UM ALGORITMO EXATO E UM ALGORITMO GENÉTICO PARA O PROBLEMA DE ROTAS DE COBERTURA MULTIVEÍCULO

Cristina Teruko Ota
Faculdade de Ciências Aplicadas, FCA, UNICAMP, Limeira, SP
R. Pedro Zaccaria, 1300 Caixa Postal 1068, CEP 13484-350 - Limeira - São Paulo
cristina.ota@gmail.com

Washington Alves de Oliveira
Faculdade de Ciências Aplicadas, FCA, UNICAMP, Limeira, SP
R. Pedro Zaccaria, 1300 Caixa Postal 1068, CEP 13484-350 - Limeira - São Paulo
washington.oliveira@fca.unicamp.br

Antonio Carlos Moretti
Faculdade de Ciências Aplicadas, FCA, UNICAMP, Limeira, SP
R. Pedro Zaccaria, 1300 Caixa Postal 1068, CEP 13484-350 - Limeira - São Paulo
moretti@ime.unicamp.br

RESUMO

Neste trabalho foi realizado o estudo do Problema de Rota de Cobertura multiveículo, que consiste em encontrar rotas para um conjunto de veículos, de maneira a minimizar a distância total percorrida, considerando um subconjunto de localidades que devem ser visitadas e um subconjunto de localidades que devem ser cobertas (próximas das rotas traçadas pelos m veículos) mas não visitadas. O modelo utilizado neste artigo considera o balanceamento entre rotas, ou seja, a diferença entre a maior e a menor rota deve estar dentro de um intervalo definido. O problema foi resolvido com duas formas de resolução, sendo que para o método exato foi utilizado o *branch-and-cut*, e para a resolução aproximada foi desenvolvido um algoritmo genético. Para os testes foram utilizados dados adaptados dos cenários da biblioteca TSPLIB. Os resultados obtidos pelos métodos são promissores.

PALAVRAS CHAVE. Problemas de rota de cobertura, Algoritmo *branch-and-cut*, Algoritmo genético.

PM-Programação Matemática, OA-Outras aplicações em PO, L&T-Logística e Transportes

ABSTRACT

In this work we studied the multivehicle Covering Tour Problem, which consists in finding routes to a set of vehicles, in order to minimize the total distance traveled, considering a subset of locations that must be visited and another subset that must be covered (close enough by the routes traced by the m vehicles) but not visited. The model used in this article considers the route balancing, that is the difference of the bigger and the smaller route must be in a defined range. The problem was solved in two ways of resolution, to the exact method we used branch-and-cut, and to get an estimated solution we developed an genetic algorithm. To the tests we used an adaptation of the data of instances from TSPLIB. The results obtained by the methods are promising.

KEYWORDS. Covering Tour Problem, Branch-and-Cut Algorithm, Genetic Algorithm.

Mathematical Programming, Others applications in OR, Logistic and Transport

1. Introdução

O problema de rota de cobertura multiveículo (m-PRC) foi introduzido por [Hachicha et al., 2000] como uma generalização do problema de roteamento de veículo (PRV). O m-PRC pode ser definido sobre um grafo $G = (V \cup W, E)$ não direcionado, em que $V = \{v_0, \ldots, v_{n-1}\}$ é o conjunto de vértices que *podem* ser visitados, v_0 é o depósito de onde m veículos idênticos partem e retornam, e $T \subset V$ é um subconjunto de vértices em V que *devem* obrigatoriamente ser visitados; $W = \{w_1, \ldots, w_l\}$ é o conjunto de vértices que *devem* ser cobertos mas não visitados, ou seja, devem estar dentro de um raio de alcance c predeterminado; e $E = \{(v_i, v_j) : v_i, v_j \in V \cup W\}$ é o conjunto de arestas. Cada aresta (v_i, v_j) está associado a uma distância d_{ij}. O objetivo é encontrar um caminho total mínimo percorrido pelos m veículos através das arestas respeitando as seguintes condições: cada rota inicia e termina em v_0; cada vértice de T é visitado exatamente uma vez, por apenas um veículo, enquanto que cada vértice de $V \setminus T$ deve ser visitado no máximo uma vez; todos os vértices de W devem ser cobertos, no sentido de que cada vértice de W deve estar dentro de um raio de cobertura de pelo menos um dos vértices de $V \setminus T$ que são visitados pelas rotas; o número de vértices visitados por cada rota, excluindo o depósito, não deve exceder o valor predeterminado p, que é o número máximo de localidades visitadas por rota; o comprimento de cada rota não deve exceder o valor predeterminado q.

Algumas aplicações para o modelo m-PRC podem ser encontradas na literatura. [Labbé e Laporte, 1986] resolveram o problema de localizar caixas de correios, sendo que cada caixa de correios serve (cobre) um número limitado de endereços (clientes em uma cidade), considerando que o comprimento total das rotas para a coleta das correspondências dos clientes seja mínimo. [Foord, 1995] apresentaram um estudo de atendimento de serviço médico no distrito de West Kiang, Gambia, em que o aumento da cobertura dos cuidados médicos aos pacientes considera vários aspectos logísticos de pessoal e material para definir um raio de cobertura mais adequado. [Oliveira et al., 2015] usaram o m-PRC para desenhar rotas de patrulhamento preventivo para veículos policiais na cidade de Vinhedo, São Paulo, em que a distância de cobertura significa o quanto um agente policial consegue vigiar um local visualmente.

O m-PRC é $\mathcal{NP}$-difícil e a sua resolução com o uso de algoritmos exatos de otimização demanda grande esforço computacional. Apesar de alguns métodos exatos terem sido elaborados para resolver o m-PRC, como em [Hà et al., 2013] e [Hà et al., 2014], eles podem não atingir boas soluções devido ao tempo computacional desfavorável e a falta de memória computacional. Por este motivo, [Hachicha et al., 2000], [Hà et al., 2013], [Hà et al., 2014], [Baldacci et al., 2005] e [Oliveira et al., 2015] propuseram algumas abordagens heurísticas e metaheurísticas para a resolução deste problema.

O principal aspecto desta contribuição é o desenvolvimento de dois métodos de resolução para uma versão balanceada do m-PRC que foi proposta em trabalho anterior [Oliveira et al., 2015], sendo um método exato através de um algoritmo *branch-and-cut* e uma metaheurística utilizando algoritmo genético. Para os testes computacionais foram utilizados dados relativos a biblioteca TSPLIB conforme descrito em [Oliveira et al., 2015]. Os resultados obtidos são comparados com a literatura e mostram-se competitivos.

O restante deste trabalho está organizado como segue. A formulação matemática para a versão balanceada do m-PRC é apresentada na Seção 2. Na Seção 3 são descritas algumas desigualdades válidas para o algoritmo *branch-and-cut*. A Seção 4 detalha o algoritmo genético. A Seção 5 apresenta os resultados computacionais e as conclusões são discutidas na Seção 6.

