

Meta-heurísticas para o Problema da Árvore Geradora com Rotulação Mínima e Restrições Orçamentárias

Bianca Lima de Carvalho Oliveira
Kervén Maciel Monteiro de Albuquerque
Emanuel Eduardo Oliveira Dionísio
Thiago Gouveia

Instituto Federal da Paraíba, João Pessoa/PB
bialimacel@gmail.com

RESUMO

O Problema da Árvore Geradora com Rotulação Mínima e Restrições Orçamentárias é um problema de conectividade em grafos com arestas rotuladas, cujo objetivo é encontrar uma árvore geradora minimizando o número de rótulos utilizados, respeitando limites sobre a quantidade máxima de arestas permitidas por rótulo. Para sua resolução, foram propostos dois métodos heurísticos baseados nas meta-heurísticas GRASP e BRKGA, além de uma formulação baseada em cortes com um algoritmo de branch-and-cut, utilizada para validar os métodos heurísticos. Os experimentos computacionais demonstraram que o método baseado em GRASP encontrou a solução ótima em todas as instâncias nas quais a formulação convergiu dentro do tempo limite. Também obteve resultados melhores e com desempenho superior ao método baseado em BRKGA.

PALAVRAS CHAVE. Grafos com arestas rotuladas. Árvores geradoras. GRASP. BRKGA.

MH - Metaheurísticas, TAG – Teoria dos Grafos e Algoritmos Relacionados.

ABSTRACT

The Budgeted Minimum Labeling Spanning Tree Problem is a connectivity problem in edge-labeled graphs, aiming to find a spanning tree that minimizes the number of labels used, while respecting upper bounds on the number of edges allowed per label. To address this problem, two heuristic methods were proposed, based on the GRASP and BRKGA metaheuristics, along with a cut-based formulation and a branch-and-cut algorithm, used to validate the heuristic methods. Computational experiments showed that the GRASP-based method found the optimal solution in all instances where the formulation converged within the time limit. It also outperformed the BRKGA-based method in both solution quality and computational performance.

KEYWORDS. Edge-labeled graphs. Spanning trees. GRASP. BRKGA.

Metaheuristics, Graph Theory and Related Algorithms.

1. Introdução

Nos últimos anos, muitos problemas em grafos com arestas rotuladas têm ganhado destaque na literatura [Zhang, 2024; Long et al., 2023; Yang et al., 2021]. Um desses problemas é o Problema da Árvore Geradora com Rotulação Mínima (PAGRM), introduzido por Chang e Leu [1997], e que objetiva encontrar uma árvore geradora minimizando o número de rótulos utilizados. Uma variação proposta por Cerulli et al. [2024] é o Problema da Árvore Geradora Mínima com Restrições Orçamentárias nos Rótulos (PAGMROR), que visa minimizar o custo total da árvore geradora, respeitando restrições orçamentárias na quantidade de arestas de cada rótulo.

Este trabalho introduz o Problema da Árvore Geradora com Rotulação Mínima e Restrições Orçamentárias (PAGRMRO), outra variação do PAGRM que permite limitar a quantidade de arestas por rótulo. Formalmente, seja um grafo com arestas rotuladas $G = (V, E, L)$, onde V representa o conjunto de vértices, E o conjunto de arestas e L o conjunto de rótulos. Cada aresta $e \in E$ está associada a um rótulo $\ell(e) \in L$. Seja, também, uma função de restrições orçamentárias $w : L \rightarrow \mathbb{Z}^+$. O objetivo do PAGRMRO é encontrar uma árvore geradora $T = (V, E', L')$ que minimize a quantidade $|L'|$ de rótulos presentes em T e tal que $|\mathcal{E}_T(l)| \leq w(l)$ para cada $l \in L$, onde $\mathcal{E}_T(l) = \{e \in E' \mid \ell(e) = l\}$ representa o conjunto de arestas em T com rótulo l . Um exemplo de instância é ilustrado na Figura 1.

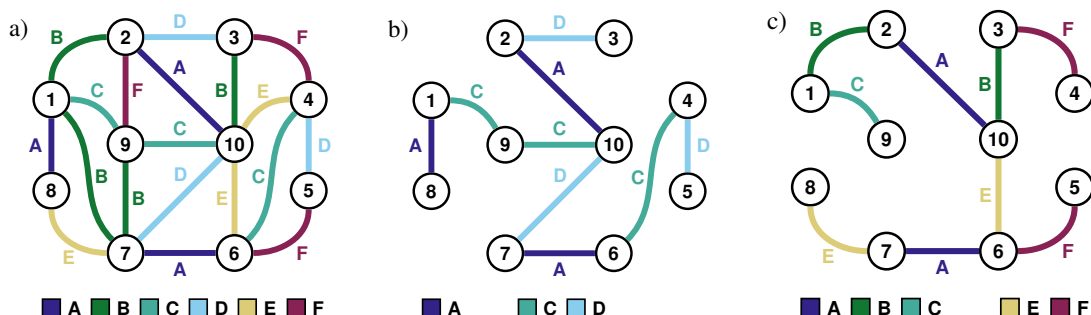


Figura 1: Grafo com arestas rotuladas com 10 vértices e 6 rótulos (a); solução do PAGRMRO com restrição de até 3 arestas por rótulo (b); e solução do PAGRMRO com restrição de até 2 arestas por rótulo (c)

A Figura 1(a) apresenta um grafo com arestas rotuladas com 10 vértices, 19 arestas e 6 rótulos. O rótulo de cada aresta é representado por uma letra maiúscula e pela cor da aresta. Para esse grafo, a Figura 1(b) exemplifica uma solução do PAGRMRO com restrição de 3 arestas por rótulo. Nessa solução, $L' = \{A, C, D\}$, com custo $|L'| = 3$. A Figura 1(c) exemplifica outra solução, para o mesmo grafo, mas com restrição de 2 arestas por rótulo. Nesse caso, a solução tem $L' = \{A, B, C, E, F\}$ e custo $|L'| = 5$.

O PAGRM possui várias aplicações práticas, como o projeto de redes de comunicação, redes de transporte multimodal e compressão de dados. Por exemplo, em redes de comunicação, busca-se conectar diversos dispositivos utilizando o menor número possível de tipos de meio de comunicação, reduzindo tanto os custos quanto a complexidade da rede [Lin et al., 2020]. A introdução de restrições no PAGRMRO torna o problema mais aplicável a cenários reais, garantindo que a solução esteja adequada ao orçamento disponível.

