A ser publicado no Livro: ***Transporte em Transformação XIV*** *– Edição especial formada pelos artigos vencedores do Prêmio CNT de Produção Acadêmica 2009*

O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETA E ENTREGA SIMULTÂNEA: UMA ABORDAGEM VIA *ITERATED LOCAL SEARCH* E GENIUS

Marcio Tadayuki Mine
Matheus de Souza Alves Silva
Luiz Satoru Ochi
Universidade Federal Fluminense
Instituto de Computação

Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas

RESUMO
Este trabalho considera o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). O PRVCES é um problema básico na área da logística reversa, a qual visa planejar o transporte de produtos aos clientes, bem como o retorno de resíduos ou produtos utilizados por esses para a reciclagem ou depósitos especializados. O PRVCES pertence à classe NP-difícil, uma vez que ele pode ser reduzido ao Problema de Roteamento de Veículos clássico quando nenhum cliente necessita de serviço de coleta. Para resolvê-lo, propõe-se um algoritmo heurístico híbrido, denominado GENILS, baseado nas técnicas *Iterated Local Search*, *Variable Neighborhood Descent* e GENIUS. O algoritmo proposto foi testado em três conjuntos consagrados de problemas-teste da literatura e se mostrou competitivo com os melhores algoritmos existentes.

ABSTRACT
This work deals with the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). The VRPSPD is a common problem in the area of reverse logistics, which aims to plan the transportation of products to customers, as well as the return of leavings or products used by them for recycling or to special depots. The VRPSPD is NP-hard, since it can be reduced to the classical Vehicle Routing Problem when no client needs the pickup service. To solve it, we propose a hybrid heuristic algorithm, called GENILS, based on Iterated Local Search, Variable Neighborhood Descent and GENIUS. The proposed algorithm was tested on three well-known sets of instances found in literature and it was competitive with the best existing approaches.

1. INTRODUÇÃO

O Problema de Roteamento de Veículos (PRV), conhecido na literatura como *Vehicle Routing Problem* (VRP), foi originalmente proposto por Dantzig e Ramser (1959) e pode ser definido da seguinte forma: Dado um conjunto N de clientes, cada qual com uma demanda d_i e uma frota de veículos homogênea com capacidade Q, tem-se como objetivo, estabelecer os trajetos de custo mínimo a serem percorridos pelos veículos, de forma a atender completamente a demanda dos clientes numa única visita.

Em 1989, Min propôs uma importante variante do PRV: o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES), em que os serviços de entrega e coleta devem ser realizados simultaneamente. Este modelo é um problema básico na área da logística reversa, a qual visa planejar o transporte de produtos aos clientes, bem como o retorno de resíduos ou produtos utilizados por esses para a reciclagem ou depósitos especializados. A logística reversa pode ser observada, por exemplo, na logística postal ou no planejamento da distribuição de indústria de bebidas.

O PRVCES pertence à classe de problemas NP-difíceis, uma vez que ele pode ser reduzido ao PRV clássico quando nenhum cliente necessita de serviço de coleta. Dessa forma, diversos trabalhos na literatura o tratam de forma heurística.

Min (1989) propôs um método de três fases para resolver o planejamento de distribuição de materiais para bibliotecas públicas. A primeira fase do método consiste em agrupar os clientes em *clusters* por meio do Método de Ligação por Médias (*Average Linkage Method*) (Anderberg, 2007). A segunda fase associa os veículos às respectivas rotas e a terceira consiste em resolver cada *cluster* por meio de uma heurística para o Problema do Caixeiro Viajante. Essa heurística atribui, iterativamente, uma penalidade aos arcos em que a carga do veículo foi excedida, procurando, dessa forma, gerar uma solução viável.

Halse (1992) aborda o PRVCES por meio de uma heurística que consiste em, primeiramente, associar os clientes aos veículos e, em seguida, gerar as rotas através de um procedimento baseado no método *3-optimal*.

Dethloff (2001) desenvolveu uma adaptação do método da Inserção Mais Barata, em que os clientes são adicionados às rotas seguindo três critérios: (*i*) distância; (*ii*) capacidade residual e (*iii*) distância do cliente ao depósito. Nesse trabalho não foi aplicado nenhum método de refinamento da solução.

Vural (2003) desenvolveu duas versões do Algoritmo Genético (Goldberg, 1989). A primeira faz a codificação dos indivíduos através de chaves aleatórias (*Random Keys*) (Bean, 1994) e a segunda foi implementada como uma heurística de refinamento, baseada na estrutura do AG desenvolvido por Topcuoglu e Sevilmis (2002).

Gökçe (2004) trata o PRVCES com a metaheurística Colônia de Formigas (Dorigo *et al.*, 1996) e utiliza o método *2-optimal* como um procedimento de pós-otimização.

Nagy e Salhi (2005) desenvolveram uma metodologia para a resolução do PRVCES com a restrição de limite de tempo para percorrer cada rota. Essa metodologia reúne diferentes heurísticas para resolver o PRV clássico, tais como, *2-optimal*, *3-optimal*, realocação, troca e, além disso, procedimentos para viabilizar a solução.

Dell'Amico *et al.* (2006) utilizaram a técnica *branch-and-price* por meio de duas abordagens: programação dinâmica e relaxação do espaço de estados (*state space relaxation*).

Crispim e Brandão (2005) propõem uma técnica híbrida, combinando as metaheurísticas Busca Tabu (Glover e Laguna, 1997) e *Variable Neighborhood Descent* – VND (Hansen e Mladenović, 2001). Para gerar uma solução foi utilizado o método da varredura (*sweep method*) e, para refiná-la, um procedimento de busca local que explora o espaço de soluções com movimentos de realocação e troca.

