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

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

[1]Departamento de Ciência da Computação - Universidade Federal de Ouro Preto
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, xmartins@decea.ufop.br

Resumo

Este trabalho tem seu foco no Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Dada sua dificuldade de solução na otimalidade, é proposto um algoritmo heurístico, chamado de GENILS-TS-CL, que combina os procedimentos heurísticos Inserção Mais Barata Rota a Rota, Inserção Mais Barata com 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 também usa uma estratégia para limitar o número de soluções avaliadas no espaço de soluções, denominada Lista de Candidatos. O algoritmo proposto foi testado em problemas-teste disponíveis na literatura e se mostrou capaz de gerar soluções de qualidade.

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

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-CL. This algorithm combines the heuristic procedures Cheapest Insertion, Cheapest Insertion with multiple routes, GENIUS, Iterated Local Search (ILS), Variable Neighborhood Descent (VND), Tabu Search (TS). The first three procedures aim to obtain an initial solution, and VND and TS are used as local search methods for ILS. TS is called after some iterations without success of the VND method. The algorithm also uses a strategy to limit the number of solutions evaluated in the exploration of the solution space, called Candidate List.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.

• INTRODUÇÃO

O Problema de Roteamento de Veículos (PRV), do inglês *Vehicle Routing Problem* (VRP), foi proposto por Dantzig e Ramser (1959) e é definido sobre 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 gerar um conjunto de rotas a serem realizadas pelos veículos, de custo mínimo, que atenda a demanda de todos os clientes.

Uma importante variação do PRV, objeto de estudo neste 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), ele se diferencia do PRV 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 um problema NP-difícil, sendo, por isso, resolvido principalmente por métodos heurísticos. Alguns deles estão comentados a seguir.

Zachariadis *et al.* (2009) usaram um método híbrido, combinando as metaheurísticas Busca Tabu e *Guided Local Search* (Voudouris e Tsang, 1996). Posteriormente, eles propuseram um algoritmo evolucionário (Zachariadis *et al*., 2010), que utiliza uma memória adaptativa para guardar as informações das soluções de alta qualidade obtidas durante a busca. Tais 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) desenvolveram um algoritmo, nomeado 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 utilizada a metaheurística *Iterated Local Search* (ILS), tendo o *Variable Neighborhood Descent* (VND) como método de busca local. Silva *et al* (2011) aperfeiçoaram o GENILS, inserindo um módulo de Busca Tabu como alternativa para a busca local do ILS. A Busca Tabu é chamada após um número fixo de iterações sem melhora do VND. Esta variante do GENILS, nomeada GENILS-TS, mostrou-se superior aos algoritmos sequenciais da literatura em termos de número de melhores soluções encontradas.

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

Neste trabalho é expandido e aprimorado o algoritmo GENILS-TS de Silva *et al* (2011). O algoritmo proposto, denominado GENILS-TS-CL, incorpora ao GENILS-TS uma Lista de Candidatos para evitar a avaliação de soluções não promissoras. Os experimentos realizados mostraram que esta estratégia faz reduzir o

tempo de resolução do problema sem perda da qualidade da solução.

O restante deste trabalho está organizado como segue. Na seção 2 é apresentado o algoritmo proposto e na seção 3 são mostrados e comparados os resultados desse algoritmo com outros da literatura. A seção 4 finaliza o trabalho e aponta propostas para aperfeiçoamento do algoritmo desenvolvido.

• DESCRIÇÃO DO PROBLEMA

O Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES) envolve um conjunto N de clientes, cada qual com uma demanda d_i e coleta p_i a serem completamente atendidas, 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.

A Figura 1 ilustra um exemplo de uma solução para um PRVCES envolvendo 19 clientes e veículos de capacidade Q = 150 unidades.

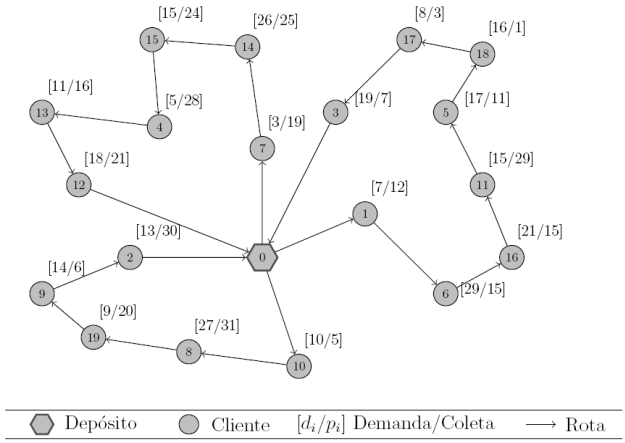

Figura 1 - Exemplo do PRVCES

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.

