SIMULATED ANNEALING APLICADO À RESOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO

Aloísio de Castro Gomes Júnior
Marcone Jamilson Freitas Souza
Alexandre Xavier Martins
Universidade Federal de Ouro Preto
Departamento de Computação

RESUMO
Este trabalho apresenta um algoritmo eficiente, baseado na metaheurística *Simulated Annealing* (SA), para resolver o Problema de Roteamento de Veículos com Janela de Tempo. Esse problema tem como objetivo determinar as rotas de custo mínimo para uma frota de veículos de mesma capacidade, atendendo à demanda de um conjunto de clientes, para os quais o atendimento somente é possível dentro de um intervalo de tempo determinado, chamado janela de tempo. A metodologia proposta, denominada SA-RAI, incorpora ao algoritmo *Simulated Annealing* clássico, mecanismos auto-adaptativos para determinação da temperatura inicial e número de iterações em uma mesma temperatura. Nesta metodologia, quando a temperatura atinge um valor limiar, a mesma é reaquecida um certo número de vezes, possibilitando escapar de ótimos locais. Além disso, ela conta com uma fase de intensificação. Sempre que uma melhor solução é encontrada, ela é submetida a um procedimento de refinamento, visando ao seu melhoramento. A metodologia foi aplicada a 168 instâncias-teste da literatura e 13 novos melhores resultados foram encontrados.

ABSTRACT
This work presents an efficient algorithm based on Simulated Annealing metaheuristic (SA) for solving the Vehicle Routing Problem with Time Windows. The goal of this problem is to find a set of routes with minimum cost (in terms of distance travelled) for a fleet of identical vehicles based at the depot, obeying the capacity of the vehicles and the demand of a set of costumers and its predefined time windows. The proposed method, named SA-RAI, uses auto-adaptive mechanisms to determine the initial temperature and number of iterations in a same temperature. When the temperature is close to zero, this one is reheating in order to escape local optimuns. In addition, SA-RAI uses a refinement phase when a new best solution is found. The proposed method was applied to 168 benchmarks and 13 new best solutions were found.

1. INTRODUÇÃO

O Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) é um dos vários problemas derivados do Problema de Roteamento de Veículos (PRV), um dos problemas mais estudados da área de Otimização Combinatória. É um problema da classe NP-difícil, ou seja, não existem algoritmos que o resolva em tempo polinomial e o uso de métodos exatos para sua resolução se torna bastante restrito.

O PRVJT pode ser definido como segue. Seja $G=\{V, A\}$ um grafo onde $V=\{v_0, v_1, ..., v_n\}$ é o conjunto de vértices e $A=\{(v_i,v_j) : v_i, v_j \in V, i\neq j\}$ o conjunto de arcos. O vértice v_0 representa o Centro de Distribuição (CD), enquanto os demais n vértices correspondem aos clientes. Com A são associados uma matriz de custos (c_{ij}) e uma matriz de tempo de viagens (t_{ij}). Se estas matrizes são simétricas, o caso mais comum, então é padrão definir o PRVJT em um grafo não direcionado $G=\{V, E\}$, onde $E=\{(v_i,v_j) : v_i, v_j \in V, i\neq j\}$ é um conjunto de arestas. Cada cliente tem uma demanda não-negativa m_i e um tempo de atendimento f_i. Uma frota de veículos idênticos de capacidade Q é atribuída ao CD. O número de veículos pode ser conhecido com antecedência ou tratado como uma variável de decisão. Neste trabalho considera-se que o número de veículos é ilimitado. O PRVJT consiste em designar um conjunto de rotas de entrega ou coleta tal que:

- cada rota inicia e termina no CD;

- cada cliente é visitado exatamente uma vez por exatamente um veículo;
- a demanda total de cada rota não exceda Q;
- a janela de tempo $[e_i, l_i]$ deve ser respeitada, ou seja, um cliente não pode ser atendido antes de e_i e nem após o tempo l_i e
- o custo total da rota seja minimizado.

Encontrar uma solução de boa qualidade para o PRVJT é de fundamental importância, uma vez que segundo Ballou (2001), o custo de transporte corresponde de um terço a dois terços dos custos logísticos de uma empresa.

Devido à importância prática do PRV e o desafio de se resolvê-lo eficientemente, várias técnicas de solução têm sido relatadas na literatura. Algumas delas são apresentadas a seguir.

Clark e Wright (1964) desenvolveram uma técnica heurística que insere, em cada iteração, um cliente a uma rota de acordo com a economia que este apresenta na função de avaliação. O cliente que gera a maior economia é inserido na rota. O processo inicia-se com n rotas. Estas inserções são feitas até o momento que nenhuma economia é mais possível. Esta técnica foi aplicada ao PRV clássico.

Golden *et al*. (1984) adaptaram a heurística de Clark e Wright (1964) ao PRV com frota heterogênea, onde os veículos possuem diferentes capacidades. Os autores desenvolveram várias fórmulas de economia tentando incorporar à fórmula de economia desenvolvida por Clark e Wright (1964) os custos fixos relativos a cada tipo de veículo que poderia ser utilizado.

Liu e Shen (1999) basearam-se no trabalho de Golden *et al*. (1984), inserindo às fórmulas desenvolvidas por estes as restrições de janela de tempo pertinentes a cada cliente. Esses autores abordaram o PRV com frota heterogênea e janelas de tempo.

