UMA REVISÃO SOBRE A METODOLOGIA HÍBRIDA ANT-TPR APLICADA AO PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELAS DE TEMPO

Marcelo Caramuru Pimentel FRAGA[*]
Sérgio Ricardo de SOUZA[**]
Luís Augusto Mattos MENDES[***]
Mirela PITELI[****]

RESUMO

Este artigo apresenta uma revisão sobre uma metodologia híbrida baseada nas meta-heurísticas de colônia de formigas, busca tabu e na técnica de reconexão por caminhos, aplicada ao problema de roteamento de veículos com janela de tempo (PRVJT). A metodologia foi avaliada em problemas-teste propostos por Solomon (1987) e gerou resultados distantes em 3% dos melhores da literatura, porém com tempo computacional desfavorável. Esta revisão foi feita com o intuito de explicar melhor o funcionamento da metodologia desenvolvida. Nesta revisão, os resultados obtidos previamente foram analisados de forma diferente para permitir uma comparação mais adequada com os demais autores. O foco não será apenas nos resultados, mas também nos tempos computacionais envolvidos, tornando a metodologia mais competitiva e expondo os pontos fortes e fracos das técnicas utilizadas.

PALAVRAS-CHAVE: ANT-TPR. Roteamento de veículos. Meta-heurísticas.

ABSTRACT

This article presents an overview of a hybrid methodology based on meta-heuristics of Ant Colony Optimization, Tabu Search and the Path Relinking intensification technique. applied to the vehicle routing problem with time window (PRVJT). The methodology was benchmarked in Solomon's VRPTW instances (1987) and produced results far in 3% of the best in literature, however with unfavorable computational time. This revision was made in order to better explain the functioning of the methodology developed. In this review, the results previously obtained were analyzed differently to allow a more appropriate comparison with other authors. The focus is not only on the results, but also on computational times involved, making the methodology more competitive and exposing the strengths and weaknesses of the techniques used.

KEYWORDS: *ANT-TPR. Vehicle routing. Meta-heuristics.*

[*] Professor Assistente do CEFET-MG – Campus Divinópolis – Curso Técnico de Informática, e-mail: caramuru@div.cefetmg.br
[**] Professor Adjunto do CEFET-MG, e-mail: sergio@dppg.cefetmg.br
[***] Professor Assistente do CEFET-MG – Campus Divinópolis – Curso Técnico de Informática, e-mail: luisaugusto@div.cefetmg.br
[****] Professor Assistente da FATEC -TQ – Campus Taquaritinga – Curso Superior em Tecnologia de Agronegócio, e-mail: mlpiteli@yahoo.com.br

INTRODUÇÃO

Neste trabalho, é apresentada uma revisão da metodologia híbrida Ant-TPR detalhada em Fraga (2006) proposta para resolução do Problema de Roteamento de Veículos com Janela de Tempo (PRVJT). O PRVJT é uma variação do Problema de Roteamento de Veículos Capacitados (PRVC) na qual, além das restrições inerentes ao PRVC, o atendimento a cada consumidor deve ser iniciado em um intervalo de tempo denominado janela de tempo.

A solução para esse problema consiste em, inicialmente, encontrar o número mínimo de veículos capaz de atender a todos os consumidores e, em seguida, determinar um conjunto de rotas que minimize a distância total percorrida pelos veículos. O PRVJT é um problema NP - Difícil, logo, o uso de métodos heurísticos, que exigem menor custo computacional, têm sido amplamente utilizados, sofrendo, no entanto, das dificuldades inerentes à indefinição da qualidade da solução determinada.
A metodologia Ant-TPR combina o uso das metaheurísticas Colônia de Formigas e Busca Tabu à técnica de intensificação de resultados Reconexão por Caminhos. A metaheurística Colônia de Formigas (ACO) se alimenta, em parte, dos próprios resultados, contribuindo, a cada interação, para convergência e precisão dos resultados gerados. A Busca Tabu (BT) é uma estratégia que depende fortemente da solução inicial e, por isso, a associação com a metaheurística ACO, capaz de produzir boas soluções, potencializa o resultado de ambas. Periodicamente, a técnica Reconexão por Caminhos (RC) é aplicada, de forma a tentar inserir bons atributos das melhores soluções previamente encontradas nas soluções correntes.

1. O problema de Roteamento de Veículos

No Problema de Roteamento de Veículos Capacitados (PRVC), uma frota de veículos, localizada inicialmente em um depósito, deve atender a um conjunto de consumidores que possuem diferentes demandas por produtos a serem distribuídos pela frota. A única restrição presente no PRVC é a capacidade de carga dos veículos que, obviamente, não pode ser violada. As demandas são conhecidas a priori e não podem ser fracionadas em sua entrega, ou seja, cada consumidor não pode ser visitado mais de uma vez para entrega de produtos. Todos os veículos possuem a mesma capacidade de carga e suas rotas iniciam e terminam no depósito. O Problema de Roteamento de Veículos com Janelas de Tempo (PRVJT) é uma variação do PRVC na qual, além da restrição de capacidade, são adicionadas restrições relacionadas ao horário em que cada consumidor deve ser atendido. Para cada consumidor i, é associado um intervalo de tempo [ai, bi], chamado de janela de tempo, que indica o horário em que deve ser iniciado o atendimento e, além disso, um tempo de serviço (si), que indica o período de tempo no qual o veículo deve aguardar a conclusão das tarefas. O PRVJT consiste em designar um conjunto de rotas que atenda a essas restrições, minimize o número total de veículos utilizados e o custo total da rota.

