Algoritmo Genético para o Problema da Árvore Geradora Generalizada de Custo Mínimo

Ernando Gomes de Sousa
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão - Campus S. R. das Mangabeiras
BR-230, Km 319, Zona Rural, CEP: 65840-000, São Raimundo das Mangabeiras - MA - Brasil
ernando.sousa@ifma.edu.br

Rafael Castro de Andrade
Departamento de Estatística e Matemática Aplicada - Universidade Federal do Ceará
Campus do Pici, Bloco 910. CEP 60.455-900 - Fortaleza, Ceará - Brasil
rca@lia.ufc.br

Andréa Cynthia Santos
ICD-LOSI, UMR CNRS 6281, Université de Technologie de Troyes
12, rue Marie Curie, CS 42060, 10004, Troyes Cedex, France
andrea.duhamel@utt.fr

RESUMO

Dado um grafo m-partido $G = (V, E)$, onde V e E correspondem respectivamente ao seu conjunto de vértices e arestas, cujo conjunto de vértices é dividido em m *clusters*, o problema da árvore geradora generalizada de custo mínimo consiste em encontrar uma árvore de custo mínimo contendo m - 1 arestas e um vértice de cada *cluster*. Este problema encontra aplicações em redes de telecomunicações, *smart-cities*, entre outros. Neste artigo é proposto um Algoritmo Genético (AG) contendo múltiplas populações independentes e um operador de cruzamento diferente dos já empregados em outros AGs para o GMSTP. Além disso, *threads* são utilizados para acelerar o processo de cálculo do custo das soluções geradas durante a execução do AG. Experimentos computacionais foram realizados para testar o seu desempenho, incluindo uma comparação com os resultados de uma formulação matemática da literatura. O AG proposto é capaz de encontrar soluções ótimas para várias instâncias e supera os melhores resultados heurísticos presentes na literatura em 12 instâncias.

PALAVRAS CHAVE. Árvores geradoras de custo mínimo, algoritmo genético, heurísticas.
Área Principal: Metaheurísticas, Otimização Combinatória.

ABSTRACT

Given a m-partite graph $G = (V, E)$, where V and E stand respectively for the vertice-set and for the edge-set, such that the set of vertices is partitioned into m *clusters*, the Generalized Minimum Spanning Tree Problem (GMSTP) consists in finding a cost minimum tree with m - 1 edges and which spans a unique vertex in each *cluster*. This problem finds application in telecommunications, smart cities, among others. In this study, a genetic algorithm (GA) containing a set of independent populations and a crossover operator different from the ones used in other GAs for the GMSTP is proposed. Furthermore, threads are applied in order to speed up the computation of the solution cost value. Computational experiments were addressed to measure the algorithm performance, including a comparison with results obtained by a mathematical formulation found in the literature. The proposed GA is able to produce optimal solutions for several instances and found better results for 12 instances.

KEYWORDS. Minimum spanning tree problems, genetic algorithm, heuristics.
Main Area: Metaheuristic, Combinatorial Optimization.

1. Introdução

Problemas de árvores geradoras tem motivado diversos estudos devido sua aplicabilidade, sua complexidade de modelagem matemática e aos seus desafios na construção de algoritmos eficientes (Santos et al., 2016). Neste trabalho, investiga-se o problema da árvore geradora generalizada de custo mínimo (denotado por GMSTP, do inglês *Generalized Minimum Spanning Tree Problem*), o qual é definido em um grafo não direcionado m-partido $G = (V, E)$, onde o seu conjunto de vértices V é particionado em m *clusters*, tal que $V = V_1 \cup V_2 \cup \cdots \cup V_m$, com $V_r \cap V_q = \varnothing$, para todo $r \neq q$, $1 \leq r \leq m$ e $1 \leq q \leq m$; e o seu conjunto E de arestas é definido como $E = \{[i, j] \mid i \in V_r;\ j \in V_q\}$ para todo $r \neq q,\ 1 \leq r \leq m$ e $1 \leq q \leq m$, com $c_{ij} \in \mathbb{R}_+$ denotando o custo de uma aresta $[i, j] \in E$. O GMSTP consiste em encontrar uma árvore de custo mínimo com o número de arestas igual a m - 1, sendo que no conjunto de extremidades dessas arestas existe exatamente um único vértice de cada *cluster*. O GMSTP tem aplicações em rede de telecomunicações, rede de distribuição de energia, agricultura de irrigação, *smart-cities* e *data science*. Uma versão simplificada desse problema ocorre quando $|V_r| = 1$, para todo $1 \leq r \leq m$, onde encontrar a GMSTP de G corresponde a encontrar uma Árvore Geradora Mínima (denotado por MST, do inglês *Minimum Spanning Tree Problem*) de G. Embora existam algoritmos polinomiais para resolver o problema MST, como por exemplo, o algoritmo de (Kruskal, 1956) e o de (Prim, 1957), o problema GMSTP é NP-difícil, conforme foi demonstrado em (Myung et al., 1995). Nesse trabalho, o GMSTP foi reduzido ao problema da cobertura de vértices de tamanho k.

Em (Myung et al., 1995), também foram apresentadas quatro formulações matemáticas para esse problema. Duas dessas formulações são aplicadas em grafos não direcionados e possuem um número exponencial de restrições. As outras duas formulações são aplicadas em grafos direcionados, sendo que uma delas é exponencial no número de restrições e a outra possui um número polinomial de restrições e variáveis. Utilizando a relaxação linear desse último modelo, (Myung et al., 1995) desenvolveu um *Branch and Bound* (B&B) para encontrar soluções ótimas de instâncias do GMSTP.

Uma análise comparativa do politopo associado à relaxação linear de oito formulações matemáticas para o GMSTP foi realizada por (Feremans et al., 2002), sendo que quatro delas eram novas formulações e as outras quatro foram propostas por (Myung et al., 1995). Desses oito modelos matemáticos, demonstrou-se que quatro deles são equivalentes na relaxação linear e dominam os demais. Dessas quatro melhores formulações, uma delas é a formulação polinomial de (Myung et al., 1995) e as outras três são exponenciais no número de restrições.