Gendreau *et al*. (1999) aplicaram a metaheurística Busca Tabu (BT) ao PRV com frota heterogênea. O procedimento desenvolvido por esses autores se baseia em um procedimento de memória adaptativa para gerar soluções iniciais para o procedimento BT. Nesse procedimento, as inserções dos clientes às rotas são baseadas na heurística GENIUS, desenvolvida por Gendreau *et al*. (1992) para resolver o Problema do Caixeiro Viajante.

Tailard (1993) aplica o mesmo algoritmo de Gendreau *et al*. (1999) ao PRV com frota heterogênea. O procedimento difere na estrutura de vizinhança usada. Ao invés de usar a heurística GENIUS, o mesmo utiliza trocas e inserções convencionais. Outro ponto diferente é que o método proposto pelo autor faz uso de um procedimento de re-otimização das rotas através da aplicação de um algoritmo exato para o Problema do Caixeiro Viajante.

Tan *et al*. (2001) aplicam diversas metaheurísticas ao PRV com janela de tempo. Todas elas utilizam um procedimento para determinação da solução inicial e um procedimento de dupla troca. O procedimento de geração da solução inicial procura gerar soluções iniciais factíveis baseadas na inserção, em cada iteração, de um cliente a uma rota, respeitando-se as restrições de capacidade do veículo e janela de tempo de cada cliente. A partir daí são aplicadas as metaheurísticas *Simulated Annealing*, Busca Tabu e Algoritmos Genéticos.

Cordeau *et al*. (2002) apresentam quatro características cruciais para um bom algoritmo para o PRV e suas variantes. As quatro características apontadas são: Precisão (que mede o quão distante a solução heurística ficou da solução ótima ou da melhor solução conhecida); Velocidade (que avalia o tempo para a tomada de decisões); Simplicidade (que avalia a facilidade de se implementar e entender o código e também o número de parâmetros que são utilizados, que podem facilitar ou dificultar a compreensão do algoritmo, além de dificultar a implementação do mesmo) e a Flexibilidade (que avalia a capacidade para incluir novas restrições comumente encontradas na maioria das aplicações da vida real). Esses autores apresentam, ainda, uma comparação entre várias técnicas aplicadas à resolução do PRV baseadas nestas quatro características.

Neste trabalho apresenta-se uma metodologia, baseada na metaheurística *Simulated Annealing*, para resolver eficientemente o PRVJT. Esta metodologia faz uso de mecanismos auto-adaptativos para determinar a temperatura inicial e o número máximo de iterações em uma dada temperatura. Por um determinado número de vezes, sempre que a temperatura atinge um valor limiar, é feito um reaquecimento para tentar escapar de ótimos locais a baixas temperaturas. Além disso, sempre que o método encontra uma melhor solução, é aplicado um mecanismo de busca local para refinar a solução.

Este trabalho está organizado como segue. Na seção 2 apresenta-se um modelo de programação matemática para o PRVJT. Na seção 3 especifica-se, em detalhe, a metodologia proposta para resolver o problema. Os resultados são apresentados e discutidos na seção 4. A seção 5 conclui o trabalho.

2. FORMULAÇÃO DE PROGRAMAÇÃO MATEMÁTICA

Apresenta-se, a seguir, uma formulação de programação matemática, descrita em Tan *et al*. (2001), para o Problema de Roteamento de Veículos com Janela de Tempo. Esta formulação considera que a frota é heterogênea.

- Variáveis de decisão:
  - t_i = hora de chegada no cliente i
  - w_i = tempo de espera no cliente i
  - $x_{ijk} = \begin{cases} 1 & \text{se há um arco ligando o cliente } i \text{ ao cliente } j \text{ usando o veículo } k \\ 0 & \text{caso contrário} \end{cases}$
- Dados de entrada:
  - K = número total de veículos
  - n = número total de clientes
  - d_{ij} = distancia euclidiana entre o cliente i e o cliente j
  - c_{ij} = custo incorrido no arco do cliente i ao cliente j
  - t_{ij} = tempo de viagem entre o cliente i e o cliente j
  - m_i = demanda do cliente i
  - q_k = capacidade do veículo k
  - e_i = início da janela de tempo do cliente i
  - l_i = final da janela de tempo do cliente i
  - f_i = tempo de serviço no cliente i
  - r_k = tempo de rota máximo permitido para o veículo k.

As equações (1) a (11) modelam o PRVJT:

$$Min \sum_{i=0}^{n} \sum_{j=0; j\neq i}^{n} \sum_{k=1}^{K} c_{ij} x_{ijk} \quad (1)$$