2. Colônias de Formigas

Os algoritmos baseados em Colônia de Formigas (ou Ant Colony Optimization - ACO) têm origem nos trabalhos de Marco Dorigo (veja Dorigo et al. (1991)), inspirados no comportamento das formigas na natureza, que propôs o algoritmo Ant System (AS) para solucionar o Problema do Caixeiro

Viajante (PCV) (veja, para uma revisão completa, Dorigo & Blum (2005)).

Também de acordo com Dorigo et al. (1991), as formigas são capazes de encontrar o caminho mais curto entre a fonte de alimento e o formigueiro, através da cooperação entre os indivíduos e comunicação indireta. O comportamento das formigas na natureza é descrito da seguinte forma: inicialmente, as formigas exploram aleatoriamente a área ao redor do formigueiro à procura de comida. Enquanto se deslocam, depositam sobre o solo uma substância volátil chamada feromônio, que as auxilia a encontrar o caminho de volta ao formigueiro. Desta forma, quando uma formiga estabelece uma trilha ou caminho entre a fonte de alimento e o formigueiro, o caminho percorrido ficará marcado por um rastro desta substância. As demais formigas à procura de alimento detectam a presença de feromônio no solo e tendem a escolher o caminho com a maior concentração do mesmo. As formigas que escolheram o caminho mais curto farão o percurso em menor tempo e o rastro de feromônio será reforçado com uma freqüência maior que nos caminhos mais longos. Assim, os caminhos mais eficientes, ou seja, de menor distância percorrida, receberão maior quantidade de feromônio e tenderão a ser os mais escolhidos. Por ser uma substância volátil, a evaporação do feromônio evita que, com o tempo, um caminho que não esteja sendo mais utilizado continue a influenciar a decisão das formigas.

2.1. Ant Colony System

O algoritmo Ant Colony System (ACS) foi introduzido em Gambardella & Dorigo (1996), Dorigo & Gambardella (1997a) e Dorigo & Gambardella (1997b), de forma a melhorar o desempenho do algoritmo Ant System. Um critério de seleção, chamado pseudo-randômico-proporcional, no qual o parâmetro q_0 controla a formação das soluções, se de forma probabilística ou de forma gulosa, conforme a expressão:

$$s = \begin{cases} \arg\max_{u \notin M} \{[\tau_{(r,u)}]^{\alpha} \cdot [\eta_{(r,u)}]^{\beta}\}, & \text{se } q \leq q_0 \\ \text{exp}ressão\ (2), & \text{caso contrário} \end{cases} \qquad (1)$$

Um número aleatório q é gerado e comparado ao parâmetro q_0 cada vez que a formiga necessita escolher o próximo nó para se mover. O algoritmo funciona de forma gulosa se o valor de q for menor que o parâmetro q_0. Desta forma, a geração da solução valoriza o aprendizado gerado pela deposição do feromônio. Caso o valor de q seja maior que q0, a geração da solução será feita de forma probabilística, segundo a expressão 2.

$$p_k(r,s) = \begin{cases} \dfrac{[\tau_{(r,s)}]^{\alpha} \cdot [\eta_{(r,s)}]^{\beta}}{\sum_{u \notin M_k} [\tau_{(r,u)}]^{\alpha} \cdot [\eta_{(r,u)}]^{\beta}} & \text{se } s \notin M_k \\ 0 & \text{caso contrário} \end{cases} \qquad (2)$$

Essa expressão é apresentada em Dorigo *et al.* (1991). Nela, tem-se que:

• p_k (r, s): probabilidade da formiga k mover-se do nó r para o nó s;
• $\tau_{(r,s)}$: quantidade de feromônio associado ao arco (r, s);
• $\eta_{(r,s)}$: atratividade do arco (r, s);
• U: conjunto de arcos não visitados;
• M_k: lista tabu da formiga k;
• α e β: parâmetros de controle da influência de $\tau_{(r,s)}$ e $\eta_{(r,s)}$. Variam na faixa [0, 1].

Na expressão 2, a cada iteração as formigas decidem, de acordo com uma medida probabilística, qual o caminho a seguir. A probabilidade de escolha de um caminho é proporcional ao rastro de feromônio e à atratividade do mesmo - que varia de acordo com o tipo e a modelagem do problema em que o algoritmo está sendo aplicado. Cada formiga tem uma memória que armazena os caminhos percorridos por ela e previne que um caminho seja visitado pela formiga mais de uma vez. Se a formiga já percorreu aquele caminho, a probabilidade de escolha é zero; caso contrário, é positiva. No roteamento de veículos, o parâmetro atratividade é chamado de visibilidade e é calculado como o inverso da distância entre os consumidores ou vértices do arco.

A atualização do feromônio no ACS é feita de duas formas diferentes: global e local. A atualização global é feita para privilegiar apenas os caminhos pertencentes à melhor solução global, ou seja, a cada iteração, após todas as formigas terem gerado suas soluções, apenas a formiga que gerou a melhor solução desde o início do algoritmo irá depositar feromônio. A atualização global do feromônio é feita pela expressão:

$$\tau_{(r,s)} = (1-\rho)\cdot\tau_{(r,s)} + \frac{\rho}{J_{\psi}^{gb}}, \quad \forall\ (r,s) \in \psi^{gb} \qquad (3)$$

na qual:

• $\tau_{(r,s)}$: quantidade de feromônio associado ao arco (r, s);
• ρ: taxa de evaporação, tal que ρ ∈ [0, 1];
• J_{ψ}^{ϕ} : menor distância percorrida, computada desde o início da aplicação do algoritmo;
• ψ^{ϕ} : melhor solução computada desde o início da aplicação do algoritmo.

