

Algoritmo Genético para o Problema da Árvore de Cobertura com Representação Mínima

David Galvão², Victor Koehler², Thiago Gouveia¹, Gilberto Farias de Sousa Filho²

¹Instituto Federal da Paraíba, João Pessoa — PB

²Universidade Federal da Paraíba, João Pessoa — PB
João Pessoa — PB

Autor para correspondência: galvao2010@icloud.com

RESUMO

O Problema da Árvore de Cobertura com Representação Mínima (PACRM) tem como objetivo, dado um grafo com arestas rotuladas $G = (V, E, L)$, sendo V o conjunto de vértices de G , E o conjunto de arestas, L o conjunto de rótulos, e $L(e)$ o rótulo associado à aresta e , encontrar uma árvore geradora, $T = (V, E', L')$, tal que $E' \subseteq E$, $L' \subseteq L$, e a soma dos rótulos representados em cada vértice seja minimizada. Neste trabalho, propomos um algoritmo genético para o problema e três novas estruturas de vizinhança adotadas na fase de educação do algoritmo. Os experimentos computacionais realizados demonstraram que a meta-heurística proposta obteve resultados melhores que as heurísticas da literatura, além de baixar o *upper bound* conhecido dos métodos exatos em um conjunto de instâncias de dimensão média com pouco esforço computacional.

PALAVRAS CHAVE. Grafos com Arestas Rotuladas, Algoritmo Genético, Árvores.

Área Principal: MH – Metaheurísticas

ABSTRACT

The Minimum Spanning Tree Problem (MRSTP) aims to, given an edge-labeled graph $G = (V, E, L)$, in which V is the set of vertices of G , E is the set of edges, L the set of labels, and $L(e)$ is the label associated with edge e , find a spanning tree, $T = (V, E', L')$, such that $E' \subseteq E$, $L' \subseteq L$, and minimizes the sum of the labels represented in each vertex. In this work, we propose a Genetic Algorithm for the problem and three new neighborhood structures applied in education phase of the algorithm. The computational experiments have shown that the proposed metaheuristic achieved better results than the heuristics of the literature, in addition to decreasing the known upper bound of the exact methods in a set of medium-sized instances with little computational effort.

KEYWORDS. Edge-Labeled Graphs, Genetic Algorithm, Trees.

Main Area: MH - Metaheuristics

1. Introdução

Um grafo com arestas rotuladas (GAR) é um grafo, que possui para cada aresta um rótulo associado. Nesta classe de grafos, é comum utilizar cores para melhorar a visualização dos rótulos, e, por este motivo, os termos rótulos e cores são utilizados de forma intercambiável neste trabalho. Nos últimos anos, diversos trabalhos foram realizados sobre problemas relacionados a GARs. Entre estes, citamos o Problema do Diâmetro Colorido (Figueirêdo et al., 2020), e o Problema da Floresta Geradora de k -rótulos (Cerulli et al., 2014). Adicionalmente, podemos relacionar com aqueles que buscam por árvores, como o Problema de Steiner de Rotulação Mínima (Cerulli et al., 2006), e a versão Fortemente Generalizada do Problema da Árvore de Cobertura de Rotulação Mínima (Cerrone et al., 2019).

Este trabalho aborda o Problema da Árvore de Cobertura com Representação Mínima (PACRM, (Silva, 2018; Linares et al., 2018)). Neste, é dado um grafo com arestas rotuladas como entrada, e objetiva-se encontrar uma árvore de cobertura que minimize a soma das cores representadas em cada vértice. Este problema possui aplicações diretas no projeto de redes de computadores e no projeto de redes de transporte multimodais, nos quais é desejável que cada nó possua a menor complexidade possível, o que o torna mais barato, além de facilitar a sua manutenção. A Figura 1 traz um exemplo do PACRM aplicado ao projeto de uma rede de transporte multimodal. Nesta, cada vértice representa um bairro, enquanto cada aresta representa uma possível ligação entre estes usando um dos meios de transporte ecológicos que podem ser utilizados, representando os rótulos. Em cada bairro deve ser criada uma estação para a integração dos transportes. Objetiva-se que as estações tenham o menor custo, representado pela menor complexidade para a sua construção.