Um algoritmo aproximativo para o GMSTP foi desenvolvido por (Pop, 2002), baseado em um modelo relaxado, que no pior caso tem uma solução limitada por $\omega(2 - 2/m) \times O^*$, onde ω é o número de vértices do maior *cluster* e O^* é o valor da solução ótima. Esse trabalho também apresentou um modelo inédito de fluxo bidirecional e uma relaxação Lagrangeana utilizando esse modelo. É importante ressaltar que essa relaxação Lagrangeana apresenta melhores limites inferiores do que a relaxação linear das oito formulações matemáticas supracitadas para o GMSTP.

Um método *Branch-and-Cut* (B&C) e várias desigualdades válidas para o GMSTP foram propostos por (Feremans et al., 2004), sendo que muitas dessas desigualdades foram provadas serem facetas. O B&C usa um modelo de (Feremans et al., 2002), onde em cada nó do B&C a metaheurística *Tabu Search* é usada na busca por uma solução melhor. Além disso, restrições são adicionadas ao B&C quando as inequações válidas apresentadas em (Feremans et al., 2004) são violadas. Testes foram realizados em instâncias Euclidianas e não Euclidianas com até 200 vértices e 40 *clusters*.

Alguns trabalhos na literatura utilizam a estratégia de converter o GSMTP para o problema da árvore de Steiner. Esse é o caso do trabalho de (Duin et al., 2004), onde GMSTP é convertido em uma árvore de Steiner e em seguida é resolvido por um algoritmo de árvore Steiner adaptado. O trabalho de (Golden et al., 2005) também modela o GMSTP como uma árvore

de Steiner e o resolve através de uma formulação matemática para árvore de Steiner, com algumas restrições adicionais. Outra estratégia para resolver o GMSTP foi apresentada por (Pop et al., 2006). Nesse trabalho, um modelo matemático para o GMSTP considera um grafo G e um grafo de suporte G', onde cada vértice de G' representa um *cluster* de G e as arestas de G' correspondem às conexões entre os *clusters* de G. Um modelo de árvore geradora é aplicado em G' e restrições adicionais são acrescentadas para gerar uma solução viável do GMSTP em G. Testes foram realizados no modelo proposto por (Pop et al., 2006) em instâncias Euclidianas e não Euclidianas com até 240 vértices e 40 *clusters*.

Várias metaheurísticas foram desenvolvidas para o GMSTP, entre elas pode-se citar: *Tabu Seach* (Feremans et al., 2001), (Ghosh, 2003), (Wang et al., 2006), (Öncan et al., 2008); *Simulated Annealing* (Pop, 2002); *Varible Neighborhood Search* (Ghosh, 2003), (Hu et al., 2005); *Greedy Randomized Adaptive Search Procedure* (GRASP) (Ferreira et al., 2012) e o Algoritmo Genético (AG) (Golden et al., 2005), (Contreras-Bolton et al., 2016). Abaixo iremos descrever os AGs, os quais são versões diferentes do AG utilizado neste trabalho, e o GRASP. O AG de (Contreras-Bolton et al., 2016) e o GRASP de (Ferreira et al., 2012) são as metaheurísticas que até o presente momento apresentaram melhor eficiência para o GMSTP.

O AG desenvolvido por (Golden et al., 2005) utiliza um operador de cruzamento simples, um operador de busca local, um operador de mutação e um operador de auto-reprodução. Durante a evolução do algoritmo, os indivíduos que irão compor a próxima geração são os melhores indivíduos elite da geração corrente, correspondendo a um total de 10% do tamanho da população, e os demais serão gerados por um dos quatro operadores genéticos citados anteriormente.

O trabalho de (Ferreira et al., 2012) aplicou a metaheurística GRASP para o GMSTP e apresentou uma estratégia de pré-processamento das instâncias para diminuir o número de arestas do grafo e consequentemente diminuir o tempo de busca por uma boa solução. Nessa metaheurística foram usadas cinco maneiras diferentes de criar uma solução viável do GMSTP. Nela, também foi empregada uma busca local, um *path-relinking* e um *iterated local search*.

Outro AG para o GMSTP foi desenvolvido por (Contreras-Bolton et al., 2016), o qual aplica dois operadores de cruzamento, 5 operadores de mutação e uma maneira diferente de classificar os indivíduos em elite e não elite. Nessa avaliação é usada uma fórmula que considera dois critérios. O primeiro é o erro relativo do valor da solução corrente em relação ao valor da solução ótima ou do melhor valor conhecido para a instância abordada. O segundo critério considera uma penalização referente à possibilidade do indivíduo que está sendo avaliado gerar soluções inviáveis.

O AG desenvolvido neste artigo utiliza múltiplas populações independentes para acelerar a convergência do algoritmo. Para diminuir o seu tempo de execução, *threads* são usados para calcular o valor de indivíduos novos e agilizar a aplicação de uma busca local. Além disso, também é usado um processo de cruzamento diferente dos já empregados em outros AGs para o GMSTP e uma nova abordagem de indivíduos mutantes é empregada. Uma estratégia de interdição de vértices é aplicada para diversificar a seleção dos vértices a compor uma solução nova e assim melhorar a exploração do espaço de solução. O restante do artigo encontra-se organizado como segue. Na seção 2 é apresentado um modelo matemático para encontrar soluções ótimas para instâncias do GMSTP. Na seção 3, o AG proposto neste trabalho é detalhado, bem como a sua busca local. Na seção 4, uma técnica de pré-processamento das instâncias para diminuir o número de arestas do grafo G é descrita. Resultados e conclusões são apresentados, respectivamente, nas seções 5 e 6.