Röpke e Pisinge (2006) desenvolveram uma heurística baseada na *Large Neighborhood Search* – LNS (Shaw, 1998). O LNS é uma busca local baseada em duas idéias para definir e explorar estruturas de vizinhança de alta complexidade. A primeira idéia é fixar uma parte da solução e assim definir o espaço de soluções. A segunda consiste em realizar a busca por meio de programação por restrições, programação inteira, técnicas *branch-and-cut*, entre outras.
Montané e Galvão (2006) utilizaram a metaheurística Busca Tabu considerando quatro tipos de estruturas de vizinhança. Essas estruturas utilizam os movimentos de realocação, troca e *crossover*. Para gerar uma solução vizinha foram desenvolvidas duas estratégias, sendo que uma considera o primeiro movimento viável e a outra, o melhor movimento viável.

Chen (2006) trata o problema por meio de uma técnica baseada nas metaheurísticas *Simulated Annealing* (SA) (Kirkpatrick *et al.*, 1983) e Busca Tabu, enquanto Chen e Wu (2006) desenvolveram uma metodologia baseada na heurística *record-to-record travel* (Dueck, 1993), a qual é uma variação do SA.

Algoritmos construtivos, heurísticas de refinamento e técnicas baseadas na metaheurística Busca Tabu são apresentados em Bianchessi e Righini (2007). Essas técnicas utilizam movimentos de troca de nós (*node-exchange-based*) e troca de arcos (*arc-exchange-based*).

Wassan *et al.* (2007) propõem uma versão reativa da metaheurística Busca Tabu. Para gerar uma solução inicial, foi utilizado o método da varredura (*sweep method*) e para explorar o espaço de soluções, movimentos de realocação, de troca e de inversão do sentido da rota.

Em Subramanian *et al.* (2008) foi desenvolvido um algoritmo baseado em *Iterated Local Search* (ILS), tendo como busca local o procedimento *Variable Neighborhood Descent* (VND). Para gerar uma solução inicial foi utilizada uma adaptação da heurística de inserção de Dethloff (2001), porém sem considerar a capacidade residual do veículo. O VND explora o espaço de soluções usando movimentos baseados em realocação, troca e *crossover*. O VND realiza, a cada melhora na solução corrente, uma intensificação nas rotas alteradas, por meio dos procedimentos de busca local *Or-opt*, *2-opt*, *exchange* e *reverse*. O procedimento *Or-opt* que foi implementado consiste em permutar um, dois ou três clientes consecutivos em uma rota. O *2-opt* e o *exchange* realizam a permutação de um par de arcos e dois clientes, respectivamente. O movimento *reverse* consiste em inverter o sentido da rota, caso haja redução na carga do veículo nos arcos. Os mecanismos de perturbação aplicados no ILS foram o *ejection chain*, o *double swap* e o *double bridge*. O *ejection chain* consiste em transferir um cliente de cada rota a outra adjacente. O *double swap* consiste em realizar duas trocas sucessivas e o *double bridge* consiste em remover quatro arcos e inserir quatro novos arcos. Uma descrição detalhada desse algoritmo, bem como uma nova formulação de programação matemática para o PRVCES pode ser encontrada em Subramanian (2008).

Zachariadis *et al.* (2009) abordada o PRVCES com uma técnica híbrida, combinando as metaheurísticas Busca Tabu e *Guided Local Search* (Voudouris e Tsang, 1996).

Para comparar as abordagens da literatura, Dethloff (2001) propôs um conjunto com 40 problemas envolvendo 50 clientes cada. Salhi e Nagy (1999) apresentaram 28 problemas-teste com 50 a 199 clientes, sendo que a metade desses têm restrições de limite de tempo. Por fim, Montané e Galvão (2006) adaptaram 18 problemas-teste de Solomon *et al.* (2005) e Gehring e Homberger (1999), envolvendo 100, 200 e 400 clientes.

Até o momento, os melhores resultados encontrados na literatura para esses problemas-teste pertencem a:

- Chen e Wu (2006): um problema-teste de Salhi e Nagy (1999);
- Röpke e Pisinger (2006): 26 problemas-teste de Dethloff (2001);
- Wassan *et al.* (2007): 6 problemas-teste de Salhi e Nagy (1999);
- Zachariadis *et al.* (2009): 6 problemas de Salhi e Nagy (1999) e 27 de Dethloff (2001);
- Subramanian (2008) e Subramanian *et al.* (2008): todos os problemas-teste de Dethloff (2001) e Montané e Galvão (2006) e 17 de Salhi e Nagy (1999).

Neste trabalho é apresentado um novo algoritmo heurístico para resolver o PRVCES. O algoritmo proposto, denominado GENILS, combina as técnicas *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND) e uma adaptação da heurística GENIUS. A heurística GENIUS foi proposta por Gendreau *et al.* (1992) para resolver o Problema do Caixeiro Viajante e possui duas fases: uma construtiva (GENI – *Generalized Insertion*) e outra de refinamento (US – *Unstringing and Stringing*). O GENILS difere do ILS de Subramanian *et al*. (2008) basicamente por incluir a heurística GENIUS e os procedimentos 3-*optimal* e 4-*optimal* na exploração do espaço de busca. Conforme mostram os resultados, estas estratégias se mostraram eficientes na resolução do problema.

O restante deste trabalho está organizado como segue. Na Seção 2 descreve-se o problema abordado. A metodologia proposta é apresentada na Seção 3. Na Seção 4 são apresentados e analisados os resultados encontrados, enquanto na Seção 5 conclui-se o trabalho, apontando trabalhos futuros.