Sujeito a:

$$\sum_{k=1}^{K} \sum_{j=1}^{n} x_{ijk} \leq K \quad \text{para } i = 0 \quad (2)$$

$$\sum_{j=1}^{n} x_{ijk} = \sum_{j=1}^{n} x_{jik} \leq 1 \text{ para } \; i = 0 \;\; e \; \forall k \in \{1, 2, ..., K\} \quad (3)$$

$$\sum_{k=1}^{K} \sum_{j=0,\, j\neq i}^{n} x_{ijk} = 1 \;\; \forall i \in \{1,..., n\} \quad (4)$$

$$\sum_{k=1}^{K} \sum_{i=0,\, i\neq j}^{n} x_{ijk} = 1 \;\; \forall j \in \{1,..., n\} \quad (5)$$

$$\sum_{i=1}^{n} m_i \sum_{j=0,\, j\neq i}^{n} x_{ijk} \leq q_k \quad \forall k \in \{1,..., K\} \quad (6)$$

$$\sum_{i=0}^{n} \sum_{j=0,\, j\neq i}^{n} x_{ijk} \left(t_{ij} + f_i + w_i\right) \leq r_k \quad \forall k \in \{1,..., K\} \quad (7)$$

$$t_0 = w_0 = f_0 = 0 \quad (8)$$

$$\sum_{k=1}^{K} \sum_{i=0,\, i\neq j}^{n} x_{ijk} \left(t_i + w_i + f_i + t_{ij}\right) \leq t_j \quad \forall j \in \{1,..., n\} \quad (9)$$

$$e_i \leq \left(t_i + w_i\right) \leq l_i \quad \forall i \in \{1,..., n\} \quad (10)$$

$$x_{ijk} \in \{0,1\} \quad \forall i, j \in \{1,..., n\}, \;\; \forall k \in \{1,..., K\} \quad (11)$$

A expressão (1) é a função objetivo do problema. A restrição (2) especifica que há no máximo *K* rotas saindo do CD. As equações (3) fazem com que todas as rotas iniciem e terminem no CD. Os conjuntos de equações (4) e (5) definem que todo nó cliente pode ser visitado somente uma vez por um veículo. Equação (6) é a restrição de capacidade. As equações (7) limitam o tempo máximo de viagem. As restrições (8) a (10) definem as janelas de tempo, enquanto as restrições (11) indicam a bivalência das variáveis de decisão.

3. METODOLOGIA

Esta seção apresenta a metodologia utilizada para resolver o problema. Mostra-se como uma solução para o problema é representada, como são geradas as soluções iniciais para o método, os tipos de movimentos usados para explorar o espaço de soluções do problema, a função de avaliação utilizada para guiar a busca e a descrição detalhada do algoritmo proposto.

3.1. Representação de uma solução

Uma solução do problema é representada por uma lista de números inteiros, sendo que os números positivos representam os clientes a serem visitados e os negativos, os veículos e suas respectivas capacidades. Essa lista é ordenada e a ordem de apresentação dos números indica o veículo usado e a ordem de visita dos clientes. Assim, a lista {-200, 1, 3, 2, 5, -200, 6, 4, 7} indica que um veículo de capacidade 200 sai do centro de distribuição (CD) e visita os

clientes 1, 3, 2 e 5, nesta seqüência, e retorna ao CD. Outro veículo de mesma capacidade sai do CD e visita os clientes 6, 4 e 7 e retorna ao CD.

3.2. Determinação de uma Solução Inicial

Para gerar uma solução inicial para o PRVJT criou-se um procedimento que, partindo do CD, insere a cada iteração um único cliente a uma rota. A inserção é feita observando o limite de capacidade do veículo e a janela de tempo de cada cliente.

O cliente a ser inserido na rota corrente é escolhido aleatoriamente dentre aqueles que ainda não foram inseridos e cuja inserção não faça ultrapassar nem a capacidade do veículo alocado à rota nem a janela de tempo pré-definida de cada cliente. Se não existem clientes satisfazendo a essas duas condições, faz-se o retorno ao CD e uma nova rota é iniciada.

Este procedimento termina quando todos os clientes forem inseridos. Observa-se que, com este procedimento, as soluções iniciais são sempre factíveis. Além disso, tem-se um limite superior para o número de veículos a serem utilizados (constante *K* da restrição 6 da seção 2).

3.3. Estrutura de Vizinhança

Dada uma solução *s*, um vizinho *s'* desta solução é obtido ou por um movimento de realocação de clientes ou por um movimento de troca entre clientes na ordem das visitas.

O movimento de realocação pode ser inter-rota, quando um cliente é passado de uma rota para outra, ou intra-rota, na qual a ordem de visita de um cliente é alterada dentro de uma mesma rota. A Figura 1 ilustra a aplicação dos dois possíveis movimentos de realocação.

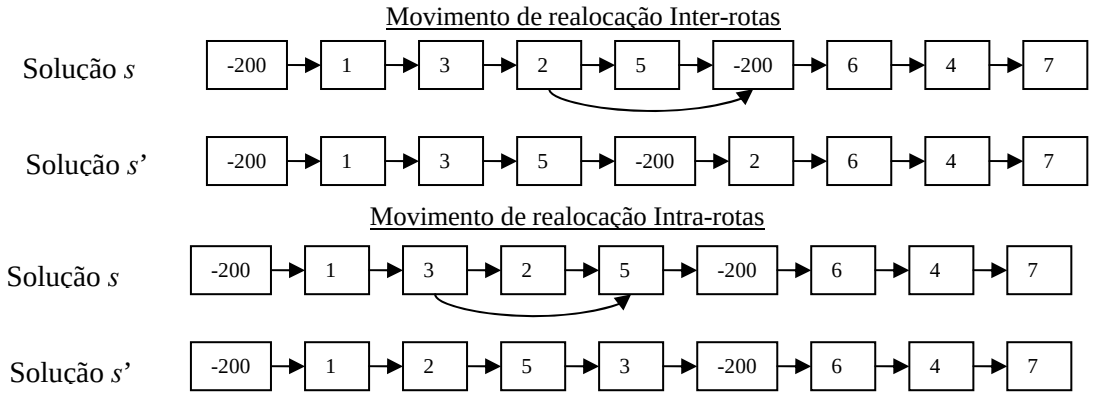

Figura 1: Estrutura de Vizinhança

Na Figura 1, o movimento inter-rotas mostra que o cliente 2 é realocado para ser o primeiro cliente a ser visitado pelo segundo veículo de capacidade 200. Já o movimento intra-rotas é ilustrado com o cliente 3 sendo realocado para ser visitado após o cliente 5 da primeira rota.