2. Um modelo matemático polinomial para o GMSTP

Um modelo proposto por (Myung et al., 1995) é fornecido nas equações de (1) à (10) e é utilizado aqui para encontrar soluções ótimas. Esse modelo usa variáveis de fluxo e possui um número polinomial de variáveis e restrições. Como esse modelo utiliza variáveis definidas em um grafo orientado, o grafo G é convertido em $H = (V, A)$ e cada aresta $[i, j] \in E$ é transformada em dois arcos (i, j) e (j, i) em A, com o mesmo custo da aresta $[i, j]$. Considerando $n = |V|$, uma

unidade de fluxo é enviada do *cluster* 1 para cada *cluster* q, com $2 \leq q \leq m$. A variável f_{ij}^q é igual a 1 se, e somente se, uma unidade de fluxo enviada do *cluster* 1 para o *cluster* q passa pelo arco (i, j). A variável binária y_i é igual a 1 se, e somente se, o vértice i estiver na solução. A variável binária w_{ij} é igual a 1 se, e somente se, o arco $(i, j) \in A$ estiver na solução e a variável binária x_{ij} é igual a 1 se, e somente se, a aresta $[i, j]$ estiver na solução.

$$(M) \min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}w_{ij} \tag{1}$$

$$\text{s.a:} \quad \sum_{i \in V_q} y_i = 1 \qquad \forall\, q = 1, ..., m \tag{2}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = m - 1 \tag{3}$$

$$\sum_{j=1}^{n} f_{ij}^q - \sum_{j=1}^{n} f_{ji}^q = \left\{ \begin{array}{cc} y_i, & i \in V_1 \\ -y_i, & i \in V_q \\ 0, & i \notin V_1 \cup V_q \end{array} \right\} \qquad \forall\, q = 2, ..., m \tag{4}$$

$$w_{ij} + w_{ji} = x_{ij} \qquad \forall (i, j) \in A, [i, j] \in E \tag{5}$$

$$f_{ij}^q \leq w_{ij} \qquad \forall (i, j) \in A,\ \forall q = 2, ..., m \tag{6}$$

$$f_{ij}^q \geq 0 \qquad \forall (i, j) \in A,\ \forall q = 2, ..., m \tag{7}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{8}$$

$$w_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A \tag{9}$$

$$x_{ij} \in \{0, 1\} \qquad \forall [i, j] \in E \tag{10}$$

As restrições (2) garantem que exatamente um vértice de cada *cluster* é escolhido. A restrição (3) assegura que exatamente m - 1 arestas são selecionadas. As restrições (4) determinam que uma unidade de fluxo q, com $2 \leq q \leq m$, deixa o *cluster* 1; uma unidade de fluxo entra em cada *cluster* $2 \leq q \leq m$ e existe uma conservação de fluxo entre *clusters* intermediários. Além disso, essas restrições evitam a formação de ciclo por causa da minimização de (M). As restrições (5) especificam que se uma aresta é selecionada, então exatamente um arco relacionado a ela deve ser selecionado. As restrições (6) determinam que existirá fluxo no arco $(i, j) \in A$ se, e somente se, ele estiver na solução. As restrições de (7) à (10) definem os domínios das variáveis.

3. Algoritmo Genético para o GMST

Um algoritmo genético é uma metaheurística baseada no conceito de evolução natural das espécies (Goldberg, 1989). Esse método tem sido aplicado com sucesso em diversos problemas de árvores geradoras (Pessoa et al., 2016),(Santos et al., 2014). A representação de uma solução no AG é chamada de indivíduo ou de cromossomo. A seguir são apresentados detalhes do algoritmo genético proposto neste trabalho.

3.1. Representação de uma solução do GSMT no AG

A representação de uma solução do GMSTP usada neste artigo é inspirada no trabalho de (Golden et al., 2005) e corresponde a um vetor de tamanho m. Um nó selecionado em cada *cluster* é guardado em uma posição desse vetor. Assim sendo, a i-ésima posição do vetor (chamada de gene em algoritmo genético) guarda um vértice representante do i-ésimo *cluster*. Existem outras formas de representação de uma solução do GMSTP, tal como um vetor binário de tamanho $|V|$

indicando os vértices selecionados para uma solução, ou um vetor binário de tamanho $|E|$ indicando as arestas selecionadas para uma solução (Golden et al., 2005). No entanto, a primeira representação é simples, fácil de construir, acessar e aplicar operadores genéticos.

3.2. Geração de soluções e população inicial do AG

Os indivíduos do AG são criados através de um algoritmo similar ao de (Prim, 1957), sendo que este tem complexidade de pior caso $O(|E|\ log\ (|V|))$ (Cormen et al., 1990). A ideia do algoritmo de Prim é construir uma MST através de uma estratégia gulosa, onde a cada iteração, a aresta de menor custo que não provoca ciclos é incluída na árvore. A adaptação realizada aqui é similar à utilizada em (Santos, 2006). Inicialmente sorteia-se um vértice $s \in V$ e o mesmo é inserido na solução parcial T. Em seguida, cria-se uma lista L de nós que se conectam a algum vértice da solução parcial T. O próximo vértice a compor a solução parcial T será sorteado e retirado da lista L. Posteriormente, a lista L é atualizada com os vértices que se ligam à nova solução parcial T. Esse procedimento é repetido até que T contenha m vértices.

Para acelerar o processo de criação de um indivíduo, o vértice de *clusters* com cardinalidade igual a 1 é diretamente inserido na solução parcial T. Isso é realizado porque necessariamente ele estará presente em qualquer solução viável, já que por definição do GMSTP, um vértice deve ser selecionado em cada *cluster*. Assim sendo, se existem tais *clusters*, não é preciso realizar o sorteio do primeiro vértice a compor a solução.

Testes de calibração foram realizados para determinar a quantidade de populações e indivíduos presentes em cada uma. Assim, no GA desenvolvido neste trabalho utilizou-se $\tau = 3$ populações independentes, contendo cada uma $p = 20$ indivíduos.