2. Formulação Matemática

Apresentamos abaixo o m-PRC balanceado, que é uma versão da formulação de [Oliveira et al., 2015] como um modelo de programação inteira de fluxo em redes. Denotamos esta versão por m-PRC-$\rho\tilde{\rho}$, em que ρ e $\tilde{\rho}$ são, respectivamente, os limites superior e inferior para o número de vértices permitidos em cada rota.

O m-PRC-$\rho\tilde{\rho}$ pode ser definido sobre um grafo $G = (V \cup W, E_1 \cup E_2)$, com $V \cup W$ o conjunto de vértices e $E_1 \cup E_2$ o conjunto de arestas. $V = \{v_0, \ldots, v_{n-1}\}$ é o conjunto de n vértices que podem ser visitados e $W = \{w_1, \ldots, w_l\}$ é o conjunto de vértices que devem ser cobertos mas não visitados. $T = \{v_0, \ldots, v_t\}$, $T \subset V$, é o conjunto dos vértices que devem obrigatoriamente ser visitados. O depósito é o vértice v_0, no qual os m veículos partem e retornam. Cada aresta em $E_1 = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ está associada a um custo c_{ij} e cada aresta em $E_2 = \{(v_i, w_j) : v_i \in V \setminus T, w_j \in W\}$ está associada a uma distância d_{ij}. Podemos considerar que existe uma demanda unitária em cada vértice de $V \setminus \{v_0\}$, assim os limites inferior $\tilde{\rho}$ e superior ρ da quantidade de vértices visitados determinam a quantidade mínima e máxima de demanda que pode ser atendida por cada veículo. Logo, o equilíbrio no número de visitas nas diferentes rotas pode ser determinado quando a diferença $r = \rho - \tilde{\rho}$ for pequena. Para o modelo de fluxo em redes com multiveículo é necessário adicionar ao grafo original um vértice artificial v_n (cópia do depósito) para redefinir o m-PRC-$\rho\tilde{\rho}$ sobre o grafo estendido $\bar{G} = (\bar{V} \cup W, \bar{E}_1 \cup E_2)$, em que $\bar{V} = V \cup \{v_n\}$ e $\bar{E}_1 = E_1 \cup \{(v_i, v_n), v_i \in V'\}$, com $V' = \bar{V} \setminus \{v_0, v_n\}$. Definimos também $c_{in} = c_{0i}, \forall v_i \in V'$ e $\lambda_{il} = 1$ se $d_{il} \leq c$ e 0 caso contrário, representando se o vértice $v_i \in V \setminus T$ cobre o vértice $w_l \in W$.

$$\text{Min.} \sum_{\{v_i,v_j\} \in \bar{E}_1} c_{ij} x_{ij} \tag{1}$$

s.a

$$\sum_{v_i \in V \setminus T} \lambda_{il} y_i \geq 1, \quad w_l \in W \tag{2}$$

$$\sum_{v_i \in \bar{V}, i \neq j} x_{ik} + \sum_{v_j \in \bar{V}, j \neq k} x_{kj} = 2y_k, \quad v_k \in V' \tag{3}$$

$$x_{ij} + x_{ji} \leq 1, \quad v_i, v_j \in V', i < j \tag{4}$$

$$\sum_{v_i \in \bar{V}, i \neq k} x_{ik} \leq 1, \quad v_k \in V' \tag{5}$$

$$\sum_{v_j \in \bar{V}, j \neq k} x_{kj} \leq 1, \quad v_k \in V' \tag{6}$$

$$\sum_{v_j \in \bar{V}} (f_{ji} - f_{ij}) = 2y_i, \quad v_i \in V' \tag{7}$$

$$\sum_{v_j \in \bar{V}} f_{0j} = \sum_{v_i \in \bar{V}} y_i \tag{8}$$

$$\sum_{v_j \in \bar{V}} f_{nj} = m\rho \tag{9}$$

$$f_{0j} \geq x_{0j}\tilde{\rho}, \quad \forall v_j \in V' \tag{10}$$

$$f_{ij} + f_{ji} = \rho(x_{ij} + x_{ji}), \quad \{v_i, v_j\} \in \bar{E}_1, i < j \tag{11}$$

$$x_{i0} = 0, \quad v_i \in V' \tag{12}$$

$$x_{nj} = 0, \quad v_j \in V' \tag{13}$$

$$y_i = 1, \quad v_i \in T \setminus \{v_0\} \tag{14}$$

$$f_{ij} \geq 0, f_{ji} \geq 0, \quad \{v_i, v_j\} \in \bar{E}_1 \tag{15}$$

$$x_{ij} \in \{0, 1\}, \quad \bar{E}_1 \tag{16}$$

$$y_i \in \{0, 1\}, \quad v_i \in V'. \tag{17}$$

Neste modelo o número de veículos m é constante, a restrição de comprimento das rotas é relaxada ($q = +\infty$) e o grafo é direcionado. Como conjunto de variáveis, consideramos: x_{ij}

a variável binária que indica se a aresta (v_i, v_j) é utilizado na solução, com valor unitário se a aresta está ativa e zero caso contrário; y_i a variável binária que recebe o valor 1 caso o vértice v_i seja visitado, e 0 caso contrário; f_{ij} e f_{ji} são variáveis de fluxo que medem o fluxo nas arestas na solução factível. A partir do depósito um veículo pode visitar até ρ vértices. Caso o veículo percorra o trecho de v_i a v_j, então o fluxo f_{ij} mede a quantidade de vértices que ainda podem ser visitados, enquanto que f_{ji} representa o número de vértices já visitados. A relação de fluxo de cada aresta é $f_{ji} = \rho - f_{ij}$. O grafo direcionado fornece a direção do fluxo em cada aresta, e restrições adicionais garantem que cada rota inicia o seu percurso no depósito e termina no depósito artificial.

Em (1) o comprimento total das distâncias percorridas pelos m veículos é minimizada. As Restrições (2) garantem a cobertura de todos os vértices de W. As Restrições (3) garantem que caso o vértice v_k seja visitado ($y_k = 1$), então existe fluxo de entrada e saída neste vértice. As Restrições (4) garantem que caso a aresta (v_i, v_j) seja ativa, apenas uma direção é percorrida. As Restrições (5) e (6) garantem que para cada vértice $v_k \in V'$ há no máximo uma aresta ativa na solução chegando e partindo do vértice v_k. Para cada vértice $v_i \in V'$ visitado, as Restrições (7) garantem que a diferença entre o fluxo total que chega em v_i (número de vértices já visitados) e o fluxo total que sai de v_i (número de vértices que ainda podem ser visitados) é 2; caso o vértice não seja visitado, os fluxos associados a este vértice serão nulos. A Restrição (8) define que o fluxo total que sai do depósito é igual ao número total de vértices que são visitados e a Restrição (9) define que o fluxo total que parte do depósito artificial v_n é a capacidade total de todos os veículos. Restrições (10) e (11) garantem uma solução balanceada. Restrições (12) e (13) forçam que não se tenha solução em que uma rota chegue no depósito ou que parta do depósito artificial e Restrições (14) definem que todos os vértices $v_i \in T \setminus \{v_0\}$ serão visitados. As variáveis de decisão pertencem aos conjuntos descritos nas Restrições (15)-(17).