2. DESCRIÇÃO DO PROBLEMA

O Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES), ou *Vehicle Routing Problem with Simultaneous Pickup and Delivery* (VRPSPD), é uma variante do PRV clássico. Neste problema existe um depósito com uma frota ilimitada de veículos de capacidade Q e um conjunto N de clientes espalhados geograficamente. Cada cliente $i \in N$ está associado a duas quantidades d_i e p_i, que representam a demanda por um determinado produto e a coleta no cliente i, respectivamente. O objetivo do problema é definir as rotas necessárias para atender a todos os clientes, de forma a minimizar os custos referentes ao deslocamento dos veículos e satisfazer as seguintes restrições: (a) cada rota deve iniciar e finalizar no depósito; (b) todos os clientes devem ser visitados uma única vez e por um único veículo; (c) as demandas por coleta e entrega de cada cliente devem ser completamente atendidas; (d) a carga do veículo, em qualquer momento, não pode superar a capacidade do veículo. Em algumas variantes desse problema, considera-se também a necessidade de cada veículo não percorrer mais que um determinado limite de distância (tempo). A Figura 1 ilustra um exemplo do PRVCES.

Na Figura 1, os clientes são representados pelos números 1 a $|N|$ e o depósito pelo número 0 (zero). Cada par $[d_i / p_i]$ denota a demanda e coleta em um cliente i, respectivamente. Nesta figura, há três rotas a serem executadas por veículos de capacidade $Q = 150$. Em uma delas, o veículo sai do depósito e atende aos clientes 10, 8, 19, 9 e 2, retornando ao depósito no final. No primeiro cliente atendido nessa rota, é feita uma entrega de 10 unidades do produto e recolhida outras 5 unidades. A última visita do veículo ocorre no cliente 2, o qual demanda 13 unidades do produto e necessita que sejam coletadas 30 unidades.

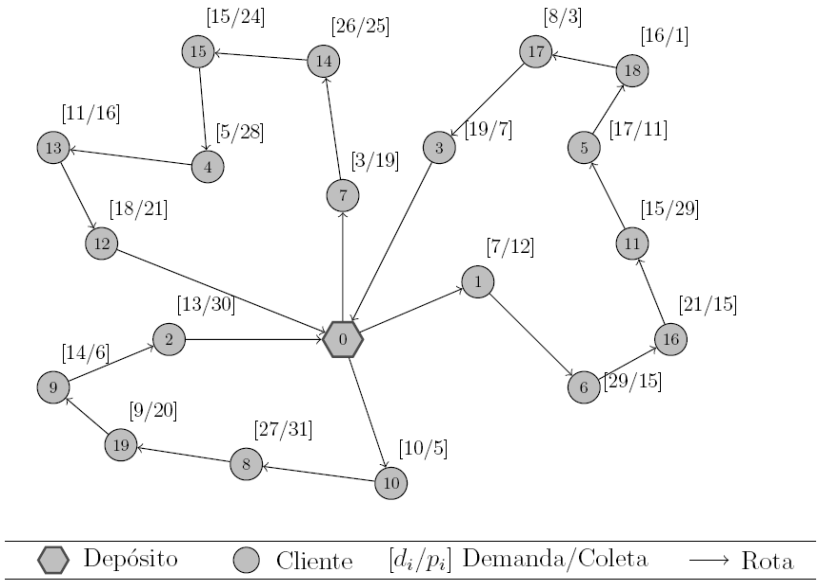

Figura 1: Exemplo do PRVCES

3. METODOLOGIA

Apresenta-se, nesta Seção, a metodologia desenvolvida para resolver o PRVCES. Na Subseção 3.1 mostra-se como gerar uma solução inicial. Na Subseção 3.2 são mostrados os movimentos usados para explorar o espaço de busca, enquanto em 3.3 é mostrado como uma solução é avaliada. Na Subseção 3.4 é apresentado o algoritmo GENILS desenvolvido para resolver o problema.

3.1. Geração de uma solução inicial

Para gerar uma solução inicial são utilizadas três heurísticas baseadas em inserção. A primeira, denominada IMB-1R, é uma adaptação da heurística de Inserção Mais Barata, a qual constrói uma solução rota a rota. A segunda, denominada IMB-NR, foi proposta por Subramanian *et al*. (2008) e baseia-se na heurística de inserção de Dethloff (2001). A última, VRGENIUS, proposta neste trabalho, é uma adaptação da heurística GENIUS (Gendreau *et al.*, 1992), desenvolvida para o Problema do Caixeiro Viajante, e possui duas fases: uma construtiva (VRGENI) e outra de refinamento (VRUS). A fase VRGENI é um método de inserção, cuja característica fundamental é que a inclusão de um cliente não é realizada necessariamente entre dois outros clientes consecutivos. No entanto, esses dois clientes tornam-se adjacentes após a inserção. Já a fase VRUS consiste em, a cada iteração, remover um cliente da solução e reinseri-lo em outra posição que vise melhorar a solução corrente. Caso essa melhora não seja possível com nenhum cliente, esta fase é finalizada. Vale ressaltar que, em ambas as fases, tanto a remoção, quanto a inserção de um cliente é realizada por procedimentos que analisam um espaço reduzido da vizinhança explorada pelas buscas locais *3-optimal* e *4-optimal*. A eficiência desses procedimentos encontra-se no fato de que o espaço analisado é restrito ao número de vizinhos de cada cliente, sendo, no máximo, igual a um parâmetro *p*.