O PAGRMRO é um problema NP-difícil, por se tratar de uma generalização do PAGRM, que também é NP-difícil [Chang e Leu, 1997]. Para abordá-lo, propomos dois métodos heurísticos baseados nas meta-heurísticas GRASP (do inglês, *Greedy Randomized Adaptive Search Procedure*) e BRKGA (do inglês, *Biased Random-Key Genetic Algorithm*), além de uma formulação baseada em cortes, com um algoritmo de *branch-and-cut*, para validar os métodos heurísticos.

Este trabalho está organizado em sete seções, incluindo esta introdução. A Seção 2 apresenta uma revisão de trabalhos relacionados ao PAGRMRO. A Seção 3 descreve a formulação proposta e o algoritmo de *branch-and-cut*. A Seção 4 detalha o método heurístico baseado em GRASP e a Seção 5 detalha o método baseado em BRKGA. Os resultados dos experimentos computacionais são discutidos na Seção 6. Por fim, a Seção 7 traz as considerações finais do trabalho.

2. Trabalhos Relacionados

Diversos métodos têm sido propostos na literatura para resolver problemas de conectividade em grafos com arestas rotuladas, incluindo o PAGRM e suas variantes. Um desses métodos é o *Carousel Greedy*, proposto por Cerrone et al. [2017], que se mostrou promissor na solução do PAGRM e de outros problemas de otimização combinatória. A construção da solução ocorre em três etapas: geração de uma solução parcial por meio de um algoritmo guloso, modificação determinística dessa solução e posterior finalização com o mesmo algoritmo.

Lin et al. [2020], por sua vez, propuseram o algoritmo *Binary Firefly* para o PAGRM. Trata-se de uma adaptação do algoritmo *Firefly*, que converge para a solução ótima dentro do espaço de soluções, usando um comportamento semelhante ao de um enxame de vaga-lumes. O algoritmo emprega um método de operação binária para atualizar a posição de um vaga-lume e um método de manipulação local para garantir uma solução viável. Essas melhorias tornaram o algoritmo mais adequado para problemas discretos, e seus resultados na resolução do PAGRM são comparáveis aos de outras meta-heurísticas.

Outra solução proposta na literatura para PAGRM é o método de *Cross-Entropy* de Vaisman [2021]. Semelhante a outros algoritmos evolutivos, utiliza os princípios da teoria da informação e da simulação estocástica. Sua eficácia foi comparada com outras estratégias, como o *(1+1) Evolutionary Algorithm*, o *Global Simple Evolutionary Multiobjective Optimizer* e o Algoritmo Genético. Os resultados da análise demonstraram que o *Cross-Entropy* exibe um desempenho comparável aos outros métodos, além de ser menos sensível às características dos grafos, conseguindo alcançar uma solução ótima ou próxima independentemente das particularidades das instâncias.

Em relação às variantes do PAGRM, o Problema da Árvore Geradora de Rotulação Mínima Generalizada foi introduzido por Chen et al. [2008] e explorado em detalhes por Silva et al. [2019]. Trata-se de uma extensão do PAGRM que permite a atribuição de múltiplos rótulos a uma única aresta. Silva et al. [2019] propuseram uma nova formulação matemática baseada no conceito de cortes coloridos e que alcançou um desempenho superior ao estado da arte tanto para o PAGRM quanto para a versão generalizada.

Outra variante é o Problema da Árvore Geradora de Rotulação Mínima Fortemente Generalizada proposto por Cerrone et al. [2019]. Nesse problema, cada aresta pode ter múltiplos rótulos associados e todos são considerados ao incluir a aresta na solução. Eles propuseram uma formulação matemática, uma heurística construtiva, um algoritmo baseado no *Carousel Greedy* e uma meta-heurística híbrida chamada *Carousel-Pilot*. Para o mesmo problema, Long et al. [2023] propuseram uma abordagem decompondo em dois subproblemas: buscar um subgrafo conectado com rotulação mínima a partir do grafo original e, a partir disso, buscar uma árvore geradora do subgrafo. Para encontrar o subgrafo, foi proposto um algoritmo de pilotagem em zigue-zague baseado em comunidades, considerando as relações de coexistência de rótulos, particionando os rótulos e encontrando uma solução inicial, melhorada pela pilotagem em zigue-zague.

Yang et al. [2021] propõem o estudo de uma classe de problemas de otimização em grafos com arestas rotuladas. Eles convertem as versões com rotulação de alguns problemas clássicos, como o Problema da Cobertura em Vértices Mínima e o Problema do Conjunto Independente Máximo, nas versões originais por meio de uma redução em tempo polinomial. Eles também generalizam uma abordagem gulosa do Problema da Cobertura Submodular para sua versão com

rotulação e usam isso para resolver a versão ponderada e rotulada do PAGRM.

Entre os trabalhos mais recentes, Cerulli et al. [2024] propuseram três formulações matemáticas para o PAGMROR, baseadas em fluxo, empacotamento e penalidades, além de uma abordagem lagrangiana com subgradiente para obtenção de limites inferiores. A formulação baseada em fluxo apresentou melhor desempenho em grande parte dos testes.

3. Formulação

Seja um grafo com arestas rotuladas $G = (V, E, L)$, onde V é o conjunto de vértices, E é o conjunto de arestas e L é o conjunto de rótulos. Cada aresta $e \in E$ possui um rótulo $\ell(e) \in L$ associado. Seja, também, uma função de restrições orçamentárias $w : L \rightarrow \mathbb{Z}^+$. Dado um rótulo $l \in L$, o conjunto $\mathcal{E}(l) = \{e \in E \mid \ell(e) = l\}$ representa as arestas com rótulo l . Dado um corte $S \subsetneq V$, $S \neq \emptyset$, o conjunto $\delta(S)$ representa as arestas do corte, ou seja, que possuem uma extremidade em S e a outra em $V \setminus S$. A partir dessas notações, uma formulação baseada em cortes para o PAGRMRO é apresentada no Programa (1) a (7).