O movimento de troca também pode ser intra-rota, quando dois clientes de uma mesma rota têm suas ordens de visita trocadas entre si, ou inter-rota, quando há a permutação entre clientes de rotas distintas.

Observa-se que, com esses dois tipos de movimentos, são consideradas na exploração do espaço de busca tanto as soluções factíveis quanto as não factíveis.

3.4. Função de Avaliação

Uma solução *s* é avaliada por uma função que leva em conta o custo de transporte do cliente *i* para o cliente *j*, a capacidade do veículo que irá atender aquela rota e a janela de tempo em que o cliente deve ser atendido. Esta função é dada pela expressão:

$$f(s) = \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} + \mu f_1(s) + \lambda f_2(s) \quad (12)$$

em que c_{ij} representa o custo de transporte do cliente *i* para o cliente *j*, $f_1(s)$ representa a sobrecarga dos veículos, ou seja, o excesso de peso nos veículos e $f_2(s)$ representa o tempo em que um veículo excede o horário máximo permitido para atendimento a determinado cliente. O cliente 0 é o centro de distribuição (CD), μ é um fator de penalidade igual ao valor da maior distância entre dois clientes quaisquer da instância considerada e λ outro fator de penalidade, fixado em λ = 1000. A comparação entre duas soluções é feita apenas com base no valor da função de avaliação, sem levar em consideração o número de veículos associado a cada solução.

3.5. Algoritmo SA-RAI

Este método consiste no algoritmo *Simulated Annealing* clássico, ao qual é incorporado um mecanismo auto-adaptativo para determinar a temperatura inicial e o número máximo de iterações em uma dada temperatura. O método conta também com um procedimento de busca local que é acionado sempre que uma nova melhor solução é gerada. Além disso, por um certo número de vezes a temperatura é reaquecida quando se aproxima de zero, de forma a escapar de mínimos locais.

O método parte de uma solução inicial gerada conforme seção 3.2 e consiste de um procedimento iterativo que gera, a cada iteração, um único vizinho *s'* da solução corrente *s*. Seja Δ a variação de valor da função objetivo ao mover-se para uma solução vizinha candidata, isto é, $\Delta = f(s') - f(s)$. Para o PRVJT, o método aceita o movimento e a solução vizinha passa a ser a nova solução corrente se Δ < 0. Caso Δ ≥ 0 a solução vizinha candidata também poderá ser aceita, mas neste caso, com uma probabilidade $e^{-\Delta/T}$, onde *T* é um parâmetro do método, chamado de temperatura, que regula a probabilidade de se aceitar soluções de pior custo. A temperatura *T* assume, inicialmente, um valor elevado T_0 (obtido de forma auto-adaptativa pelo procedimento descrito na seção 3.6). Após um número fixo de iterações (que representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico), a temperatura é gradativamente diminuída por uma razão de resfriamento α, tal que $T_k \leftarrow \alpha \times T_{k-1}$, sendo $0<\alpha<1$. Com esse procedimento, dá-se, no início uma chance maior para escapar de mínimos locais e, à medida que *T* se aproxima de zero, o algoritmo se comporta como um método de descida, uma vez que diminui a probabilidade de se aceitar movimentos de piora.

De forma a escapar de ótimos locais a baixas temperaturas, quando a temperatura se aproxima de seu valor mínimo, no caso 0,01, ela é reaquecida. Este procedimento é aplicado por *NR* vezes, sendo a temperatura de reinício calculada com base na fórmula (13):

$$T_r = \frac{T_{r-1}}{e^{\ln(T_{r-1}/100)/(NR-r+1)}} \quad (13)$$

em que T_r é a r-ésima temperatura de reinício ($r > 1$) e NR é o número de reinícios. Esta fórmula obriga o algoritmo em seu último reinício a trabalhar com uma temperatura inicial igual a 100, que foi o parâmetro utilizado por Tan *et al*. (2001).

O método SA-RAI conta, ainda, com um mecanismo de busca local (MBL), que é acionado sempre que uma nova melhor solução é gerada. No caso, é aplicado um procedimento de descida usando movimentos de troca (intra e inter-rotas).