3.3. Operadores de cruzamento e mutação

Dois operadores genéticos são utilizados neste artigo: um operador de cruzamento e um operador de mutação. O primeiro favorece a passagem de genes das soluções elites de uma geração para a geração seguinte (genes correspondem a um vértice selecionado de um *cluster* especifíco) e o segundo é um mecanismo para garantir um nível de diversidade dos indivíduos. Os dois são descritos a seguir.

Dadas duas soluções, sendo uma elite e a outra não, o cruzamento funciona da seguinte forma: para definir se o i-ésimo gene do cromossomo da nova solução herda o i-ésimo gene do cromossomo elite, um valor $\alpha \in [0, 1)$ é sorteado. Se $\alpha \leq \delta$, com $\delta > 0.5$, então o cromossomo filho herda o i-ésimo gene da solução elite. Após percorrer todos os genes do cromossomo elite, se ainda existir genes vazios na nova solução, estes genes herdam os respectivos genes do cromossomo não elite, se a nova solução permanecer viável. Se o algoritmo percorrer todo o cromossomo não elite e ainda existir algum gene vazio na nova solução, então um gene qualquer que mantenha a nova solução viável é utilizado para concluir a sua construção.

Nesse algoritmo, o operador de mutação corresponde à criação de um novo indivíduo. Esse operador de mutação ajuda a manter um nível de diversificação das soluções, utilizando um mecanismo de interdição da seleção dos vértices que foram muitas vezes escolhidos até a iteração corrente. Assim sendo, o controle de escolha dos vértices é realizado tendo como base a frequência de seleção de cada vértice. Após a execução de um certo número de iterações pré-determinado, vértices que tem um percentual de seleção maior ou igual a β é impedido de entrar em novas soluções criadas pelo operador de mutação até que o seu percentual de seleção seja inferior a β. Com essa diversificação na seleção dos vértices o espaço de solução de uma instância é melhor explorado. A quantidade de iterações e β foram calibrados nos experimentos.

3.4. Funcionamento global do AG para o GMSTP

O AG evolui até que o número máximo de iterações $max\_ger = 100$ seja alcançado, ou o algoritmo execute 30 gerações sem atualizar o melhor indivíduo do AG, ou o melhor valor

conhecido para a instância abordada seja alcançado. Essa última condição de parada é ignorada se o AG superar o melhor valor de solução conhecido para essa instância. Os parâmetros do AG propostos para o GMSTP são descritos na Tabela 1.

Tabela 1: Descrição dos parâmetros do AG.

Parâmetros	Valores	Descrição
m		número de genes de um cromossomo
p	20	tamanho de uma população
p_e	4	tamanho do conjunto elite de toda população
p_m	4	número de cromossomos mutantes inseridos em toda população, de qualquer iteração, exceto na inicialização ou reinicialização de populações
δ	60%	probabilidade de um gene ser herdado do cromossomo elite
τ	3	número de populações independentes
max_ger	100	guarda o número máximo de iterações do AG
$reinicie$	15	guarda o número de gerações a serem executadas até que toda população seja reinicializada, preservando somente os indivíduos elite de cada população
num_crom	2	número dos primeiros indivíduos elite de cada população a serem inseridos nas demais, substituindo os piores indivíduos dessas
$troca\_crom$	10	guarda o número de iterações a serem executadas até que os num_chro melhores indivíduos de toda população sejam inseridos nas demais
$\varphi\_threads$	8	número de $threads$ utilizados para calcular concorrentemente o custo de todos os cromossomos de uma população.

Em uma dada geração ger, exceto no processo de inicialização e reinicialização das populações, uma nova população é formada de indivíduos provenientes do conjunto elite da iteração precedente e indivíduos gerados a partir dos operadores de cruzamento e mutação descritos anteriormente. Sendo assim, a geração ger é composta por p_e indivíduos do conjunto elite da geração $ger-1$, p_m novos indivíduos mutantes gerados aleatoriamente e $p_o = p-p_m-p_e$ novos indivíduos gerados pelo cruzamento de dois cromossomos, um proveniente do conjunto elite e o outro não, ambos selecionados aleatoriamente da geração $ger-1$. Sempre após a execução de $troca\_crom$ interações, o algoritmo copia os num_crom primeiros indivíduos elite de cada população para as demais populações, substituindo-os pelos piores indivíduos destas. As populações são trabalhadas de forma sequencial.

O custo de cada solução é calculado através do algoritmo de Kruskal usando os vértices selecionados de cada cluster, com complexidade $O(|E|\ log\ |V|)$ (Golden et al., 2005). Determinar o valor de uma solução é um processo demorado. Para acelerar o tempo de cálculo do custo de todas as novas soluções de uma população, todas elas são distribuídas entre $\varphi\_threads$. Cada *thread* está associado a um núcleo do computador que foi utilizado neste trabalho e por isso todos eles são executadas simultaneamente. A partir de um número k de novos cromossomos de uma população, cada *thread* trabalha com $k/\varphi\_threads$ cromossomos.

Todos os k novos indivíduos gerados na iteração corrente são submetidos à aplicação de uma busca local. Essa busca local, descrita na Seção 3.5, é aplicada se, e somente se, o valor da solução do indivíduo abordado encontra-se no máximo 25% pior do que o valor da melhor solução encontrada até a geração corrente. Além disso, *threads* são usados durante a aplicação dessa busca local e cada *thread* trabalha com $k/\varphi\_threads$ indivíduos. O uso de *threads* no AG desenvolvido neste artigo reduziu o seu tempo de execução, em média, 42%.

3.5. Busca local

A busca local utilizada neste artigo é inspirada do trabalho de (Ferreira et al., 2012). Sendo T uma solução do GMSTP, a vizinhança de T, descrita a seguir, é investigada e assim que uma

solução vizinha de custo menor que o de T for encontrada, T recebe essa solução. Isso caracteriza uma busca local que utiliza uma estratégia primeiro aprimorante.

