57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

Algoritmos para o Problema da Árvore Geradora com Quantidade Mínima de Vértices *Branch*

Filipe Gomes Arante de Souza
Universidade Federal do Espírito Santo
Avenida Fernando Ferrari, 514, Goiabeiras, Vitória, Espírito Santo, CEP 29-075.910
filipe.gomes.arante@gmail.com

Maria Claudia Silva Boeres
Universidade Federal do Espírito Santo
Avenida Fernando Ferrari, 514, Goiabeiras, Vitória, Espírito Santo, CEP 29-075.910
maria.boeres@ufes.br

Renato Elias Nunes de Moraes
Universidade Federal do Espírito Santo
Avenida Fernando Ferrari, 514, Goiabeiras, Vitória, Espírito Santo, CEP 29-075.910
renato.moraes@ufes.br

RESUMO

Dado um grafo conexo e não direcionado $G = (V, E)$, o problema da Árvore Geradora com Quantidade Mínima de Vértices *Branch* (MBV) consiste em encontrar uma árvore geradora de G que possua a menor quantidade de vértices com grau maior que 2. Como esse problema é NP-*hard*, é inviável resolvê-lo de forma exata num tempo hábil, a não ser que P = NP. Portanto, é necessário a utilização de técnicas heurísticas para encontrar soluções com qualidade satisfatória. Este artigo apresenta adaptações de algoritmos propostos na literatura para este problema, abordando heurísticas construtivas, buscas locais e meta-heurísticas. Os resultados obtidos pelos algoritmos construtivos apresentaram resultados equivalentes em relação as heurísticas construtivas já existentes na literatura, enquanto as meta-heurísticas não superaram, mas se aproximaram do estado da arte.

PALAVRAS CHAVE. árvore geradora, vértices *branch*, otimização combinatória.
Tópicos: Otimização combinatória

ABSTRACT

Given a connected and undirected graph $G = (V, E)$, the Minimum Branch Vertices Spanning Tree problem (MBV) consists of finding a spanning tree from G that has the fewest vertices with degree greater than 2. As this problem is NP-*hard*, it is not feasible to solve it exactly in a timely manner, unless P = NP. It is therefore necessary to use heuristic techniques to find solutions with satisfactory quality. This article presents adaptations of algorithms proposed in the literature for this problem, covering constructive heuristics, local searches and meta-heuristics. The results obtained by the constructive algorithms were equivalent to those of the constructive heuristics available in the literature, while the meta-heuristics did not surpass, but came close to the state of the art.

KEYWORDS. spanning tree, branch vertices, combinatorial optimization
Paper topics: Combinatorial optimization

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

1. Introdução

Problemas de otimização combinatória considerados NP-completos apresentam grandes desafios para encontrar soluções ótimas. A complexidade intrínseca desses problemas frequentemente impede a aplicação de métodos exatos, tornando inviável a busca exaustiva pelo ótimo global em um tempo razoável, a não ser que P = NP. Nesse cenário, as meta-heurísticas têm oferecido abordagens que, apesar de não garantirem a solução ótima, conseguem de maneira eficaz gerar soluções de alta qualidade em um tempo computacional viável [Talbi, 2009].

Muitos problemas da área de otimização combinatória podem ser modelados usando representações por grafos, pois eles permitem a visualização e a resolução de problemas complexos de maneira estruturada. Um exemplo clássico são os problemas relacionados a árvores geradoras mínimas, onde se busca a construção de uma árvore que conecta todos os vértices de um grafo com o menor custo possível. Esse tipo de problema possui várias aplicações, como redes de transporte, telecomunicações e distribuição, geralmente com o objetivo de planejar a redução de custos [De Almeida et al., 2012].