Os parâmetros de controle do método SA-RAI são o número de reinícios (fixado em 2), a razão de resfriamento α (fixada em 0,998), o número de iterações para cada temperatura ($SAmax = n$, sendo n o número de clientes) e a temperatura inicial T_0, a qual é determinada pelo procedimento auto-adaptativo descrito na seção 3.6. A Figura 3 apresenta o pseudo-código do método proposto.

```
Procedimento SA-RAI (NR, α, SAmax, T0, s)
1       s* ← s;     {melhor solução obtida até então}
2       IterT ← 0; {Número de iterações na temperatura T}
3       T ← T0;     {Temperatura Corrente}
4       r ← 1;      {Número corrente de reinícios}
5       Tr ← T0;
6       enquanto (r ≤ NR) faça
7           enquanto (T > 0) faça
8               enquanto (IterT < SAmax) faça
9                   IterT← IterT + 1;
10                  Gere um vizinho qualquer s' ∈ N(s);
11                  Δ = f(s') - f(s);
12                  se (Δ < 0)
13                      então
14                        s ← s';
15                        se (f(s') < f(s*))
16                            então
17                                  Melhore s' usando MBL;
18                                  s* ← s';
19                      senão
20                        Escolha x ∈ [0, 1];
21                        se (x < e^(-Δ/T)) então s ← s';
22                  fim-se;
23              fim-enquanto;
24              T←α×T;
25              IterT ← 0;
26          fim-enquanto;
27          T ← Tr / e^(ln(Tr/100)/(NR−r+1));
28          Tr ← T;
29          r ← r + 1;
30      fim-enquanto;
31      s ← s*;
32      Retorne s;
fim SA-RAI;
```

Figura 3: Pseudo-código do algoritmo SA-RAI

3.6. Determinação da Temperatura Inicial

Utilizou-se um mecanismo auto-adaptativo para determinar a temperatura inicial do algoritmo SA-RAI. Tal mecanismo, descrito em Souza (2005), consiste em partir de uma solução *s* qualquer e de uma temperatura de partida baixa. A seguir, contam-se quantos vizinhos dessa solução *s* são aceitos em um determinado número de iterações (*SAmax*) nessa temperatura. Caso esse número de vizinhos aceitos seja elevado, no caso 90% dos vizinhos, então retorna-se a temperatura corrente como a temperatura inicial para o processo de refinamento. Caso o número de vizinhos aceitos não atinja o valor mínimo requerido, aumenta-se a temperatura segundo uma certa taxa, no caso 10%, e repete-se a contagem do número de vizinhos aceitos naquela temperatura. O procedimento prossegue até que se obtenha o número mínimo de vizinhos aceitos. A temperatura na qual essa condição ocorre representa a temperatura inicial para o método. A Figura 2 apresenta o pseudo-código deste procedimento. Os parâmetros utilizados por este mecanismo foram: β – taxa de aumento da temperatura (no caso $\beta = 1{,}1$), γ – taxa mínima de aceitação de soluções vizinhas (no caso $\gamma = 0{,}90$) e T_0 – temperatura de partida para o método ($T_0 = 2$).

```
Procedimento TemperaturaInicial (β, γ, SAmax, T0, s)
1.  T ← T0; {Temperatura Corrente}
2.  Continua ← TRUE;
3.  enquanto (Continua) faça
4.      Aceitos←0; {Número de vizinhos aceitos na temperatura T}
5.      Para IterT = 1 até SAmax faça
6.          Gere um vizinho qualquer s' ∈ N(s);
7.          Δ = f(s')-f(s);
8.          se (Δ < 0)
9.              então
10.                 Aceitos ← Aceitos + 1;
11.             senão
12.                 Tome x ∈ [0, 1];
13.                 se (x < e^(-Δ/T)) então Aceitos ← Aceitos + 1;
14.         fim-se;
15.     fim-para;
16.     se (Aceitos ≥ γ × SAmax)
17.         então Continua ← FALSE;
18.         senão T ← β × T;
19.     fim-se
20. fim-enquanto;
21. T0 ← T;
22. Retorne T0;
fim TemperaturaInicial;
```

Figura 2: Pseudo-código para determinação da temperatura inicial

4. APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

O algoritmo proposto foi desenvolvido na linguagem C, utilizando o compilador C++ Builder 6.0 da Borland, e executado em um computador PENTIUM® IV 1.8 GHz com 512 MB de memória RAM, sob plataforma Windows XP Pro.

Os problemas-teste adotados para a validação do algoritmo são as 168 instâncias apresentadas em Solomon (1987). Essas instâncias se dividem em 3 grupos: C – onde os clientes se encontram clusterizados, ou seja, estão distribuídos geograficamente em grupos de clientes próximos uns dos outros; R – onde os clientes são distribuídos aleatoriamente sem formar

grupos e distantes uns dos outros e RC – onde se tem uma mistura dos dois grupos anteriores. O algoritmo proposto foi executado trinta vezes, cada qual partindo de uma semente diferente de números aleatórios.

Os resultados obtidos são apresentados nas tabelas a seguir. Nessas tabelas, os melhores resultados da literatura advêm de duas fontes, cujo acesso se deu em 16/05/2005: (1) http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html e (2) http://w.cba.neu.edu/~msolomon.

As tabelas 2 e 3 mostram as características das soluções obtidas, bem como o desempenho do método SA-RAI em todas as instâncias de 50 e 100 cidades. Nestas tabelas, a coluna "DT" representa a distância total percorrida pelos veículos e "NV" o número de veículos da solução. A coluna "Desvio Médio" é calculada com base na seguinte fórmula:

$$Desvio\,Médio = \frac{\frac{1}{30}\left(\sum_{i=1}^{30} f(s_i)\right) - f(s^*)}{f(s^*)} \tag{14}$$

sendo $f(s^*)$ o melhor resultado encontrado na literatura e $f(s_i)$ o resultado encontrado na i-ésima execução do algoritmo. Já a coluna "Desvio da Melhor Solução" é calculada com base na seguinte fórmula:

$$Desvio\ da\ Melhor\ Solução = \frac{\min\{f(s_i);\ i = 1,...,30\} - f(s^*)}{f(s^*)} \tag{15}$$

A Tabela 1 apresenta os resultados relativos ao desempenho do algoritmo aplicado aos grupos de instâncias envolvendo 25, 50 e 100 clientes. Nesta tabela são apresentadas a média dos desvios médios encontrados para cada grupo de instância, bem como a média dos desvios das melhores soluções encontradas nesse grupo.