• ALGORITMO PROPOSTO

Nesta Seção é apresentado o algoritmo proposto para resolver o PRVCES e detalhados todos os seus módulos.

• Algoritmo GENILS-TS-CL

O algoritmo proposto, nomeado GENILS-TS-CL, é um aperfeiçoamento do algoritmo GENILS-TS de Silva *et al.* (2011). Ele combina os procedimentos heurísticos Inserção Mais Barata Rota a Rota (Dethloff, 2001), Inserção Mais Barata com Múltiplas Rotas (Subramanian, 2008) e GENIUS (Gendreau *et al*., 1992) para gerar uma solução inicial e ILS, VND (Hansen e Mladenovic, 2001) e Busca Tabu – *TS*, do inglês *Tabu Search* (Glover e Laguna, 1997) para o refinamento. Adicionalmente, ele utiliza uma Lista de Candidatos durante a fase de refinamento para evitar a geração de movimentos não promissores. Seu pseudocódigo é apresentado no Algoritmo 1.

Como pode ser observado no Algoritmo 1, os três métodos construtivos

utilizados (brevemente descritos na Subseção 3.3) produzem três soluções distintas, as quais, por sua vez, são refinadas por um procedimento VND. A melhor solução gerada se transforma na solução inicial *s*, sendo esta refinada pelo ILS. Como forma de explorar melhor o espaço de soluções, o ILS utiliza os métodos VND e Busca Tabu (descritos nas Subseções 3.7 e 3.8, respectivamente) como procedimentos de busca local. A Busca Tabu é acionada somente após *iterMaxVND* iterações sem melhora com o procedimento VND. Para não ficar preso em ótimos locais e se dirigir para outras regiões do espaço de busca, foram utilizadas perturbações descritas na Seção 3.9.

Algoritmo GENILS-TS-CL
Entrada*:* Conjunto de buscas locais, Número máximo de iterações ($iter_{max}$)
Tamanho da lista tabu TamListaTabu (*iterMaxTS*)
Número máximo de iterações da Busca Tabu (*iterMaxTS*)
Número máximo de iterações sem melhora com o VND (*iterMaxVND*)
Saída*:* solução *s*

```
1     número aleatório no intervalo [0; 0,7]
2     s^A  IMB-1R()
3     nrotas  número de rotas da solução s^A
4     s^B  IMB-MR(nrotas, )
5     s^C  VRGENIUS(nrotas)
6     s^A VND(s^A)
7     s^B VND(s^B)
8     s^C VND(s^C)
9     s  argmin{f(s^A), f(s^B), f(s^C)}
10    iter  0
11    enquanto (iter < iter_max) faça
12        iter  iter + 1
13        s'  perturbação(s)
14        se ( iter < iterMaxVND ) faça
15            s"  VND(s')
16        senão
17            s"   TS(s' ; TamListaTabu; iterMaxTS)
18        fim-se
19        se ( f(s") < f(s) ) faça
20            s  s"; iter  0;
21        fim-se
22    fim-enquanto
23    retorne s
```

Algoritmo 1: GENILS-TS-CL

- **Função de Avaliação**

Uma solução é avaliada pela função *f* definida pela Equação (1), a qual deve ser minimizada. 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. 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.

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

Na Eq. (1) é utilizada a seguinte notação:
N: conjunto dos clientes, incluindo o depósito;
E: conjunto das arestas $(i, j) \in E$, 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 $(i, j) \in E$ for utilizada (x_{ij}= 1) e valor zero (x_{ij} = 0), caso contrário;
d_j: valor da demanda do cliente $j \in N$;
p_j: quantidade a ser coletada no cliente $j \in N$;
Q: capacidade máxima do veículo.

- **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, 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 que distante do depósito.

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

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, e são adaptações do algoritmo GENIUS de Gendreau *et al*. (1992). Ambos utilizam como busca local os procedimentos *3-opt* e *4-opt*.

- **Estruturas de Vizinhança**

Para explorar o espaço de soluções foram aplicados sete tipos diferentes de movimentos: *Shift*, *Shift* (2,0), *Swap*, *Swap* (2,1), *Swap* (2,2), *M2-Opt* e *kOr-Opt*. O *Shift* é um movimento de realocação e consiste em transferir um cliente de uma rota para outra. 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 movimento *M2-Opt* consiste em remover dois arcos e inserir dois novos arcos de forma a gerar uma nova rota. Finalmente, o movimento *kOr-Opt* consiste em realocar uma sequência de *k* clientes consecutivos para outra posição na mesma rota, sendo *k* um parâmetro do método; no nosso caso, *k* é fixado no valor 3.