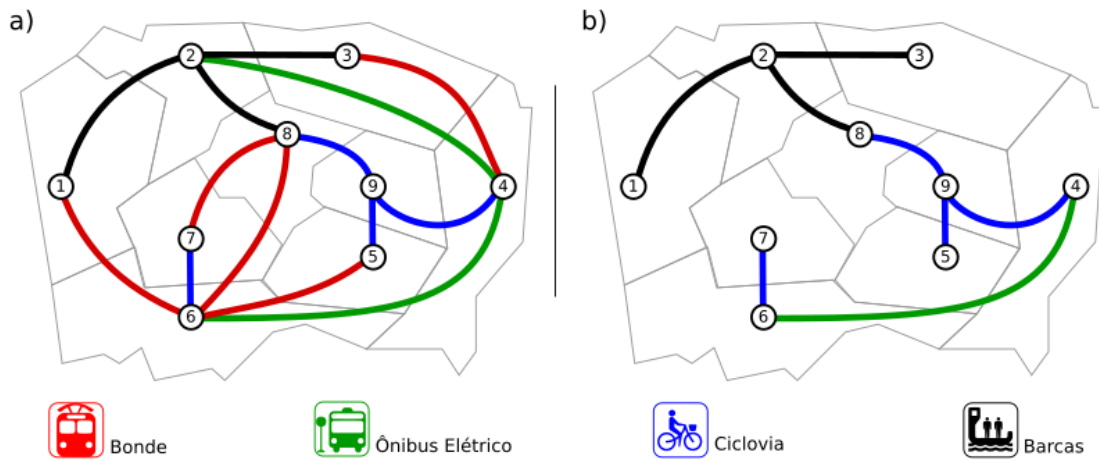


Figura 1: Exemplo de aplicação prática do PACRM

Formalmente, considere $G = (V, E, L)$ um GAR simples e não orientado, sendo V o conjunto de vértices, E o conjunto de arestas, L o conjunto de rótulos (ou cores), e $L(e)$ uma função que retorna o rótulo associado à aresta e . Seja $E(v)$ o conjunto de arestas que incidem sobre o vértice v , $c_T(v) = \bigcup_{e \in E(v)} L(e)$ o conjunto de rótulos que incidem sobre o vértice v , e $Rt(G) = \sum_{v \in V} (|c_T(v)| - 1)$ a função de *representação*. O objetivo do PACRM é encontrar uma árvore geradora $T = (V, E', L')$, tal que $E' \subseteq E$, $L' \subseteq L$ e $Rt(T)$ é minimizado. A Figura 2 ilustra os conceitos de GAR, de representação e o Problema da Árvore de Cobertura com Representação Mínima. A Figura 2(a) ilustra um GAR $G = (V, E, L)$ com 6 vértices, 10 arestas e 6 rótulos. A Figura 2(b) exhibe uma árvore de cobertura T com $Rt(T) = 2$, menor valor possível, e 3 cores.

Em detalhes, os vértices 1, 3, 5 e 6 possuem $|c_T(v)| = 1$ enquanto os vértices 2 e 4 possuem $|c_T(v)| = 2$. Logo, $Rt(G) = \sum_{v \in V} (|c_T(v)| - 1) = 2$. Adicionalmente, observe que a árvore de cobertura de G com o menor número de cores utiliza apenas o conjunto de rótulos $\{A, B\}$.

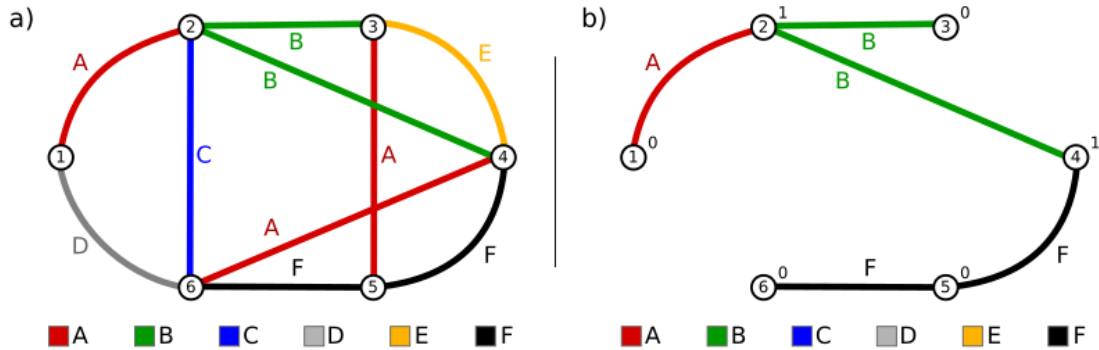


Figura 2: (a) Um pequeno GAR G como Instância-exemplo para o PACRM, (b) Solução ótima para o PACRM em G com valor $Rt(T) = 2$.

Neste trabalho, propomos um método baseado nos Algoritmos Genéticos e são definidas três novas vizinhanças aplicadas na busca local na fase de educação do algoritmo. As demais seções deste trabalho estão organizadas como segue: a Seção 2 apresenta trabalhos relacionados ao tema; a Seção 3 descreve o algoritmo genético e as novas estruturas de vizinhança propostas; os experimentos computacionais são discutidos na Seção 4; enquanto que as considerações finais e propostas de trabalhos futuros são trazidas na Seção 5.

2. Trabalhos Relacionados

Silva (2018) propõe o PACRM e demonstra que o problema é NP-Completo tanto para multigrafos quanto para grafos simples. Neste, são propostos uma formulação exata, e duas heurísticas construtivas gulosas para o problema, estas denominadas *KBased* e *PBased*, inspiradas nos algoritmos de Kruskal e Prim (Cormen et al., 2009), respectivamente. Considere $G = (V, E, L)$ o GAR de entrada, e $\mathcal{K}$ o conjunto de componentes monocromáticas maximais conexas (CMMC) de G , construído com o auxílio do algoritmo de busca em profundidade aplicado sobre cada subgrafo induzido $G = (V, E(\{l\}), \{l\}), \forall l \in L$, onde $E(\{l\}) = \{e, \forall e \in E \mid L(e) = l\}$. Note que a operação de obtenção de todos os CMMCs de G pode ser realizada em $O(|L| \times |V| \times (|V| + |E|))$.

Heurística *KBased*: Inicialmente, o conjunto de arestas que representa a solução $E' = \emptyset$. Enquanto $G(V, E', L)$ não for conexo, cada iteração do laço principal do algoritmo busca o CMMC $K \in \mathcal{K}$ que minimiza o número de componentes conexos de $G(V, E' \cup E(K), L)$. No laço interno, dado o conjunto de arestas de K , uma aresta apenas é adicionada à E' se esta não vier a causar ciclos no mesmo. Seu funcionamento é descrito pelo Algoritmo 1.

Heurística *PBased*: Seu funcionamento está descrito no Algoritmo 2. Inicia o conjunto de arestas solução $E' = \emptyset$ e $S = \{r\}$, onde $r \in V$ é um nó raiz escolhido arbitrariamente. Enquanto $S \neq V$, encontre o CMMC $K \in \mathcal{K}$ que minimize o incremento da representação mínima dos vértices de S , adicione os vértices de K em S sem repetição e adicione em E' as arestas de K que não formem ciclo.