Tabela 1: Desempenho médio do algoritmo SA-RAI nos grupos de instâncias

Grupo de instâncias	Média do "Desvio Médio"	Média do "Desvio da Melhor Solução"
C25	0,13%	0,00%
R25	0,88%	0,00%
RC25	0,98%	0,02%
C50	1,56%	0,15%
R50	2,94%	0,56%
RC50	4,04%	0,89%
C100	8,23%	2,72%
R100	3,62%	0,07%
RC100	5,78%	1,67%
Média	3,13%	0,68%

Como pode ser observado na Tabela 1, as melhores soluções geradas pelo procedimento SA-RAI desviam dos melhores resultados da literatura em apenas 0,68%, na média. Considerando os valores médios gerados nas diversas execuções dos algoritmos, o procedimento tem um desvio médio de 3,13%. Assim, tendo em vista que as melhores soluções da literatura foram obtidas por diferentes algoritmos, cada qual explorando as especificidades de uma dada instância ou conjunto de instâncias, o algoritmo desenvolvido mostrou-se capaz de produzir soluções finais bastante satisfatórias.

5. CONCLUSÕES

Neste trabalho é proposto um algoritmo, baseado na metaheurística *Simulated Annealing*, para resolver o problema de roteamento de veículos com janela de tempo. A metodologia proposta, denominada SA-RAI, incorpora ao algoritmo *Simulated Annealing* clássico, mecanismos auto-adaptativos para determinação da temperatura inicial e número de iterações em uma mesma temperatura. Neste método, quando a temperatura atinge um valor limiar, a mesma é reaquecida um certo número de vezes, possibilitando escapar de ótimos locais. Além disso, ele conta com uma fase de intensificação. Sempre que uma melhor solução é encontrada, ela é submetida a um procedimento de refinamento, visando ao seu melhoramento.

A metodologia foi aplicada a 168 instâncias-teste da literatura e 13 novos melhores resultados foram encontrados. Além desses resultados obtidos, a metodologia produziu soluções finais que, em média, diferem pouco das melhores soluções da literatura.

REFERÊNCIAS BIBLIOGRÁFICAS

Ballou, R. (2001) *Gerenciamento da Cadeia de Suprimentos: Planejamento, Organização e Logística Empresarial.* São Paulo: Bookman.

Clark, G. e J. Wright (1964) Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, v. 12, p. 568-581.

Cordeau, J. F.; M. Gendreau; G. Laporte; J. Y. Potvin e F. Semet (2002) A guide to Vehicle Routing Problem. *Journal of the Operational Research Society,* v. 53, p. 512-522.

Gendreau, M.; G. Laporte; C. Musaraganyi e E. D. Taillard (1999) A Tabu Search Heuristic for the Heterogeneous Fleet Vehicle Routing Problem. *Computers & Operations Research***,** v. 26, p. 1153-1173.

Gendreau, M.; A. Hertz e G. Laporte (1992) New Insertion and Postoptimization Procedures for the Traveling Salesman Problem. *Operations Research*, v. 40, p. 1086-1093.

Golden, B.; A. Assad; L. Levy e F. Gheysens (1984) The Fleet Size and Mix Vehicle Routing. *Computers & Operations Research*. Grã-Bretanha, v. 11, n. 1, p. 49-66.

Liu, F. e S. Y. Shen (1999) A Metohd for Vehicle Routing Problem with Multiple Vehicle Types and Time Windows. *Proc. Natl. Sci. Counc***.**. Hsinchu, Taiwan, v. 23, p. 526-536.

Souza, M. J. F. (2005) *Inteligência Computacional para Otimização*. Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, Disponível em http://www.decom.ufop.br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional.ps.

Taillard, É. (1993) Parallel iterative search methods for vehicle routing problems. *Networks*, v. 23, p. 661-673.

Tan, K.C.; L. H. Lee; Q. L. Zhu e K. Ou (2001) Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering*, v. 15, p. 281-295.

Aloísio de Castro Gomes Júnior (algomesjr2004@yahoo.com.br)
Marcone Jamilson Freitas Souza (marcone@iceb.ufop.br)
Alexandre Xavier Martins (xmartins@uai.com.br)
Departamento de Computação, Universidade Federal de Ouro Preto
Campus Universitário, Morro do Cruzeiro
35.400.000 – Ouro Preto, MG, Brasil.