Segundo Ellabib *et al.* (2003), a fase de evaporação do algoritmo AS é substituída no ACS por uma atualização local do feromônio, como pode ser visto na expressão 4. Cada vez que uma formiga se move de um arco para outro, a quantidade de feromônio associado ao arco é reduzida. O resultado da atualização local é a modificação dinâmica da possibilidade de escolha do caminho pelas formigas. Logo, cada vez que um arco é percorrido por uma formiga, ele se torna ligeiramente menos desejável para as demais.

$$\tau_{(r,s)} = (1-\rho)\cdot\tau_{(r,s)} + \rho\cdot\tau_0 \qquad (4)$$

Para:

• $\tau_{(r,s)}$: quantidade de feromônio associado ao arco (r, s);
• ρ: taxa de evaporação, de modo que $\rho \in [0, 1]$;
• τ_0: quantidade de feromônio inicial em todos os arcos.

De acordo com Gambardella *et al.* (1999), um bom valor inicial para τ_0 é dado por $\tau_0 = \frac{1}{n \cdot J_{\psi}^{h}}$ sendo J_{ψ}^{h} a distância percorrida, calculada pela heurística do Vizinho mais Próximo, introduzida por Flood (1956), e n é o número de nós.

Um procedimento chamado *Daemon Actions* pode ser usado, opcionalmente, para um controle centralizado da tomada de decisões pelo ACS, em situações como ativar um procedimento de busca local ao final de cada iteração ou depositar uma quantidade extra de feromônio sobre a melhor solução. Esses são exemplos de decisões que não podem ser tomadas pelas formigas individualmente.

3. Busca Tabu

A metaheurística Busca Tabu (BT) foi proposta, de forma independente, por dois autores, Glover (1986) e Hansen (1986), e é conhecida como uma das metaheurísticas mais robustas para a resolução de problemas combinatoriais.

A partir de uma solução inicial qualquer, a BT vasculha toda a vizinhança da mesma a procura pelo melhor vizinho. O melhor vizinho encontrado, mesmo que seja uma solução pior que a atual, se torna a nova solução corrente e o processo é repetido até que algum critério de parada seja atingido.

Um problema que surge ao aceitar movimentos de piora e escolher sempre o melhor vizinho é a ciclagem. Um exemplo de ciclagem ocorre quando um ótimo local é atingido, a BT moverá então para seu melhor vizinho, o qual possui uma solução de pior qualidade. Quando este melhor vizinho se tornar a solução corrente e a BT vasculhar a vizinhança a partir do mesmo, o novo melhor vizinho será novamente o ótimo local encontrado na iteração anterior.

O uso de uma estrutura de memória, conhecida como lista tabu, onde são armazenadas as soluções visitadas, evita que a ciclagem ocorra ao proibir que a busca retorne às soluções que constam na lista. Entretanto, o tempo de processamento de todos os dados pertencentes à solução pode tornar o processo demasiado lento ou até mesmo o inviabilizar. Para tornar o processo menos custoso computacionalmente, a lista tabu, geralmente, é feita de forma a armazenar apenas os movimentos efetuados na exploração da vizinhança, reduzindo a quantidade de parâmetros a serem comparados e, consequentemente, o tempo computacional necessário. Uma lista tabu de movimentos armazena os movimentos proibidos (ou tabus) que, quando realizados, podem levar a uma solução já analisada anteriormente.

O número de iterações que um movimento estará proibido de ser realizado é definido pelo tamanho da lista tabu. Uma lista tabu muito grande pode restringir demais a busca, pois ao tornar tabu um grande número de movimentos, o procedimento termina de forma prematura por não existirem movimentos

factíveis a realizar. Por outro lado, uma lista tabu muito pequena aumenta as chances do processo ciclar, pois reduz o número de iterações em que o movimento é proibido de ser realizado.

Por isso, a BT deve manter um rigoroso controle sobre o tamanho da lista tabu e uma forma de se contornar estes problemas é a implementação de uma lista de tamanho variável. Também chamada de lista tabu dinâmica, ela tem seu tamanho alterado periodicamente no decorrer do processo, tentando evitar que a busca fique muito restrita ou cicle.

Apesar de todos os esforços para manter uma lista tabu de tamanho adequado, é possível que um movimento considerado tabu leve a uma solução não analisada ainda. Neste caso, uma função de aspiração pode tornar possível a aceitação de um movimento pertencente à lista tabu, desde que leve a uma solução desconhecida e com bons atributos. O critério de aspiração mais comum na Busca Tabu é baseado na função objetivo, ou seja, o movimento tabu é aceito somente se levar a uma solução melhor do que a melhor solução existente até o momento. Para uma referência mais detalhada sobre Busca Tabu, consultar Glover & Laguna (1997). Aplicações recentes dessa metaheurística ao PRVJT podem ser vistas em Fraga *et al.* (2005) e Costa (2005).

4. Reconexão por Caminhos

A técnica de Reconexão por Caminhos (Path Relinking) foi introduzida por Glover (1996) como uma estratégia de intensificação dos resultados, através da exploração de trajetórias que conectam soluções elite (melhores soluções geradas). A idéia principal do método é unir os bons atributos pertencentes às melhores soluções em uma única solução. Ele funciona gerando e explorando caminhos no espaço de soluções, partindo de uma solução corrente até uma solução elite, através de movimentos que introduzem atributos da solução elite na solução corrente. Reconexão por Caminhos pode ser usada como um processo de intensificação a cada ótimo local encontrado e, de acordo com Glover & Laguna (1997), essa é a forma mais eficiente de utilização do método.