3. Algoritmo *branch-and-cut*

Para obter soluções exatas para o *m*-PRC-$\rho\tilde{\rho}$ foi utilizado um algoritmo *branch-and-cut*, que é um método de otimização para resolver problemas de programação linear inteira.

O *branch-and-cut* é um método iterativo que consiste na combinação do método de plano de corte e do algoritmo *branch-and-bound*. Esta combinação, quando comparado ao *branch-and-bound* puro, pode acelerar a resolução do problema, reduzindo a quantidade de vértices visitados (diminuição do tamanho da árvore de busca), além de melhorar os limites obtidos da programação linear relaxada.

Este método resolve o problema linear inteiro de forma relaxada (sem as restrições de integralidade) e caso a solução ótima não seja inteira, um algoritmo de plano de corte é utilizado de maneira a encontrar desigualdades válidas que sejam satisfeitas pelos pontos factíveis inteiros mas que sejam violadas pela solução fracionária atual. Essas desigualdades são então adicionadas ao problema, de forma a apertar a área da região factível, eliminando soluções que possuam valores fracionários. Com essas desigualdades adicionais, o programa é então reotimizado e o processo continua até que não se tenha solução a ser verificada pelo *branch-and-bound* ou o tempo limite de processamento seja atingido.

[Gendreau et al., 1997] apresentaram algumas desigualdades válidas para o *m*-PRC (Inequações 18 - 20), sendo utilizadas no *m*-PRC-$\rho\tilde{\rho}$.

$$x_{ij} \leq y_i \text{ e } x_{ij} \leq y_j \quad (v_i \text{ ou } v_j \in V \backslash T) \tag{18}$$

$$y_i + y_j \leq 1, \tag{19}$$

se v_i domina v_j ou reciprocamente, com $v_i, v_j \in V \backslash T$.

$$y_i + y_j + y_k \leq 2, \tag{20}$$

se dois de v_i, v_j, v_k dominam o outro vértice ou um destes vértices domina os outros dois ($v_i, v_j, v_k \in V \backslash T$).

As Inequações (18) verificam se, caso uma aresta x_{ij} esteja ativo, seus vértices incidentes v_i e v_j (com v_i ou $v_j \in V \setminus T$) sejam visitados. As Inequações (19) estão relacionadas à dominância de cobertura, ou seja, se o vértice $v_i \in V \setminus T$ domina o vértice $v_j \in V \setminus T$ então apenas o vértice v_i será solução, sendo que um vértice v_i é dito dominante em relação ao vértice v_j quando v_i pode cobrir todos os vértices em W que v_j pode cobrir. Isto garante que apenas o vértice dominante estará na solução. Estendendo a dominância em relação a três vértices (Inequações (20)), caso a junção de dois vértices resultem na dominância de um terceiro vértice, então a solução conterá no máximo 2 desses três vértices; o mesmo ocorre quando um vértice domina os outros dois vértices.

De acordo com [Hà et al., 2013], todas as desigualdades válidas para o conjunto do politopo de cobertura $conv\{y : \sum b_a y_a \geq 1, y_a \in \{0,1\}\}$, com b_a coeficiente binário, são válidas para *m*-PRC-ρ, então estenderemos para *m*-PRC-$\rho\tilde{\rho}$. Então, seja S um subconjunto não nulo de W e o coeficiente α_k^S definido para cada $v_k \in V$, como se segue

$$\alpha_k^S = \begin{cases} 0 & \text{se } \lambda_{kl} = 0 \quad \forall w_l \in S, \\ 2 & \text{se } \lambda_{kl} = 1 \quad \forall w_l \in S, \\ 1 & \text{caso contrário.} \end{cases}$$

Com a definição de α_k^S, outra desigualdade válida é

$$\sum_{v_k \in V} \alpha_k^S y_k \geq 2, \tag{21}$$

sendo que o conjunto S foi limitado a $|S| = 3$ por conta dos esforços computacionais.

[Baldacci et al., 2005] introduziram as desigualdades de fluxo

$$f_{ij} \geq x_{ij}, f_{ji} \geq x_{ij}, \text{ se } i, j \neq v_0 \text{ e } i, j \neq v_n, \tag{22}$$

em que caso uma aresta esteja ativa, pode haver fluxo de carregamento, e se não houver fluxo de carregamento, então a aresta não poderá estar ativa.

[Laporte et al., 1985] propuseram uma restrição de capacidade de *m*-PRC-p e então consideramos a capacidade ρ de *m*-PRC-$\rho\tilde{\rho}$ em

$$\sum_{v_i, v_j \in S} x_{ij} \leq |S| - \left\lceil \frac{|S|}{\rho} \right\rceil \quad (S \subseteq T, |S| \geq 2). \tag{23}$$

4. Algoritmo genético

Os algoritmos genéticos são métodos metaheurísticos, que têm o objetivo de obter soluções subótimas de qualidade com baixo custo computacional na resolução de problemas de grande complexidade. Diferentemente de um método exato, as metaheurísticas não garantem o ótimo global, mas podem obter soluções próximas do ótimo global com menor esforço computacional.

A metaheurística implementada utiliza a rotina Rota Primeiro/ *Cluster* Segundo, o método da varredura e o *Best Cost Route Crossover* (BCRC) ([Ombuki-Berman e Hanshar, 2009]) adaptado.

A rotina Rota Primeiro/*Cluster* Segundo constrói primeiramente uma rota única $R = (h_0, \ldots, h_k)$, com R uma solução 1-PRC factível e então separa esta rota única em m rotas. Neste algoritmo genético, a primeira solução, ou rota gigante, é determinada por uma variação da rotina da varredura.

O método da varredura modificada é aplicado aos vértices em $\overline{T} \cup W$, gerando várias soluções. No caso do nosso problema, $\overline{T} = T \cup N$, sendo N o conjunto obtido na primeira fase da metaheurística. Esta rota inicia com $R = (h_0)$, em que $h_0 = v_0$, e $L = T^* \cup W$, em que $T^* = \overline{T} \setminus \{v_0\}$. Escolhe-se então um nó arbitrário $\bar{h} \in L$ e considera-se uma meia linha de h_0 passando por $\bar{h}$. O conjunto L é gradualmente esvaziado utilizando o critério que varre os vértices $h \in L$ de acordo com a ordem crescente, pelo ângulo $\theta_h = \widehat{\bar{h}h_0h}$ ou no sentido anti-horário.