Um dos problemas de otimização combinatória definidos a partir de uma árvore geradora é o Problema da Árvore Geradora com Quantidade Mínima de Vértices *Branch* [Gargano et al., 2002], que consiste em encontrar uma árvore geradora de um dado grafo com a menor quantidade de vértices *branch*, isto é, vértices com grau maior que 2. Sua resolução tem implicações práticas importantes, especialmente em cenários onde a simplicidade da rede é desejada ou onde a minimização da complexidade estrutural pode levar a ganhos em termos de manutenção, custos e eficiência operacional. O estudo desse problema e de soluções para ele é o foco deste artigo, que busca explorar métodos heurísticos para resolver este problema de forma eficiente, utilizando técnicas de otimização combinatória.

Neste trabalho foram implementados dois algoritmos construtivos: *Randomized Branch Expanding Prim* (R-BEP) [Beltrão et al., 2023] e *Alpha Branch Expanding Prim* (α-BEP), que é uma proposta de variação do R-BEP. Ambos os algoritmos foram escolhidos para este artigo pois desempenharam melhor que a heurística construtiva de Moreno et al. [2018], que atualmente é o estado da arte. Além disso, também foi proposta a busca local *Turbo Treevial Search* (TTVS). Estes algoritmos foram utilizados para compor a proposta de duas versões da meta-heurística GRASP.

Neste e nos próximos parágrafos, destacamos alguns trabalhos da literatura que exploram diferentes abordagens de solução para o problema MBV e suas variações. Cerulli et al. [2009] propuseram heurísticas construtivas como as estratégias *Edge Weighting Strategy* (EWS), *Node Coloring Strategy* (NCS) e uma combinação híbrida das duas. As heurísticas seguem um processo similar ao algoritmo de Kruskal [Cormen et al., 2009] para construção de árvores geradoras. A EWS calcula um peso para cada aresta, selecionando a de menor valor para manter a solução viável, enquanto a NCS atribui cores aos nós com base na proximidade de se tornarem vértices *branch*, selecionando arestas cujas extremidades estejam menos propensas a formar vértices *branch*. A abordagem híbrida combina os dois métodos, obtendo resultados superiores às estratégias individuais. Silva et al. [2011] propuseram uma busca local do tipo *Iterative Refinement* (IR) [Deo e Kumar, 1997], tradicionalmente associada a algoritmos do tipo *multistart*, mas adaptada para realizar trocas de arestas. A estratégia utiliza uma função de *blacklisting* para determinar o quão vantajoso é manter uma aresta na solução. Em vez de penalizar as arestas com altos valores na função, o algoritmo remove as que maximizam o valor da função e as substitui por outras que a minimizam. A abordagem mostrou-se eficaz, superando os algoritmos EWS e NCS em 78% das instâncias testadas. Sundar et al. [2012] propuseram um algoritmo construtivo que aceita temporariamente soluções inviáveis ao permitir a inserção de arestas que geram ciclos na árvore solução, identificando posteriormente as arestas que, ao serem removidas, melhoram a quantidade de vértices

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

branch, recuperando a viabilidade da solução.

Cerrone et al. [2014] adaptaram um algoritmo genético para o problema, selecionando duas soluções através de torneio e realizando *crossover*, mutação e busca local na solução gerada. Essa abordagem se mostrou eficiente ao combinar características de uma solução de alta qualidade com uma de qualidade inferior, promovendo diversificação ao longo das iterações.

Uma extensão do problema MBV foi proposta por Merabet e Molnár [2016], introduzindo o problema k-MBV, que define um vértice *k-branch* como aquele com grau maior que $k+2$, onde $k \geq 0$. O autor provou que o problema é NP-difícil para qualquer valor de k, além de mostrar que, à medida que o valor de k e a densidade do grafo aumentam, o número de vértices *branch* diminui. Para $k \geq 4$, a quantidade de nós *k-branch* tende a ser próxima de zero, independentemente do tamanho e da densidade do grafo. Moreno et al. [2018] simplificaram a notação introduzida por Merabet e Molnár [2016], ao definir o parâmetro $d = k+2$, onde $k \geq 0$, e renomeando o problema para d-MBV. Além disso, propos uma estratégia de solução baseada na decomposição do grafo original em componentes conexas obtidas a partir da retirada das pontes do grafo, aplicando o algoritmo *Iterated Local Search* (ILS) com busca local *first improvement*. Este algoritmo busca reduzir gradualmente o grau dos nós do grafo, removendo e substituindo arestas de forma estratégica. O estudo reafirmou que, para valores de $d \geq 4$, a solução ótima foi frequentemente atingida, enquanto para $d \geq 6$, os valores encontrados estavam consistentemente próximos de zero, corroborando os achados anteriores.