Um movimento da busca local troca o vértice representante de um *cluster* por outro vértice desse *cluster* que está fora da solução. Somente movimentos que retornem uma solução viável são aceitos. Seja λ o vetor (indivíduo) que representa a solução T e λ' o vetor associado a sua solução vizinha T'. Uma solução vizinha de T é aquela em que $\lambda[i] \neq \lambda'[i]$ e $\lambda[j] = \lambda'[j]$, para todo $j \in \{1, ..., m\} \setminus i$. Para cada *cluster*, essa busca local considera todas as $|V_q| - 1$ possíveis soluções vizinhas de T, com $1 \leq q \leq m$.

4. Pré-processamento de instâncias

No início da execução do AG, um pré-processamento da instância tratada é realizado. O pré-processamento de uma instância do GMSTP remove arestas do conjunto E que garantidamente não compõem uma solução ótima. Diminuir o número de arestas de uma instância implica em diminuir o tempo de cálculo do valor de suas soluções. Para esclarecer essa afirmação, considere P_{uv} o conjunto de todos os caminhos entre os vértices u e v de G, compostos somente pelos vértices presentes na solução ótima, exceto a própria aresta $[u, v]$; $c(Q)$ o valor da aresta de maior peso do caminho $Q \in P_{uv}$; c_{ij} o custo de uma aresta $[i, j] \in E$ e $b_{uv} = min\{c(Q)|Q \in P_{uv}\}$. O trabalho de (Uchoa, 2006) prova que dada uma instância do GMSTP cuja solução ótima é uma árvore geradora generalizada mínima $T = (V', E')$, se $b_{uv} < c_{uv}$, então $[u, v] \notin E'$. Além disso, se $b_{uv} \leq c_{uv}$, então a aresta $[u, v]$ é redundante e também pode ser removida de G, pois mesmo com a sua remoção ainda existirá um caminho entre u e v sem alterar o valor da solução ótima. Todavia, o número de caminhos entre um par de vértices é exponencial.

Uma outra eliminação de arestas apresentada por (Uchoa, 2006), agora polinomial, é reproduzida neste trabalho. Nessa eliminação, a remoção de uma aresta $[u, v] \in E$ é feita considerando todos os caminhos entre os vértices u e v de V, excluindo a própria aresta $[u, v]$, para um determinado número de *clusters* intermediários entre esses dois vértices. O tempo computacional para identificação desses caminhos aumenta exponencialmente com o número σ de *clusters* intermediários entre os vértices u e v. Dessa forma, trabalha-se neste artigo com $\sigma = 1$ e $\sigma = 2$. Quando $\sigma = 1$, uma aresta $[u, v]$ é dita redundante se existe um *cluster* V_r, com $1 \leq r \leq m$, no qual c_{uv} é maior ou igual ao custo de cada aresta $[u, w]$ e $[w, v]$ para todo $w \in V_r$. Quando $\sigma = 2$, uma aresta $[u, v]$ é redundante se existirem dois *cluster* V_r e V_q, com $1 \leq r \leq m$ e $1 \leq q \leq m$, tal que c_{uv} é maior ou igual ao custo de cada aresta contida em $\{[u, w],\ [w, z],\ [z, v]\}$ para todo $w \in V_r$ e para todo $z \in V_q$.

5. Experimentos computacionais

O AG criado neste trabalho foi desenvolvido na linguagem C++, executado em uma máquina com processador Intel Core i7-3770 3.40 GHz de 8 núcleos, memória RAM de 16 GB e o sistema operacional Linux Ubuntu 14.04 LTS. A formulação matemática foi testada no IBM CPLEX 12.6.1.

As instâncias utilizadas neste trabalho, as quais são euclidianas, provêm do trabalho (Öncan et al., 2008), contendo de 229 à 783 vértices. O pré-processamento foi aplicado em todas as instâncias e a média de redução do número de arestas é da ordem de 85% usando $\sigma = 1$ e $\sigma = 2$. A média de tempo de processamento para realizar o pré-processamento foi de 1,37 segundos.

Os clusters de cada instância são definidos de acordo com um dos dois procedimento de clusterização definidos em (Öncan et al., 2008): clusterização em grade ou centro de clusterização. Na Tabela 2, para um grupo de 27 instâncias, compara-se os valores das soluções produzidas pelo AG com os valores das soluções ótimas produzidas pela relaxação linear do modelo matemático da Seção 2. Em todas elas foi usado o procedimento de clusterização em grade. Na primeira coluna da Tabela 2 é apresentado os nomes das instâncias. Na segunda coluna é informado o valor do μ usado

na clusterização em grade para cada instância. Nas colunas três, quatro e cinco são apresentados, para toda instância, o valor da sua solução ótima, seu tempo de execução em segundos utilizando matemático da Seção 2 e o seu percentual de arestas eliminadas em relação ao seu grafo completo, respectivamente. Nas colunas seis e sete têm-se o custo e a diferença relativa em porcentagem entre o valor de solução obtido pelo AG e o valor ótimo, respectivamente, para cada instância. Nas colunas oito e nove são apresentados, em segundos, o tempo do pré-processamento e da execução do AG, respectivamente, para cada instância. De 27 instâncias analisadas na Tabela 2, em 25 delas o AG proposto neste trabalho alcança o valor da solução ótima.