ANEXOS

Tabela 2: Desempenho do algoritmo SA-RAI nas instâncias de 50 clientes

Instância	Melhor Publicado		SA-RAI				
	DT	NV	DT	NV	Desvio Médio	Desvio da Melhor Solução	Tempo médio (s)
C101-50	362,4 [2]	5	**362,4**	5	0,51%	**0,00%**	38,90
C102-50	361,4 [2]	5	**361,4**	5	0,54%	**0,00%**	36,69
C103-50	361,4 [2]	5	**361,4**	5	0,64%	**0,00%**	33,52
C104-50	358,0 [2]	5	**358,0**	5	2,47%	**0,00%**	30,51
C105-50	362,4 [2]	5	**362,4**	5	0,28%	**0,00%**	37,14
C106-50	362,4 [2]	5	**362,4**	5	**0,00%**	**0,00%**	38,33
C107-50	362,4 [2]	5	**362,4**	5	0,93%	**0,00%**	35,34
C108-50	362,4 [2]	5	**362,4**	5	0,25%	**0,00%**	33,38
C109-50	362,4 [2]	5	**362,4**	5	0,52%	**0,00%**	31,86
C201-50	360,2 [2]	3	**360,2**	3	4,90%	**0,00%**	36,96
C202-50	360,2 [2]	3	**360,2**	3	4,19%	**0,00%**	34,73
C203-50	359,8 [2]	3	**359,8**	3	1,17%	**0,00%**	33,07
C204-50	350,1 [2]	2	359,1	2	4,42%	2,57%	28,16
C205-50	359,8 [2]	3	**359,8**	3	0,76%	**0,00%**	34,43
C206-50	359,8 [2]	3	**359,8**	3	1,30%	**0,00%**	32,75
C207-50	359,6 [2]	3	**359,6**	3	1,10%	**0,00%**	32,29
C208-50	350,5 [2]	2	**350,5**	2	2,58%	**0,00%**	32,09
R101-50	1044,0 [2]	12	1045,3	13	0,74%	0,12%	42,02
R102-50	909,0 [2]	11	**909,0**	11	0,89%	**0,00%**	38,35
R103-50	772,9 [2]	9	773,8	9	1,51%	0,12%	36,06
R104-50	625,4 [2]	6	629,5	6	3,65%	0,66%	33,04
R105-50	899,3 [2]	9	904,4	10	2,55%	0,57%	38,47
R106-50	793,0 [2]	5	**793,0**	8	2,45%	**0,00%**	36,85
R107-50	711,1 [2]	7	715,5	7	3,70%	0,62%	34,85
R108-50	617,7 [2]	6	619,1	6	3,45%	0,23%	32,15
R109-50	786,8 [2]	8	798,7	8	3,04%	1,51%	34,96
R110-50	697,0 [2]	7	717,4	8	4,80%	2,93%	33,57
R111-50	707,2 [2]	7	708,8	7	3,40%	0,23%	33,71
R112-50	630,2 [2]	6	639,4	6	4,69%	1,46%	31,94
R201-50	791,9 [2]	6	799,7	5	3,08%	0,98%	32,37
R202-50	698,5 [2]	5	709,2	6	3,95%	1,53%	31,63
R203-50	603,5 [2]	5	605,3	5	3,31%	0,30%	30,32
R204-50	509,5 [1]	2	***508,1***	3	2,28%	***-0,27%***	28,09
R205-50	690,1 [2]	4	698,2	4	3,31%	1,17%	30,28
R206-50	632,4 [2]	4	640,6	4	3,27%	1,30%	29,68
R207-50	584,6 [1]	4	***576,8***	3	1,55%	***-1,33%***	29,26
R208-50	487,7 [1]	2	493,9	2	3,54%	1,27%	27,72
R209-50	600,6 [2]	4	600,6	4	3,13%	0,00%	29,76
R210-50	645,6 [2]	4	652,7	5	3,48%	1,10%	29,87
R211-50	551,3 [1]	3	***541,9***	3	1,78%	***-1,71%***	28,25
RC101-50	944,0 [2]	8	955,1	9	2,97%	1,18%	38,38
RC102-50	822,5 [2]	7	838,9	8	5,35%	1,99%	37,67
RC103-50	710,9 [2]	6	711,8	6	7,27%	0,13%	35,71
RC104-50	545,8 [2]	5	**545,8**	5	1,93%	**0,00%**	33,39
RC105-50	855,3 [2]	8	888,9	9	6,86%	3,93%	37,55
RC106-50	723,2 [2]	6	765,7	7	16,62%	5,88%	35,15
RC107-50	642,7 [2]	6	**642,7**	6	3,55%	**0,00%**	34,84
RC108-50	598,1 [2]	6	**598,1**	6	0,51%	**0,00%**	33,56
RC201-50	684,8 [2]	5	**684,8**	5	3,47%	**0,00%**	33,37
RC202-50	613,6 [2]	5	**613,6**	5	0,05%	**0,00%**	32,11
RC203-50	555,3 [2]	4	**555,3**	4	6,30%	**0,00%**	30,88
RC204-50	444,2 [1]	3	**444,2**	3	4,65%	**0,00%**	28,85
RC205-50	630,2 [2]	5	**630,2**	5	0,52%	**0,00%**	32,44
RC206-50	610,0 [2]	5	**610,0**	5	0,15%	**0,00%**	30,83
RC207-50	558,6 [2]	4	560,2	5	0,35%	0,29%	30,30
RC208-50	-	-	489,1	4	-	-	28,56

[1] http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html ; [2] http://w.cba.neu.edu/~msolomon.