Trabalhos mais recentes como os de Beltrão et al. [2023] investigam versões determinísticas e randomizadas como algoritmos construtivos inspirados no algoritmo de Prim [Cormen et al., 2009] para o problema MBV. Os resultados obtidos superaram significativamente os alcançados pelos construtivos propostos em Moreno et al. [2018] em quase todos os grupos de instâncias testadas. O trabalho de Boeres et al. [2023] realizou uma análise empírica sobre métricas de centralidade de grafos, a fim de identificar vértices *branch*, destacando as centralidades de grau e *Pagerank* como as mais eficazes. Por fim, de Lima et al. [2024] investigaram a relação entre a densidade do grafo e o número de vértices *branch*, concluindo que, quanto maior a densidade do grafo, menor a quantidade de vértices *branch* na solução ótima do problema.

Esse artigo está organizado como se segue. Na Seção 2, o problema MBV é brevemente definido. As versões do algoritmo GRASP propostas neste trabalho para a solução do problema MBV são descritas na Seção 3. Em seguida, na Seção 4 são descritos os experimentos computacionais realizados e a análise de seus resultados. Por fim, a Seção 5 apresenta as conclusões e sugestões de trabalhos futuros.

2. O problema da árvore geradora com quantidade mínima de vértices *branch*

Dado um grafo G conexo e não direcionado, o problema da árvore geradora com quantidade mínima de vértices *branch* (MBV) tem o objetivo de encontrar uma árvore geradora de G que possua a menor quantidade de vértices com grau maior que 2. Tais vértices são denominados *branch*. Esse problema foi proposto por Gargano et al. [2002] para alocar *switches* da melhor forma em redes óticas. Os autores provaram que o problema MBV pertence à classe NP-*hard*.

3. O algoritmo GRASP para solução do problema MBV

Este trabalho propõe duas versões do algoritmo *Greedy Randomized Adaptive Search Procedure* (GRASP) para o problema MBV. O GRASP [FEO e RESENDE, 1995], consiste num processo *multistart*, composto por uma fase de construção, utilizando um algoritmo construtivo, seguida de uma fase de melhoria, utilizando uma busca local. Ambas recebem como entrada um grafo $G = (V_G, E_G)$ e tem como saída uma árvore geradora $T^* = (V_{T^*}, E_{T^*})$. Seu pseudocódigo está apresentado no Algoritmo 1. Para a primeira versão do algoritmo foram escolhidos o algoritmo construtivo *Alpha Branch Expanding Prim* (α-BEP) e a busca local *Turbo Treevial Search* (TTVS),

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

Algoritmo 1: Greedy Randomized Adaptive Search Procedure (GRASP)

Entrada: Grafo $G = (V_G, E_G)$, Parâmetro $\alpha \in [0, 1]$
Saída : Árvore Geradora $T^* = (V_{T^*}, E_{T^*})$
1 **enquanto** *critério de parada não é atingido*
2 $T \leftarrow ConstruçãoAleatória(G, \alpha)$
3 $T \leftarrow BuscaLocal(T)$
4 **se** $f(T) < f(T^*)$ **então**
5 $T^* \leftarrow T$
6 **retorne** T^*

dando origem ao algoritmo **GRASP-FIXED-α-BEP-TTVS**. Já para a segunda versão foram escolhidos o algoritmo construtivo *Randomized Branch Expanding Prim* (R-BEP) [Beltrão et al., 2023] e a busca local *Turbo Treevial Search* (TTVS), dando origem ao algoritmo **GRASP-R-BEP-TTVS**. Os algoritmos α-BEP, R-BEP e TTVS são descritos, respectivamente, nas Seções 3.1, 3.2 e 3.3. As notações utilizadas em suas descrições estão reunidas na Tabela 1.