Tabela 2: Comparando os resultados do AG com os valores de soluções ótimas

Instância	μ	Valor Ótimo	Tempo	% Eliminação de arestas	AG			
					Custo	GAP	T-1	T-2
81gr229	3	74819	5999,96	89,86 %	74819	0,0000	2,34	6631
95gil262	3	1255	23772,20	91,67 %	1255	0,0000	284,00	4
101pr264	3	29199	7603,99	95,55 %	29199	0,0000	0,15	1
102pr299	3	23096	16619,60	95,55 %	23096	0,0000	0,21	6
108lin318	3	24092	35942,10	94,17 %	24092	0,0000	0,29	3
47gr229	5	54236	2555,10	82,05 %	54236	0,0000	0,11	55
63gil262	5	984	24719,00	83,19 %	984	0,0000	0,29	6
55pr264	5	21351	2022,29	92,07 %	21351	0,0000	0,10	2
69pr299	5	18582	9231,12	91,72 %	18582	0,0000	0,18	7
64lin318	5	17667	14247,20	90.02 %	17667	0,0000	0,26	2
34gr229	7	45960.5	2089,43	75,27 %	46030	0.1512	1,53	406
49gil262	7	802	9605,16	80,34 %	802	0,0000	0,36	3
43pr264	7	20438	1831,42	88,50 %	20438	0,0000	0,10	3
47pr299	7	15238	7152,64	87,46 %	15238	0,0000	0,28	22
49lin318	7	14909	5985,29	88,30 %	14909	0,0000	0,33	4
64rd400	7	4576	44095,80	81,47 %	4581	0,1100	2,00	2990
61fl417	7	7446	12243,90	87,07 %	7446	0,0000	0,83	189
23gr229	10	39797	836,24	62,35 %	39797	0,0000	0,14	31
36gil262	10	639	7278,78	71,23 %	639	0,0000	0,38	2
27pr264	10	16546	381,89	86,68 %	16546	0,0000	0,96	2
35pr299	10	11624	3334,99	82,22 %	11624	0,0000	0,29	2
36lin318	10	10119	5859,57	81,48 %	10119	0,0000	0,37	2
49rd400	10	3825	29174,40	71,62 %	3825	0,0000	1,49	175
42fl417	10	6986	3736,64	83,99 %	6986	0,0000	0,74	2
52gr431	10	62688	22691,70	73,23 %	62688	0,0000	0,92	150
48pcb442	10	11941	22355,90	86,11 %	11941	0,0000	0,65	6
52d493	10	11121	109970,00	71,89 %	11121	0,0000	1,34	5

Da Tabela 3 à 7, o AG é comparado com as metaheurísticas GRASP de (Öncan et al., 2008) e M-GA de (Contreras-Bolton et al., 2016). Nessas tabelas, nas colunas um e dois têm-se os nomes e os melhores valores de limite superior (LS) encontrado na literatura para cada instância, respectivamente. Em alguns casos esse LS pode ser o valor da solução ótima. Na coluna três é dado o custo da melhor solução encontrada pelo AG. Nas colunas quatro, sete e nove têm-se a diferença relativa em porcentagem entre o valor de solução obtido pelo AG, GRASP e M-GA, nessa ordem, e o valor de LS para cada instância. Nas colunas três e quatro, os campos em negrito e em branco sinalizam onde o AG desenvolvido neste trabalho superou o LS da literatura. Nas colunas cinco e seis são fornecidos, em segundos, os tempos de pré-processamento e execução do AG, respectivamente, para cada instância. Na coluna oito é dado o tempo de execução do GRASP, em segundos, para cada instância. Na décima coluna tem-se apenas o símbolo "-"porque o tempo de execução de cada instância para o M-GA não é apresentado em (Contreras-Bolton et al., 2016).

Os resultados indicam que nos grupos de clusterização com $\mu = 3$, $\mu = 5$, $\mu = 7$ e $\mu = 10$,

o AG supera os LSs da literatura em 3, 2, 3 e 3 instâncias, respectivamente. No grupo de centro de clusterização o AG supera os LSs da literatura de 2 instâncias. Em 72 de 98 instâncias o AG supera ou alcança os LSs da literatura. Para as demais instâncias, quando o AG não alcança os melhores LSs da literatura, os seus resultados são competitivos.

Tabela 3: Resultados com as instâncias usando clusterização em grade, com $\mu = 3$

		AG				GRASP		M-GA	
Instância	LS	Custo	GAP	T-1	T-2	GAP	T	GAP	T
181ali535	134362	**134354**		2,34	6631	0,0000	1576	0,0000	-
182att532	15652	15652	0,0000	0,97	54	0,0000	1569	0,0000	-
171d493	20269	20269	0,0000	1,30	36	0,0000	1104	0,0000	-
221d657	25703	25704	0,0039	2,06	7193	0,0039	3027	0,0000	-
142fl417	8353	8353	0,0000	0,89	66	0,0000	681	0,0000	-
95gil262	1255	1255	0,0000	284,00	4	0,0000	204	0,0000	-
81gr229	74792	74819	0,0361	0,14	262	0,0000	67	0,0000	-
149gr431	103844	**103805**		0,79	3009	0,0000	1697	0,0000	-
224gr666	174655	174858	0,1162	2,71	7838	0,0000	3105	0,0000	-
108lin318	24092	24092	0,0000	0,29	3	0,0000	206	0,0000	-
230p654	23547	23548	0,0042	2,82	8556	0,0000	4127	0,0000	-
156pcb442	27513	27513	0,0000	0,71	100	0,0000	810	0,0000	-
101pr264	29199	29199	0,0000	0,15	1	0,0000	102	0,0000	-
102pr299	23096	23096	0,0000	0,21	6	0,0000	399	0,0000	-
163pr439	64953	64953	0,0000	0,61	16	0,0000	932	0,0000	-
196rat575	2880	2880	0,0000	1,28	5101	0,0000	2080	0,1389	-
285rat783	4203	4204	0,0238	2,96	9936	0,0000	7180	0,0714	-
135rd400	7632	7632	0,0000	788,00	20	0,0000	548	0,0000	-
198u574	19696	**19629**		1,52	408	0,0000	1340	0,0000	-
266u724	21739	21747	0,0368	3,53	2508	0,0368	5201	0,0124	-

6. Conclusão

Este artigo descreve um AG com múltiplas populações independentes para o GMSTP. Os resultados mostram que o AG desenvolvido neste trabalho possui um bom desempenho quando comparado à outras metaheurísticas, superando em 12 de 98 instâncias testadas, os melhores resultados heurísticos já apresentados na literatura. Além disso, em 60 de 98 instâncias, o AG alcança os melhores resultados da literatura, ou seja, em 72 de 98 o AG supera ou alcança os LSs da literatura. Para as demais instâncias, quando o AG não alcança os melhores LSs da literatura, os seus resultados são competitivos. Dessas 98 instâncias, para 27 delas foram encontrados os seus valores ótimos e em 25 delas o AG alcança o valor da solução ótima. Como trabalhos futuros, pretende-se implementar novas técnicas de buscas locais para explorar melhor o espaço de solução e um paradigma de programação paralela.