$$\min \sum_{l \in L} z_l \quad (1)$$

sujeito a

$$z_{\ell(e)} \geq y_e, \quad \forall e \in E, \quad (2)$$

$$\sum_{e \in \mathcal{E}(l)} y_e \leq w(l), \quad \forall l \in L, \quad (3)$$

$$\sum_{e \in E} y_e = |V| - 1, \quad (4)$$

$$\sum_{e \in \delta(S)} y_e \geq 1, \quad \forall S \subsetneq V, S \neq \emptyset, \quad (5)$$

$$z_l \in \{0, 1\}, \quad \forall l \in L, \quad (6)$$

$$y_e \in \{0, 1\}, \quad \forall e \in E. \quad (7)$$

O objetivo do programa é minimizar a quantidade de rótulos presentes na solução, como mostrado na Equação (1). As variáveis binárias z_l representam a presença de cada rótulo na solução, com domínio descrito na Equação (6). As variáveis binárias y_e representam a presença de cada arco na solução, com domínio descrito na Equação (7). As Restrições (2) garantem que cada aresta só será utilizada caso o rótulo correspondente esteja presente na solução. Já as Restrições (3) limitam a quantidade de arcos por rótulo às restrições orçamentárias do problema.

A definição da árvore geradora é descrita pelas Restrições (4) e (5). As Restrições (5) garantem que as arestas utilizadas formem um subgrafo gerador conexo. Para isso, pelo menos uma aresta de cada corte no grafo deve ser incluída. Caso contrário, o corte desconectaria o grafo. Além disso, um subgrafo gerador conexo com exatamente $|V| - 1$ arestas, como garantido pela Restrição (4), é uma árvore geradora. Note-se, no entanto, que há uma quantidade exponencial de cortes no grafo, sendo impraticável adicionar todas as restrições ao modelo. Para isso, é proposta uma abordagem de *branch-and-cut*, como descrito no Algoritmo 1.

O Algoritmo 1 recebe um grafo com arestas rotuladas G e uma função w que define as restrições orçamentárias da instância. A linha 2 inicia com o modelo matemático proposto, mas ainda sem as Restrições (5), referentes aos cortes do grafo. A solução do problema é representada por um conjunto de arestas S , inicialmente vazio. Então, o laço das linhas 4–8 é repetido enquanto o número de componentes conexas no subgrafo gerador $G[S]$ for maior que 1, ou seja, enquanto $G[S]$

não for conexo. Inicialmente, cada vértice tem a sua própria componente conexa, o que garante que o laço será executado pelo menos uma vez, desde que $|V| > 1$.

Algoritmo 1: Algoritmo de *branch-and-cut* para a formulação baseada em cortes

```

1 Procedimento BRANCH-AND-CUT ( $G = (V, E, L)$ ,  $w : L \rightarrow \mathbb{Z}^+$ )
2   Seja  $P$  um modelo de programação linear inteira de acordo com o Programa (1) a (7), exceto (5);
3    $S \leftarrow \emptyset$ ;
4   Enquanto número de componentes conexas de  $G[S] > 1$  faça
5     Para cada componente conexa  $C = (V', E', L')$  de  $G[S]$  faça
6       Adicione ao modelo  $P$  a restrição (5) referente ao corte  $(V', V \setminus V')$ ;
7     Resolva o modelo  $P$ ;
8      $S \leftarrow$  conjunto de arestas obtido como solução de  $P$ ;
9   Retorne  $S$ ;
```

Enquanto $G[S]$ não for conexo, faz-se necessário adicionar arestas que conectem as componentes conexas entre si. Como cada componente define um corte, essa necessidade pode ser satisfeita pela Restrição (5) correspondente, que é adicionada ao modelo, para cada componente, nas linhas 5 e 6. A linha 7 resolve o modelo e, caso não haja solução, encerra o algoritmo. Havendo solução, o processo é repetido até que o subgrafo seja conexo. Cada Restrição (5) adicionada ao modelo garante que o conjunto de arestas do corte correspondente em $G[S]$ não é vazio, logo a linha 6 será executada no máximo uma vez para cada corte em G e a condição da linha 4 será eventualmente satisfeita, desde que a instância tenha solução.

4. GRASP

Esta seção discute o primeiro dos métodos heurísticos propostos neste trabalho, baseado na meta-heurística GRASP e descrito no Algoritmo 2. A meta-heurística GRASP consiste em uma construção gulosa aleatorizada e uma busca local, e novas soluções são geradas até um critério de parada ser atingido [Saad et al., 2018]. Neste caso, o critério é o tempo limite.

Algoritmo 2: Método heurístico baseado na meta-heurística GRASP

```

1 Procedimento GRASP ( $G = (V, E, L)$ ,  $w : L \rightarrow \mathbb{Z}^+$ ,  $\alpha$ ,  $n_T$ )
2    $S^* \leftarrow$  SOLUÇÃO-NÃO-ENCONTRADA;
3   Enquanto tempo < tempo limite faça
4      $S \leftarrow$  BUSCA-LOCAL ( $G$ ,  $w$ , CONSTRUÇÃO ( $G$ ,  $w$ ,  $\alpha$ ),  $n_T$ );
5     Se  $f(S) < f(S^*)$  então
6        $S^* \leftarrow S$ ;
7   Retorne  $S^*$ ;
```

A entrada do algoritmo é um grafo com arestas rotuladas $G = (V, E, L)$; uma função de restrições orçamentárias w ; um parâmetro $\alpha \in [0, 1]$, que representa o nível de aleatoriedade do GRASP, onde 0 seria totalmente guloso e 1, totalmente aleatório; e um parâmetro $n_T \in \mathbb{Z} \geq 1$ que determina o número de tentativas de melhoria feitas na busca local. Como saída, é retornado um conjunto S^* de arestas, tal que o subgrafo gerador $G[S^*]$ é uma solução viável do problema, uma árvore geradora que respeita as restrições orçamentárias.

Inicialmente, tem-se S^* como solução não encontrada, definido na linha 2. Então, uma solução S é gerada por meio dos procedimentos de construção e busca local, na linha 4, e a melhor solução é mantida em S^* , através do condicional das linhas 5–6. O processo repete-se enquanto o tempo limite não for atingido. A comparação da melhor solução é feita por meio da função objetivo f . Dado um subconjunto de arestas $S \subset E$, o conjunto $\mathcal{L}(S) = \{l \in L \mid \exists e \in S, \ell(e) = l\}$

representa os rótulos presentes nas arestas em S e o valor da função objetivo é definido como $f(S) = |\mathcal{L}(S)|$, se $G[S]$ é uma árvore geradora, ou $f(S) = \infty$, caso contrário.