A técnica é aplicada em pares de soluções (s1; s2), sendo s1 a solução corrente, obtida após um procedimento de busca local e s2 uma solução escolhida aleatoriamente de um limitado conjunto de soluções elite, gerado durante a construção das soluções. Cada solução obtida ao final de uma busca local é considerada como uma candidata a ser inserida no conjunto elite, desde que seja melhor que a solução de pior qualidade do conjunto e apresente um percentual mínimo de diferença em relação a cada solução do conjunto elite. Se o conjunto estiver vazio, a solução é simplesmente inserida no conjunto. Se o conjunto elite já possui o número máximo de soluções e a solução corrente é candidata a ser inserida neste conjunto, então esta substitui a solução de pior qualidade.

O método de Reconexão por Caminhos inicia-se computando a diferença de simetria, dada por $\Delta(s1; s2)$, entre duas soluções s1 e s2, resultando em um conjunto de movimentos que deve ser aplicado a uma delas, dita solução inicial, para ”transformar-se” na outra, dita solução guia. A partir da solução inicial, o melhor movimento ainda não executado de $\Delta(s1; s2)$ é aplicado à solução corrente, até que a solução guia seja atingida. A melhor solução encontrada ao longo dessa trajetória é considerada como candidata à inserção no conjunto elite e a melhor solução já encontrada é atualizada. Uma aplicação da técnica de Reconexão por Caminhos ao PRVJT pode ser vista em Ho & Gendreau (2006).

Metodologia

A metodologia Ant-TPR proposta consiste em, inicialmente, definir os movimentos de exploração da vizinhança e tipo de refinamento a serem utilizados. Em seguida, é definida uma função de avaliação, para controle da qualidade das soluções geradas. Por fim, a descrição geral do algoritmo desenvolvido é realizada. Para maiores detalhes sobre a metodologia, consultar Fraga (2006).

Um movimento de realocação consiste em retirar um consumidor de uma rota qualquer e inseri-lo em outra posição, que pode ser tanto dentro da mesma rota quanto em outra rota distinta. Na fase de refinamento foi utilizada a técnica de busca local descida com primeira melhora.

A função de avaliação utilizada é lexicográfica, ou seja, formada por um conjunto de objetivos distintos, classificados segundo um critério de prioridades. Nesta metodologia, dois objetivos são considerados na seguinte ordem de prioridade: primeiro, deseja-se minimizar o número de veículos utilizados e segundo, minimizar a distância total percorrida. Cada nova solução gerada é comparada com a anterior, avaliando-se primeiramente o critério de maior prioridade e, somente caso a avaliação com relação a este critério produza um resultado melhor ou igual à solução anterior, o próximo objetivo é avaliado.

A Reconexão por Caminhos foi implementada da seguinte forma. São inseridos, gradativamente, arcos da solução guia s2 em uma solução corrente s1. Em seguida, procura-se na solução s1, a partir do ponto de inserção, os consumidores pertencentes ao arco incluído e os remove. Por fim, uma heurística de busca local é aplicada, somente a partir do ponto em que o arco foi inserido.

No algoritmo criado, a cada iteração, uma lista restrita de consumidores candidatos (LRCC) é gerada, contendo todos os consumidores (com exceção do depósito) da instância a ser solucionada. Além disso, uma lista tabu para nascimento de novas formigas (LTPNNF) e uma lista tabu geral das formigas (LTGF) (que representa a memória da formiga) são geradas, contendo, inicialmente, apenas o depósito. Um vetor de rotas geral (VRG) é criado e, inicialmente, não contém nenhuma rota. Em seguida, uma formiga é atribuída a um consumidor qualquer, escolhido aleatoriamente dentre os consumidores da LRCC, de modo que o consumidor selecionado nunca pertencerá a LTPNNF. A LTPNNF evita que, em uma futura iteração, uma formiga seja colocada ou em uma posição que não permita a formação de novas rotas ou em uma posição já colocada anteriormente.

Um vetor de rotas para a formiga é então iniciado e, caso o consumidor escolhido já pertença a alguma rota, o vetor rota da formiga (VRF) receberá, desse modo, a rota do consumidor. A formiga escolhe, dentre os consumidores da LRCC, para qual consumidor se mover. Cada formiga possui sua própria lista tabu (LTF) que, inicialmente, é uma cópia da LTGF e para onde serão movidos os consumidores analisados. A função da LTGF é evitar que as formigas visitem consumidores que estejam no meio de rotas, ou seja, que não estejam ou na primeira ou na última posição dos vetores de rotas das formigas. A união entre rotas geradas pelas formigas só é possível quando os consumidores envolvidos estão nas extremidades de cada rota. Quando o consumidor escolhido pela formiga já pertencer a uma rota, o algoritmo selecionará a rota do consumidor escolhido e tentará realizar a união.

A cada tentativa mal-sucedida de movimento da formiga (violação de qualquer uma das restrições existentes), a LTF receberá o consumidor envolvido, para que ela não retorne aos consumidores já analisados. Caso o movimento da formiga de um consumidor para outro res- peite todas as restrições, a LTGF e o VRG são atualizados, e a LTF recebe novamente a LTGF. A formiga então move-se para o último consumidor da rota gerada e continua seus movimentos, até que não existam mais movimentos possíveis, ou seja, até que todos os consumidores pertençam à LTF. A formiga então é eliminada; sua rota gerada fica armazenada no VRG; a LRCC e a LTPNNF são atualizadas de acordo com a LTGF; e uma nova formiga será gerada. Quando não existirem mais consumidores na LRCC, o custo da solução gerado pelo conjunto de formigas é analisado e a matriz de feromônio é atualizada.