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

Os métodos de busca local *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 do valor da maior carga do veículo nessa rota.

- **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 (CL, do inglês *Candidate List*). Esta Lista permite que apenas movimentos promissores sejam analisados, descartando aqueles considerados desnecessários.

Para reduzir o número de soluções analisadas no espaço de soluções, um movimento somente é aplicado se pelo menos uma aresta gerada por ele for de comprimento menor que o valor *dist* dado pela Eq. (3).

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

Na Eq. (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 clientes existentes.

- **VND**

O VND implementado é uma adaptação do mesmo método descrito em Mine

et al. (2010). Tal como esses autores, a cada iteração a ordem das vizinhanças a serem exploradas pode variar, pois há uma ordenação aleatória das sete vizinhanças descritas na seção 3.4. Entretanto, após cada melhor vizinho escolhido aleatoriamente no passo anterior, é aplicada uma sequência aleatória de procedimentos de otimização, no caso, envolvendo a busca local *Best Improvement* com os sete movimentos descritos na seção 3.4, e os procedimentos *G3-opt*, *G4-opt* e *reverse*, apresentados na seção 3.5. Em Mine *et al.* (2010), tais procedimentos de otimização são aplicados em uma sequência predeterminada. O pseudocódigo do VND implementado é apresentado no Algoritmo 2.

```
Procedimento VND
Entrada: solução s, conjunto de r vizinhanças distintas
Saída: solução refinada s
1     RN  conjunto de t vizinhanças, em uma ordem aleatória
2     k 1
3     enquanto (i  r) faça
4           Seja s0 o melhor vizinho na vizinhança RN(k)(s)
5        se f(s') < f(s) então
6              {Intensificação nas rotas alteradas}
7              Seja BL o conjunto de procedimentos de busca (no caso, 7 buscas locais associadas a cada
                 um dos movimentos descritos na Seção 2.4 e os 3 procedimentos descritos na Seção 2.5), em
                 ordem aleatória;
8              Seja r_BL a cardinalidade do conjunto BL, no caso, 10;
9              s  s'
10             k  1
11             para (j=1 até r_BL ) faça
12                 s' ∈ BL(j)(s)
13                 s  s'
14             fim-para
15        senão
16           k  k + 1
17        fim-se
18    fim-enquanto
19    retorne s
```

Algoritmo 2: VND

- **Busca Tabu**

A Busca Tabu (TS) é uma metaheurística que utiliza uma estrutura de memória para explorar o espaço de soluções. Seu principal mecanismo é a chamada Lista Tabu, que é usada para evitar o retorno a uma solução já gerada anteriormente.

No Algoritmo 3 é apresentado o pseudocódigo da Busca Tabu implementada. Nesta implementação 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, armazenado na variável *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. O procedimento é interrompido quando o número máximo de iterações sem melhora 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. Utiliza-se, também, um critério de aspiração, que permite que um movimento tabu seja realizado, caso a solução produzida seja melhor que a melhor solução encontrada até então. Esta estratégia foi implementada porque a Lista Tabu pode não somente impedir o retorno a uma solução já gerada anteriormente, mas também outras soluções ainda não geradas.

Para representar computacionalmente a Lista Tabu, utilizou-se uma matriz quadrada, em que cada célula (i, j) representa até que iteração está proibido o cliente j ficar como sucessor do cliente i na rota. Esta representação tem a vantagem de requerer complexidade $O(1)$ para consultar se um movimento é tabu ou não. Caso fosse utilizada uma lista encadeada, a consulta seria $O(m)$ no pior caso, sendo m o tamanho da lista.