3.2. Estruturas de vizinhança

Para explorar o espaço de soluções do problema, aplicam-se, neste trabalho, sete tipos diferentes de movimentos, a saber: (a) *Shift*: movimento de realocação que consiste em transferir um cliente de uma rota para outra; (b) *Shift*(2,0): movimento semelhante ao *Shift*,

porém realocando dois clientes consecutivos de uma rota para outra; (c) *Swap*: consiste em trocar um cliente i de uma rota r_1 com um outro cliente j de uma rota r_2; (d) *Swap*(2,1): é análogo ao *Swap*, porém trocando dois clientes consecutivos de uma rota com um cliente de outra rota; (e) *Swap*(2,2): consiste em realizar a troca de dois clientes consecutivos de um rota com dois outros clientes consecutivos de outra rota; (f) *M2-Opt*: consiste em remover dois arcos e inserir dois novos arcos; (g) *kOr-Opt*: consiste em remover k clientes consecutivos de uma rota r e, em seguida, reinseri-los em uma outra posição nessa mesma rota. O valor de k é um parâmetro. Esse movimento é uma generalização do *Or-Opt* proposto por Or (1976), em que é realizada a remoção de no máximo três clientes consecutivos. Destaca-se que não são permitidos movimentos que conduzam a soluções inviáveis.

3.3. Função de avaliação

Uma solução s é avaliada pela função f apresentada pela Equação (1), que determina o custo total de deslocamento.

$$f(s) = \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

em que N: conjunto dos clientes, incluindo o depósito;
A: conjunto dos arcos (i, j), com $i, j \in N$;
c_{ij}: custo de deslocamento ou distância de um cliente i a um cliente j;
x_{ij}: assume valor 1 ($x_{ij} = 1$) se o arco $(i,j) \in A$ estiver na solução ou assume valor 0 ($x_{ij} = 0$), caso contrário.

3.4. Algoritmo GENILS

Para resolver o PRVCES propõe-se um algoritmo híbrido, denominado GENILS. Este algoritmo utiliza o método de geração de solução inicial descrito na Subseção 3.2, e combina os procedimentos heurísticos *Iterated Local Search* – ILS (Stützle e Hoos, 1999), *Variable Neighborhood Descent* – VND (Hansen e Mladenović, 2001) e uma adaptação da heurística GENIUS (Gendreau *et al.*, 1992). Seu pseudocódigo é apresentado na Figura 1.

```
Algoritmo GENILS
    s^A ← construa uma solução com a IMB-1R
    s^B ← construa uma solução com a IMB-NR
    s^C ← construa uma solução com a VRGENIUS
    s^A ←VND(s^A)
    s^B ←VND(s^B)
    s^C ←VND(s^C)
    s ← s' | f(s') = min{f(s^A), f(s^B), f(s^C)}
    iter ← 0
    enquanto (iter < iter_max)
        iter ← iter + 1
        s' ← perturbação(s)
        s" ← VND(s)
        se ( f(s") < f(s) ) faça
            s ← s"
            iter ← 0
        fim-se
    fim-enquanto
    retorne s
```

Figura 1: GENILS

O algoritmo GENILS inicia gerando três soluções iniciais s^A, s^B, e s^C, cada qual por um dos métodos descritos na Subseção 3.1. Essas soluções são, a seguir, refinadas pelo VND e a melhor solução obtida é usada como a solução inicial *s*. Para escapar do ótimo local *s*, é feita uma perturbação, gerando uma nova solução *s*'. Em seguida, essa solução perturbada é refinada pela busca local VND, obtendo-se um novo ótimo local *s*". Esta solução torna-se a nova solução corrente caso *s*" seja melhor que *s*; caso contrário, ela é descartada e nova perturbação é feita a partir da solução *s*. Esse procedimento é repetido até o número máximo de iterações sem melhora na solução corrente ($iter_{\max}$) seja atingido.

As perturbações são realizadas por um dos três mecanismos descritos a seguir, escolhidos aleatoriamente:

- Múltiplos *Shift*: Consiste em realizar *k* movimentos *Shift* (descrito na Subseção 3.2) sucessivamente. O valor de *k* é definido aleatoriamente entre 1, 2 ou 3;
- Múltiplos *Swap*: Segue a mesma idéia da perturbação com múltiplos *Shift*, porém utilizando movimentos *Swap*;
- *Ejection chain*: Essa perturbação foi proposta por Rego e Roucairol (1996). Inicialmente, seleciona-se um subconjunto de *m* rotas $R = \{r_1, r_2, ..., r_m\}$ 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.