Em lugar de gerar um conjunto fixo de soluções por iteração, são geradas novas soluções, até que surja uma solução melhor que a melhor solução da iteração anterior ou até um valor máximo escolhido. A solução encontrada é refinada e, após certo número de iterações, o processo de Busca Tabu é acionado sempre que ocorra uma melhora no valor da função objetivo. Periodicamente, o processo de reconexão por caminhos é solução existente para a solução corrente.

A atuação da metodologia Ant-TPR será demonstrada nas figuras a seguir. A Figura 1 mostra uma instância fictícia do PRVJT com 7 consumidores a ser solucionada. Na Figura 2, uma formiga foi posicionada, de forma aleatória, no consumidor 2. As linhas partindo desse consumidor para cada um dos demais representam suas opções de movimento. Baseado nas equações de escolha de consumidores e respeitando-se as restrições existentes, supõe-se que o movimento executado foi partir do consumidor 2 para o consumidor 3. Nessa posição, a formiga analisa suas novas opções, como visto na Figura 3. Novamente, como pode ser visto na Figura 4, partindo da cidade 3 e supondo que o novo consumidor escolhido seja o consumidor 4, a formiga analisa suas opções de movimento. No momento em que não mais existirem movimentos possíveis para a formiga executar (devido às restrições ou caso todos os consumidores já tenham sido visitados), as ligações ao depósito são feitas e a rota gerada pela formiga será integrada ao conjunto de rotas (FIGURA 4). Por fim, na figura 5 uma nova formiga é aleatoriamente posicionada em um consumidor que não pertença a nenhuma rota, analisa suas opções de movimento e continua o processo, até que não existam mais consumidores fora de rotas.

Logo, uma solução completa e factível é gerada pelo conjunto de formigas e o algoritmo entrará na fase de refinamento (*Daemon actions*). A fase de refinamento quase sempre é capaz de melhorar as soluções geradas pelas formigas.

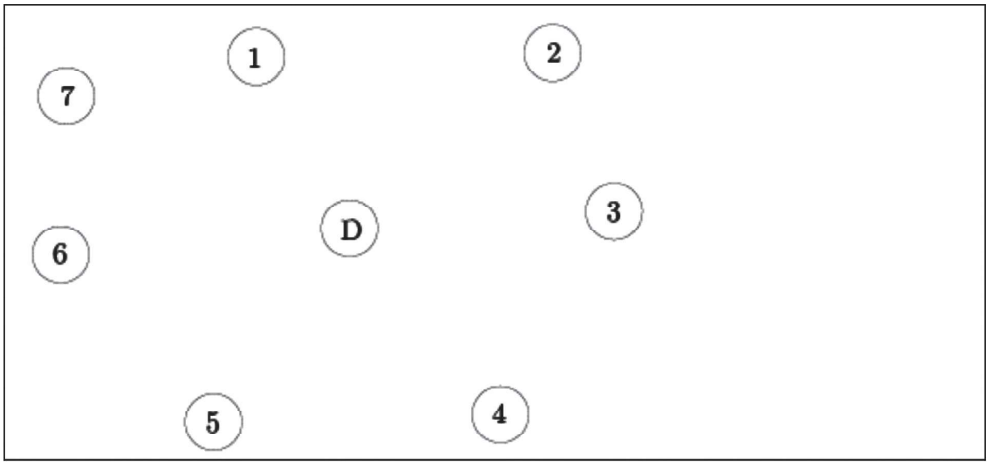

Figura 1 - Instância do PRVJT a ser solucionada.

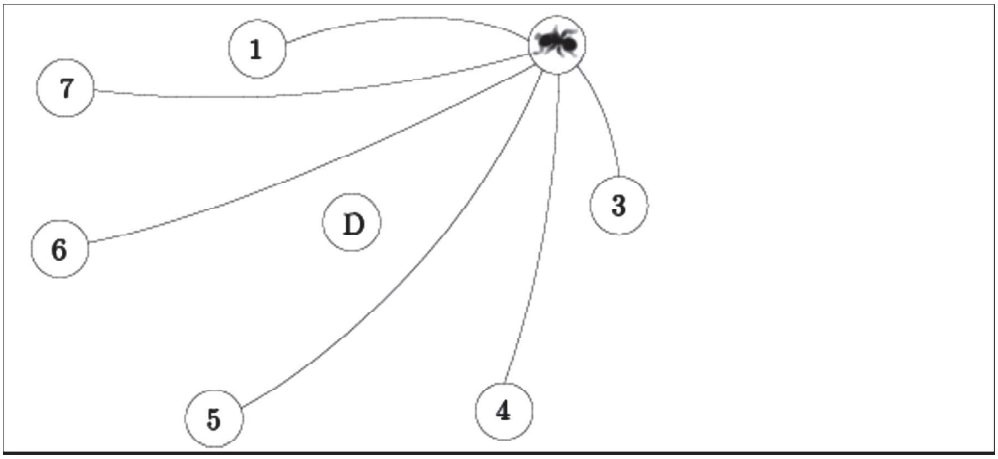

Figura 2 - A formiga analisa suas opções de movimento.