Algoritmo 1: KBased($G = (V, E, L), \mathcal{K}$)

```

1  $E' \leftarrow \emptyset$ 
2 while  $|E'| < |V| - 1$  do
3    $K \leftarrow \text{CMMC} \in \mathcal{K}$  que minimize o número de componentes conexos
4   Seja  $E(K)$  o conjunto de arestas pertencentes ao CMMC  $K \in \mathcal{K}$ 
5   Para cada  $e \in E(K)$  faça
6     Se  $E' \cup e$  não possui ciclos então
7        $E' \leftarrow E' \cup e$ 
8 retorne  $E'$ 

```

Algoritmo 2: PBased($G = (V, E, L), \mathcal{K}$)

```

1  $E' \leftarrow \emptyset$ 
2 Escolha arbitrariamente um vértice  $r \in V$ 
3  $S \leftarrow \{r\}$ 
4 while  $|E'| < |V| - 1$  do
5   Seja  $\mathcal{K}' \subseteq \mathcal{K}$  o conjunto de CMMCs que fazem  $S$  crescer
6    $K \leftarrow \text{CMMC} \in \mathcal{K}'$  que minimize o incremento da representação do conjunto de
     vértices  $S$ 
7   Sejam  $E(K)$  e  $V(K)$ , respectivamente, o conjunto de arestas e vértices
     pertencentes ao CMMC  $K \in \mathcal{K}'$ 
8   Para cada  $e \in E(K)$  faça
9     Se  $E' \cup e$  não possui ciclos então
10       $E' \leftarrow E' \cup e$ 
11   Para cada  $v \in V(K)$  faça
12     Se  $v \notin S$  então
13       $S' \leftarrow S \cup v$ 
14 retorne  $E'$ 

```

Adicionalmente, Linares et al. (2018) demonstram que o PACRM permanece NP-Completo mesmo no caso de $|L| = 2$, isto é, mesmo se o grafo de entrada possuir apenas duas cores. E propõem um método baseado na meta-heurística GRASP que utilizou um mapeamento do PACRM para o Problema de Steiner de Rotulação Mínima e uma adaptação da heurística MVCA, a heurística foi denominada, Algoritmo do Ramo de Máxima Cobertura (ARMC). Nesta heurística, inicia-se com um conjunto de arestas solução $E' = \emptyset$, e são inseridas arestas em E' enquanto E' não for uma árvore geradora do GAR de entrada. Dado um ramo monocromático, que é um conjunto de arestas, onde toda aresta é incidente em um vértice v e possui o mesmo rótulo l , cada iteração do laço principal do ARMC seleciona, entre todos os ramos do GAR, o maior ramo que minimize o aumento de representação de E' .

3. Algoritmo Genético para o PACRM

O algoritmo genético (AG) proposto neste trabalho faz uso de uma abordagem híbrida (El-mihoub et al., 2006), que incorpora a estrutura tradicional deste método (Holland, 1992), acrescido de heurísticas para geração de novos indivíduos, gerenciamento populacional e melhoria dos indivíduos. Uma solução para o PACRM, é obrigatoriamente uma árvore geradora com $|V| - 1$ arestas. Cada indivíduo p no AG é representado por uma lista de arestas que constitui a árvore

geradora. No exemplo da Figura 2 (b), a solução exemplo seria representada como o indivíduo $p = \{(1, 2), (2, 3), (2, 4), (4, 5), (5, 6)\}$. Seja $E(v, p)$ o conjunto de arestas do indivíduo p que incidem sobre o vértice v , então a função avaliação de p é dada por:

$$fitness(p) = \sum_{v \in V} \left| \bigcup_{e \in E(v, p)} L(e) \right|.$$

O Algoritmo 3 descreve o AG proposto. Seja $G = (V, E, L)$ o GAR de entrada do algoritmo, a geração inicial do conjunto de indivíduos P é composta por $\alpha \times |V| - 2$ indivíduos aleatórios construídos pela função *construirArvore*(E), e 2 indivíduos determinísticos construídos pelas heurísticas *KBased* e *PBased*. Enquanto o tempo de execução do algoritmo não exceder T_{max} , novas gerações de indivíduos são construídas, onde em cada geração um subconjunto $P' \subset P$ é selecionado através de estratégias de elitismo e roleta controlados pelos parâmetros β e γ (linha 6). Este subconjunto P' passa por uma fase de reprodução e educação (linha 7), criando a nova geração de indivíduos P . Ao final, o algoritmo retorna o melhor indivíduo p^* avaliado pela função *fitness*.

A função *construirArvore*(.), descrita pelo Algoritmo 4, parte do pressuposto que o conjunto de arestas fornecido como entrada contém pelo menos uma árvore geradora. A lógica dele é baseada no algoritmo de Kruskal (Cormen et al., 2009), esta função adiciona $|V| - 1$ arestas à um conjunto de arestas T , de forma que T seja uma árvore geradora. Para tal, se visita o conjunto de arestas E de forma aleatória, e cada iteração do laço das linhas 5-10 realiza uma inspeção de uma aresta e , caso a adição de e à T não venha a causar ciclos neste, esta é adicionada à T . Uma estrutura de dados União-Busca (Cormen et al., 2009) é utilizada no laço das linhas e 3-4, e nas linhas 9 e 11, com fins de detecção de ciclos. Ao final do algoritmo T é retornado.