O VND explora o espaço de soluções por meio dos movimentos descritos na Subseção 3.2. Esse método possui duas estratégias: a primeira consiste em determinar aleatoriamente a ordem das vizinhanças a serem pesquisadas, enquanto que a segunda é a intensificação da busca nas rotas modificadas em cada iteração do método. Essa intensificação é realizada por meio de buscas locais baseadas nos movimentos *Shift*, *Shift*(2,0), *Swap*, *2-Opt*, *Swap*(2,1), *Swap*(2,2), *kOr-Opt* com $k = 3, 4, 5$, apresentados na Subseção 3.2. Além dessas buscas locais, a intensificação é realizada por dois outros procedimentos inspirados na heurística GENIUS, denominados *G3-Opt* e *G4-Opt*, que representam adaptações das buscas locais 3-*optimal* e 4-*optimal*. A adaptação consiste em analisar parcialmente o espaço de soluções, limitando o número de vizinhos de cada cliente a serem explorados. Além disso, a inserção de um arco $(v_i; v_j)$ é realizada somente se os clientes v_i e v_j estiverem relativamente próximos. Para isso, define-se $N_p(v)$ como o conjunto dos *p* vizinhos mais próximos ao cliente *v* em uma rota *r* da solução *s*, sendo *p* um parâmetro. Além disso, considere as seguintes definições: N^r, conjunto dos clientes pertencentes à rota *r*; v_i: cliente $v_i \in N^r$; v_{h+1}; v_{h-1}: clientes, pertencentes à rota *r*, sucessor e antecessor ao cliente $v_h \in N^r$, respectivamente; v_j : cliente $v_j \in N_p(v_i)$; v_k: cliente $v_k \in N_p(v_{i+1})$ no caminho de v_j para v_i; v_l: cliente $v_l \in Np(v_{j+1})$ no caminho de v_i para v_j. O procedimento *G3-Opt* funciona da seguinte forma: a cada passo, é feita a remoção dos arcos $(v_i; v_{i+1})$, $(v_j; v_{j+1})$ e $(v_k; v_{k+1})$ e a inserção dos arcos $(v_i; v_j)$, $(v_{i+1}; v_k)$ e $(v_{j+1}; v_{k+1})$ na rota *r*, de forma a melhorar a solução *s* e tal que o custo seja o menor possível. Ressalta-se que ambos os sentidos da rota *r* são analisados. Este procedimento é repetido até que não seja possível melhorar a solução *s*. Já o procedimento *G4-Opt* é semelhante ao *G3-Opt*, com a diferença de que, a cada iteração, são removidos os arcos $(v_i; v_{i+1})$, $(v_{l-1}; v_l)$, $(v_j ; v_{j+1})$ e $(v_{k-1}; v_k)$ e adicionados os arcos $(v_i; v_j)$, $(v_l; v_{j+1})$, $(v_{k-1}; v_{l-1})$ e $(v_{i+1}; v_k)$. Por fim, é realizado o movimento *Reverse* nas rotas modificadas, que consiste em inverter o sentido da rota, caso haja redução na carga máxima.

4. RESULTADOS COMPUTACIONAIS

Apresentam-se, nesta Seção, os resultados computacionais obtidos pelo algoritmo heurístico híbrido proposto para resolver o PRVCES. O sistema foi desenvolvido na linguagem C++, utilizando o ambiente Microsoft Visual C++, versão 2005. Foi utilizado um computador Intel Core 2 Duo com 1,66 GHz e 2 GB de memória RAM e sistema operacional Windows Vista Home Premium de 32 bits.

Para validar o algoritmo, foram utilizados três conjuntos de problemas-teste apresentados na Seção 1, a saber, os de Salhi e Nagy (1999), Dethloff (2001) e Montané e Galvão (2006). No conjunto de Salhi e Nagy (1999), não foram tratados os problemas que possuem restrições de limite de tempo. O número máximo de iterações do GENILS utilizado foi 10.000.

As Tabelas 1, 2 e 3 comparam o desempenho do GENILS com diferentes algoritmos da literatura. Nessas tabelas, a coluna *Problema* indica o problema-teste considerado, *Melhor* é o melhor valor encontrado pelo algoritmo do respectivo autor e *Tempo* é o tempo, em segundos, de processamento do algoritmo. Na coluna *Gap*, mostra-se o desvio percentual das soluções médias do GENILS em relação aos melhores resultados existentes. O *Gap* é calculado pela expressão $Gap = 100 \times (Média - MelhorValor) / MelhorValor$.

Em relação aos problemas-teste propostos por Dethloff (2001), o GENILS foi capaz de alcançar todas as melhores soluções da literatura. Dos 14 problemas-teste de Salhi e Nagy (1999), o algoritmo proposto encontrou quatro melhores soluções da literatura, enquanto que nos demais problemas, o *gap* máximo foi de 3,16%. É importante ressaltar que neste conjunto nenhum algoritmo vence na maioria das instâncias, ou seja, não existe uma dominância clara de nenhum método. O melhor desempenho do GENILS se deu nos problemas-teste de Montané e Galvão (2006), em que, dos 18 problemas desse conjunto, em 9 foram geradas novas melhores soluções, em 6 foram encontrados os melhores resultados da literatura e nos 3 restantes, o *gap* foi inferior a 0,58%.

Tabela 1: Resultados obtidos pelo GENILS nos problemas-teste de Salhi e Nagy (1999)

Problema	Wassan *et al.*		Zachariadis *et al.*		Subramanian *et al.*		GENILS		
	Melhor	Tempo[(1)] (s)	Melhor	Tempo[(2)] (s)	Melhor	Tempo[(3)] (s)	Melhor	Tempo[(4)] (s)	*Gap* (%)
CMT1X	468,30	48	469,80	2,89	466,77	1,10	466,77	7,82	0,00
CMT1Y	458,96	69	469,80	3,85	466,77	1,08	466,77	7,61	1,68
CMT2X	668,77	94	684,21	7,42	684,21	6,99	684,21	17,62	2,31
CMT2Y	663,25	102	684,21	8,02	684,21	5,84	684,21	20,10	3,16
CMX3X	729,63	294	721,27	11,62	721,40	6,80	721,40	59,61	0,02
CMT3Y	745,46	285	721,27	13,53	721,40	7,37	721,27	58,72	0,00
CMT12X	644,70	242	662,22	11,80	662,22	8,02	662,22	22,89	2,72
CMT12Y	659,52	254	662,22	7,59	662,22	7,32	663,50	22,33	0,60
CMT11X	861,97	504	838,66	17,78	839,39	12,58	846,23	48,85	0,90
CMT11Y	830,39	325	837,08	14,26	841,88	14,80	836,04	287,30	0,68
CMT4X	876,50	558	852,46	27,75	852,46	50,72	852,46	134,26	0,00
CMT4Y	870,44	405	852,46[a]	31,20	852,46	46,06	862,28	266,76	1,17[b]
CMT5X	1044,51	483	1030,55	51,67	1030,55	53,51	1033,51	768,94	0,29
CMT5Y	1054,46	533	1030,55	58,81	1031,17	58,74	1036,14	398,75	0,54

(1) Tempo de CPU em um computador Sun-Fire-V440 com um processador UltraSPARC-IIIi 1062 MHz.;
(2) Tempo de CPU em um computador Pentium IV 2,4 GHz.
(3) Tempo de CPU em um computador Intel Core 2 Quad 2,5 GHz.;
(4) Tempo de CPU em um computador Intel Core 2 Duo 1,6 GHz.
[a] Um resultado melhor de valor 852,35 foi obtido por Chen e Wu (2006).
[b] *Gap* em relação ao valor encontrado por Chen e Wu (2006).