Procedimento *TS*
Entrada: TamListaTabu, iterMaxTS, Solução s
Saída*:* solução *s*

```
1     s* ← s {Melhor solução obtida até então}
2     TamOriginal ← TamListaTabu {Tamanho original da Lista Tabu}
3     iter ← 0; MelhorIter ← 0
4     enquanto (iter – MelhorIter < iterMaxTS) faça
5
        se (iter – MelhorIter < iterAux) faça
6           TamListaTabu ← TamListaTabu +1
7       fim-se
8       s0 ← Shift(s)
9       s1 ← Swap(s)
10      s2 ← Shift(2,0)(s)
11      s3 ← Swap(2,1)(s)
12      s4 ← Swap(2,2)(s)
13      s ← argmin{ f(s0) , f(s1) , f(s2) , f(s3) , f(s4) }
14      AtualizaListaTabu(TamListaTabu, Lista, Ant, Prox, iter, Δ)
15      se ( f(s) < f(s*) ) faça
16         s* ← s; MelhorIter ← iter
17         TamListaTabu ← TamOriginal
18      fim-se
19      iter ← iter +1
20    fim-enquanto
21    retorne s*
```

Algoritmo 3: Busca Tabu (TS)

- **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 e Roucairol (1996). Inicialmente, seleciona-se um subconjunto de m rotas $R = r_1, r_2, \ldots, r_m$ 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.

- **RESULTADOS COMPUTACIONAIS**

Os algoritmos GENILS-TS e GENILS-TS-CL foram codificados em C++ usando o compilador Visual C++ 2005. Para testá-los, foi usado um microcomputador com processador *Intel Core 2 Quad*, 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 validá-los, 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: $TamListaTabu = 10$, $iterMaxTS = 300$, $iter\_{max} = 10000$, $iterMaxVND = 500$ e $\Delta = 3$.

Na Subseção 4.1, o algoritmo GENILS-TS, de Silva *et al*. (2011), é comparado com sua variante GENILS-TS-CL, que usa uma lista de candidatos na exploração do espaço de soluções. Posteriormente, na Subseção 4.2, o algoritmo GENILS-TS-CL é comparado com os principais algoritmos da literatura.

- **GENILS-TS GENILS-TS-CL**

Apresenta-se, nesta Seção, a comparação entre os algoritmos GENILS-TS e GENILS-TS-CL.

As Tabelas 1, 2 e 3 mostram os resultados desses algoritmos nos problemas-teste de Salhi e Nagy (1999), Montané e Galvão (2006) e Dethloff (2001). 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 em 30 execuções. A coluna $Desv^{Avg}$ apresenta o desvio percentual do valor das soluções médias em relação ao melhor resultado da literatura, calculado pela Equação (4). A coluna $Desv^{Best}$ apresenta o desvio percentual do melhor resultado encontrado pelo algoritmo em relação ao melhor conhecido na literatura, sendo

calculado pela Equação (5). Já a coluna Tempo apresenta o tempo médio, em segundos, das 30 execuções.

$$Desv^{Avg} = \frac{Média - MelhorLit}{MelhorLit} x100 \quad (4)$$

$$Desv^{Best} = \frac{Melhor - MelhorLit}{MelhorLit} x100 \quad (5)$$

Avaliando as Tabelas 1, 2 e 3, observa-se que o GENILS-TS-CL requereu um menor tempo de processamento quando comparado com o GENILS-TS, sem prejuízo da qualidade das soluções geradas. Porém, quando são analisadas as médias das soluções, nota-se que houve uma pequena piora. Acredita-se que este resultado se dá não só pelo fato de o algoritmo ser estocástico, mas também por um menor número de soluções terem sido geradas, já que a Lista de Candidatos limita os movimentos da busca local. Entretanto, mesmo assim, os melhores resultados foram encontrados, mostrando que a restrição adicionada não exclui do espaço de soluções a melhor solução.

Tabela 1 - Comparação GENILS-TS GENILS-TS-CL nos problemas-teste de Salhi e Nagy (1999)

Problema	MelhorLit	GENILS-TS					GENILS-TS-CL				
-	-	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo(s)	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Te
CMT1X	**466,77**	**466,77**	472,23	1,17	0,00	2,25	**466,77**	472,23	1,17	0,00	
CMT1Y	**466,77**	**466,77**	472,23	1,17	0,00	62,20	**466,77**	472,23	1,17	0,00	
CMT2X	**668,77**	684,11	693,40	3,68	2,29	127,97	684,11	694,4	3,83	2,29	
CMT2Y	**663,25**	684,11	693,40	4,55	3,15	132,64	684,11	695,11	4,80	3,15	
CMT3X	**721,27**	**721,27**	726,34	0,70	0,00	265,77	**721,27**	727,06	0,80	0,00	
CMT3Y	**721,27**	**721,27**	730,56	1,29	0,00	256,88	**721,27**	731,32	1,39	0,00	
CMT12X	**644,70**	662,22	666,77	3,42	2,72	269,68	662,22	666,79	3,43	2,72	
CMT12Y	**659,52**	662,22	673,91	2,18	0,41	208,54	662,22	673,91	2,18	0,41	
CMT11X	**833,92**	**833,92**	867,96	4,08	0,00	416,26	**833,92**	867,95	4,08	0,00	
CMT11Y	**830,39**	833,92	856,32	3,12	0,43	488,47	833,92	856,41	3,13	0,43	
CMT4X	**852,46**	**852,46**	865,03	1,47	0,00	858,95	**852,46**	865,28	1,50	0,00	
CMT4Y	**852,35**	855,52	866,06	1,61	0,37	406,01	855,52	866,11	1,61	0,37	
CMT5X	**1029,25**	1030,50	1052,67	2,28	0,12	1655,14	1030,50	1052,71	2,28	0,12	
CMT5Y	**1029,25**	1030,50	1053,73	2,38	0,12	790,88	1030,50	1053,98	2,40	0,12	
Média	-	-	-	**2,36**	**0,69**	**424,40**	-	-	**2,41**	**0,69**	