Algoritmo 3: AG($G = (V, E, L)$, α , β , γ , T_{max})

```

1  $P \leftarrow \emptyset, P' \leftarrow \emptyset$ 
2 Para  $i \leftarrow 0$  até  $\alpha \times |V| - 2$  faça
3    $P \leftarrow P \cup \{\text{construirArvore}(E)\}$ 
4  $P \leftarrow P \cup \{PBased(E), KBased(E)\}$ 
5  $p^* \leftarrow$  Indivíduo  $p \in P$  com menor fitness
6 Enquanto critério de parada  $T_{max}$  não atendido faça
7    $P' \leftarrow \text{selecionar}(P, \beta, \gamma)$ 
8    $P \leftarrow \text{gerarFilhos}(G, P', \beta, \gamma)$ 
9   Se  $fitness(p) < fitness(p^*)$ , sendo  $p \in P$  o indivíduo com menor fitness
     então
10     $p^* \leftarrow p$ 
11 retorne  $p^*$ 

```

O processo de seleção de indivíduos por meio da função *selecionar*(.) faz uso de dois métodos:

- **elitismo:** $\beta \times |P|$ indivíduos com menor *fitness* são selecionados, assegurando que as soluções de melhor qualidade prossigam para as futuras gerações; e
- **roleta:** $(1 - \beta - \gamma) \times |P|$ indivíduos são selecionados utilizando o método da roleta, para determinar a chance de um indivíduo ser selecionado por este método, é utilizada a função

Algoritmo 4: construirArvore(V, E)

```

1  $Candidatos \leftarrow E, T \leftarrow \emptyset, i \leftarrow 0$ 
2 Ordene o vetor  $Candidatos$  de forma aleatória
3 Para cada  $v \in V$  faça
4    $construcao(v)$ 
5 while  $|T| < |V| - 1$  do
6    $e \leftarrow Candidatos[i]$ 
7    $i \leftarrow i + 1$ 
8   Para cada  $(v_1, v_2) \in e$  faça
9     Se  $busca(v_1) \neq busca(v_2)$  então
10       $T \leftarrow T \cup e$ 
11       $união(v_1, v_2)$ 
12 retorne  $T$ 

```

rating. Seja p_i o i -ésimo indivíduo pertencente a P , sendo os elementos de P ordenados de forma decrescente pela função *fitness*, a função *rating* é dada por:

$$rating(p_i) = \frac{\sum_{k=0}^i fitness(p_k)}{\sum_{p \in P} fitness(p)},$$

a aplicação desta função tem como resultado atribuir uma chance maior de ser selecionado à aqueles indivíduos que possuem um valor de *fitness* menor.

Com base no conjunto de indivíduos $P', P' \subset P$, obtido pela função *selecionar(.)*, a função *gerarFilhos(.)*, ilustrado pelo algoritmo 5, é responsável por criar um novo conjunto de indivíduos C a partir do conjunto P' , C se encontra particionado em 3 partes formadas por estratégias de cruzamento e elitismo, sendo o tamanho destas partes controlados pelos parâmetros β e γ .

Algoritmo 5: gerarFilhos($G = (V, E, L), P', \beta, \gamma$)

```

1  $C \leftarrow \{\beta \times |P'| \text{ indivíduos } \in P' \text{ com o menor } fitness\}$ 
2 Para  $i \leftarrow 0$  até  $(1 - (\beta + \gamma)) \times |P'|$  faça
3    $p_1 \leftarrow$  indivíduo  $\in P'$  selecionado pela roleta
4    $p_2 \leftarrow$  indivíduo  $\in P'$  selecionado pela roleta e diferente de  $p_1$ 
5    $p \leftarrow construirÁrvore(p_1 \cup p_2)$ 
6    $C \leftarrow C \cup VND(p)$ 
7 Para  $i \leftarrow 0$  até  $\gamma \times |P'|$  faça
8    $p_1 \leftarrow construirÁrvore(E)$ 
9    $p_2 \leftarrow$  indivíduo  $\in P'$  selecionado aleatoriamente
10   $p \leftarrow construirÁrvore(p_1 \cup p_2)$ 
11   $C \leftarrow C \cup VND(p)$ 
12 retorne  $C$ 

```

O conjunto C é inicializado com os $\beta \times |P'|$ indivíduos, onde por meio da estratégia de elitismo, os indivíduos com menor *fitness* de P' prosseguem para a próxima geração (linha 1). Em seguida estratégias de cruzamento completam a geração. Na primeira estratégia, $(1 - (\beta + \gamma)) \times |P'|$ indivíduos são gerados, sendo cada indivíduo p formado pelo cruzamento entre dois indivíduos $p_1, p_2 \in P'$, que foram selecionados pela fase da roleta, este cruzamento é realizado pela função *construirArvore*($p_1 \cup p_2$) (linhas 3-7). Na segunda estratégia, $\gamma \times |P'|$ indivíduos são gerados, sendo cada indivíduo p formado pelo cruzamento de um indivíduo p_1 aleatório de P' e outro indivíduo p_2 aleatoriamente construído a partir do conjunto de arestas E de G , novamente, p recebe as arestas retornadas por *construirArvore*($p_1 \cup p_2$) (linhas 8-12).

Em ambos os casos, o procedimento construtivo *construirArvore*(.) é utilizado para obter uma árvore do conjunto de arestas resultante da operação de união, observe que se p_1 e p_2 são árvores geradoras de G , $p_1 \cup p_2$ contém pelo menos uma árvore geradora, sendo assim, *construirArvore*($p_1 \cup p_2$) tem como resultado um indivíduo válido. Adicionalmente, em cada indivíduo p obtido pelo cruzamento é realizada uma busca local, que denominamos de fase de educação.

Esta busca local implementa o método *Variable Neighborhood Descent* (VND), proposto por Mladenović e Hansen (1997), aplica sobre o indivíduo de entrada o conjunto de vizinhanças $N = \{Folha, Ciclo, Reconexão\}$, nesta ordem, enquanto puder trocar uma aresta da solução por outra que não esteja na solução e venha a resultar em um melhor indivíduo por meio da estratégia de primeira melhora.