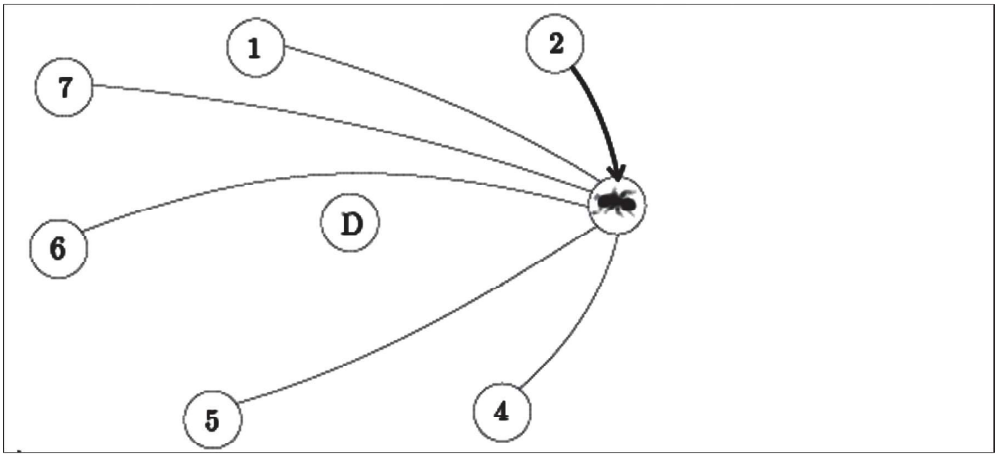

Figura 3 - Após movimentar-se, a formiga analisa suas novas opções de movimento.

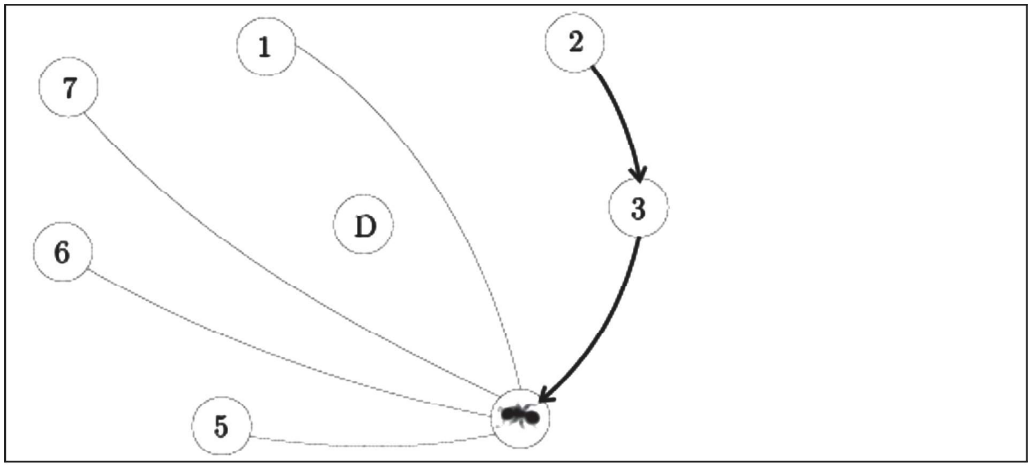

Figura 4 - A formação da rota continua, enquanto gerar soluções factíveis.

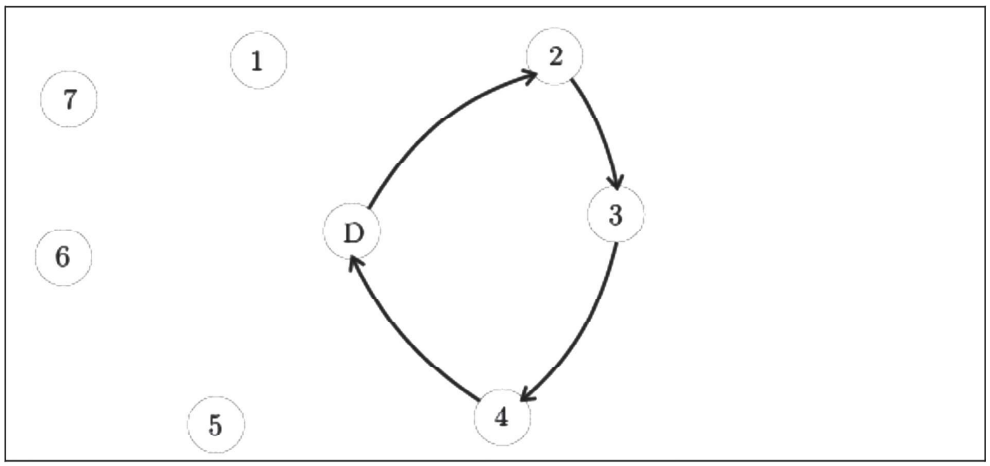

**Figura 5 - Ao final da excursão da formiga,
são adicionadas ligações ao depósito.**

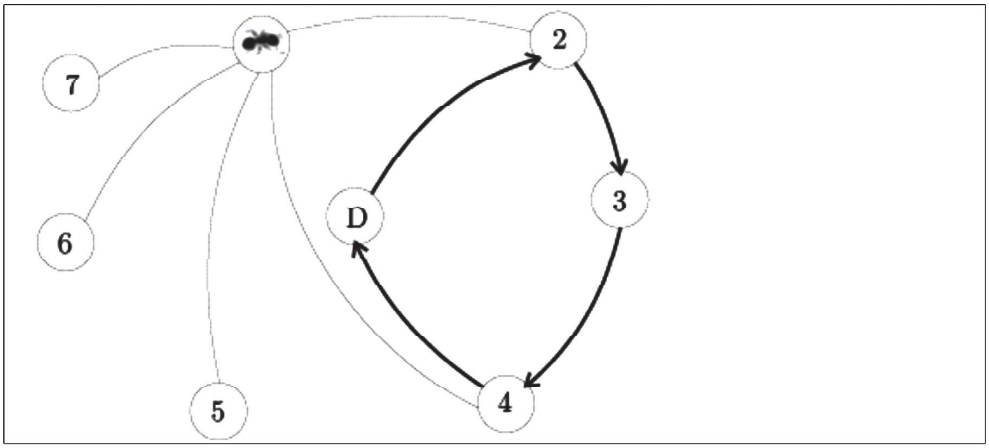

Figura 6 - Uma nova formiga é posicionada e o processo continua.