Tabela 2: Resultados obtidos pelo GENILS nos problemas-teste de Dethloff (2001)

Problema	Röpke e Pisinger		Zachariadis *et al.*		Subramanian *et al.*		GENILS		
	Melhor	*Tempo*[1] (s)	Melhor	Tempo[2] (s)	Melhor	Tempo[3] (s)	Melhor	Tempo[4] (s)	*Gap* (%)
SCA3-0	636,10	232,00	636,06	2,83	635,62	0,90	635,62	6,77	0,00
SCA3-1	697,80	170,00	697,84	2,12	697,84	1,12	697,84	8,49	0,00
SCA3-2	659,30	160,00	659,34	2,58	659,34	1,19	659,34	8,13	0,00
SCA3-3	680,60	182,00	680,04	3,13	680,04	1,13	680,04	8,45	0,00
SCA3-4	690,50	160,00	690,50	2,68	690,50	1,32	690,50	8,09	0,00
SCA3-5	659,90	178,00	659,90	2,56	659,90	1,17	659,90	8,19	0,00
SCA3-6	651,10	171,00	651,09	4,40	651,09	1,23	651,09	8,21	0,00
SCA3-7	666,10	162,00	659,17	2,98	659,17	1,69	659,17	6,76	0,00
SCA3-8	719,50	157,00	719,47	3,98	719,47	1,08	719,48	8,85	0,00
SCA3-9	681,00	167,00	681,00	3,86	681,00	1,03	681,00	8,63	0,00
SCA8-0	975,10	98,00	961,50	3,21	961,50	2,52	961,50	5,65	0,00
SCA8-1	1052,40	95,00	1050,20	3,55	1049,65	2,98	1049,65	5,67	0,00
SCA8-2	1039,60	83,00	1039,64	4,67	1039,64	3,42	1039,64	5,92	0,00
SCA8-3	991,10	94,00	983,34	3,29	983,34	3,44	983,34	4,58	0,00
SCA8-4	1065,50	84,00	1065,49	2,68	1065,49	2,74	1065,49	5,98	0,00
SCA8-5	1027,10	96,00	1027,08	4,50	1027,08	3,44	1027,08	6,62	0,00
SCA8-6	972,50	93,00	971,82	2,67	971,82	2,48	971,82	6,57	0,00
SCA8-7	1061,00	92,00	1052,17	4,32	1051,28	5,39	1051,28	5,56	0,00
SCA8-8	1071,20	85,00	1071,18	3,43	1071,18	2,05	1071,18	5,57	0,00
SCA8-9	1060,50	86,00	1060,50	4,12	1060,50	3,10	1060,50	5,62	0,00
CON3-0	616,50	171,00	616,52	3,89	616,52	2,02	616,52	6,77	0,00
CON3-1	554,50	190,00	554,47	2,97	554,47	1,83	554,47	7,76	0,00
CON3-2	521,40	176,00	519,26	3,32	518,00	2,10	518,01	9,28	0,00
CON3-3	591,20	177,00	591,19	2,78	591,19	1,34	591,19	9,18	0,00
CON3-4	588,80	173,00	589,32	3,12	588,79	1,79	588,79	6,29	0,00
CON3-5	563,70	179,00	563,70	3,45	563,70	1,71	563,70	9,16	0,00
CON3-6	499,10	195,00	500,80	2,98	499,05	1,93	499,05	7,33	0,00
CON3-7	576,50	226,00	576,48	2,40	576,48	1,52	576,48	6,96	0,00
CON3-8	523,10	174,00	523,05	5,02	523,05	1,51	523,05	8,75	0,00
CON3-9	578,20	163,00	580,05	3,14	578,24	1,58	578,25	6,87	0,00
CON8-0	857,20	86,00	857,17	3,40	857,17	3,74	857,17	6,36	0,00
CON8-1	740,90	81,00	740,85	3,73	740,85	2,82	740,85	4,88	0,00
CON8-2	716,00	84,00	713,14	2,87	712,89	2,46	712,89	6,95	0,00
CON8-3	811,10	91,00	811,07	3,82	811,07	2,82	811,07	5,87	0,00
CON8-4	772,30	87,00	772,25	2,98	772,25	3,37	772,25	5,01	0,00
CON8-5	755,70	94,00	756,91	5,76	754,88	3,30	754,88	5,82	0,00
CON8-6	693,10	96,00	678,92	4,00	678,92	3,04	678,92	5,67	0,00
CON8-7	814,80	94,00	811,96	2,46	811,96	2,73	811,96	4,71	0,00
CON8-8	774,00	94,00	767,53	4,21	767,53	3,42	767,53	5,23	0,00
CON8-9	809,30	92,00	809,00	3,87	809,00	3,60	809,00	5,86	0,00

[1] Tempo de CPU em um computador Pentium IV 1.5 GHz.
[2] Tempo de CPU em um computador Pentium IV 2.4 GHz.
[3] Tempo de CPU em um computador Intel Core 2 Quad 2.5 GHz.
[4] Tempo de CPU em um computador Intel Core 2 Duo 1,6 GHz.