Tabela 2 - Comparação GENILS-TS GENILS-TS-CL nos problemas-teste de Montané e Galvão (2006)

Problema	MelhorLit	GENILS-TS					GENILS-TS-CL				
-	-	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo(s)	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Te
r101	**1009,95**	**1009,95**	1013,71	0,37	0,00	208,56	**1009,95**	1013,74	0,38	0,00	
r201	**666,20**	**666,20**	666,81	0,09	0,00	165,84	**666,20**	666,88	0,10	0,00	
c101	**1220,18**	**1220,18**	1222,81	0,22	0,00	203,19	**1220,18**	1222,93	0,23	0,00	
c201	**662,07**	**662,07**	664,31	0,34	0,00	91,87	**662,07**	665,16	0,47	0,00	
rc101	**1059,32**	**1059,32**	1064,49	0,49	0,00	93,86	**1059,32**	1064,52	0,49	0,00	
rc201	**672,92**	**672,92**	677,49	0,68	0,00	189,50	**672,92**	677,94	0,75	0,00	
r1_2_1	**3357,64**	**3357,64**	3450,31	2,76	0,00	890,19	**3357,64**	3451,13	2,78	0,00	4
r2_2_1	**1665,58**	**1665,58**	1670,35	0,29	0,00	1038,79	**1665,58**	1672,02	0,39	0,00	9
c1_2_1	**3629,89**	3634,65	3657,16	0,75	0,13	800,47	3634,65	3657,68	0,77	0,13	6
c2_2_1	**1726,59**	**1726,58**	1745,10	1,07	0,00	773,48	**1726,58**	1745,95	1,12	0,00	2
rc1_2_1	**3306,00**	3312,92	3327,22	0,64	0,21	977,57	3312,92	3329,01	0,70	0,21	7
rc2_2_1	**1560,00**	**1560,00**	1584,84	1,59	0,00	914,61	**1560,00**	1584,18	1,55	0,00	9
r1_4_1	**9605,75**	9627,88	9706,91	1,05	0,23	5415,90	9627,88	9706,96	1,05	0,23	2
r2_4_1	**3551,38**	3582,08	3612,11	1,71	0,86	4027,96	3582,08	3612,67	1,73	0,86	1
c1_4_1	**11098,21**	**11098,21**	11133,59	0,32	0,00	3757,66	**11098,21**	11134,64	0,33	0,00	3
c2_4_1	**3546,10**	3592,71	3617,86	2,02	1,31	3293,82	3592,71	3618,91	2,05	1,31	2
rc1_4_1	**9535,46**	**9535,46**	9638,30	1,08	0,00	6004,00	**9535,46**	9638,52	1,08	0,00	5
rc2_4_1	**3403,70**	3419,76	3502,01	2,89	0,47	3600,39	3419,76	3502,68	2,91	0,47	3
Média	-	-	-	**1,02**	**0,18**	**1802,65**	-	-	**1,05**	**0,18**	**1**

Tabela 3 - Comparação GENILS-TS GENILS-TS-CL nos problemas-teste de Dethloff (2001)