4.1. Primeira fase

A primeira fase consiste em gerar θ_1 subconjuntos de $V \setminus T$, tal que todos os vértices de W sejam cobertos. Os vértices de cada subconjunto selecionado criam o conjunto $N \subset V$ dos vértices que devem ser visitados. Com os vértices selecionados mais os vértices em T, o problema passa a ser então um PRV, ou seja, todos os vértices de $N \cup T$ devem ser visitados.

Para criar os θ_1 subconjuntos de $V \setminus T$ que cobrem todos os vértices de W é resolvido o seguinte problema de programação linear:

$$\text{Minimizar} \sum_{v_i \in V \setminus T} b_i y_i \tag{24}$$

s.a

$$\sum_{v_i \in V \setminus T} \lambda_{il} y_i \geq 1, \quad w_l \in W \tag{25}$$

$$y_i = \{0, 1\} \qquad \forall v_i \in V \setminus T \tag{26}$$

em que b_i é um número aleatório de $\{1, 2, 3\}$. [Hà et al., 2013] consideraram valores de 1 e 2 para b_i. Estendemos esta classe com o intuito de obter subconjuntos de $V \setminus T$ mais aleatórios.

4.2. Segunda fase

Para cada solução obtida pela primeira fase ($\overline{T} = N \cup T$, em que $N = \{v_i | y_i = 1, v_i \in V \setminus T\}$) é gerada uma rota gigante com os vértices em $\overline{T}$, utilizando-se a rotina da varredura modificada. O subconjunto de vértices em W que são cobertos por $v_i \in V$ é denotado por $C_i = \{j \in W | i \in S_h\}$, em que $S_h = \{l \in N | c_{lh} \leq c\}$. Os critérios para gerar o conjunto R e atualizar L são:

- Se o nó escolhido h pertence a $\overline{T}$, este é simplesmente acrescentado a R e $\{h\} \cup C_h$ é removido de L;
- Se h pertence a W, o nó de S_h que cobre o maior número de vértices ainda não cobertos, assumindo que seja o nó l, então este é acrescentado a R. Então C_l é removido de L.

Dividimos aleatoriamente a rota gigante em m rotas, de até $\theta_2 = m$ formas distintas, mantendo sempre o balanceamento, ou seja, com $\rho - \tilde{\rho} \leq r$.

Após a geração das m rotas selecionam-se as θ_3 melhores soluções da segunda fase para cada conjunto $\overline{T}$ da primeira fase como indivíduos da população inicial do algoritmo genético. Assim, a população do algoritmo genético contém $\theta_1 \times \theta_3$ indivíduos. Nesta etapa, temos então uma população inicial $P = \{p_{1N_1}, \ldots, p_{\theta_3 N_1}, \ldots, p_{1N_{\theta_1}}, \ldots, p_{\theta_3 N_{\theta_1}}\}$, em que p_{1N_1} representa o trajeto do indivíduo 1 com os genes obtidos pela resolução N_1 da primeira fase.

4.3. Terceira fase

Formada a população inicial P, é realizada a terceira fase com o objetivo de melhorar a solução. O primeiro passo consiste no elitismo, selecionando os 30% melhores indivíduos de cada tipo de amostra da população pai, ou seja, para cada subconjunto $\overline{T}$ da primeira fase, são selecionados os 30% melhores indivíduos para a nova população.

Após o elitismo, são selecionados dois indivíduos da população anterior, do mesmo conjunto $\overline{T}$, nos quais é realizado o *Best Cost Route Crossover* (BCRC) desenvolvido por [Ombuki et al., 2006].

Na Figura 1 temos um exemplo de possível BCRC entre dois indivíduos escolhidos aleatoriamente de um mesmo conjunto $\overline{T}$. Neste exemplo, a última rota do primeiro indivíduo e a segunda rota do segundo indivíduo foram selecionadas. Estas rotas são então retiradas do outro indivíduo, ou seja, os vértices 7 e 5 são removidos do indivíduo p_2 e os vértices 9 e 2 são removidos de p_1. Após a retirada dos vértices, verifica-se em qual posição há menor custo, quando inserido

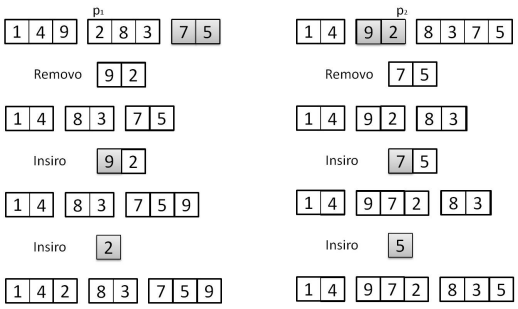

Figura 1: Exemplo de operador de melhoria *Best Cost Route Crossover* (Ombuki et al. [2006])

cada um dos vértices retirados anteriormente. Neste processo não há garantia do balanceamento entre as rotas, que será realizada no próximo passo.

Após o BCRC verifica-se a quantidade de vértices em cada rota. Caso a diferença entre a maior e a menor rota seja maior que r, então é realizada a mutação, em que um vértice da maior rota é retirada e incluída em uma das rotas que possui menor quantidade de vértices. Da maior rota, verificamos qual vértice possui maior custo, considerando-se as arestas ativas. E na menor rota verificamos em qual posição a inclusão do vértice que será deslocado resulta em menor custo, com a alteração das arestas ativas.

Repete-se o BCRC e a mutação até que a melhor solução de pelo menos γ gerações consecutivas não apresente melhoria. Assim, o número de gerações depende deste critério de parada ou do limite do tempo de processamento, o que ocorrer antes.

5. Experimentos Computacionais

Para os testes computacionais foram utilizados dados relativos a biblioteca TSPLIB, utilizando os mesmos valores de r e m e as mesmas localidades como sendo o depósito, apresentados em [Oliveira et al., 2015]. Para a resolução do método exato, foram utilizados valores de ρ e $\tilde{\rho}$ do melhor resultado obtido pela metaheurística implementada. Tanto a formulação do método exato quanto o algoritmo genético, foram implementados em AIMMS® versão 4.28.1, com o *solver* CPLEX versão 12.6.3.

Nesta seção serão apresentados os resultados obtidos pelo método exato e pelo algoritmo genético. Os algoritmos foram rodados em CPU de 2.30GHz com 4 GB de RAM. O tempo de execução do algoritmo *branch-and-cut* e da metaheurística foi limitado a uma hora para cada exemplar.