Resultados computacionais

O algoritmo Ant-TPR foi implementado em linguagem C++, com o compilador Builder 5.0 da Borland, e executado em um computador Intel Core 2 Duo T5550 1.83GHz, com 2Gb de memória RAM, sob plataforma Windows XP Pro. Foram submetidos aos problemas-teste introduzidos por Solomon (1987), que são utilizados como referência de desempenho no PRVJT. O algoritmo Ant-TPR desenvolvido foi avaliado em cinco intervalos de tempo pré-definidos: 100, 600, 1200 e 3600 segundos com a finalidade de conhecer o comportamento da metodologia em diversas situações. Para cada intervalo de tempo, foram feitas dez execuções por instância, cada uma delas partindo de uma semente diferente de números aleatórios.

Nesta revisão, os resultados produzidos pela metodologia Ant-TPR podem ser vistos na Tabela 1 (os resultados dos demais autores foram obtidos de Chen & Ting (2005)), onde (a) é o valor médio do número de veículos da solução; (b) é a média da distância total percorrida; e (c) é o tempo computacional gasto, em segundos; (d) é a soma do número de veículos de todas as soluções; (e) é a distância total percorrida; (f) é o tempo computacional total; (M1) Método antigo de contabilidade do tempo computacional e (M2) método atual. Os valores são referentes às melhores soluções encontradas durante as execuções. Como o hardware utilizado foi diferente entre as metodologias, a comparação direta dos resultados de tempo computacional não é recomendável. Os melhores resultados da metodologia Ant-TPR ficaram distantes a menos de 3% dos melhores da literatura, demonstrando a robustez do método.

Tabela 1 - Comparativo de resultados.

	R1	R2	C1	C2	RC1	RC2	Total
	12,6[a]	3,1	10	3	12,6	3,4	427
TS-P	1294,7[b]	1185,9	861	602,5	1465	1476,1	64679
(Potvin, 1996)	639[c]	722	435	431	586	662	32957
	12,17	2,82	10	3	11,5	3,38	410
TS-T	1209,35	980,27	828,38	589,86	1389,22	1117,44	57953
(Taillard, 1997)	13774	20232	14630	16375	11264	11596	833390
	12	2,73	10	3	11,63	3,25	407
MACS	1217,73	967,75	828,38	589,86	1382,42	1129,19	57525
(Gambardella, 1999)	1800	1800	1800	1800	1800	1800	100800
	11,92	2,73	10	3	11,5	3,25	405
GA	1221,1	975,43	828,48	589,93	1389,89	1159,37	57523
(Berger, 2001)	-	-	-	-	-	-	-
	13,2	5	10,1	3,25	13,5	5	478
HGA	1227	980	861	619	1427	1223	59405
(Chen, 2001)	-	-	-	-	-	-	84000

	12,08	2,91	10	3	11,75	3,25	411
TESA	1215,14	953,43	828,38	589,86	1385,47	1142,48	57467
(Li, 2003)	1474	3882	201	1220	916	2669	100639
	12,25	2,82	10	3	12,13	3,38	416
Ant-TPR (2006)	1211,61	968,77	828,46	589,86	1358,84	1119,93	57201
	2857,2 / 1662,89	3142,1 /2040,8	847,8 / 284,45	606,8 / 166,14	2420,0 / 745,58	2635,3 / 1489,6	121776^{M1} / 64170^{M2}

CONCLUSÕES

Neste trabalho, foi feita uma revisão da metodologia Ant-TPR utilizada para solucionar o Problema de Roteamento de Veículos com Janela de Tempo (PRVJT). O algoritmo desenvolvido é baseado nas metaheurísticas Ant Colony Optimization e Busca Tabu e associado ainda à técnica de Reconexão por Caminhos. Na metodologia desenvolvida, as soluções para o PRVJT são geradas pela metaheurística ACO, refinadas com busca local e, caso ocorram melhoras, também com Busca Tabu. Periodicamente, a técnica de Reconexão por Caminhos é acionada, para intensificar a busca sobre regiões promissoras. O método foi avaliado em 56 instâncias teste, produzindo alguns resultados melhores que os relatados na literatura. O resultado final obtido ficou a menos de 3% dos melhores encontrados na literatura, ressaltando a competitividade da metodologia na resolução do PRVJT.

Antes da revisão e das mudanças na forma de avaliação, o tempo computacional estava acima da média e, levando-se em consideração a diferença na capacidade de processamento entre os hardwares utilizados pelos demais autores, isto tornava a metodologia menos atraente.

Contudo, a análise por um novo ponto de vista na medição do tempo computacional – os resultados não foram mais obtidos a partir da média do tempo computacional utilizado e, sim, a partir do tempo necessário até o encontro da melhor solução.