Notação	Significado
$d(v, G)$	Grau do vértice v no grafo G
$CConexa(v, G)$	Componente conexa do vértice v no grafo G.
$adj(v, G)$	Conjunto dos vértices adjacentes a v no grafo G.
$B(G)$	Conjunto de vértices *branch* do grafo G.
$Pontas(T(G), G)$	Conjunto de vértices do grafo G que possuem grau 1 na floresta $T(G)$ e grau maior que 1 em G.
$[p]$, onde p é uma expressão booleana	$[p] = 1$, se x é verdadeiro, e 0 caso contrário.

Tabela 1: Notações que serão utilizadas na descrição dos algoritmos.

3.1. Algoritmo Construtivo *Alpha Branch Expanding Prim* (α-BEP)

O algoritmo *Alpha Branch Expanding Prim* (α-BEP) é uma variação do algoritmo *Randomized Branch Expanding Prim* (R-BEP) [Beltrão et al., 2023], possuindo diferenças nos critérios de escolhas de arestas a serem inseridas na árvore. Ele recebe como entrada um grafo $G = (V_G, E_G)$ e um parâmetro $\alpha \in [0, 1]$ e tem como saída uma árvore geradora $T = (V_T, E_T)$. Seu pseudocódigo está representado no Algoritmo 2.

A heurística construtiva inicia executando o pré-processamento, como indicado na linha 1, responsável por inserir na árvore as arestas que são pontes e vértices que são ramificações. Pontes são arestas que quando removidas do grafo o separam em duas componentes conexas, enquanto ramificações são vértices que quando removidos do grafo o separam em três ou mais componentes conexas. Esse passo é importante pois as pontes estão presentes em qualquer árvore geradora de G, e as ramificações são vértices *branch* também em qualquer árvore geradora de G. Além disso, nessa etapa também é determinado o conjunto $Pontas(T, G)$, que é composto pelos vértices de T que podem ser cultivados sem mudar o valor da função objetivo. Um vértice é cultivável se é possível inserir arestas nele de modo que não sejam criados novos *branches*. Desse modo, o pré-processamento retorna uma floresta T, o conjunto de ramificações $B(T)$ e o conjunto $Pontas(T, G)$. Os conjuntos $B(T)$ e $Pontas(T, G)$ são definidos pelas Equações (1) e (2), respectivamente.

$$B(T) = \{v \in T | d(v, T) > 2\} \tag{1}$$

$$Pontas(T, G) = \{v \in T | d(v, T)) = 1 \wedge d(v, G) > 1\} \tag{2}$$

O laço das linhas 2 a 19 do Algoritmo 2 é executado enquanto $|E_T| < |V_G| - 1$, ou seja, enquanto a árvore não estiver completa. A partir daí, temos dois casos para tratar: (i) quando o conjunto $Pontas(T, G)$ não é vazio (condicional das linhas 3 a 11) e (ii) quando $Pontas(T, G)$ é vazio (condicional das linhas 12 a 19).

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

Algoritmo 2: *Alpha Branch Expanding Prim* (α-BEP)