Os dados utilizados para os testes da nova metaheurística são baseados no TSPLIB. Cada classe é definida por um total de $V \cup W$ vértices. Os dados kroA100, kroB100, kroC100, kroD100 e kroE100 são utilizados para criar conjuntos de $|V| = 50$ e $|W| = 50$, enquanto que kroA150 e kroB150 criam conjuntos de $|V| = 50$ e $|W| = 100$. Cada classe é subdividida em três subclasses, com diferentes cardinalidades para T, em que a primeira subclasse tem $|T| = \lceil |V|/8 \rceil$, a segunda subclasse tem $|T| = \lceil |V|/4 \rceil$ e a última subclasse tem $|T| = \lceil |V|/2 \rceil$. Dos vértices utilizados em cada classe, temos que os primeiros correspondem aos vértices $T \backslash \{v_0\}$, seguidos dos vértices $V \backslash T$ e os últimos são os vértices em W. O número de rotas m varia de 2 a 4 e a diferença entre os comprimentos das rotas não pode exceder $r = \rho - \tilde{\rho}$, sendo $r = 2$. Os exemplares utilizados são denominados por $C\_SC\_TSPLIB\_m$, sendo que C representa a classe, SC a subclasse, $TSPLIB$ o nome do exemplar de TSPLIB e m o número de rotas. Como exemplo, temos 100_2_4_2, que corresponde à Classe 1, subclasse 2, exemplar *kroD100* e $m = 2$.

O raio de cobertura dos vértices em $V \setminus T$ que cobrem os vértices em W é representado pela constante $c = \max\{\max_{w_l \in W}\{d_{l,h(l)}\}, \max_{v_h \in V \setminus T} \min_{w_l \in W}\{d_{lh}\}\}$, em que $h(l)$ é o índice do segundo vértice de $V \setminus T$ mais próximo de w_l ([Hachicha et al., 2000]). Esta regra garante que cada vértice de $V \setminus T$ cubra pelo menos um vértice de W e cada vértice de W possa ser coberto por pelo menos dois vértices de $V \setminus T$. Neste caso, $v_k \in V$ cobre $w_l \in W$ quando $d_{kl} \leq c$.

Para reduzir o tamanho do problema de cobertura realizamos um pré-processamento, em que não consideramos todos os vértices em W e em $V \setminus T$, como proposto por [Hachicha et al., 2000], como: (i) cada vértice w_l com $\lambda_{lh} = 1 \; \forall v_h \in V \setminus T$ é eliminado; (ii) dados dois vértices $w_h, w_e \in W$, com $w_h \neq w_e$ e $\lambda_{eh} = \lambda_{lh} \; \forall v_h \in V \setminus T$, um deles é eliminado; (iii) dados $w_i, w_j \in W$, se $\lambda_{ih} \leq \lambda_{jh}, \forall v_h \in V \setminus T$ então w_i é dominada por w_j e é eliminada; (iv) se um vértice $v_h \in V \setminus T$ não cobre nenhum vértice $w_l \in W$ devido a uma redução anterior, então este vértice também é eliminado.

No algoritmo *branch-and-cut* limitamos o subconjunto S a $|S| = 3$ nas Inequações (23). No algoritmo genético cada população possui $\theta_1 \times \theta_3$ indivíduos, o número de gerações depende do critério de parada (limite do tempo de processamento ou até que a melhor solução de pelo menos γ gerações consecutivas não apresente melhoria), as mutações ocorrem sempre que a diferença da maior rota e da menor rota de cada novo indivíduo for maior que r e o BCRC ocorre em 70% dos $\theta_1 \times \theta_3$ indivíduos de cada geração, sendo que o restante é definido por elitismo.

Um critério de parada para ambos os métodos é o tempo de processamento limitado em uma hora. Na configuração "AIMMS *Options*" é possível alterarmos os parâmetros do *solver* CPLEX. Para a resolução do método exato foram realizadas as seguintes alterações dos parâmetros do *solver*: *MIP search strategy*: "*apply branch-and-cut*"; *MIP start algorithm*: "*barrier*"; *cover cuts*: "*Generate cuts very aggressively*"; *Gomory cuts*: "*Generate cuts aggressively*" e *Implied bound cuts*: "*Generate cuts moderately*".

Os valores de ρ e $\tilde{\rho}$ utilizados no método exato foram obtidos a partir da melhor solução obtida pelo algoritmo genético. Na Tabela 2 temos o valor final de r de cada exemplar, ou seja, a diferença entre ρ e $\tilde{\rho}$, considerando o valor máximo de r para cada grupo de dados, definidos anteriormente.

O GAP do *branch-and-cut* (GAP_{BC}), na Tabela 1, pode ser obtido pela Equação (27).

$$GAP_{BC} := \frac{|\text{solução linear} - \text{melhor solução objetiva}|}{\text{melhor solução objetiva} + 1} * 100. \tag{27}$$

Já o GAP do valor obtido com o algoritmo genético em relação ao obtido no problema exato, é apresentado na Tabela 2. Seja C_{AG} o valor da solução do algoritmo genético e C_{BC} o valor da solução do *branch-and-cut*, então GAP_{AG} é o desvio percentual do algoritmo genético e é computado como na Equação (28).

$$GAP_{AG} = \frac{|C_{AG} - C_{BC}|}{C_{BC}} * 100. \tag{28}$$

A Tabela 1 apresenta dois blocos: os resultados de tempo de processamento, custo total de todas as rotas, número de vértices na árvore de busca e iterações e GAP e informação sobre os cortes utilizados no modelo, ou seja, o número total de desigualdades válidas referentes a arestas incidentes (Inc) (Inequações (18)), dominância (Dom) (Inequações (19) e (20)), fluxo (Inequações (22)) e capacidade (Capac) (Inequações (23)). As Inequações (21) não apresentaram grande alteração nos cortes de plano do modelo, logo não foram consideradas nos testes. Nesta tabela, percebeu-se que todos os exemplares puderam ser resolvidas com o algoritmo *branch-and-cut* utilizando-se ρ e $\tilde{\rho}$ obtidos do algoritmo genético.

A Tabela 2 apresenta informações relativas ao algoritmo genético, com o melhor custo, o custo médio de todas as soluções obtidas, o desvio da melhor solução com o custo médio (ΔCusto), a diferença da maior e menor rota da melhor solução (r), o GAP do algoritmo genético em relação ao *branch-and-cut*, o número de gerações, quantidade de mutações realizadas e a quantidade de iterações do BCRC.

Observamos pela Tabela 1 que conforme o conjunto T aumenta, o número de restrições relativas a capacidade aumentam também. Os custos que estão destacados apresentaram valor inferior ao obtido com o algoritmo genético.