Referências

Contreras-Bolton, C., Gatica, G., Barra, C. R., e Parada, V. (2016). A multi-operator genetic algorithm for the generalized minimum spanning tree problem. *Expert Systems With Applications*, 50:1–8.

Cormen, T., Leiserson, C., e Rivest, R. (1990). Introduction to algorithms. *McGraw-Hill, New York*.

Duin, C. W., Volgenant, A., e Voss, S. (2004). Solving group Steiner problems as steiner problems. *European Journal of Operational Research*, 154:323–329.

Feremans, C., Labbe, M., e Laporte, G. (2001). On generalized minimum spanning trees. *European Journal of Operational Research*, 134(2):457–458.

Tabela 4: Resultados com as instâncias usando clusterização em grade, com $\mu = 5$

		AG				GRASP		M-GA	
Instância	LS	Custo	GAP	T-1	T-2	GAP	T	GAP	T
108ali535	108201	**107936**		1,79	9109	0,0000	632	0.0000	-
110att532	11896	11896	0,0000	0,90	493	0,0000	641	0,0000	-
102d493	16132	16132	0,0000	1.22	510	0,0000	725	0.0000	-
137d657	19826	19838	0,0605	1.89	1390	0,0000	1249	0,0000	-
93fl417	7952	7952	0,0000	0.91	3	0,0000	230	0.0000	-
63gil262	984	984	0,0000	0.29	6	0,0000	63	0.0000	-
47gr229	54305	**54236**		0,11	55	0,0000	43	0.0000	-
87gr431	81856	81963	0,1307	1,08	305	3,6650	244	0,0000	-
139gr666	144077	145369	0,8967	2,31	861	0,0000	2192	0,0000	-
64lin318	17667	17667	0,0000	0,26	2	0,0000	116	0,0000	-
134p654	22377	22377	0,0000	2,54	121	0,0000	1316	0,0000	-
95pcb442	19383	19383	0,0000	0,58	537	0,0000	357	0,0000	-
55pr264	21351	21351	0,0000	0,10	2	0,0000	67	0,0000	-
69pr299	18582	18582	0,0000	0,18	7	0,0000	105	0,0000	-
96pr439	54230	54230	0,0000	0,57	10	0,0000	587	0,0000	-
121rat575	2014	2018	0,1986	1,05	1130	0,0000	705	0,0000	-
169rat783	2823	2823	0,0000	2,30	1699	0,0000	2424	0,0708	-
81rd400	5433	5434	0,0184	0,90	481	0,0331	284	0,0184	-
127u574	15240	15255	0,0984	1,35	974	0,0787	1037	0,3609	-
166u724	15435	15435	0,0000	3,01	419	0,0000	2064	0,0000	-

Tabela 5: Resultados com as instâncias usando clusterização em grade, com $\mu = 7$

		AG				GRASP		M-GA	
Instância	LS	Custo	GAP	T-1	T-2	GAP	T	GAP	T
83ali535	94149	**93905**		1,67	1149	0,0000	1012	0,0000	-
80att532	10204	10204	0,0000	1,15	83	0,0000	992	0,0000	-
78d493	14152	14152	0,0000	1,39	29	0,0000	212	0,0000	-
96d657	16542	16542	0,0000	2,26	2675	0,0036	440	0,0000	-
61fl417	7446	7446	0,0000	0,83	189	0,0000	99	0,0000	-
49gil262	802	802	0,0000	0,36	212	0,0748	63	0,0000	-
34gr229	45989	46030	0,0892	0,18	59	0,1305	13	0,1305	-
64gr431	71415	**71318**		1,53	406	0,0000	111	0,0000	-
96gr666	119271	120120	0,7118	2,61	1074	0,0000	452	0,0000	-
49lin318	14909	14909	0,0000	0,33	4	0,0000	44	0,0000	-
100p654	21770	21770	0,0000	3	121	0,0000	462	0,0000	-
64pcb442	14639	14644	0,0342	1,44	758	0,0424	117	0,0342	-
43pr264	20438	20438	0,0000	0,1	3	0,0000	28	0,0000	-
47pr299	15238	15238	0,0000	0,28	22	0,0000	37	0,0000	-
74pr439	47101	47101	0,0000	0,76	6	0,0000	140	0,0000	-
100rat575	1734	1735	0,0577	1,31	1685	0,0923	601	0,0577	-
121rat783	2228	2229	0,0449	2,5	3281	0,0449	850	0,0449	-
64rd400	4576	4581	0,1093	2	2990	0,1093	97	0,1093	-
92u574	12186	12186	0,0000	1,39	433	0,0000	304	0,0000	-
11u724	13101	**13086**		3,5	2847	0,0611	710	0,0000	-

Tabela 6: Resultados com as instâncias usando clusterização em grade, com $\mu = 10$