Comparando o GENILS com outros algoritmos, verifica-se que o mesmo tem desempenho bastante próximo ao de Subramanian *et al.* (2008). De fato, tanto nos problemas-teste de Dethloff (2001) quanto nos de Montané e Galvão (2006), são esses os únicos algoritmos que têm todos os melhores resultados da literatura. Nesse segundo conjunto de problemas-teste, o GENILS foi superior ao de Subramanian *et al.* (2008) em 9 problemas e inferior em 3. Já no conjunto de problemas de Salhi e Nagy (1999), o GENILS superou o algoritmo de

Subramanian *et al*. (2008) em 2 casos e teve desempenho pior em 5.

Tabela 3: Resultados obtidos pelo GENILS nos problemas-teste de Montané e Galvão (2006)

Problema	Montané e Galvão		Zachariadis *et al.*		Subramanian *et al.*		GENILS		
	Melhor	Tempo[1] (s)	Melhor	Tempo[2] (s)	Melhor	Tempo[3] (s)	Melhor	Tempo[4] (s)	*Gap* (%)
r101	1042,62	12,20	1019,48	10,50	1010,90	10,51	1009,95	35,65	-0,09
r201	671,03	12,02	666,20	8,70	666,20	6,24	666,20	39,62	0,00
c101	1259,79	12,07	1220,99	10,20	1220,26	12,73	1220,18	18,34	-0,01
c201	666,01	12,40	662,07	5,70	662,07	4,18	662,07	16,62	0,00
rc101	1094,15	12,30	1059,32	12,90	1059,32	9,48	1059,32	12,79	0,00
rc201	674,46	12,07	672,92	10,50	672,92	4,21	672,92	24,03	0,00
r1_2_1	3447,20	55,56	3393,31	61,80	3371,29	95,79	3357,64	175,81	-0,40
r2_2_1	1690,67	50,95	1673,65	47,40	1665,58	24,13	1665,58	103,44	0,00
c1_2_1	3792,62	52,21	3652,76	66,30	3640,20	95,17	3636,74	117,62	-0,10
c2_2_1	1767,58	65,79	1753,68	60,90	1728,14	41,94	1726,59	127,81	-0,09
rc1_2_1	3427,19	58,39	3341,25	45,30	3327,98	76,30	3312,92	299,30	-0,45
rc2_2_1	1645,94	52,93	1562,34	62,40	1560,00	34,28	1560,00	77,48	0,00
r1_4_1	10027,81	330,42	9758,77	315,30	9695,77	546,39	9627,43	2928,31	-0,71
r2_4_1	3685,26	324,44	3606,72	273,60	3574,86	231,73	3582,08	768,60	0,20
c1_4_1	11676,27	287,12	11207,37	283,50	11124,30	524,35	11098,21	1510,44	-0,23
c2_4_1	3732,00	330,20	3630,72	336,00	3575,63	293,18	3596,37	569,01	0,58
rc1_4_1	9883,31	286,66	9697,65	145,80	9602,53	550,90	9535,46	2244,18	-0,70
rc2_4_1	3603,53	328,16	3498,30	345,00	3416,61	291,15	3422,11	3306,84	0,16

[1] Tempo de CPU em um computador Athlon XP 2.0 GHz.
[2] Tempo de CPU em um computador Pentium IV 2.4 GHz.
[3] Tempo de CPU em um computador Intel Core 2 Quad 2.5 GHz.
[4] Tempo de CPU em um computador Intel Core 2 Duo 1,6 GHz.

Uma comparação em termos de tempos computacionais não foi feita porque os resultados dos outros algoritmos da literatura foram obtidos em máquinas distintas.

5. CONSIDERAÇÕES FINAIS

Este trabalho abordou o Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Para resolvê-lo, foi proposto um algoritmo heurístico híbrido, denominado GENILS. Para gerar uma solução inicial foram utilizadas adaptações do método da Inserção Mais Barata e da heurística GENIUS. Para refinar essa solução, foi utilizada a metaheurística *Iterated Local Search* (ILS), tendo o *Variable Neighborhood Descent* (VND) como método de busca local. O VND explora a vizinhança de uma solução utilizando os movimentos *Shift*, *Shift*(2,0), *Swap*, *Swap*(2,1), *Swap*(2,2), *M2-Opt* e *kOr-Opt*, apresentados na Subseção 3.2. Além disso, ele realiza uma intensificação da busca sempre que ocorre uma melhora na solução corrente. Essa intensificação é feita somente nas rotas modificadas e é realizada pelas buscas locais apresentadas na Seção 3.2, pelos procedimentos *G3-opt* e *G4-opt* (os quais são baseados na heurística GENIUS) e pelo movimento *Reverse*.

De acordo com os resultados obtidos, verifica-se que o algoritmo proposto é competitivo com as melhores abordagens da literatura, sendo capaz de produzir soluções de qualidade. De fato, em um conjunto consagrado de problemas-teste, foram alcançados todos os melhores resultados da literatura; em outro, foram gerados nove melhores resultados e seis resultados iguais aos melhores da literatura; e no terceiro, foram igualados três resultados da literatura, tendo-se um *gap* máximo igual a 3,16% para os demais problemas desse conjunto. Além disso, o GENILS obteve soluções com variabilidade inferior a 1% em 67 dos 72 problemas-

teste, o que corresponde a 93% dos casos. Um comportamento interessante do algoritmo GENILS é o fato deste obter o melhor desempenho nos problemas de maior porte, os de Montané e Galvão (2006), o que mostra o seu potencial em resolver aplicações reais, onde geralmente defrontamos com problemas de elevadas dimensões.