Tabela 1: Resultados computacionais do algoritmo *branch-and-cut* nos exemplares com $|V| + |W| = 100$ e $|V| + |W| = 150$

Exemplar	Tempo(s)	Custo	Vértices	Iterações	$GAP_{BC}(\%)$	Inc	Dom	Fluxo	Capac
100-1-1-2	383.0	**9283**	1199	96081	**0.00**	558	134	1648	25
100-1-1-3	3600.7	**11706**	4752	642972	5.41	2172	908	11076	25
100-1-1-4	3600.1	**13205**	4011	484008	6.93	1395	1620	14844	21
100-1-2-2	65.9	**9710**	196	34335	**0.00**	131	23	524	17
100-1-2-3	3600.3	13651	4560	811635	13.46	1812	479	8876	25
100-1-2-4	2957.5	**14280**	5072	393832	**0.00**	1098	171	7062	25
100-1-3-2	30.3	**9453**	74	9590	**0.00**	94	12	212	21
100-1-3-3	3601.1	**10235**	4400	599555	5.81	2526	436	9754	21
100-1-3-4	1919.5	**11286**	4134	283013	**0.00**	1769	281	4224	25
100-1-4-2	3600.5	**9880**	4657	830412	3.74	2023	288	10786	21
100-1-4-3	3601.1	**12815**	4127	940790	12.75	2594	215	12240	21
100-1-4-4	3600.6	**14089**	4591	821306	10.06	2799	364	11716	21
100-1-5-2	3645.5	**9557**	0	3505	15.38	65	0	18624	25
100-1-5-3	2010.5	**10584**	4240	417740	**0.00**	1463	412	7444	25
100-1-5-4	3600.4	**12397**	4897	397176	4.23	1835	444	13498	25
100-2-1-2	**1.8**	**10634**	0	581	**0.00**	1	0	28	55
100-2-1-3	3603.5	**13113**	5843	874633	9.24	1294	1217	5064	121
100-2-1-4	3600.9	**14416**	4245	661052	6.42	3	0	64	121
100-2-2-2	3600.5	**11455**	4622	857007	15.59	1	0	54	120
100-2-2-3	3601.2	**12100**	6054	688676	3.43	1380	291	11312	111
100-2-2-4	19.1	16785	0	7377	0.00	2	1	82	121
100-2-3-2	31.4	**12386**	74	8946	**0.00**	20	2	111	121
100-2-3-3	3601.2	**14286**	4220	701228	9.01	1714	510	13788	121
100-2-3-4	3600.6	**16759**	5531	628103	3.25	601	1650	12148	121
100-2-4-2	107.3	**11100**	595	62853	**0.00**	78	35	415	121
100-2-4-3	3613.2	13653	3203	289377	7.26	270	261	4724	121
100-2-4-4	3600.5	**14600**	5408	683988	6.24	938	421	5082	121
100-2-5-2	3601.0	**10909**	5620	880413	4.61	785	396	3781	121
100-2-5-3	3600.5	**13478**	5654	1196262	10.26	1431	845	7376	121
100-2-5-4	3600.4	**16334**	4241	713888	19.35	1293	496	6714	121
100-3-1-2	3600.3	**12893**	4911	680263	5.92	530	495	15916	528
100-3-1-3	3600.2	**14942**	4210	492485	9.98	697	588	12728	529
100-3-1-4	3600.2	**17309**	4118	743524	8.80	466	249	12784	529
100-3-2-2	1762.0	**12863**	3956	355348	**0.00**	135	94	5670	511
100-3-2-3	3600.3	**17535**	3767	702903	18.06	352	383	12696	528
100-3-2-4	3600.1	**17570**	4160	656260	8.04	530	495	15916	528
100-3-3-2	3600.5	**15111**	4670	622843	6.48	442	161	11228	526
100-3-3-3	3600.3	**17563**	4221	616726	11.41	419	199	14442	529
100-3-3-4	3603.7	26879	0	0	0.00	0	0	0	0
100-3-4-2	3600.5	**13555**	5541	693913	5.73	216	213	10800	529
100-3-4-3	3600.9	**17012**	3996	684557	16.79	284	361	11980	529
100-3-4-4	3600.5	**17886**	4317	617173	10.46	277	336	11882	529
100-3-5-2	**10.2**	**12859**	15	3920	**0.00**	5	3	126	514
100-3-5-3	3600.7	**18476**	4439	768891	22.98	212	179	13120	529
100-3-5-4	3600.1	**19760**	1645	406107	18.55	187	97	2917	529
150-1-1-2	3600.0	**9683**	6071	525659	**0.49**	3285	2661	9334	25
150-1-1-3	3601.2	**11407**	6414	535797	2.62	2933	1487	8048	21
150-1-1-4	2789.6	**13226**	5803	429192	**0.00**	2199	507	6412	21
150-1-2-2	3600.3	**8859**	5600	984061	6.23	3849	362	12112	21
150-1-2-3	3600.3	**10162**	4747	645439	5.98	3067	404	9756	21
150-1-2-4	3601.2	**10978**	4898	799661	4.75	3885	848	11240	21
150-2-1-2	283.3	**10328**	1269	64583	**0.00**	216	57	1468	121
150-2-1-3	3600.9	**13051**	5243	984860	7.61	1813	866	12210	121
150-2-1-4	3600.7	**14402**	5076	489526	5.90	666	411	9568	121
150-2-2-2	153.3	**12216**	164	33004	**0.00**	35	10	432	121
150-2-2-3	3603.7	14118	4836	684765	8.34	1614	416	13924	121
150-2-2-4	3600.0	**16570**	2428	241793	2.79	489	808	4302	121
150-3-1-2	3601.2	**13225**	6073	605179	5.51	79	250	8450	513
150-3-1-3	3600.2	**15334**	4958	863483	10.76	336	420	12326	529
150-3-1-4	3600.4	**18266**	4258	814805	15.15	445	815	13844	529
150-3-2-2	3600.9	**14668**	6600	740482	1.72	295	91	10150	520
150-3-2-3	3602.5	**17156**	5354	789197	6.06	266	78	13954	529
150-3-2-4	3600.2	**18934**	4686	752682	8.28	335	92	14668	529

Tabela 2: Comparação entre o algoritmo *branch-and-cut* e o algoritmo genético nos exemplares com $|V| + |W| = 100$ e $|V| + |W| = 150$