Tabela 3: Desempenho do algoritmo SA-RAI nas instâncias de 100 clientes

Instância	Melhor Publicado		SA-RAI				
	DT	NV	DT	NV	Desvio Médio	Desvio da Melhor Solução	Tempo médio (s)
C101-100	827,3 [2]	10	**827,3**	10	2,65%	**0,00%**	216,93
C102-100	827,3 [2]	10	**827,3**	10	3,43%	**0,00%**	195,78
C103-100	826,3 [2]	10	**826,3**	10	5,93%	**0,00%**	176,10
C104-100	822,9 [2]	10	834,0	10	6,68%	1,35%	153,89
C105-100	827,3 [2]	10	**827,3**	10	3,64%	**0,00%**	199,60
C106-100	827,3 [2]	10	**827,3**	10	2,20%	**0,00%**	191,75
C107-100	827,3 [2]	10	**827,3**	10	2,67%	**0,00%**	189,05
C108-100	827,3 [2]	10	**827,3**	10	3,23%	**0,00%**	173,74
C109-100	827,3 [2]	10	**827,3**	10	3,76%	**0,00%**	160,79
C201-100	589,1 [2]	3	624,0	4	16,79%	5,92%	187,29
C202-100	589,1 [2]	3	626,9	4	15,25%	6,42%	173,76
C203-100	588,7 [2]	3	617,8	4	14,38%	4,94%	159,53
C204-100	588,1 [2]	3	635,1	4	12,96%	7,99%	144,93
C205-100	586,4 [2]	3	619,8	4	13,26%	5,70%	165,34
C206-100	586,0 [2]	3	613,1	4	10,43%	4,62%	157,78
C207-100	585,8 [2]	3	613,3	4	11,75%	4,69%	158,04
C208-100	585,8 [2]	3	612,9	4	10,95%	4,63%	151,83
R101-100	1637,7 [2]	20	1675,9	21	4,00%	2,33%	228,56
R102-100	1466,6 [2]	18	1491,1	19	3,88%	1,67%	212,50
R103-100	1208,7 [2]	14	1236,0	15	4,85%	2,26%	191,48
R104-100	971,5 [2]	11	1021,5	12	8,42%	5,15%	171,01
R105-100	1355,3 [2]	15	1399,1	17	6,38%	3,23%	201,05
R106-100	1234,6 [2]	13	1267,7	14	6,15%	2,68%	196,39
R107-100	1064,6 [2]	11	1107,7	13	7,74%	4,05%	181,88
R108-100	960,88 [2]	9	967,85	11	5,18%	0,73%	167,82
R109-100	1146,9 [2]	13	1185,3	13	7,26%	3,35%	180,77
R110-100	1068,0 [2]	12	1119,8	12	7,93%	4,85%	175,59
R111-100	1048,7 [2]	12	1101,2	13	7,75%	5,01%	179,53
R112-100	982,14 [2]	9	1000,26	11	5,31%	1,84%	162,85
R201-100	1143,2 [2]	8	1188,2	10	6,50%	3,94%	160,34
R202-100	1191,7 [2]	3	***1062,06***	9	***-6,99%***	***-10,88%***	156,87
R203-100	939,54 [2]	3	***889,69***	6	0,26%	***-5,31%***	149,18
R204-100	825,52 [2]	2	***759,70***	5	***-3,17%***	***-7,97%***	139,16
R205-100	994,42 [2]	3	1005,73	7	3,99%	1,14%	147,07
R206-100	906,14 [2]	3	917,41	6	5,15%	1,24%	144,24
R207-100	893,33 [2]	2	***832,38***	6	***-1,98%***	***-6,82%***	141,05
R208-100	726,75 [2]	2	732,60	4	6,37%	0,80%	134,48
R209-100	909,16 [2]	3	***897,53***	6	2,51%	***-1,28%***	143,38
R210-100	939,34 [2]	3	952,22	8	3,90%	1,37%	147,08
R211-100	892,71 [2]	2	***786,80***	6	***-8,10%***	***-11,86%***	134,52
RC101-100	1619,8 [2]	15	1686,0	18	7,14%	4,09%	210,34
RC102-100	1457,4 [2]	14	1515,7	16	7,37%	4,00%	205,81
RC103-100	1258,0 [2]	11	1335,4	12	10,80%	6,15%	192,22
RC104-100	1135,48 [2]	10	1204,05	12	10,19%	6,04%	178,44
RC105-100	1513,7 [2]	15	1544,0	16	7,15%	2,00%	202,19
RC106-100	1424,73 [2]	11	1454,80	14	4,33%	2,11%	188,23
RC107-100	1207,8[2]	12	1292,2	13	10,73%	6,99%	182,61
RC108-100	1114,2[2]	11	1143,5	12	9,09%	2,63%	176,49
RC201-100	1261,8 [2]	9	1311,8	10	7,65%	3,96%	165,48
RC202-100	1092,3 [2]	8	1140,3	10	7,33%	4,39%	159,54
RC203-100	1049,62 [2]	3	***977,38***	7	***-3,20%***	***-6,88%***	152,58
RC204-100	798,41 [2]	3	813,38	5	7,20%	1,87%	142,80
RC205-100	1154,0 [2]	7	1183,7	8	6,36%	2,57%	159,85
RC206-100	1146,32 [2]	3	***1107,20***	7	***-0,43%***	***-3,41%***	151,13
RC207-100	1061,14 [2]	3	***993,42***	7	***-2,01%***	***-6,38%***	145,64
RC208-100	828,14 [2]	3	***800,46***	5	2,71%	***-3,34%***	138,82

[1] http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html ; [2] http://w.cba.neu.edu/~msolomon.