Seja $l \in L$ um rótulo, e $v \in V$ um vértice, l é considerado um rótulo com representação única em v caso exista apenas uma única aresta incidente à v com o rótulo l . Seja $(i, j) \in E$ uma aresta, esta é considerada como:

- **Aresta com representação única:** se o rótulo l associado a (i, j) seja de representação única em um dos vértices i ou j ;
- **Aresta com representação duplamente única:** caso l seja de representação única em ambos os vértices i e j ;
- **Aresta sem representação única:** caso l não atenda a nenhum dos casos anteriores;

Sendo assim, os três movimentos de vizinhança utilizados são definidos como:

Folha: dado um vértice folha v e seu pai p , caso o rótulo associado a aresta (v, p) tenha representação única em p , esta aresta será substituída por outra que conecte v a árvore, e que o rótulo associado a esta não seja de representação única no novo pai de v após o movimento. A complexidade de tempo deste algoritmo é descrito como $O(|V|^2)$ no pior caso, para tal, se supõe que existam $O(|V|)$ folhas na solução, e para cada uma delas existem $O(|V|)$ arestas incidentes passíveis de serem utilizadas para o movimento. A Figura 3 mostra a realização deste movimento, para efeitos de ilustração os vértices folha estão marcados, em (a) a solução possui $Rt = 7$ e o vértice folha 6 está conectado à árvore pela aresta $(3, 6)$, cujo rótulo é de representação única em 3, $(3, 6)$ é removida em (b) e substituída em (c) por $(1, 6)$, cujo rótulo não é de representação única em 1 após o movimento, adicionalmente, a solução passa a ter $Rt = 6$;

Ciclo: adiciona uma aresta e que não pertencente a solução, desde que a adição de e não resulte em um aumento no custo na solução. No ciclo formado pela adição de e é removida uma aresta com representação única ou duplamente única, caso não exista tal aresta, e é removida. No pior caso,

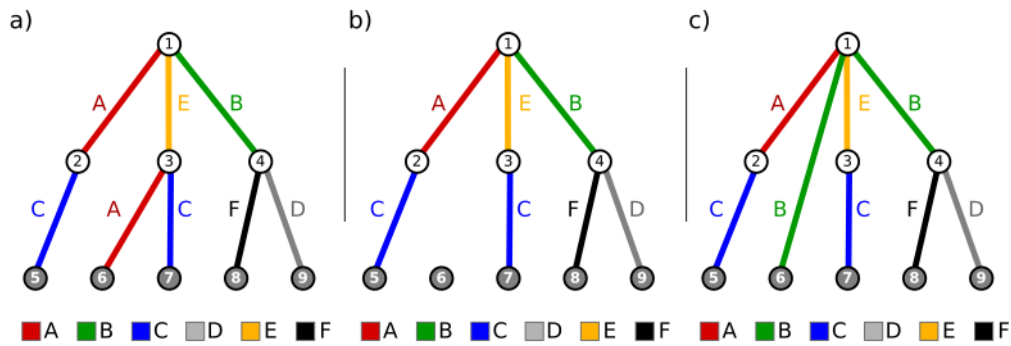


Figura 3: Movimento de troca de aresta de vértice folha.

considerando que existam $O(|E|)$ arestas possíveis de serem adicionadas, e que cada adição venha a resultar ciclo hamiltoniano, o movimento de vizinhança possui uma complexidade de $O(|V| \times |E|)$. A Figura 4 demonstra o movimento de ciclo, em (a) tem-se a solução inicial com $Rt = 6$, onde em (b) é feita a adição da aresta (5, 7) que não resulta em um incremento do Rt e forma um ciclo envolvendo as arestas (1, 3), (3, 7), (7, 5), (5, 2), (2, 1), das arestas do ciclo, é removida em (c) a aresta (1, 3), de representação duplamente única, o que faz da solução assumir $Rt = 4$;

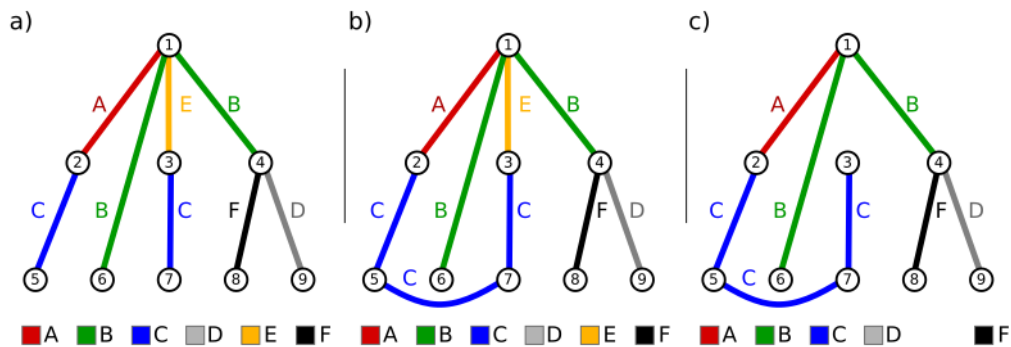


Figura 4: Movimento de troca de aresta por ciclo.