	Branch-and-cut			Algoritmo genético								
Exemplar	**Tempo(s)**	**Custo**	**r**	**Tempo(s)**	**Custo**	**Custo médio**	**ΔCusto**	**r**	GAP_{AG}**(%)**	**Geração**	**Mutação**	**Iteração**
100-1-1-2	383	**9283**	1	131	9356	16816	1.80	1	0.78	10	643	245
100-1-1-3	3601	**11706**	1	210	11763	16060	1.37	1	0.49	9	1031	617
100-1-1-4	3600	**13205**	1	387	13726	17235	1.26	2	3.95	29	3077	1887
100-1-2-2	66	**9710**	0	147	10313	18278	1.77	2	6.22	15	260	91
100-1-2-3	3600	13651	0	230	**12743**	16672	1.31	2	**6.65**	23	2952	1634
100-1-2-4	2957	**14280**	0	357	14618	17753	1.21	1	2.36	12	1571	1016
100-1-3-2	30	9453	0	104	**9315**	14086	1.51	1	**1.45**	13	1392	554
100-1-3-3	3601	**10235**	1	192	10590	14550	1.37	2	3.47	18	1964	1052
100-1-3-4	1919	**11286**	1	290	11581	15186	1.31	2	2.61	23	1818	1249
100-1-4-2	3600	**9880**	1	195	10673	17357	1.63	1	8.02	18	3537	1115
100-1-4-3	3601	**12815**	0	345	13056	17787	1.36	2	1.88	11	2445	1105
100-1-4-4	3601	**14089**	1	598	15186	19423	1.28	1	7.79	48	9233	4882
100-1-5-2	3646	**9557**	1	144	10029	17225	1.72	2	4.94	15	1738	756
100-1-5-3	2011	**10584**	2	282	11629	17736	1.53	2	9.87	26	3568	1569
100-1-5-4	3600	**12397**	2	365	14319	18520	1.29	2	15.50	17	1959	1264
100-2-1-2	2	**10634**	0	234	10977	20090	1.83	0	3.23	22	1656	425
100-2-1-3	3603	**13113**	2	457	14229	18275	1.28	1	8.51	13	2839	1260
100-2-1-4	3601	14416	0	696	**14286**	18742	1.31	2	**0.90**	14	2568	1225
100-2-2-2	3600	**11455**	0	289	12257	19913	1.62	2	7.00	29	9218	2724
100-2-2-3	3601	**12100**	1	440	13575	17228	1.27	2	12.19	22	4467	1539
100-2-2-4	19	16785	0	669	**14458**	21297	1.47	2	**13.86**	10	1957	941
100-2-3-2	31	**12386**	0	239	13327	21940	1.65	0	7.59	12	2506	614
100-2-3-3	3601	**14286**	1	457	15678	21001	1.34	1	9.75	18	3784	1646
100-2-3-4	3601	16759	0	727	**16521**	21593	1.31	1	**1.42**	11	2037	973
100-2-4-2	107	**11100**	2	284	12139	18360	1.51	2	9.36	12	2659	817
100-2-4-3	3613	**13653**	0	561	14215	19414	1.37	2	4.12	15	3913	1467
100-2-4-4	3601	**14600**	2	898	16244	20885	1.29	2	11.26	23	5759	2396
100-2-5-2	3601	**10909**	1	290	13196	21379	1.62	1	20.96	16	4684	1416
100-2-5-3	3600	**13478**	1	468	14525	21691	1.49	2	7.77	9	2403	744
100-2-5-4	3600	16334	2	806	**15284**	21691	1.42	2	**6.43**	36	7113	3142
100-3-1-2	3600	**12893**	2	1045	18970	27942	1.47	1	47.14	253	16962	3572
100-3-1-3	3600	**14942**	2	2172	23935	28044	1.17	1	60.19	1226	65315	18214
100-3-1-4	3600	**17309**	1	2520	26110	28485	1.09	2	50.85	455	19437	6550
100-3-2-2	1762	**12863**	1	1198	21079	28417	1.35	1	63.87	687	50632	9674
100-3-2-3	3600	**17535**	0	1875	22196	27877	1.26	1	26.58	312	18729	4978
100-3-2-4	3600	**17570**	2	2730	25072	28699	1.14	2	42.70	453	22628	6871
100-3-3-2	3601	**15111**	1	2294	21351	28667	1.34	1	41.30	4356	291103	61710
100-3-3-3	3600	**17563**	0	1821	24873	29126	1.17	2	41.62	270	14895	4193
100-3-3-4	3604	26879	0	3355	**24557**	30500	1.24	1	**8.64**	2792	127373	41466
100-3-4-2	3601	**13555**	0	1611	18702	29045	1.55	1	37.97	1054	76995	13782
100-3-4-3	3601	**17012**	1	2036	22285	28315	1.27	2	30.99	239	14711	3854
100-3-4-4	3601	**17886**	2	2895	23255	29961	1.29	2	30.02	168	8877	2585
100-3-5-2	10	**12859**	0	1168	20708	29814	1.44	2	61.03	27	1899	370
100-3-5-3	3601	**18476**	0	3371	23737	30441	1.28	2	28.48	4052	219683	61017
100-3-5-4	3600	**19760**	0	2288	25039	32012	1.28	2	26.72	1449	67531	21444
150-1-1-2	3600	**9683**	1	258	10103	15164	1.50	2	4.33	12	749	346
150-1-1-3	3601	**11407**	1	507	12427	16691	1.34	2	8.94	34	3205	1785
150-1-1-4	2790	**13226**	1	692	13294	17779	1.34	2	0.51	12	1014	689
150-1-2-2	3600	**8859**	1	191	9581	13433	1.40	1	8.15	13	1126	474
150-1-2-3	3600	**10162**	1	332	10605	13957	1.32	1	4.36	22	1684	991
150-1-2-4	3601	**10978**	2	542	11912	14491	1.22	2	8.51	30	1446	963
150-2-1-2	283	**10328**	0	531	11268	21069	1.87	1	9.10	22	1112	278
150-2-1-3	3601	**13051**	1	868	14790	20299	1.37	1	13.32	12	2971	1194
150-2-1-4	3601	**14402**	1	1239	14705	18831	1.28	2	2.10	34	6372	2803
150-2-2-2	153	**12216**	0	471	12840	21509	1.68	1	5.11	28	5601	1397

Pela Tabela 2 temos que o valor de GAP do valor do custo obtido pelo algoritmo genético em relação ao método exato variou de 0.49% (solução do algoritmo genético foi melhor em relação ao *branch-and-cut*) a 63.87% no pior caso.