O procedimento de construção é apresentado no Algoritmo 3. Ele recebe um grafo, uma função de restrições orçamentárias e o parâmetro α . Inicia-se com uma solução vazia na linha 2, à qual são adicionadas arestas no laço das linhas 3–12, mantendo uma floresta até ela se tornar apenas uma árvore, quando $|S| = |V| - 1$. Para isso, as linhas 4–5 ordenam os rótulos pela quantidade de arestas que ainda podem ser adicionadas à solução, ou seja, a diferença entre o número de componentes conexas no subgrafo atual, $G[S]$, e no subgrafo obtido incluindo todas as arestas do rótulo l em questão, $G[S \cup \mathcal{E}(l)]$, limitada à restrição orçamentária do rótulo.

Algoritmo 3: Construção gulosa aleatorizada

```

1  Procedimento CONSTRUÇÃO ( $G = (V, E, L)$ ,  $w : L \rightarrow \mathbb{Z}^+$ ,  $\alpha$ )
2   $S \leftarrow \emptyset$ ;
3  Enquanto  $|S| < |V| - 1$  faça
4      Seja  $n(l) = \min \{w(l), \text{diferença no número de componentes de } G[S] \text{ e } G[S \cup \mathcal{E}(l)]\}$ ,  $\forall l \in L$ ;
5      Seja  $L'$  uma ordenação de  $\{l \in L \mid n(l) > 0\}$  não decrescente em  $n(l)$ ;
6      Se  $|L'| = 0$  então
7           $\quad$  Retorne SOLUÇÃO-NÃO-ENCONTRADA;
8       $k \leftarrow \lceil \alpha |L'| \rceil$ ;
9      Enquanto  $k < |L'|$  e  $L'[k] = L'[k + 1]$  faça
10          $\quad$   $k \leftarrow k + 1$ ;
11     Seja  $l'$  um rótulo tomado aleatoriamente de  $L'[1 : k]$ ;
12      $S \leftarrow \text{FLORESTA-GERADORA}(G, w, S, \mathcal{E}(l'))$ ;
13 Retorne  $S$ ;

```

Dentre os k melhores rótulos, as linhas 11 e 12 selecionam um rótulo l' aleatório, dentre os que com maior valor n , e adicionam arestas aleatórias de $\mathcal{E}(l')$, enquanto possível, desde que não incluam ciclos no subgrafo. O número k de rótulos é definido de forma a representar os $\alpha\%$ rótulos com maior valor de n , incluindo repetições, como mostrado nas linhas 8–10. Se não houver mais rótulos que possam ser selecionados, o condicional das linhas 6–7 retorna que não encontrou uma solução. Caso contrário, a linha 13 retorna a solução construída S . O procedimento que adiciona arestas à solução atual, baseado no algoritmo de Kruskal [1956], é descrito no Algoritmo 4.

Algoritmo 4: Construção de floresta geradora com restrições orçamentárias

```

1  Procedimento FLORESTA-GERADORA ( $G = (V, E, L)$ ,  $w : L \rightarrow \mathbb{Z}^+$ ,  $F \subseteq E$ ,  $E' \subseteq E$ )
2  Enquanto  $|E'| > 0$  faça
3      Seja  $e = (u, v)$  uma aresta tomada aleatoriamente de  $E'$ ;
4       $E' \leftarrow E' \setminus \{e\}$ ;
5      Se  $|\mathcal{E}(\ell(e)) \cap F| < w(\ell(e))$  e  $u, v$  estão em componentes distintas em  $G[F]$  então
6           $\quad$   $F \leftarrow F \cup \{e\}$ ;
7  Retorne  $F$ ;

```

O Algoritmo 4 recebe como entrada um grafo, uma função de restrições orçamentárias, uma solução inicial F tal que $G[F]$ é uma floresta geradora e um subconjunto $E' \subset E$ de arestas que podem ser adicionadas à solução. Enquanto houver aresta disponível em E' , as linhas 4–5 escolhem uma aresta aleatoriamente e a removem de E' . A linha 5 verifica se ainda há orçamento disponível para o rótulo da aresta escolhida e se a aresta conecta duas componentes conexas distintas em $G[F]$. Note-se que é mantida a estrutura de uma floresta, com um número não crescente de árvores.

O procedimento de busca local é apresentado no Algoritmo 5, que recebe como entrada um grafo, uma função de restrições orçamentárias, uma solução inicial S e o parâmetro n_T . O algoritmo consiste em dois níveis de melhoria. As linhas 2–6 tentam melhorar a solução inicial reconstruindo uma solução com todos os rótulos, exceto um. As linhas 7–13 tentam construir uma nova solução removendo dois rótulos e adicionando um novo à solução atual. Em ambos os casos, retorna-se a primeira melhoria alcançada, como pode-se observar nas linhas 5–6 e 12–13.

Algoritmo 5: Busca local

```

1  Procedimento BUSCA-LOCAL ( $G = (V, E, L)$ ,  $w : L \rightarrow \mathbb{Z}^+$ ,  $S$ ,  $n_T$ )
2      Para cada  $l \in \mathcal{L}(S)$  faça
3          Para cada  $i = 1 : n_T$  faça
4               $S' \leftarrow \text{FLORESTA-GERADORA}(G, w, \emptyset, \mathcal{E}(\mathcal{L}(S) \setminus \{l\}))$  ;
5              Se  $|S'| = |V| - 1$  então
6                  Retorne  $S'$  ;
7      Se  $|\mathcal{L}(S)| > 1$  então
8          Para cada  $i = 1 : n_T (|L \setminus \mathcal{L}(S)|)$  faça
9              Sejam  $l_1, l_2$  rótulos distintos tomados aleatoriamente de  $\mathcal{L}(S)$ ;
10             Seja  $l_3$  um rótulo tomado aleatoriamente de  $L \setminus \mathcal{L}(S)$ ;
11              $S' \leftarrow \text{FLORESTA-GERADORA}(G, w, \emptyset, \mathcal{E}((\mathcal{L}(S) \setminus \{l_1, l_2\}) \cup \{l_3\}))$  ;
12             Se  $|S'| = |V| - 1$  então
13                 Retorne  $S'$  ;
14  Retorne  $S$  ;

```

Convém destacar que a busca local utiliza o procedimento descrito no Algoritmo 4 para construir uma nova solução a partir de uma solução inicial vazia, conforme linhas 4 e 11, considerando as arestas em $\mathcal{L}(S)$, com as remoções e adições de cada nível. Observe, ainda, que o número total de tentativas de melhoria é dado por $n_T |L|$, sendo $n_T |\mathcal{L}(S)|$ tentativas no primeiro nível, nos laços das linhas 2 e 3, e $n_T |L \setminus \mathcal{L}(S)|$ tentativas no segundo nível, na linha 8.