Com isso, apesar de todo o esforço computacional e dos bons resultados obtidos percebe-se que o uso da metaheurística Colônia de Formigas (ACS) aplicada ao PRVJT, apesar de sua qualidade de adaptação, aprendizado e convergência de resultados não é capaz de competir diretamente com algumas técnicas heurísticas aplicadas ao problema. As formigas, durante a construção de uma solução, necessitam realizar para cada consumidor a ser adicionado à rota, o cálculo e ordenamento de todos os consumidores de acordo com a equação 1. Dependendo do valor de q_0, a probabilidade de escolha de todos os consumidores também deve ser calculada de acordo com a equação 2. Tudo isto demanda um alto custo computacional que acaba por aumentar o tempo necessário para a geração da solução. Levando em consideração que a formação da solução inicial é apenas parte do processo (em seguida entra em ação o *Daemon Actions* – com o refinamento das soluções) observa-se que o uso de Colônia de Formigas aplicado ao PRVJT mesmo com a capacidade de gerar soluções de boa qualidade, possui um alto custo computacional. Comparando com as técnicas de criação e refinamento das soluções LNS (Large Neighborhood Search) e VLNS (Very Large Scale Neighborhood search) vistas em Pisinger & Ropke (2007), é possível perceber que os tempos computacionais limitam o uso da heurísti-

ca ACS em determinadas aplicações. Também de acordo com este último autor, o uso destas técnicas vem demonstrando resultados impressionantes na qualidade e no tempo de obtenção de soluções em problemas de transporte e agendamento de tarefas.

Por fim, foi apresentada, neste trabalho, uma revisão mais detalhada sobre a metodologia Ant-TPR, destacando suas principais qualidades e falhas, desde a escolha das heurísticas utilizadas até os pormenores da implementação. Apesar dos bons resultados obtidos, o tempo computacional ainda pode ser considerado alto quando comparado com outras heurísticas mais adequadas ao problema, sendo necessário mais estudos para aumento da performance das técnicas empregadas.

REFERÊNCIAS

BLUM, C.; ROLI, A.. 2003. *Metaheuristics in combinatorial optimization: Overview and con- ceptual comparison*. ACM Computing Surveys, vol. 35, n. 3, pp. 268–308.

DORIGO, M.; MANIEZZO, V.; COLORNI, A.. 1991. *The ant system: An autocatalytic optimization process*. Technical Report 91-016 Revised. Dipartimento di Elettronica, Politecnico di Milano, Italy.

DORIGO, M.; GAMBARDELLA, L. M.. 1997a. *Ant colonies for the traveling salesman problem.* BioSystems, vol. 43, pp. 73–81.

DORIGO, M.; GAMBARDELLA, L. M.. 1997b. *Ant colony system: A cooperative learning approach to the traveling salesman problem*. IEEE Transactions on Evolutionary Computation, vol. 1, pp. 53–66.

DORIGO, M.; BLUM, C.. 2005. *Ant colony optimization theory: A survey*. Theoretical Computer Science, vol. 344, pp. 243–278.

ELLABIB, I.; BASIR, O. A.; CALAMAI, P.. 2003. *A new ant colony system updating strategy for vehicle routing problem with time windows*. In MIC2003: The Fifth Metaheuristics International Conference, pp. 18–1 – 18–6.

FRAGA, M. C. P.; SOUZA, S. R.; COSTA, T. A.; SOUZA, M. J. F.. 2005.
Algoritmos baseados em grasp e busca tabu para resolução do problema de roteamento de veículos com janela de tempo. In VIII Encontro de Modelagem Computacional. ISBN 85-904971-2-7.

FRAGA, M. C. P.. 2006. *Uma metodologia híbrida Colônia de Formigas - Busca Tabu - Re- conexã o por Caminhos para resolução do Problema de Roteamento de Veículos com Janelas de Tempo.* Dissertação de mestrado, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, MG.

GAMBARDELLA, L. M.; DORIGO, M.. 1996. *Solving symmetric and asymmetric tsps by ant colonies*. In ICEC96 Proceedings of the IEEE Conference on Evolutionary Computation, pp. 622–627. IEEE Press.

GAMBARDELLA, L. M.; TAILLARD, É . D.; AGAZZI, G.. 1999. *Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows*. In D. Corne, M. D. & Glover, F., eds, New Ideas in Optimization, pp. 63–76. McGraw-Hill.

GLOVER, F.. 1986. *Future paths for integer programming and links to artificial intelligence*. vol. 5, pp. 553–549.

GLOVER, F.. 1996. *Tabu search and adaptative memory programming - advances, applications and challenges*. In Barr, R., Helgason, R., & Kennington, J., eds, Interfaces in Computer Sciences and

Operations Research, pp. 1–75. Kluwer Academic Publishers.
GLOVER, F.; LAGUNA, M.. 1997. *Tabu Search*. Kluwer Academic Publishers, Boston.
HANSEN, P.. 1986. *The steepest ascent mildest descent heuristic for combinatorial programming*. In Proceedings of Congress on Numerical Methods in Combinatorial Optimization, Capri, Itália.
HO, S. C.; GENDREAU, M.. 2006. *Path relinking for the vehicle routing problem.* Journal of Heuristics, vol. 12, pp. 55–72.
PISINGER, D.; ROPKE, S. (2007), *A general heuristic for vehicle routing problems.* Computers & Operations Research 34, 2403-2435.
SOLOMON, M. M.. 1987. *Algorithms for vehicle routing and scheduling problems with time windows constraints*. European Journal of Operational Research, vol. 35, pp. 254–266.