Problema	MelhorLit	GENILS-TS					GENILS-TS-CL				
-	-	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	Tempo(s)	Melhor	Média	$Desv^{Avg}$	$Desv^{Best}$	T
SCA3-0	**635,62**	**635,62**	**635,62**	0,00	0,00	10,32	**635,62**	**635,62**	0,00	0,00	
SCA3-1	**697,84**	**697,84**	**697,84**	0,00	0,00	13,30	**697,84**	**697,84**	0,00	0,00	
SCA3-2	**659,34**	**659,34**	**659,34**	0,00	0,00	9,31	**659,34**	**659,34**	0,00	0,00	
SCA3-3	**680,04**	**680,04**	**680,04**	0,00	0,00	10,29	**680,04**	**680,04**	0,00	0,00	
SCA3-4	**690,50**	**690,50**	**690,50**	0,00	0,00	12,23	**690,50**	**690,50**	0,00	0,00	
SCA3-5	**659,90**	**659,90**	**659,90**	0,00	0,00	15,21	**659,90**	**659,90**	0,00	0,00	
SCA3-6	**651,09**	**651,09**	**651,09**	0,00	0,00	13,41	**651,09**	**651,09**	0,00	0,00	
SCA3-7	**659,17**	**659,17**	**659,17**	0,00	0,00	10,25	**659,17**	**659,17**	0,00	0,00	
SCA3-8	**719,48**	**719,48**	**719,48**	0,00	0,00	21,17	**719,48**	**719,48**	0,00	0,00	
SCA3-9	**681,00**	**681,00**	**681,00**	0,00	0,00	16,17	**681,00**	**681,00**	0,00	0,00	
SCA8-0	**961,50**	**961,50**	**961,50**	0,00	0,00	19,65	**961,50**	**961,50**	0,00	0,00	
SCA8-1	**1049,65**	**1049,65**	**1049,65**	0,00	0,00	11,81	**1049,65**	**1049,65**	0,00	0,00	
SCA8-2	**1039,64**	**1039,64**	**1039,64**	0,00	0,00	9,27	**1039,64**	**1039,64**	0,00	0,00	
SCA8-3	**983,34**	**983,34**	**983,34**	0,00	0,00	15,17	**983,34**	**983,34**	0,00	0,00	
SCA8-4	**1065,49**	**1065,49**	**1065,49**	0,00	0,00	10,26	**1065,49**	**1065,49**	0,00	0,00	
SCA8-5	**1027,08**	**1027,08**	**1027,08**	0,00	0,00	11,21	**1027,08**	**1027,08**	0,00	0,00	
SCA8-6	**971,82**	**971,82**	**971,82**	0,00	0,00	18,13	**971,82**	**971,82**	0,00	0,00	
SCA8-7	**1051,28**	**1051,28**	**1051,28**	0,00	0,00	12,26	**1051,28**	**1051,28**	0,00	0,00	
SCA8-8	**1071,18**	**1071,18**	**1071,18**	0,00	0,00	10,07	**1071,18**	**1071,18**	0,00	0,00	
SCA8-9	**1060,50**	**1060,50**	**1060,50**	0,00	0,00	10,55	**1060,50**	**1060,50**	0,00	0,00	
CON3-0	**616,52**	**616,52**	**616,52**	0,00	0,00	11,49	**616,52**	**616,52**	0,00	0,00	
CON3-1	**554,47**	**554,47**	**554,47**	0,00	0,00	10,36	**554,47**	**554,47**	0,00	0,00	
CON3-2	**518,00**	**518,00**	**518,00**	0,00	0,00	15,11	**518,00**	**518,00**	0,00	0,00	
CON3-3	**591,19**	**591,19**	**591,19**	0,00	0,00	20,30	**591,19**	**591,19**	0,00	0,00	
CON3-4	**588,79**	**588,79**	**588,79**	0,00	0,00	15,62	**588,79**	**588,79**	0,00	0,00	
CON3-5	**563,70**	**563,70**	**563,70**	0,00	0,00	21,70	**563,70**	**563,70**	0,00	0,00	
CON3-6	**499,05**	**499,05**	**499,05**	0,00	0,00	14,39	**499,05**	**499,05**	0,00	0,00	
CON3-7	**576,48**	**576,48**	**576,48**	0,00	0,00	12,51	**576,48**	**576,48**	0,00	0,00	
CON3-8	**523,05**	**523,05**	**523,05**	0,00	0,00	11,10	**523,05**	**523,05**	0,00	0,00	
CON3-9	**578,25**	**578,25**	**578,25**	0,00	0,00	21,25	**578,25**	**578,25**	0,00	0,00	
CON8-0	**857,17**	**857,17**	**857,17**	0,00	0,00	10,43	**857,17**	**857,17**	0,00	0,00	
CON8-1	**740,85**	**740,85**	**740,85**	0,00	0,00	13,60	**740,85**	**740,85**	0,00	0,00	
CON8-2	**712,89**	**712,89**	**712,89**	0,00	0,00	10,05	**712,89**	**712,89**	0,00	0,00	
CON8-3	**811,07**	**811,07**	**811,07**	0,00	0,00	14,24	**811,07**	**811,07**	0,00	0,00	
CON8-4	**772,25**	**772,25**	**772,25**	0,00	0,00	8,17	**772,25**	**772,25**	0,00	0,00	
CON8-5	**754,88**	**754,88**	**754,88**	0,00	0,00	11,49	**754,88**	**754,88**	0,00	0,00	
CON8-6	**678,92**	**678,92**	**678,92**	0,00	0,00	17,08	**678,92**	**678,92**	0,00	0,00	
CON8-7	**811,96**	**811,96**	**811,96**	0,00	0,00	10,03	**811,96**	**811,96**	0,00	0,00	
CON8-8	**767,53**	**767,53**	**767,53**	0,00	0,00	19,36	**767,53**	**767,53**	0,00	0,00	
CON8-9	**809,00**	**809,00**	**809,00**	0,00	0,00	16,25	**809,00**	**809,00**	0,00	0,00	
Média	-	-	-	**0,00**	**0,00**	**13,61**	-	-	**0,00**	**0,00**	