5. BRKGA

Esta seção discute o segundo método heurístico proposto, baseado na meta-heurística BRKGA e descrito no Algoritmo 6. A meta-heurística BRKGA consiste em uma população de soluções, ou cromossomos, onde cada uma é um vetor de n números reais aleatórios, ou genes, do intervalo $[0, 1]$. A população evolui através da recombinação dos cromossomos e introdução de novos cromossomos mutantes, e usa-se um algoritmo decodificador para associar cada cromossomo a uma solução viável do problema. Também adota-se uma estratégia elitista, dividindo a população entre elite e não-elite, preservando a elite para a próxima geração e selecionado sempre um cromossomo da elite e outro da não-elite para recombinação, com maior probabilidade do cromossomo gerado herdar os genes do cromossomo da elite [Gonçalves e Almeida, 2002; Ericsson et al., 2002; Gonçalves e Resende, 2004].

A entrada do algoritmo é um grafo com arestas rotuladas $G = (V, E, L)$; uma função de restrições orçamentárias w , o número de genes por cromossomo $n = |L|$, o tamanho da população $p \in \mathbb{Z}^+$, o número de cromossomos da elite $p_e \in \mathbb{Z}^+$, o número de cromossomos mutantes introduzidos a cada geração $p_m \in \mathbb{Z}^+$, a probabilidade de herdar um gene da elite na recombinação $\rho \in (0, 5, 1)$, e o número de tentativas $n_T \in \mathbb{Z}^+$ por configuração durante a decodificação, semelhante ao parâmetro homônimo no método baseado na meta-heurística GRASP.

O algoritmo inicial gerando a população inicial P , composta por p cromossomos com tomados do conjunto $[0, 1]^n$. Em seguida, a linha 3 decodifica cada cromossomo para obter a

solução correspondente e a linha 4 ordena a população com base no valor da função objetivo da solução. Note-se que, assim como no método baseado na meta-heurística GRASP, a solução é um subconjunto de arestas e a função objetivo f representa o número de rótulos presentes na solução, desde que se trate de uma solução viável, ou ∞ , caso contrário.

Algoritmo 6: Método heurístico baseado na meta-heurística BRKGA

```

1  Procedimento BRKGA ( $G = (V, E, L)$ ,  $w : L \rightarrow \mathbb{Z}^+$ ,  $n$ ,  $p$ ,  $p_e$ ,  $p_m$ ,  $\rho$ ,  $n_T$ )
2  Seja  $P$  uma população inicial de  $p$  cromossomos gerados aleatoriamente de  $[0, 1]^n$ ;
3  Seja  $S(x) = \text{DECODIFICAÇÃO}(G, w, x, n, n_T)$  para cada  $x \in P$ ;
4  Ordene  $P$  de forma não decrescente em  $f(S(x))$ ;
5  Enquanto tempo < tempo limite faça
6     $P' \leftarrow P[1 : p_e]$ ;
7    Para cada  $j = 1 : p - p_e - p_m$  faça
8      Seja  $x_e$  um cromossomo tomado aleatoriamente de  $P[1 : p_e]$ ;
9      Seja  $x_n$  um cromossomo tomado aleatoriamente de  $P[p_e + 1 : p]$ ;
10     Adicione RECOMBINAÇÃO( $x_e, x_n, n, \rho$ ) a  $P'$ ;
11   Para cada  $j = 1 : p_m$  faça
12     Seja  $x \in [0, 1]^n$  um novo cromossomo gerado aleatoriamente;
13     Adicione  $x$  a  $P'$ ;
14   Seja  $S(x) = \text{DECODIFICAÇÃO}(G, w, x, n, n_T)$  para cada  $x \in P'$ ;
15   Ordene  $P'$  de forma não decrescente em  $f(S(x))$ ;
16    $P \leftarrow P'$ ;
17 Retorne  $S(P[0])$ ;

```

Gerada a população inicial, o laço principal do algoritmo prossegue até que o tempo limite seja atingido. Na linha 6, a população sobrevivente para a próxima geração P' é iniciada com os p_e melhores cromossomos da geração anterior, a elite. Nas linhas 7–10, são gerados $p - p_e - p_m$ cromossomos por recombinação. Para cada um, escolhe-se um pai da elite x_e e um pai da não-elite x_n . Em seguida, as linhas 11–13 inserem p_m novos cromossomos mutantes na população, gerados aleatoriamente. Então, todos os cromossomos são decodificados na linha 14, a população é novamente ordenada com base nos valores da função objetivo, na linha 15, e avança-se para a próxima geração do algoritmo, substituindo a população atual P pela nova população gerada P' , conforme a linha 16. Ao final de cada geração, o primeiro indivíduo da população, $P[0]$ corresponde à melhor solução encontrada, que é retornada na linha 17.

Em relação à recombinação, o objetivo é gerar um novo cromossomo x , a partir de dois cromossomos pais, x_e da elite e x_n da não-elite. Para isso, é gerado um vetor de chaves aleatórias $k \in [0, 1]^n$ e, para cada $i = 1, 2, \dots, n$, $x[i]$ é definido como $x_e[i]$, se $k[i] \leq \rho$, ou $x_n[i]$, caso contrário. Desse modo, cada gene do cromossomo filho é herdado da elite, com probabilidade $\rho > 0,5$, ou da não-elite, com probabilidade $1 - \rho < 0,5$, de modo que é uma escolha enviesada.