Como trabalho futuro, pretende-se aprimorar os procedimentos *G3-opt* e *G4-opt*, baseados na heurística GENIUS, de forma a considerar a recombinação de múltiplas rotas. Além disso, é estratégico combinar o algoritmo GENILS com a metaheurística Busca Tabu, sendo esta acionada em substituição ao VND, por exemplo, após certo número de iterações do ILS. Isso se deve ao fato de que a Busca Tabu é o algoritmo base de Wassan *et al*. (2007) e Zachariadis *et al.* (2009), os quais têm a maioria dos melhores resultados dos problemas-teste de Salhi e Nagy (1999), conjunto em que o GENILS teve o pior desempenho.

Agradecimentos
Os autores agradecem a CAPES, CNPq, FAPERJ e FAPEMIG pelo apoio parcial ao desenvolvimento deste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

Anderberg, M. R. (2007) *Cluster analysis for applications*. Monographs and Textbooks on Probability and Mathematical Statistics. Academic Press, Inc., New York.

Bean, J. C. (1994) Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154-160.

Bianchessi, N.; Righini, G. (2007) Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, 34(2):578-594.

Chen, J. F. (2006) Approaches for the vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Chinese Institute of Industrial Engineers*, 23(2):141-150.

Chen, J. F.; Wu, T. H. (2006) Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society*, 57(5):579-587.

Crispim, J.; Brandão, J. (2005) Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the Operational Research Society*, 56(7):1296-1302.

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

Dell'Amico, M.; Righini, G.; Salanim, M. (2006) A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235-247.

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

Dorigo, M.; Maniezzo, V.; Colorni, A. (1996) The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, v 26, p 29-41.

Dueck, G. (1993) New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104:86-92.

Gehring, H; Homberger, J. (1999) A parallel hybrid evolutionary metaheristic for the vehicle routing problem with time windows. In: Miettinen K, Mäkelä M, Toivanen J, editors. *Proceedings of EUROGEN99*, v A2(S), Springer, Berlin, p 57-64.

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

Glover, F.; Laguna, M. (1992) *Tabu Search*, Kluwer Academic Publishers, Boston.

Gökçe, E. I. (2004) A revised ant colony system approach to vehicle routing problems. Master's thesis, Graduate School of Engineering and Natural Sciences, Sabanci University.

Goldberg, D. E. (1989) Genetic Algorithms in Search. *Optimization and Machine Learning*. Addison-Wesley, Berkeley.

Halse, K. (1992) *Modeling and solving complex vehicle routing problems*. PhD thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, Denmark.

Hansen, P.; Mladenović, N. (2003) Variable neighborhood search: Principles and applications, *European Journal of Operations Research*, 130:449-467.

Kirkpatrick, S; Gellat, D.C.; Vecchi, M. P. (1983) Optimization by Simulated Annealing, *Science*, 220:671–680.

Min, H. (1989) The multiple vehicle routing problems with simultaneous delivery and pick-up points.

Transportation Research A, 23(5):377-386.

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

Montané, F. A. T.; 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.

Nagy, G.; Salhi, S. (2005) Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162:126-141.

Or, I. (1976) *Traveling salesman-type combinational problems and their relation to the logistics of blood banking*. PhD thesis, Northwestern University, USA.

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

Röpke, S.; Pisinger, D. (2006) A unified heuristic for a large class of vehicle routing problems with backhauls. Technical Report 2004/14, University of Copenhagen.

Salhi, S.; 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.

Shaw, P. (1998) Using constraint programming and local search methods to solve vehicle routing problems. In: *CP-98 Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming*, p 417-431, London.

Solomon, M. M.; Joachim, I.; Desrosiers, J.; Dumas, Y.; Villeneuve, D. (1995) A request clustering algorithm for door-to-door handicapped transportation, *Transportation Science*, 29:63-78.

Stützle, T.; Hoos, H. H. (1999) Analyzing the run-time behaviour of iterated local search for the tsp. In *Proceedings of the Third Metaheuristics International Conference*, p 449-453, Angra dos Reis, Rio de Janeiro.

Subramanian, A. (2008) Metaheurística *Iterated Local Search* aplicada ao problema de roteamento de veículos com coleta e entrega simultânea. Dissertação de mestrado, Universidade Federal da Paraíba, João Pessoa.

Subramanian, A.; Cabral, L. A. F.; Ochi, L. S. (2008) An efficient ILS heuristic for the vehicle routing problem with simultaneous pickup and delivery. Relatório Técnico, Universidade Federal Fluminense, disponível em http://www.ic.uff.br/~satoru/ index.php?id=2.

Topcuoglu, H. and Sevilmis, C. (2002) Task scheduling with conicting objectives. In Yakhno, T. M., editor, ADVIS, volume 2457 of *Lecture Notes in Computer Science*, p 346-355. Springer.

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

Vural, A. V. A (2003) GA based meta-heuristic for capacited vehicle routing problem with simultaneous pick-up and deliveries. Master's thesis, Graduate School of Engineering and Natural Sciences, Sabanci University.

Wassan, N. A.; Wassan, A. H.; 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.; 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.

Marcio Tadayuki Mine (mmine@ic.uff.br)
Matheus de Souza Alves Silva (msalves@ic.uff.br)
Luiz Satoru Ochi (satoru@ic.uff.br)
Instituto de Computação, Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, 3º Andar, CEP 24210-240 – Niterói, Rio de Janeiro, Brasil

Marcone Jamilson Freitas Souza (marcone@iceb.ufop.br)
Departamento de Computação, Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto
Campus Universitário, Morro do Cruzeiro, CEP 35400-000 – Ouro Preto, Minas Gerais, Brasil