Para comparar a convergência do algoritmo GENILS-TS-CL frente à do GENILS-TS, foi realizada uma análise de probabilidade empírica utilizando o procedimento indicado em Aiex *et al.* (2002). Vários problemas-teste foram considerados e, em todos eles, o comportamento foi semelhante. Para ilustrar essa

análise de convergência foi escolhido o problema-teste CON8-7 de Dethloff (2001). O critério de parada do algoritmo foi alterado para finalizar quando fosse encontrado o valor alvo, no caso, o melhor encontrado na literatura para o respectivo problema-teste. A Figura 2 ilustra a convergência dos dois algoritmos neste problema-teste.

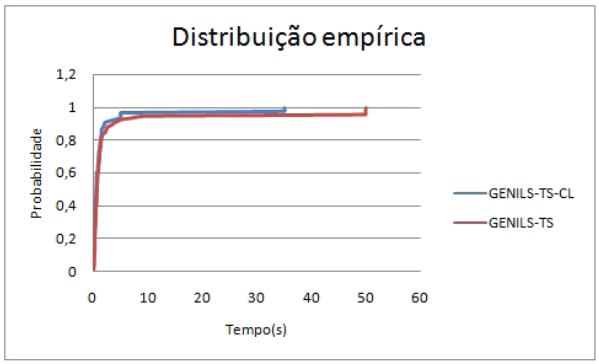

Figura 2- Probabilidade acumulada dos algoritmos GENILS-TS e GENILS-TS-CL no problema-teste CON8-7

Ao analisar as curvas de probabilidades empíricas pode-se perceber um desempenho bastante semelhante nos instantes iniciais. Entretanto, observa-se que o algoritmo GENILS-TS-CL foi o primeiro a alcançar o alvo desejado com uma probabilidade de quase 100% em aproximadamente 35 segundos, enquanto o GENILS-TS consumiu cerca de 50 segundos.

- **GENILS-TS-CL Algoritmos da Literatura**

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

A Tabela 4 apresenta a comparação entre os melhores resultados encontrados pelos algoritmos analisados. Nesta Tabela são apresentados os conjuntos de problemas-teste de Dethloff (2001), Salhi e Nagy (1999) e Montané e Galvão (2006), sendo que a coluna "Número de problemas-teste" apresenta o número de problemas-teste que fazem parte de cada conjunto. As colunas "Média $Desv^{Best}$"e "Nº sol. iguais às melhores Lit." correspondem, respectivamente, à média dos valores $Desv^{Best}$ de cada conjunto de problemas-teste analisados, e a quantidade de soluções encontradas por cada algoritmo iguais aos melhores conhecidos na literatura. O detalhamento dos dados analisados podem ser observados pelas Tabelas 1, 2 e 3.

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 GENILS-TS-CL frente aos algoritmos de Zachariadis *et al.* (2010) e Mine *et al*. (2010) em termos de qualidade da solução, já que em todos os problemas-teste o GENILS-TS-CL nunca encontra um menor número de soluções iguais aos melhores da literatura, além de ter uma variabilidade menor nas demais soluções encontradas em relação à melhor conhecida,

observada pela coluna "Média $Desv^{Best}$". Já o algoritmo ILS-RVND paralelo 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.