Reconexão: remove uma aresta e que liga dois vértices não folha, v_1 e v_2 , sendo e preferencialmente uma aresta com representação duplamente única, ou alternativamente na ausência de tal, uma aresta com representação única. A remoção de e resulta em duas partes de V , e em uma diminuição de um valor c do custo da solução. Após a remoção, é adicionada uma aresta $e' \neq e$ que reconecte as duas partes de V de forma que o custo de adição, c' , seja menor que c , caso não exista tal aresta, e é colocada de volta à solução. A complexidade do algoritmo é no pior caso $O(|V| \times |E|)$, neste caso, supõe-se que existam $O(|V|)$ arestas possíveis de serem removidas na árvore, e para cada uma delas há outras $O(|E|)$ arestas possíveis para se fazer a reconexão. A Figura 5 ilustra este movimento de vizinhança, para efeitos de ilustração, as duas partições de V criadas pelo movimento estão em cores distintas, inicialmente em (a) a solução possui $Rt = 4$, em (b) a aresta de representação duplamente única (1, 2) é removida, o que gera duas partições de V e uma redução temporária de 2 do custo, em (c) a aresta (3, 4) é adicionada para reconectar as partes de V , (3, 4) é de representação única, e portanto adiciona 1 ao custo da solução, o que faz de $Rt = 3$ ao fim do movimento;

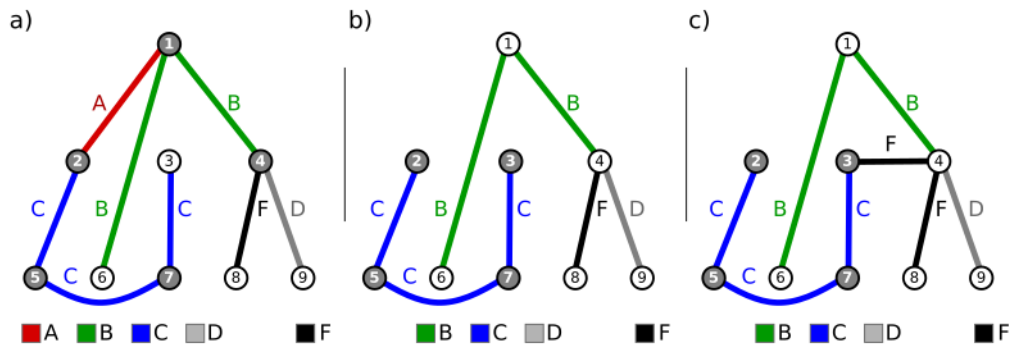


Figura 5: Movimento de troca de aresta por reconexão.

4. Experimentos computacionais

Os métodos estudados neste trabalho foram desenvolvidos em linguagem C++, utilizando o compilador Apple clang versão 12.0.0 (opção de compilação -O3). Os experimentos computacionais foram executados em um computador com processador Intel Core i5, de 4 núcleos de 2,3 GHz, adicionalmente, a máquina dispõe de 8 GB de memória RAM, e possui como sistema operacional o MacOS Catalina.

Para os experimentos realizados, GARs gerados de forma aleatória por Cerulli et al. (2005) foram utilizados. Para cada $n = |V| = \{20, 30, 40, 50\}$, as instâncias possuem 30 grafos-problema, que são divididos em 3 grupos definidos pelos valores de $d = \{ld = 20\%; md = 50\%; hd = 80\%\}$, representando a densidade do grafo. Cada grupo possui 10 instâncias distintas de mesma dimensão, totalizando 120 grafos-problema utilizados nos experimentos computacionais.

4.1. Estudo de parâmetros

Efetuamos testes de parametrização de (α, β, γ) , em um conjunto aleatório de instâncias teste, cada instância é denotada pela tupla (n, d, i) , sendo i a i -ésima instância de tamanho n , e densidade d :

$$teste = \{(20, hd, 1), (30, hd, 0), (40, hd, 4), (50, ld, 1), (50, md, 8), (50, hd, 5)\}.$$

Inicialmente, fixamos o parâmetro $\alpha = 1$, e foram testadas todas as combinações de valores $\gamma, \beta \in \{0, 0.25, 0.5, 0.75, 1\}$, com a restrição de que $0 \leq \beta + \gamma \leq 1$. Para cada combinação possível, 10 execuções foram realizadas para cada instância, com um tempo limite de 10 s para cada execução. A combinação $(\beta, \gamma) = (0.25, 0.25)$ se mostrou a melhor em termos do valor da solução. Em seguida, utilizando o procedimento descrito anteriormente, fixamos $(\beta, \gamma) = (0.25, 0.25)$, e analisamos os valores $\alpha \in \{0.5, 1, 2, 3\}$. Como resultado, o que se notou é que a combinação, $(\alpha, \beta, \gamma) = (1, 0.25, 0.25)$ se mostrou a melhor em relação ao valor da solução obtido.

4.2. Estudo de vizinhanças

A Figura 6 ilustra o número de vezes que os movimentos de vizinhança foram responsáveis por gerar a última melhoria de uma solução em uma execução. Os resultados foram obtidos por meio de 10 execuções em cada instância, e demonstram que todas as três vizinhanças conseguiram realizar a última melhoria pelo menos uma vez em todas as densidades, sendo o movimento de ciclo o maior responsável, note que para densidades maiores, os movimentos de reconexão e folha possuem uma participação maior na obtenção da última melhoria.