Entrada: Grafo $G = (V_G, E_G)$, Parâmetro $\alpha \in [0, 1]$
Saída : Árvore Geradora $T = (V_T, E_T)$
1 $(T, B(T), Pontas(T, G)) \leftarrow PreProcessamento(G)$
2 **enquanto** $|E_T| < |V_G| - 1$
3 **se** $Pontas(T, G) \neq \emptyset$ **então**
4 $R_1 \leftarrow \{a \in Pontas(T, G)/d(a, G) \in [d_{1_{min}}, d_{1_{min}} + \alpha \cdot (d_{1_{max}} - d_{1_{min}})]\}$
5 $v \leftarrow SortearRoleta(R_1)$
6 $Pontas(T, G) \leftarrow Pontas(T, G) - \{v\}$
7 $N(v) = \{u \in adj(v, G)/u \notin V_T \vee (CConexa(u, T) \neq CConexa(v, T) \wedge (u \in B(T) \vee d(u, T) = 1))\}$
8 **se** $N(v) \neq \emptyset$ **então**
9 $R_2 \leftarrow \{b \in N(v)/d(b, G) \in [d_{2_{min}}, d_{2_{min}} + \alpha \cdot (d_{2_{max}} - d_{2_{min}})]\}$
10 $u \leftarrow SortearRoleta(R_2)$
11 $T, Pontas(T, G) \leftarrow InserirNaArvoreEAtualizarPontas(T, v, u, Pontas(T, G))$
12 **senão**
13 Seja $v \in V_T$ um vértice que maximiza $C(v) = |\{u \in adj(v, G)/u \notin V_T \vee (CConexa(u, T) \neq CConexa(v, T) \wedge (u \in B(T) \vee d(u, T) = 1))\}|$
14 **se** $C(v) > 0$ **então**
15 $B(T) \leftarrow B(T) \cup \{v\}$
16 $T, Pontas(T, G) \leftarrow ExpandirRamificaçãoEAtualizarPontas(G, T, v, Pontas(T, G))$
17 **senão**
18 Seja $v \in V_T$ um vértice que maximiza $CC(v) = |\{u \in adj(v, G)/CConexa(u, T) \neq CConexa(v, T)\}|$
19 $B(T) \leftarrow B(T) \cup \{v\}$
20 **retorne** T

No caso (i) existem arestas na árvore cuja inserção não gera novas ramificações, então será realizado o sorteio de dois vértices para determinar uma aresta a ser inserida em T. O primeiro sorteio escolhe um vértice do conjunto R_1, definido na Equação 3:

$$R_1 = \{a \in Pontas(T, G)|d(a, G) \in [d_{1_{min}}, d_{1_{min}} + \alpha \cdot (d_{1_{max}} - d_{1_{min}})]\} \tag{3}$$

onde α é um valor contido no intervalo $[0, 1]$ e $d_{1_{min}}$ e $d_{1_{max}}$ representam o menor e maior grau no grafo de um vértice no conjunto $Pontas(T, G)$, respectivamente. Se v é o vértice escolhido, então é necessário escolher um vizinho de v, isto é, $u \in adj(v, G)$ tal que a inserção da aresta (v, u) em T não gere novos vértices *branch* e nem torne a solução inviável, ou seja, que pertença ao conjunto $N(v)$, definido na Equação 4.

$$N(v) = \{u \in adj(v, G)|u \notin V_T \vee (CConexa(u, T) \neq CConexa(v, T) \wedge (u \in B(T) \vee d(u, T) = 1))\} \tag{4}$$

Para escolher o vizinho u de v é realizado então outro sorteio com os vértices do conjunto R_2, de maneira análoga ao anterior. Este conjunto está definido conforme a Equação 5:

$$R_2 = \{b \in N(v)|d(b, G) \in [d_{2_{min}}, d_{2_{min}} + \alpha \cdot (d_{2_{max}} - d_{2_{min}})]\} \tag{5}$$

onde $d_{2_{min}}$ e $d_{2_{max}}$ representam respectivamente, o menor e maior grau no grafo G de um vértice pertencente a $N(v)$. A escolha do vértice u está descrita nas linhas 9 e 10.

Em seguida, a aresta (v, u) é inserida na árvore, como podemos ver na linha 11. Essa inserção também atualiza o conjunto $Pontas(T, G)$, pois o vértice u pode ter adquirido ou perdido as características de uma ponta. Portanto, u é inserido no conjunto $Pontas(T, G)$ caso possua grau 1 em T e $u \notin Pontas(T, G)$; ou ele é removido desse conjunto caso $u \in Pontas(T, G)$.