O Algoritmo 7 descreve o procedimento de decodificação de um cromossomo em uma solução do PAGRMRO. Como entrada, tem-se um grafo G , uma função de restrições orçamentárias w ; um cromossomo x ; o número de genes n ; e o número de tentativas por configuração n_T . O algoritmo inicia ordenando a lista de rótulos, na linha 2, com base na prioridade definida pelos valores dos genes de x , ou seja, $L' = L_{i_1}, L_{i_2}, \dots, L_{i_{|L|}}$ será tal que $x[i_1] \leq x[i_2] \leq \dots \leq x[i_{|L|}]$. Então, o laço principal das linhas 3–7 tenta construir uma nova solução usando os rótulos $L'[1 : i]$, com n_T tentativas para cada i , até encontrar a primeira solução viável, que é retornada imediatamente. O processo de tentar construir uma nova solução utiliza o procedimento descrito no Algoritmo 4 a partir de uma solução inicial vazia e considerando o conjunto de arestas $\mathcal{E}(L'[1 : i])$.

Algoritmo 7: Decodificação de um cromossomo em uma solução do PAGRMRO

```

1 Procedimento DECODIFICAÇÃO ( $G = (V, E, L)$ ,  $w : L \rightarrow \mathbb{Z}^+$ ,  $x, n, n_T$ )
2   Seja  $L'$  uma permutação de  $L$  em ordem não decrescente de  $x[i]$  para cada rótulo  $L_i$ ;
3   Para cada  $i = 1 : n$  faça
4     Para cada  $k = 1 : n_T$  faça
5        $S \leftarrow \text{FLORESTA-GERADORA}(G, w, \emptyset, \mathcal{E}(L'[1 : i]))$ ;
6       Se  $|S| = |V| - 1$  então
7         Retorne  $S$ ;
8   Retorne SOLUÇÃO-NÃO-ENCONTRADA;

```

6. Experimentos Computacionais

Todos os métodos analisados neste estudo foram implementados na linguagem C++, utilizando o compilador g++ versão 13.3.0 (opção de compilação -O3), com o auxílio do resolvidor matemático CPLEX versão 22.1.2, em um computador com 16 GB de RAM DDR4, processador Intel Core i7-7500U (64 bits), com CPU contendo núcleos de 2,70 GHz e frequência máxima de núcleo único de 3,50 GHz, executando o sistema operacional Ubuntu 24.04.

Foram utilizados dois conjuntos de instâncias gerados por Cerulli et al. [2005]. O primeiro conjunto possui 120 instâncias, com tamanho $|V| \in \{20, 30, 40, 50\}$, quantidade de rótulos $|L| = |V|$ e densidade do grafo $d \in \{0.2, 0.5, 0.8\}$. Cada configuração contém 10 instâncias. Do segundo conjunto, foram utilizadas 240 instâncias, com tamanho $|V| \in \{100, 200\}$, quantidade de rótulos $|L| \in \{0.25|V|, 0.5|V|, |V|, 1.25|V|\}$ e densidade do grafo $d \in \{0.2, 0.5, 0.8\}$. Cada configuração também contém 10 instâncias. Cada instância foi executada com restrições orçamentárias de 50%, 60% e 70% do número de arestas de cada rótulo. No total, foram 1080 instâncias do PAGRMRO, sendo 360 do primeiro conjunto e 720 do segundo.

Em relação aos parâmetros do método baseado em GRASP, foram definidos empiricamente como $\alpha = 0,05$ e $n_T = 10$. Os valores foram avaliados dentro dos conjuntos $\alpha \in \{0,00, 0,05, 0,10, \dots, 1,00\}$ e $n_T \in \{1, 2, 5, 10, 20, 50, 100, 200, 500, 1000\}$. No caso do método baseado em BRKGA, os parâmetros foram definidos empiricamente como $p = 2|L|$, $p_e = 0,25p$, $p_m = 0,1p$, $\rho = 0,6$ e $n_T = 10$, a partir dos conjuntos $p \in \{|L|, 2|L|, 3|L|, 4|L|, 5|L|\}$, $p_e \in \{0,1p, 0,15p, 0,2p, 0,25p\}$, $p_m \in \{0,1p, 0,2p, 0,3p\}$ e $\rho \in \{0,6, 0,7, 0,8\}$. Os resultados para o primeiro grupo de instâncias são apresentados na Tabela 1.

Tabela 1: Resultados dos experimentos para grafos com 20, 30, 40 e 50 vértices

V	d	50%						60%						70%					
		Exato		GRASP		BRKGA		Exato		GRASP		BRKGA		Exato		GRASP		BRKGA	
		obj	t (s)	obj	t _T (s)	obj	t _T (s)	obj	t (s)	obj	t _T (s)	obj	t _T (s)	obj	t (s)	obj	t _T (s)	obj	t _T (s)
20	0,2	12,0	37,446	12,0	0,000	12,0	0,005	8,9	0,289	8,9	0,000	8,9	0,001	7,1	0,015	7,1	0,000	7,1	0,004
20	0,5	5,1	TLE	5,1	0,000	5,1	0,005	4,0	0,207	4,0	0,000	4,0	0,065	3,8	0,110	3,8	0,000	3,8	0,001
20	0,8	3,7	454,732	3,7	0,000	3,7	0,001	3,0	0,346	3,0	0,000	3,0	0,043	2,8	0,205	2,8	0,000	2,8	0,004
30	0,2	11,9	TLE	11,9	0,005	11,9	0,051	9,3	28,290	9,3	0,006	9,3	0,049	8,0	0,107	8,0	0,008	8,0	0,279
30	0,5	5,3	TLE	5,3	0,001	5,3	0,073	4,5	TLE	4,5	0,000	4,5	0,214	4,0	4,741	4,0	0,000	4,0	0,073
30	0,8	3,8	TLE	3,8	0,001	3,8	0,013	3,0	1,185	3,0	0,000	3,0	0,372	3,0	1,257	3,0	0,000	3,0	0,030
40	0,2	*12,1	TLE	12,0	0,022	12,0	0,176	*9,8	TLE	9,7	0,106	9,9	0,552	8,5	73,084	8,5	0,019	8,7	0,028
40	0,5	5,5	TLE	5,5	0,001	5,5	0,014	4,5	TLE	4,5	0,002	4,5	1,581	4,0	8,638	4,0	0,007	4,1	0,769
40	0,8	4,0	TLE	4,0	0,000	4,0	0,003	3,0	81,194	3,0	0,000	3,3	0,875	3,0	2,375	3,0	0,000	3,0	1,258
50	0,2	*13,1	TLE	12,5	1,032	12,6	0,996	*10,3	TLE	10,2	0,674	10,4	0,801	9,2	TLE	9,2	0,312	9,7	0,316
50	0,5	*6,0	TLE	5,8	0,017	5,9	0,425	4,9	TLE	4,9	0,003	5,0	0,333	4,3	TLE	4,3	0,044	4,6	0,806
50	0,8	4,0	TLE	4,0	0,000	4,0	0,006	3,1	TLE	3,1	0,003	3,9	0,831	3,0	7,203	3,0	0,000	3,6	2,296