Nos gráficos apresentados nas Figuras 2 - 6 temos a média dos exemplares com as mesmas classes e subclasses. Observando-se a Figura 2, percebemos uma diferença de tempo de processa-

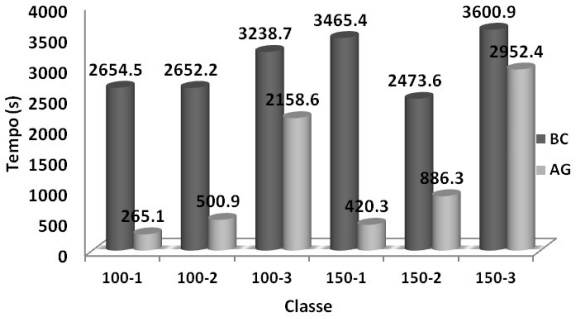

Figura 2: Tempo de processamento dos métodos exato e algoritmo genético

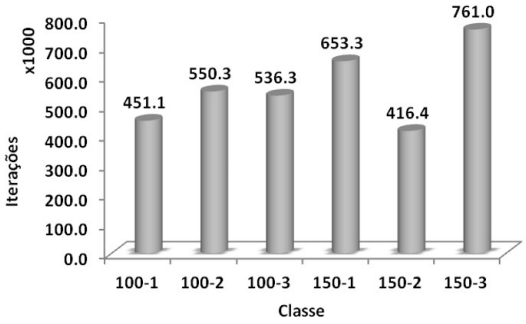

Figura 3: Número de iterações do modelo exato

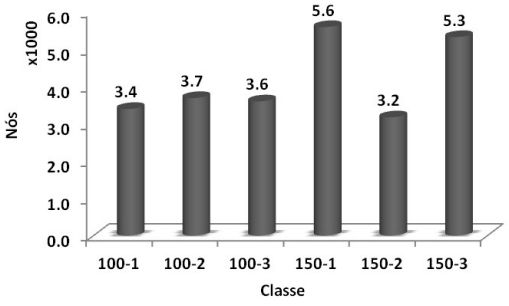

Figura 4: Número de nós da árvore de busca do BC

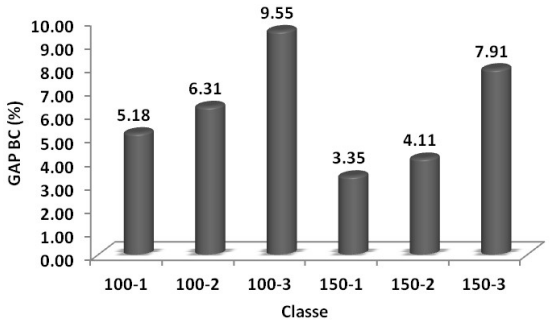

Figura 5: GAP do custo do modelo exato

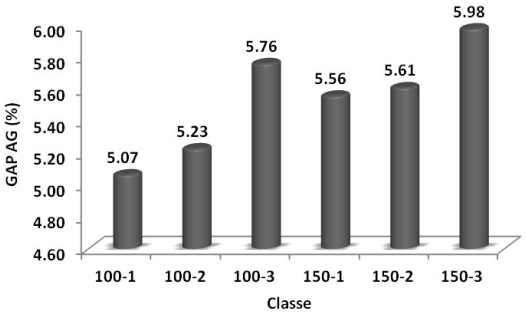

Figura 6: GAP do AG em relação ao BC

mento do *branch-and-cut* e do algoritmo genético, principalmente para os menores exemplares, sendo um fato já esperado. Percebemos que quando aumentamos $|T|$, temos uma tendência de aumento no tempo de processamento (Figura 2). Os gráficos de número de iterações (Figura 3) e número de nós na árvore de busca (Figura 4) no método exato apresentam comportamentos semelhantes. Pelas Figuras 5 e 6, temos que a média do GAP do método exato em relação ao problema relaxado, aumenta conforme aumenta a subclasse, ocorrendo o mesmo com o GAP do algoritmo genético em relação ao método exato, apresentando também aumento do GAP conforme o aumento da quantidade de vértices.

6. Conclusão

Os dois métodos propostos são capazes de retornar boas soluções para o m-PRC, sendo que o *branch-and-cut* resolve, dentro do tempo limite de execução, 25% dos exemplares na otimalidade. Enquanto que o algoritmo genético apresentou 8% dos exemplares com melhor custo em relação ao método exato (além do tempo limite de execução). Para os exemplares resolvidos na otimalidade pelo algoritmo *branch-and-cut*, a média do GAP_{AG} em relação ao algoritmo genético foi

de 12.85%. Esse resultado mostra que o algoritmo genético desenvolvido é promissor. Além disso, acreditamos que a inclusão de novos cortes válidos pode acelerar e melhorar a nossa abordagem *branch-and-cut*, uma vez que a versão atual já consegue resolver exemplares com até 150 vértices.

Para manter o modelo matemático linear utilizamos ρ e $\tilde{\rho}$ como parâmetros, e seus valores foram obtidos das soluções do algoritmo genético. No entanto, outra versão para o *branch-and-cut* está sendo estudada para o caso em que ρ e $\tilde{\rho}$ são variáveis do problema. Neste caso, é necessário fazer uma linearização do modelo matemático para aplicar o algoritmo.

Agradecimentos

Os autores agradecem o apoio financeiro da FAPESP e FAEPEX-UNICAMP.

Referências

Baldacci, R., Boschetti, M. A., Maniezzo, V., e Zamboni, M. (2005). *Scatter Search Methods for the Covering Tour Problem*, p. 59–91. Springer US, Boston, MA.

Foord, F. (1995). Gambia: evaluation of the mobile health care service in west kiang district. *World Health Stat Q*, 48(1):18–22. ISSN 0379-8070.

Gendreau, M., Laporte, G., e Semet, F. (1997). The covering tour problem. *Operations Research*, 45(4):568–576.

Hà, M. H., Bostel, N., Langevin, A., e Rousseau, L.-M. (2013). An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *European Journal of Operational Research*, 226(2):211–220.

Hà, M. H., Bostel, N., Langevin, A., e Rousseau, L.-M. (2014). An exact algorithm and a metaheuristic for the generalized vehicle routing problem with flexible fleet size. *Comput. Oper. Res.*, 43: 9–19.

Hachicha, M., Hodgson, M. J., Laporte, G., e Semet, F. (2000). Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, 27(1):29 – 42.

Labbé, M. e Laporte, G. (1986). Maximizing user convenience and postal service efficiency in post box location. *Belgian Journal of Operations Research, Statistics and Computer Science*, 26: 21–35. Language of publication: en.

Laporte, G., Nobert, Y., e Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5):1050–1073.

Oliveira, W. A., Moretti, A. C., e Reis, E. F. (2015). The multi-vehicle covering tour problem: building routes for urban patrolling. *Brazilian Operations Research Society*, 35(3):617–644.

Ombuki, B., Ross, B. J., e Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24(1):17–30. ISSN 1573-7497. URL `http://dx.doi.org/10.1007/s10489-006-6926-z`.

Ombuki-Berman, B. e Hanshar, F. T. (2009). *Using Genetic Algorithms for Multi-depot Vehicle Routing*, p. 77–99. Springer Berlin Heidelberg, Berlin, Heidelberg.