		AG				GRASP		M-GA	
Instância	LS	Custo	GAP	T-1	T-2	GAP	T	GAP	T
57ali535	73359	**73207**		1,18	305	0,0000	213	0,0000	-
57att532	8494	8497	0,0353	1,47	388	0,0353	187	0,0353	-
52d493	11121	11121	0,0000	1,34	5	0,0000	180	0,0000	-
73d657	13495	13495	0,0000	3,17	305	0,0000	186	0,0000	-
42fl417	6986	6986	0,0000	0,74	2	0,0000	75	0,0000	-
36gil262	639	639	0,0000	0,38	2	0,0000	33	0,0000	-
23gr229	39793	39797	0,0101	0,14	31	0,0000	28	0,0000	-
52gr431	62722	**62688**		0,92	150	0,0000	110	0,0000	-
70gr666	97198	**96992**		3,5	493	0.0000	315	0,0000	-
36lin318	10119	10119	0,0000	0,37	2	0,0000	31	0,0000	-
69p654	20736	20739	0,0145	2,43	235	0,0145	274	0,0145	-
48pcb442	11941	11941	0,0000	0,65	6	0,0000	135	0,0000	-
27pr264	16546	16546	0,0000	0,96	2	0,0000	28	0,0000	-
35pr299	11624	11624	0,0000	0,29	2	0,0000	63	0,0000	-
48pr439	40518	40518	0,0000	0,64	4	0,0000	117	0,0000	-
64rat575	1235	1235	0,0000	1,29	10	0,0000	284	0,0000	-
81rat783	1682	1682	0,0000	2,48	230	0,0000	468	0,0000	-
49rd400	3825	3825	0,0000	1,49	175	0,0000	99	0,0000	-
64u574	9755	9755	0,0000	1,87	97	0,0000	234	0,0000	-
80u724	9608	9608	0,0000	4,22	483	0,0000	365	0,0000	-

Tabela 7: Resultados com as instâncias usando centro de clusterização

		AG				GRASP		M-GA	
Instância	LS	Custo	GAP	T-1	T-2	GAP	T	GAP	T
107att532	12001	12001	0,0000	1,27	1442	0,0000	597	0,0000	-
99d493	16493	16493	0,0000	2,56	17	0,0000	587	0,0000	-
132d657	19427	19428	0,0051	2,19	3206	0,0000	1056	0,0000	-
84fl417	7935	7935	0,0000	1,24	5	0,0000	233	0,0000	-
53gil262	887	887	0,0000	0,51	2	0,0000	74	0,0000	-
87gr431	86885	**86842**		2,05	676	0,0000	233	0,0000	-
134gr666	144737	146006	0,8768	5,48	6629	0,0249	1365	0,0000	-
64lin318	18471	18471	0,0000	0,26	45	0,0000	130	0,0000	-
131p654	22207	22207	0,0000	5,24	3015	0,0000	1045	0,0000	-
89pcb442	19571	19571	0,0000	0,82	41	0,0000	266	0,0000	-
53pr264	21872	21872	0,0000	0,19	2	0,0000	72	0,0000	-
60pr299	20290	20290	0,0000	0,23	6	0,0000	94	0,0000	-
88pr439	51749	51749	0,0000	0,79	32	0,0000	574	0,0000	-
115rat575	2168	2170	0,0923	1,20	2592	0,0923	762	0,0923	-
157rat783	3009	3012	0,0997	2,53	5740	0,0997	1916	0,0997	-
80rd400	5868	5868	0,0000	1,47	17	0,0000	208	0,0000	-
115u574	15027	15032	0,0333	1,34	4806	0,0333	517	0,0399	-
145u724	15904	15904	0,0000	3,02	6648	0,0000	1290	0,0000	-

Feremans, C., Labbé, M., e Laporte, G. (2002). A comparative analysis of several formulations for the generalized minimum spanning tree problem. *Networks*, 39(1):29–34.

Feremans, C., Labbé, M., e Laporte, G. (2004). The generalized minimum spanning tree problem: Polyhedral analysis and branch-and-cut algorithm. *Networks*, 43(2):71–86.

Ferreira, C. S., Ochi, L. S., Parada, V., e Uchoa, E. (2012). A GRASP-based approach to the generalized minimum spanning tree problem. *Expert Systems with Applications*, 39(3):3526–3536.

Ghosh, D. (2003). Solving medium to large sized Euclidean generalized minimum spanning tree problems. *Technical report NEP-CMP-2003-09-28 Indian Institute of Management*.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Golden, B., Raghavan, S., e Stanojevic, D. (2005). Heuristic search for the generalized minimum spanning tree problem. *INFORMS Journal on Computing*, 17(3):290–304.

Hu, B., Leitner, M., e Raidl, G. R. (2005). Computing generalized minimum spanning trees with variable neighborhood search. *In: Proceedings of the 18th Mini-Euro Conference on Variable Neighborhood Search*.

Kruskal, J. B. J. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

Myung, Y. S., Lee, C. H., e Tcha, D. W. (1995). On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241.

Pessoa, L. S., Santos, A. C., e Resende, M. G. C. (2016). A biased random-key genetic algorithm for the tree of hubs location problem. *Optimization Letters*, pages 1–14.

Pop, P. C. (2002). The generalized minimum spanning tree problem. *Ph.D. Thesis, University of Twente, The Netherlands*.

Pop, P. C., Kern, W., e Still, G. (2006). A new relaxation method for the generalized minimum spanning tree problem. *European Journal of Operational Research*, 170:900–908.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401.

Santos, A. C. (2006). *Modelos e algoritmos para o problema da árvore geradora de custo mínimo com restrição de diâmetro*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro.

Santos, A. C., Duhamel, C., e Andrade, R. (2016). *Trees and forests*, pages 1–27. Springer International Publishing.

Santos, A. C., Lima, D. R., e Aloise, D. J. (2014). Modeling and solving the bi-objective minimum diameter-cost spanning tree problem. *Journal of Global Optimization*, 60(2):195–216.

Uchoa, E. (2006). Reduction tests for the prize-collecting steiner problem. *Operations Research Letters*, 34(4):437–444.

Wang, Z., Che, C. H., e Lim, A. (2006). Tabu search for generalized minimum spanning tree problem. *In: Yang, Q., Webb, G. (Eds.), PRICAI 2006, LNAI*, 4099:918–922.

Öncan, T., Cordeau, J., e Laporte, G. (2008). A tabu search heuristic for the generalized minimum spanning tree problem. *European Journal of Operational Research*, 191(2):306–319.