A tabela apresenta os resultados divididos de acordo com o orçamento da quantidade de arestas por rótulo de 50%, 60% ou 70%. Em cada caso, são apresentados os resultados médios do valor da função objetivo (colunas **obj**) e do tempo de execução (colunas **t** (s)), no caso do método exato, ou do tempo até a melhor solução (*time-to-target*, colunas **t_T** (s)), sempre em segundos. Em várias instâncias, o método exato não conseguiu convergir dentro do tempo limite de 3600 s (1 h), sendo marcadas com TLE na coluna de tempo. Nesses casos, a solução retornada foi a melhor solução encontrada, não necessariamente ótima. Por conta disso, o exato não obteve o melhor resultado em alguns casos, marcados com “*” na tabela. Os métodos heurísticos foram executados com tempo limite de 20 s. O método baseado em GRASP obteve o melhor resultado em todas as instâncias avaliadas. A Figura 2 exibe o gráfico de convergência de uma dessas instâncias.

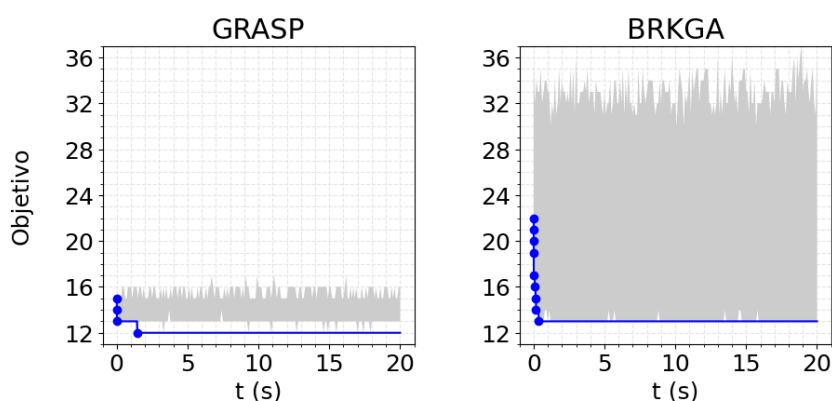


Figura 2: Gráfico de convergência dos métodos heurísticos baseados em GRASP e BRKGA para uma instância com 50 vértices, 50 rótulos, densidade de 20%, orçamento de 50% e solução ótima com 12 rótulos

Os gráficos representam o valor da função objetivo em relação ao tempo de execução do algoritmo. A área cinza representa a variação entre as melhores e piores soluções geradas, enquanto a linha azul representa a melhor solução encontrada até o momento. Observa-se que o método baseado em GRASP permanece sempre bem próximo ao ótimo, o que pode ser um reflexo da busca local, que obteve melhora em 81,75% dos casos. Também sugere que o método poderia beneficiar-se com maior variabilidade, seja de um α adaptativo ou uso em conjunto com outra meta-heurística. Em relação ao método baseado no BRKGA, observa-se uma grande diversificação das soluções geradas, mas uma dificuldade de convergência para o ótimo global, que pode ser beneficiada por uma busca local ou utilização de outro método para decodificação dos cromossomos.

Em relação às instâncias do segundo conjunto, os resultados são apresentados na Tabela 2. Neste caso, não foi executado o método exato, que já falhou em convergir nas instâncias menores. Considerou-se um tempo limite de 60 s, definido empiricamente. Observa-se que o método baseado em GRASP obteve os melhores resultados na maioria das instâncias, exceto em poucos casos com orçamento de 60% e 50%.

Como análise estatística, foi realizado o teste de Shapiro-Wilk para verificar a normalidade das diferenças entre todos os resultados dos métodos baseados em GRASP e BRKGA e, como esperado, os dados não seguem uma distribuição normal, com p-valor de $2,91 \cdot 10^{-36}$. Sendo assim, aplicou-se o teste não paramétrico de Wilcoxon para dados pareados, que resultou em um p-valor de $1,39 \cdot 10^{-97}$. Este resultado indica uma diferença estatisticamente significativa entre os métodos, com evidência robusta de que o método baseado em GRASP produziu resultados melhores que o método baseado em BRKGA.

