	**591,19**	**591,19**	0,00	**591,19**	0,00	**591,19**	0,00	**591,19**	0,00	200,17
CON3-4	**588,79**	**588,79**	0,00	**588,79**	0,00	**588,79**	0,00	**588,79**	0,00	200,30
CON3-5	**563,70**	**563,70**	0,00	**563,70**	0,00	**563,70**	0,00	**563,70**	0,00	200,30
CON3-6	**499,05**	**499,05**	0,00	**499,05**	0,00	**499,05**	0,00	**499,05**	0,00	200,17
CON3-7	**576,48**	**576,48**	0,00	**576,48**	0,00	**576,48**	0,00	**576,48**	0,00	200,17
CON3-8	**523,05**	**523,05**	0,00	**523,05**	0,00	**523,05**	0,00	**523,05**	0,00	200,31
CON3-9	**578,25**	**578,25**	0,00	**578,25**	0,00	**578,25**	0,00	**578,25**	0,00	200,31
CON8-0	**857,17**	**857,17**	0,00	**857,17**	0,00	**857,17**	0,00	**857,17**	0,00	200,15
CON8-1	**740,85**	**740,85**	0,00	**740,85**	0,00	**740,85**	0,00	**740,85**	0,00	200,15
CON8-2	**712,89**	**712,89**	0,00	**712,89**	0,00	**712,89**	0,00	**712,89**	0,00	200,17
CON8-3	**811,07**	**811,07**	0,00	**811,07**	0,00	**811,07**	0,00	**811,07**	0,00	200,17
CON8-4	**772,25**	**772,25**	0,00	**772,25**	0,00	**772,25**	0,00	**772,25**	0,00	200,17
CON8-5	**754,88**	**754,88**	0,00	**754,88**	0,00	**754,88**	0,00	**754,88**	0,00	200,30
CON8-6	**678,92**	**678,92**	0,00	**678,92**	0,00	**678,92**	0,00	**678,92**	0,00	200,30
CON8-7	**811,96**	**811,96**	0,00	**811,96**	0,00	**811,96**	0,00	**811,96**	0,00	200,31
CON8-8	**767,53**	**767,53**	0,00	**767,53**	0,00	**767,53**	0,00	**767,53**	0,00	200,31
CON8-9	**809,00**	**809,00**	0,00	**809,00**	0,00	**809,00**	0,00	**809,00**	0,00	200,31
Média	**-**	-	0,00	-	0,00	-	0,00	-	0,00	-

5 Conclusões

Este trabalho teve seu foco no Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Dada a dificuldade de resolução desse problema em tempos computacionais aceitáveis no caso geral, foi proposto um algoritmo heurístico que combina os procedimentos *Iterated Local Search*, *Variable Neighborhood Descent*, Busca Tabu, Inserção Mais Barata Rota a Rota, Inserção Mais Barata com Múltiplas Rotas e uma adaptação da heurística GENIUS.

O algoritmo, nomeado GENILS-TS, foi testado em três conjuntos de problemas-teste da literatura e comparado com três algoritmos eficientes da literatura. Claramente, o algoritmo de Subramanian et al. (2010) foi o de melhor desempenho. Entretanto, esse algoritmo usa 256 núcleos de processamento para explorar o espaço de soluções, enquanto os demais usam apenas um. Esses últimos, por sua vez, têm desempenho bastante seme-

Tabela 2. Resultados para os problemas-teste de Salhi and Nagy (1999)

Problema	MelhorLit	Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		GENILS-TS		
-	-	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Tempo(s)**
CMT1X	**466,77**	469,8	0,65	**466,77**	0,00	**466,77**	0,00	**466,77**	0,00	200,26
CMT1Y	**466,77**	469,8	0,65	**466,77**	0,00	**466,77**	0,00	**466,77**	0,00	200,26
CMT2X	**668,77**	684,21	2,31	684,21	2,31	684,11	2,29	684,11	2,29	200,55
CMT2Y	**663,25**	684,21	3,16	684,21	3,16	684,11	3,15	684,11	3,15	200,56
CMT3X	**721,27**	**721,27**	0,00	**721,27**	0,00	**721,40**	0,02	**721,27**	0,00	201,10
CMT3Y	**721,27**	**721,27**	0,00	**721,27**	0,00	**721,27**	0,00	**721,27**	0,00	201,09
CMT12X	**644,7**	662,22	2,72	662,22	2,72	662,22	2,72	662,22	2,72	201,08
CMT12Y	**659,52**	662,22	0,41	662,22	0,41	663,50	0,60	663,50	0,60	501,09
CMT11X	**833,92**	**833,92**	0,00	**833,92**	0,00	846,23	1,48	**833,92**	0,00	501,67
CMT11Y	**830,39**	833,92	0,43	833,92	0,43	836,04	0,68	833,92	0,43	501,60
CMT4X	**852,46**	**852,46**	0,00	**852,46**	0,00	**852,46**	0,00	**852,46**	0,00	502,55
CMT4Y	**852,35**	852,46	0,01	**852,46**	0,01	862,28	1,17	855,52	0,37	502,57
CMT5X	**1029,25**	1030,55	0,13	**1029,25**	0,00	1033,51	0,41	1030,5	0,13	504,87
CMT5Y	**1029,25**	1030,55	0,13	**1029,25**	0,00	1036,14	0,67	1030,5	0,13	504,86
Média	**-**	-	0,76	-	0,65	-	0,94	-	0,70	-