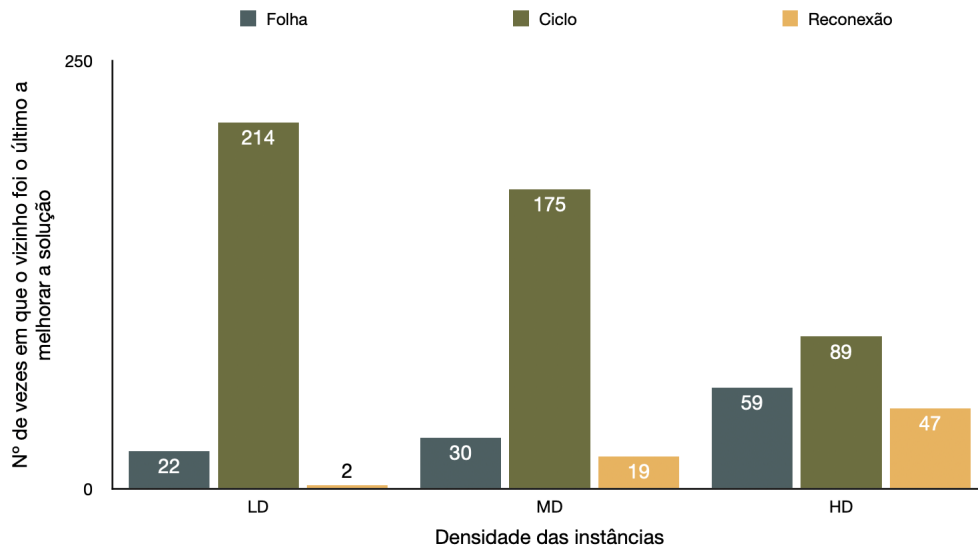


Figura 6: Gráfico de barras com o número de vezes em que uma vizinhança foi a última a melhorar a solução, com as instâncias agrupadas pela sua densidade.

4.3. Comparação com outros métodos

Implementamos e comparamos o Algoritmo Genético (AG) proposto na Seção 3 com as heurísticas determinísticas *KBased* e *PBased* propostas por Silva (2018) e a meta-heurística, intitulada MBCA, proposta por Linares et al. (2018). Todas as estratégias tiveram um tempo limite de 10 s para a execução de cada instância. As estratégias não determinísticas AG e MBCA foram executadas 10 vezes para cada instância.

A Tabela 1 apresenta a comparação destas heurísticas para os conjuntos de vértices de tamanho $|V| = \{20, 30, 40, 50\}$. Cada linha descreve resultados médios de 10 instâncias. O conjunto de colunas Inst. contém as mesmas colunas da tabela anterior com a adição da coluna BKS que descreve a média dos valores das melhores soluções encontradas pelo método exato em Silva (2018). Os conjuntos de colunas *KBased*, *PBased*, MBCA e AG representam os resultados de cada método heurístico, para o AG, os parâmetros utilizados são $(\alpha, \beta, \gamma) = (1, 0.25, 0.25)$. Os conjuntos *KBased* e *PBased* são divididos em: $t(ms)$ que representa o tempo computacional médio, em milissegundos, das 10 instâncias; Rt que representa o valor médio das soluções encontradas e $\Delta\% = (Rt - BKS) / BKS * 100$. Os conjuntos MBCA e AG são divididos em: $t(ms)$ que representa o tempo computacional médio, em milissegundos, em que o método encontrou a melhor solução em cada execução; Rt_{min} que representa o valor médio da melhor solução encontrada nas 10 execuções de cada instância; Rt_{med} que representa a média, computada das 10 instâncias, dos valores médio das soluções encontradas em 10 execuções de cada instância, e $\Delta\% = (Rt_{med} - BKS) / BKS * 100$.

O método AG, comparado aos métodos heurísticos da literatura, obteve as melhores soluções médias Rt_{med} em todas as 120 instâncias testadas, observando seus melhores valores médios (Rt_{min}), o AG encontrou o resultado do método exato em 80 instâncias e melhorou o *upper bound* médio dos outros 4 grupos de 10 instâncias, precisando de no máximo 2 s para encontrar estas soluções em todas as instâncias.

Adicionalmente, comparamos o AG com o GRASP proposto em Linares et al. (2018), foram utilizadas as mesmas instâncias descritas nesta dissertação, onde cada instância foi executada 10 vezes. Os resultados estão apresentados na Tabela 2, a coluna Rt é o valor da melhor

Tabela 1: Resultados dos métodos heurísticos para grafos com $|V| \in \{20, 30, 40, 50\}$

$ V $	Inst.		KBased			PBased			MBCA				AG			
	d	BKS	t(ms)	Rt	$\Delta\%$	t(ms)	Rt	$\Delta\%$	t(ms)	Rt _{min}	Rt _{med}	$\Delta\%$	t(ms)	Rt _{min}	Rt _{med}	$\Delta\%$
20	ld	32,3	1,2	32,4	0,3	1,5	32,5	0,6	53,2	33,1	33,1	2,5	2,7	32,3	32,3	0
	md	25,7	1,6	28,3	10,1	1,4	26,1	1,6	51,4	28	28	8,9	49,6	25,7	25,7	0
	hd	22,8	1,4	26,4	15,8	1,3	23,4	2,6	51,6	25,8	25,8	13,2	377	22,8	22,8	0,1
30	ld	45,3	3,2	46,9	3,5	3,7	45,8	1,1	164,9	46,3	46,3	2,2	55,1	45,3	45,3	0
	md	36,7	3,8	42	14,4	3,5	38,1	3,8	161,0	40,9	40,9	11,4	1s	36,7	36,8	0,3
	hd	33,8	3,2	38,8	14,8	3,1	34,7	2,7	168,7	38,8	38,8	14,8	1s	33,8	33,9	0,4
40	ld	58,9	7,4	61,9	5,1	8,2	59,9	1,7	404,8	61,3	61,3	4,1	556	58,9	58,9	0
	md	49	8,2	57,2	16,7	7,2	50	2,0	435,7	51,2	51,2	4,5	1,8 s	48,7	48,9	-0,2
	hd	47,3	6,5	53,1	12,3	6,5	46,1	-2,5	435,7	51,2	51,2	8,2	2 s	44,9	45,3	-4,3
50	ld	71,7	13,4	77,7	8,4	13,4	73,7	2,8	772,8	75,3	75,3	5,0	1,8 s	71,7	71,9	0,3
	md	61,3	15,4	71	15,8	12,0	61,5	0,3	738,5	67	67	9,3	1,6 s	60,2	60,8	-0,9
	hd	65,2	11,0	66,5	2,0	11,6	57,2	-12	820,5	63,9	63,9	-2,0	1,1 s	56,6	56,9	-13