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

No caso (ii), quando o conjunto $Pontas(T, G)$ é vazio, não é possível adicionar uma aresta em T sem gerar uma nova ramificação. Portanto, devemos escolher um vértice para se tornar *branch* que possua a maior quantidade de vizinhos que podem ser explorados no futuro. Para isso foi definida a métrica $C(v)$, Equação 6, que calcula a quantidade de adjacentes de v que são cultiváveis.

$$C(v) = |\{u \in adj(v, G) | u \notin V_T \vee (CConexa(u, T) \neq CConexa(v, T) \wedge (u \in B(T) \vee d(u, T) = 1))\}| \tag{6}$$

Na linha 13 do Algoritmo 2, o vértice v que maximiza $C(v)$ é escolhido. Se $C(v) > 0$, v possui vizinhos cultiváveis. Portanto, esse nó pode ser marcado como *branch* e o processo de expansão de ramificações pode ser aplicado em v. A expansão é responsável por maximizar o grau de um vértice *branch*, pois a inserção de arestas nele não aumenta o valor da função objetivo. Estas inserções são realizadas de modo a não gerarem ciclo na árvore. Uma vez que novas arestas foram inseridas na árvore, o conjunto $Pontas(T, G)$ é atualizado a fim de identificar os vizinhos de v que adquiriram características de uma ponta. Essa atualização ocorre implicitamente na linha 16. Entretanto, caso $C(v) = 0$, a inserção de qualquer aresta em T criará duas novas ramificações. Desse modo, o algoritmo usa $CC(v)$, que determina a quantidade de vizinhos de v que não geram ciclo na árvore. Formalmente, podemos expressar $CC(v)$ como descreve a Equação 7:

$$CC(v) = |\{u \in adj(v, G) | CConexa(u, T) \neq CConexa(v, T)\}| \tag{7}$$

$CC(v)$ é uma versão menos restrita da métrica $C(v)$, que visa identificar a quantidade de vizinhos $u \in adj(v, G)$ cuja inserção da aresta (v, u) mantém a viabilidade da solução. O vértice que maximiza $CC(v)$ é determinado na linha 18 e é marcado como ramificação na linha 19. Isso fará com que a condição $C(u) > 0$ seja satisfeita para algum vizinho u de v na próxima iteração de modo que uma segunda ramificação seja escolhida. A linha 20 retorna a saída do algoritmo.

Como o algoritmo não permite a ocorrência de ciclos e a condição de parada consiste na árvore possuir $|V_G| - 1$ arestas, temos a garantia de que o α-BEP retorna uma árvore no final de sua execução.

3.2. Algoritmo Construtivo *Randomized Branch Expanding Prim* (R-BEP)

O algoritmo *Randomized Branch Expanding Prim* (R-BEP), desenvolvido por Beltrão et al. [2023], é a base do α-BEP e foi reimplementado neste trabalho. A diferença entre ambos os algoritmos está no no fato do R-BEP não depender do parâmetro α e no critério de escolha das arestas a serem inseridas na árvore.

Ao invés de fazer o primeiro sorteio, das linhas 4 e 5 do Algoritmo 2, o R-BEP escolhe um dos nós do conjunto $Pontas(T, G)$ e atribui pesos para cada um deles com base em seu grau no grafo G, onde vértices com grau menor possuem maior peso, e portanto maior chance de serem escolhidos. Ao invés de fazer o segundo sorteio, correspondente às linhas 9 e 10 do Algoritmo 2, o vértice selecionado pelo R-BEP será algum dos pertencentes ao conjunto $N(v)$, com a mesma ideia de atribuição de peso dos nós.

3.3. Busca Local *Turbo Treevial Search* (TTVS)

A busca local TTVS recebe como entrada uma árvore geradora $T = (V_T, E_T)$, o seu respectivo grafo $G = (V_G, E_G)$ e o conjunto $B(T)$ de vértices *branch* em T. O pseudocódigo da busca local TTVS está representado no Algoritmo 3.

O algoritmo é executado até que não seja mais possível melhorar a solução, como indica a condicional da linha 27 referente ao laço das linhas 2 a 27. O laço começa com uma cópia do conjunto $B(T)$ para $B'(T)$ na linha 3 para poder iterar sobre seus nós no laço das linhas 5 a 26.