Tabela 2: Resultados dos experimentos para grafos com 100 e 200 vértices

V	L	d	50%				60%				70%			
			GRASP		BRKGA		GRASP		BRKGA		GRASP		BRKGA	
			obj	t _T (s)	obj	t _T (s)	obj	t _T (s)	obj	t _T (s)	obj	t _T (s)	obj	t _T (s)
100	25	0,2	4,9	0,002	5,4	5,908	4,5	0,003	5,4	5,789	4,5	0,002	5,2	5,183
100	25	0,5	2,0	0,001	3,0	0,007	2,0	0,000	3,0	0,002	2,0	0,000	3,0	0,002
100	25	0,8	2,0	0,000	2,8	0,007	1,8	0,000	2,8	0,002	1,8	0,000	2,8	0,002
100	50	0,2	8,0	1,613	8,4	3,675	7,1	0,172	7,6	1,549	6,7	0,121	7,4	6,818
100	50	0,5	4,0	0,000	4,0	0,287	3,1	0,245	4,0	0,160	3,0	0,059	4,0	0,363
100	50	0,8	2,8	0,001	3,0	0,008	2,0	0,001	3,0	0,004	2,0	0,001	3,0	0,004
100	100	0,2	13,8	9,614	13,9	6,149	11,7	6,624	12,1	5,628	10,5	6,552	11,4	3,527
100	100	0,5	6,2	0,776	6,3	0,542	5,1	0,387	5,8	8,602	5,0	0,069	5,8	0,080
100	100	0,8	4,1	0,016	4,4	0,662	3,9	0,016	4,0	2,648	3,0	0,241	4,0	1,520
100	125	0,2	16,7	7,555	16,1	12,800	13,9	6,507	13,9	5,106	12,3	2,782	12,8	0,852
100	125	0,5	7,3	5,664	7,5	3,491	6,0	4,246	6,9	0,166	5,7	0,959	6,2	8,145
100	125	0,8	5,0	0,007	5,0	0,139	4,0	0,070	5,0	0,026	4,0	0,009	5,0	0,054
200	50	0,2	5,4	0,188	6,3	0,392	5,2	0,244	6,3	1,047	5,2	0,093	6,2	5,455
200	50	0,5	2,2	0,004	3,2	0,028	2,2	0,002	3,2	0,029	2,2	0,005	3,2	0,008
200	50	0,8	2,0	0,000	3,0	0,015	2,0	0,001	3,0	0,008	2,0	0,000	3,0	0,008
200	100	0,2	9,0	3,762	10,2	2,334	8,2	5,031	9,6	1,692	8,1	5,461	9,8	1,809
200	100	0,5	4,0	0,065	4,4	3,693	3,6	0,085	4,4	2,352	3,4	0,165	4,5	1,135
200	100	0,8	3,0	0,001	3,6	0,378	2,7	0,041	3,6	0,320	2,6	0,031	3,6	0,203
200	200	0,2	16,3	7,393	16,1	15,896	14,0	14,778	14,5	9,347	13,0	14,244	14,5	4,600
200	200	0,5	7,0	0,458	7,3	0,861	6,1	0,916	7,0	3,865	5,4	9,406	7,0	0,897
200	200	0,8	4,5	12,756	5,0	0,178	4,0	0,041	5,0	0,269	4,0	0,023	5,0	0,118
200	250	0,2	19,4	5,725	19,1	17,994	16,7	12,093	16,6	12,881	15,4	11,543	15,8	7,802
200	250	0,5	8,1	9,017	8,4	5,752	7,1	1,909	7,9	7,070	6,4	4,086	7,9	8,729
200	250	0,8	5,2	3,261	5,9	0,364	5,0	0,050	5,5	4,952	4,0	3,816	5,6	8,117

7. Considerações Finais

Este trabalho aborda o PAGRMRO, um problema de conectividade em grafos com arestas rotuladas que objetiva encontrar uma árvore geradora utilizando uma quantidade mínima de rótulos com restrições orçamentárias para o número máximo de arestas por rótulo. Foram propostos dois métodos heurísticos baseados nas meta-heurísticas GRASP e BRKGA e uma formulação baseada em cortes, com um algoritmo de *branch-and-cut*, para validar os métodos heurísticos.

Os experimentos computacionais mostraram que o método baseado em GRASP encontrou a solução ótima em todas as instâncias nas quais o método exato convergiu dentro do tempo limite. Na maioria das demais instâncias, também conseguiu resultados melhores que o método baseado no BRKGA, com uma diferença estatisticamente significativa.

Como proposta de trabalhos futuros, sugere-se um estudo mais aprofundado dos métodos exatos, para permitir uma melhor comparação com os resultados dos métodos heurísticos. Além disso, pode-se avaliar o uso de novas meta-heurísticas em conjunto com os métodos propostos, de forma a aumentar a variabilidade do método baseado em GRASP e melhorar a convergência do método baseado no BRKGA.

Referências

- Cerrone, C., Cerulli, R., e Golden, B. (2017). Carousel greedy: A generalized greedy algorithm with applications in optimization. *Computers and Operations Research*, 85:97–112.
- Cerrone, C., D'Ambrosio, C., e Raiconi, A. (2019). Heuristics for the strong generalized minimum label spanning tree problem. *Wiley*.

- Cerulli, R., D'Ambrosio, C., Serra, D., e Sorgente, C. (2024). The budgeted labeled minimum spanning tree problem. *Mathematics*, 12(2).
- Cerulli, R., Fink, A., Gentili, M., e Voß, S. (2005). Metaheuristics comparison for the minimum labelling spanning tree problem. In *The Next Wave in Computing, Optimization, and Decision Technologies*, p. 93–106. Springer.
- Chang, R.-S. e Leu, S.-J. (1997). The minimum labeling spanning trees. *Information Processing Letters*, 63(5):277–282.
- Chen, Y., Cornick, N., Hall, A. O., Shajpal, R., Silberholz, J., Yahav, I., e Golden, B. L. (2008). Comparison of heuristics for solving the gmlst problem. *Telecommunications modeling, policy, and technology*, p. 191–217.
- Ericsson, M., Resende, M. G. C., e Pardalos, P. M. (2002). A genetic algorithm for the weight setting problem in ospf routing. *Journal of combinatorial optimization*, 6:299–333.
- Gonçalves, J. F. e Almeida, J. R. (2002). A hybrid genetic algorithm for assembly line balancing. *Journal of heuristics*, 8:629–642.
- Gonçalves, J. F. e Resende, M. G. (2004). An evolutionary algorithm for manufacturing cell formation. *Computers & Industrial Engineering*, 47(2):247–273. ISSN 0360-8352.
- Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50.
- Lin, M., Liu, F., Zhao, H., e Chen, J. (2020). A novel binary firefly algorithm for the minimum labeling spanning tree problem. *Computer Modeling in Engineering & Sciences*, 125(1):197–214.
- Long, H., Li, X., Long, Z., e Wu, F. (2023). Community-based zigzag piloting algorithm for the strong generalized minimum label spanning tree problem. *Soft Comput.*
- Saad, A., Kafafy, A., Abd-El-Raof, O., e El-Hefnawy, N. (2018). A grasp-genetic metaheuristic applied on multi-processor task scheduling systems. In *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, p. 109–115. IEEE.
- Silva, T. G., Gueye, S., Michelon, P., Ochi, L. S., e Cabral, L. A. F. (2019). A polyhedral approach to the generalized minimum labeling spanning tree problem. *EURO Journal on Computational Optimization*, 7:47–77.
- Vaisman, R. (2021). Finding minimum label spanning trees using cross-entropy method. *Networks*, 79(2):220–235.
- Yang, Z., Wang, W., e Shi, M. (2021). Algorithms and complexity for a class of combinatorial optimization problems with labelling. *Journal of Optimization Theory and Applications*.
- Zhang, P. (2024). Combining capacity and length: Finding connectivity bottleneck in a layered network. *IEEE/ACM Transactions on Networking*.