solução encontrada para dada instância, $t(s)$ indica o tempo computacional médio, em segundos. $N<|V|>C<|L|><D><i>$ é o formato do nome de cada instância, onde $|V|$ é o número de vértices, $|L|$, o de rótulos, D denota a densidade, $D \in \{LD, MD, HD\}$, e i é a i -ésima instância de densidade D com $|V|$ vértices, e $|L|$ rótulos. Para instâncias com $|V| \leq 50$, o GRASP foi melhor que o AG em uma delas, sendo o valor de Rt idêntico para ambos os métodos nos outros casos, em relação a instâncias de tamanho maior ($|V| > 50$), o AG se mostrou melhor que o GRASP em todas as instâncias, tanto em relação ao Rt quanto ao tempo computacional.

Tabela 2: Resumo dos resultados dos métodos GRASP e AG.

Inst.	GRASP		AG		Inst.	GRASP		AG	
	Rt	t(s)	Rt	t(s)		Rt	t(s)	Rt	t(s)
N020C020HD1	2	0,08	2	0,40	N100C100MD1	21	279,85	19	1,48
N020C020MD1	4	0,02	4	0,00	N100C100MD6	23	227,57	19	0,00
N030C030HD0	4	0,33	4	0,71	N100C100MD9	22	11,73	20	0,00
N030C030MD0	7	0,10	7	1,34	N100C050HD4	7	190,40	6	0,00
N030C030LD1	15	0,48	15	0,09	N100C050HD7	7	59,72	6	0,00
N040C040HD0	5	4,70	5	2,80	N100C050HD9	6	6,47	4	0,00
N050C050LD0	23	1,87	23	2,18	N100C100HD0	16	81,03	13	0,00
N050C050LD1	25	21,66	25	2,48	N100C100HD1	16	137,26	12	0,00
N050C050MD8	9	9,79	10	0,33	N100C100HD5	15	180,41	12	0,00
N050C050HD8	5	14,89	5	3,87	N100C100HD9	14	21,37	13	0,00

5. Considerações finais e proposta de trabalhos futuros

Neste trabalho, abordamos o Problema da Árvore de Cobertura com Representação Mínima (PACRM), neste, objetiva-se encontrar uma árvore de cobertura que minimize as cores representadas em cada vértice, tendo este problema aplicações, por exemplo, em projetos de redes de transporte multimodais.

Propomos uma adaptação do Algoritmo Genético (AG) com uso de conceitos de heurísticas de construção de árvores espalhadas, além de novas estruturas de movimentos de vizinhança para refinar estas soluções. Comparado aos métodos heurísticos da literatura, o AG encontrou, na

média, melhores soluções em todas as instâncias, mantendo um tempo médio de no máximo 1,9 s, o que demonstra sua eficiência, quanto a qualidade das soluções o AG melhorou o *upper bound* encontrado pelos métodos exatos em 4 grupos de 10 instâncias.

Como planejamento de trabalhos futuros, pretendemos implementar meta-heurísticas com o conceito de trajetória de soluções como o ILS e VNS. Adicionalmente, como visto na Seção 4, o AG proposto, comparado ao método baseado na meta-heurística GRASP, proposto em Linares et al. (2018), obteve, para instâncias de $|V| \leq 50$, resultados iguais em 9 instâncias, perdendo em apenas uma, entretanto, para instâncias com $|V| > 50$, o AG foi melhor em todas as 10 instâncias testadas. Por isto, pretendemos elaborar melhorias para o GRASP, incorporando os movimentos de vizinhanças, e heurísticas construtivas propostas neste trabalho.

Referências

- Cerrone, C., D'Ambrosio, C., e Raiconi, A. (2019). Heuristics for the strong generalized minimum label spanning tree problem. *Networks*, 74.
- Cerulli, R., Fink, A., Gentili, M., e Raiconi, A. (2014). The k-labeled spanning forest problem. *Procedia - Social and Behavioral Sciences*, 108:153–163. ISSN 1877-0428.
- Cerulli, R., Fink, A., Gentili, M., e Voss, S. (2005). Metaheuristics comparison for the minimum labelling spanning tree problem. In Golden, B., Raghavan, S., e Wail, E., editors, *The Next Wave in Computing, Optimization, and Decision Technologies*, volume 29 of *Operations Research/Computer Science Interfaces Series*, p. 93–106. Springer US. ISBN 978-0-387-23528-8.
- Cerulli, R., Fink, A., Gentili, M., e Voss, S. (2006). Extensions of the minimum labelling spanning tree problem. *Journal of Telecommunications and Information Technology*.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition. ISBN 0262033844.
- El-mihoub, T., Hopgood, A., Nolle, L., e Alan, B. (2006). Hybrid genetic algorithms: A review. *Engineering Letters*, 3(2).
- Figueirêdo, C. O. d., Gouveia, T., Fragoso, F. C., e Filho, G. F. d. S. (2020). Métodos exatos para o problema do diâmetro colorido. *52º Simpósio Brasileiro de Pesquisa Operacional*.
- Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.
- Linares, E. D. Z., Satoru, L., Silva, T. G., e Interian, R. (2018). The minimum representation spanning tree problem. *Anais do Simpósio Brasileiro de Pesquisa Operacional*, 274.
- Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100. ISSN 0305-0548. URL <http://www.sciencedirect.com/science/article/pii/S0305054897000312>.
- Silva, T. G. (2018). *The minimum labeling spanning tree and related problems*. PhD thesis, Universidade Federal Fluminense.