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

Algoritmo 3: Turbo Treevial Search (TTVS)

Entrada: Grafo $G = (V_G, E_G)$, Árvore Geradora $T = (V_T, E_T)$ de G, Conjunto $B(T)$ de vértices *branch* em T
Saída : Mínimo Local T^*

1 $T^* \leftarrow T$
2 **repita**
3 $B'(T) \leftarrow B(T)$
4 $houveMelhora \leftarrow Falso$
5 **enquanto** $B'(T) \neq \emptyset \wedge houveMelhora = Falso$
6 Seja v o vértice de menor grau $d(v, T)$ em $B'(T)$
7 $N(v) \leftarrow \{u \in adj(v, T) | u \in B'(T)\}$
8 **se** $N(v) = \emptyset$ **então**
9 Seja u o vértice de maior grau $d(u, T)$ em $adj(v, T)$
10 **senão**
11 Seja u o vértice de menor grau $d(u, T)$ em $N(v)$
12 $B'(T) \leftarrow B'(T) - \{v\}$
13 $T, B(T) \leftarrow RemoverAresta(T, u, v, B(T))$
14 Sejam S_1, S_2 os conjuntos de vértices que pertencem a primeira e segunda componente conexa de T, respectivamente
15 Seja X o conjunto de vértices que possuem extremidades em S_1 e S_2
16 **enquanto** $X \neq \emptyset \wedge houveMelhora = Falso$
17 Seja $(p, q) \in X$ a aresta que minimiza $[d(p, T) = 2] + [d(q, T) = 2]$ (Se ocorrer empate, o algoritmo escolhe a aresta que minimiza $d(p, T) + d(q, T)$)
18 $T, B(T) \leftarrow AdicionarAresta(T, p, q, B(T))$
19 $X \leftarrow X - \{(p, q)\}$
20 **se** $f(T) < f(T^*)$ **então**
21 $T^* \leftarrow T$
22 $houveMelhora \leftarrow Verdadeiro$
23 **senão**
24 $T, B(T) \leftarrow RemoverAresta(T, p, q, B(T))$
25 **se** $houveMelhora = Falso$ **então**
26 $T, B(T) \leftarrow AdicionarAresta(T, u, v, B(T))$
27 **até que** $houveMelhora = Falso$;
28 **retorne** T^*

A busca local inicia percorrendo os vértices de $B'(T)$ no laço da linha 5 enquanto não ocorrer melhora e houver vértices *branch* para serem explorados. A cada iteração é escolhido o vértice v que possui o menor grau $d(v, T)$ na árvore em $B'(T)$, como podemos observar na linha 6. Agora, é necessário determinar um vizinho u de v que forme uma boa aresta (u, v) a ser retirada da árvore. Seja $N(v)$ os vizinhos de v que são ramificações, conforme a Equação 8.

$$N(v) = \{u \in adj(v, T) | u \in B'(T)\} \tag{8}$$

O conjunto $N(v)$ é computado na linha 7 do pseudocódigo. Se $N(v)$ for vazio, ou seja, se não houver *branches* na adjacência de v na árvore, o nó u escolhido é o que possuir o maior grau na adjacência de v. Caso contrário, o nó u será o de menor grau $d(u, T)$ em $N(v)$. A ideia desse critério é identificar vértices que estão mais próximos de terem grau 2. Assim, vértices *branch* tendem a ficar mais próximos de se tornarem nós comuns, enquanto vértices comuns tendem a ficar mais longe de se tornarem *branch*. A determinação de u está indicada na condicional das linhas 8 a 11. Em seguida, na linha 12, removemos o vértice v de $B'(T)$, a fim de que não seja escolhido novamente em iterações futuras.