Tabela 3. Resultados para os problemas-teste de Montané and Galvão (2006)

Problema	MelhorLit	Zachariadis *et al.* (2010)		Subramanian *et al.* (2010)		Mine *et al.*(2010)		GENILS-TS		
-	-	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Melhor**	**Desv^{Best}**	**Tempo(s)**
r101	**1009,95**	**1009,95**	0,00	**1009,95**	0,00	**1009,95**	0,00	**1009,95**	0,00	200,85
r201	**666,20**	**666,20**	0,00	**666,20**	0,00	**666,20**	0,00	**666,20**	0,00	201,17
c101	**1220,18**	1220,99	0,07	**1220,18**	0,00	**1220,18**	0,00	**1220,18**	0,00	200,71
c201	**662,07**	**662,07**	0,00	**662,07**	0,00	**662,07**	0,00	**662,07**	0,00	201,14
rc101	**1059,32**	**1059,32**	0,00	**1059,32**	0,00	**1059,32**	0,00	**1059,32**	0,00	205,38
rc201	**672,92**	**672,92**	0,00	**672,92**	0,00	**672,92**	0,00	**672,92**	0,00	200,99
r1_2_1	**3357,64**	3376,30	0,55	3360,02	0,07	**3357,64**	0,00	**3357,64**	0,00	204,22
r2_2_1	**1665,58**	**1665,58**	0,00	**1665,58**	0,00	**1665,58**	0,00	1672,5	0,42	205,34
c1_2_1	**3629,89**	3643,82	0,38	**3629,89**	0,00	3636,74	0,19	3645,45	0,43	203,69
c2_2_1	**1726,59**	**1726,59**	0,00	**1726,59**	0,00	**1726,59**	0,00	1731,74	0,30	204,73
rc1_2_1	**3306,00**	3323,56	0,53	**3306,00**	0,00	3312,92	0,21	3344,79	1,17	204,08
rc2_2_1	**1560,00**	**1560,00**	0,00	**1560,00**	0,00	**1560,00**	0,00	**1560,00**	0,00	503,34
r1_4_1	**9605,75**	9691,60	0,89	**9605,75**	0,00	9627,43	0,23	9627,88	0,23	524,81
r2_4_1	**3551,38**	3572,38	0,59	**3551,38**	0,00	3582,08	0,86	3595,88	1,25	526,92
c1_4_1	**11098,21**	11179,36	0,73	11099,54	0,01	**11098,21**	0,00	11125,55	0,25	505,34
c2_4_1	**3546,10**	3549,27	0,09	**3546,10**	0,00	3596,37	1,42	3605,22	1,67	525,94
rc1_4_1	**9535,46**	9645,27	1,14	9536,77	0,01	**9535,46**	0,00	**9535,46**	0,00	221,51
rc2_4_1	**3403,70**	3423,62	0,58	**3403,70**	0,00	3422,11	0,54	3432,41	0,84	526,74
Média	**-**	-	0,31	-	0,01	-	0,19	-	0,36	-

lhante, com destaque para o GENILS-TS pelo fato de produzir soluções com o segundo menor desvio médio. Além disso, o GENILS-TS é um algoritmo mais simples de ser implementado que o de Zachariadis et al. (2010) e requerer a calibragem de muito menos parâmetros. O GENILS-TS é, assim, uma alternativa mais simples e também de menor custo, já que pode ser aplicado em computadores comuns e não exige uma estrutura computacional mais sofisticada.

Muitos aperfeiçoamentos ainda podem ser feitos no algoritmo proposto com o intuito de melhorar a qualidade de seus resultados. Como trabalho futuro pretende-se sofisticar a Busca Tabu com a inserção de outras estruturas de vizinhança, memória de longo prazo e Reconexão por Caminhos.

Agradecimentos

Os autores agradecem à UFOP, ao CNPq (processos 482765/2010-0 e 306458/2010-1) e à FAPEMIG (processos CEX-PPM-00357/09 e CEX-APQ-01201-09) pelo apoio ao desenvolvimento deste trabalho.

Referências

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6:80–91.

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

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

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publisher, Boston.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Lourenço, H. R., Martin, O., and Stützle, T. (2003). Iterated local search. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 321–353. Kluwer Academic Publishers, Norwell, MA.

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

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

Montané, F. A. T. and Galvão, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers and Operations Research*, 33(3):595–619.

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

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

Subramanian, A. (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.

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

Voudouris, C. and Tsang, E. (1996). Partial constraint satisfaction problems and guided local search. In *In The Second International Conference on the Practical Application of Constraint Technology (PACT'96)*, pages 337–356.

Wassan, N. A., Wassan, A. H., and Nagy, G. (2007). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, 15(4):368–386.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2009). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, 36(2):1070–1081.

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