Tabela 4 -GENILS-TS-CL melhores algoritmos da Literatura

Conjunto de Problemas-teste	Número de Problemas-teste	Zachariadis et al. (2010)		Subramanian et al. (2010)		Mine et al. (2010)		GENILS-TS	
		Média $Desv^{Best}$	Nº sol. iguais as melhores Lit.	Média $Desv^{Best}$	Nº sol. iguais as melhores Lit.	Média $Desv^{Best}$	Nº sol. iguais as melhores Lit.	Média $Desv^{Best}$	Nº sol. melh
Dethloff (2001)	40	0	40	0	40	0	40	0	
Salhi e Nagy (1999)	14	0,76	4	0,65	8	0,94	4	0,69	
Montané e Galvão (2006)	18	0,31	8	0,01	15	0,19	12	0,18	

• 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, ele é normalmente resolvido por métodos heurísticos. Entre eles, destaca-se o algoritmo GENILS-TS, de Silva *et al.* (2011), que se mostrou superior a outros algoritmos sequenciais da literatura. No presente trabalho, este algoritmo é aperfeiçoado com a inclusão de uma estratégia de lista de candidatos para explorar o espaço de soluções, com o objetivo de evitar a avaliação de soluções não promissoras e, assim, reduzir o tempo de processamento do algoritmo.

Mostrou-se, por meio de uma bateria de testes, que o tempo médio de processamento foi significativamente reduzido quando comparado com a versão sem a lista de candidatos. Além disso, verificou-se que o algoritmo GENILS-TS-CL manteve a capacidade de encontrar as melhores soluções. Por outro lado, a variabilidade das soluções foi pouco maior.

O algoritmo proposto também foi comparado a três outros importantes algoritmos da literatura. O algoritmo de Subramanian *et al.* (2010) foi, claramente, o de melhor desempenho. Entretanto, ele utiliza 256 núcleos de processamento para explorar o espaço de soluções, enquanto os demais usam apenas um. O GENILS-TS-CL, por sua vez, teve o segundo melhor desempenho, em termos de desvio médio e quantidade de melhores soluções, superando, desta forma, os algoritmos sequenciais de Zachariadis *et al.* (2010) e Mine *et al.* (2010).

Como trabalhos futuros sugere-se o desenvolvimento de outras estratégias de Lista de Candidatos, de forma a excluir do espaço de busca um maior de soluções sem prejuízo para a obtenção da solução ótima, assim como paralelizar o algoritmo, aproveitando os núcleos de processamento existentes nas máquinas atuais.

Agradecimentos

Os autores agradecem à Universidade Federal de Ouro Preto, e às agências FAPEMIG, CAPES e CNPq, pelo apoio ao desenvolvimento deste trabalho.

• REFERÊNCIAS BIBLIOGRÁFICAS

AIEX, R. M.; RESENDE, M. G. C.; RIBEIRO, C. C. (2002). Probability distribution of solution time in grasp: An experimental investigation.

Journal of Heuristics, 8(3):343–373, 2002.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

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

GENDREAU, M.; HERTZ, A.; LAPORTE, G. New insertion and post optimization procedures or the traveling salesman problem. *Operations Research*, 40:1086–1094, 1992.

GLOVER, F.; LAGUNA, M. *Tabu Search*. Kluwer Academic Publisher, Boston, 1997.

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

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

MINE, M. T.; SILVA, M. S. A.; OCHI, L. S.; SOUZA, M. J. F.; SILVA, T. C. B. 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, 2010.

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

REGO, C.; ROUCAIROL, C. *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, 1996

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

SILVA, T. C. B.; CRUZ, R. C.; MINE, M. T.; SOUZA, M. J. F.; SANTIBANEZ, E. R. GENILS-TS: um algoritmo heurístico híbrido para resolução do Problema de Roteamento de Veículos com Coleta e Entrega Simultânea. XLIII Simpósio Brasileiro de Pesquisa Operacional, 2011, Ubatuba (SP). Anais do XLIII SBPO. Rio de Janeiro: SOBRAPO. p. 1883-1894, 2011.

SUBRAMANIAN, A. *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, Universidade Federal da Paraíba, João Pessoa, 2008.

SUBRAMANIAN, A.; DRUMMOND, L. M. A.; BENTES, C.; OCHI, L. S.; FARIAS, R. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers and Operations Research*, 37:1899–1911, 2010.

VOUDOURIS, C.; TSANG, E. Partial constraint satisfaction problems and guided local search. *Proceedings for the Second International Conference on the Practical Application of Constraint Technology*

(PACT'96), p. 337–356, 1996.

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

ZACHARIADIS, E. E.; TARANTILIS, C. D.; KIRANOUDIS, C. T. An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research*, 202:401–411, 2010.