Uma vez definidos os vértices u e v, podemos remover a aresta (u, v) da árvore T, como é possível ver na linha 13, e em seguida, escolher alguma outra aresta para substituí-la. Seja X o conjunto das arestas de G que não geram ciclo ao serem inseridas em T, determinado nas linhas 14 e 15. A TTVS percorre o conjunto X e escolhe em cada iteração a aresta (p, q) que minimiza as

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

Equações 9 e 10:

$$\rho((p,q)) = [d(p,T) = 2] + [d(q,T) = 2] \tag{9}$$

$$\delta((p,q)) = d(p,T) + d(q,T) \tag{10}$$

A Equação 9 diz quantas extremidades de grau 2 a aresta (p,q) possui na árvore, indicando quantos *branches* serão criados caso essa aresta seja inserida em T. A Equação 10 indica a soma dos graus das extremidades em T, priorizando vértices que possuem menor grau para serem inseridos em caso de empate do primeiro critério. A determinação de (p,q) está descrita na linha 17. Em seguida, a aresta (p,q) é adicionada em T na linha 18.

Caso o movimento de remoção da aresta (u,v) e inserção de (p,q) em T diminua o valor da função objetivo f, a melhor solução encontrada até o momento é atualizada. Caso contrário, a aresta (p,q) é removida e o algoritmo parte para a próxima iteração, como podemos ver na condicional das linhas 20 a 24. Se nenhuma aresta de X diminuir a quantidade de vértices *branch*, a aresta (u,v) é reinserida em T nas linhas 25 e 26 para que outra aresta seja removida na próxima iteração.

Vale ressaltar que todas as adições e remoções de arestas executadas no Algoritmo 3 mantém o conjunto $B(T)$ atualizado, pois um vértice pode se tornar *branch* numa adição e pode deixar de ser *branch* numa remoção.

4. Experimentos computacionais

Os experimentos computacionais realizados consistem na execução dos algoritmos construtivos, **R-BEP** [Beltrão et al., 2023] e **α-BEP**, e das versões do GRASP, **GRASP-R-BEP-TTVS** e **GRASP-FIXED-α-BEP-TTVS**, em três conjuntos de instâncias: Spd RF2, Leighton e Like Merabet, cujas especificações estão presentes na Tabela 2. Neste artigo serão apresentados apenas os resultados do conjunto Spd RF2, pois os demais apresentaram comportamentos semelhantes. Entretanto, um resumo de suas conclusões será apresentado na seção 4.3.

Conjunto de Instâncias	Qtd. Instâncias	Qtd. Vértices	Características	Disponibilizado por
Spd RF2	525	20 a 1000	Grafos conexos e esparsos.	Carrabs et al. [2013]
Leighton	12	450	Grafos conexos, densos e com ótimo de valor 0.	Marín [2015]
Like Merabet	810	50 a 800	Grafos conexos e esparsos.	Merabet e Molnár [2016]

Tabela 2: Grupos de instâncias utilizados para realização dos experimentos computacionais.

Para as heurísticas construtivas foi considerado, para cada instância, o melhor resultado de 100 execuções. Já as versões do GRASP tiveram como condição de parada o tempo de 10 minutos ou 50 iterações sem melhora. Para determinar o valor de α para os algoritmos α-BEP e GRASP-FIXED-α-TTVS, foram feitos testes para $\alpha = 0,1$, $\alpha = 0,2$ e $\alpha = 0,3$. O valor escolhido foi $\alpha = 0,3$, pois se demonstrou ligeiramente melhor que os demais.

Os experimentos foram conduzidos num computador de processador AMD Ryzen 5 com 24 Gb de RAM no sistema operacional Ubuntu 20.04. Os algoritmos foram implementados na linguagem de programação Python 3.8.10.

Para a exibição dos resultados, as instâncias do grupo Spd RF2 foram divididas da mesma forma que Beltrão et al. [2023] fez na análise de seus resultados: em instâcias de tamanho médio, possuindo de 20 a 500 vértices, e instâncias de tamanho grande, contendo de 600 a 1000 vértices. Para cada valor n de vértices, há cinco conjuntos de grafos com o mesmo número m de arestas. Cada par (n,m) forma um grupo contendo cinco grafos distintos, todos com n vértices e m arestas. As instâncias de tamanho médio foram agrupadas de acordo apenas com a quantidade de vértices, de modo que cada um desses grupos corresponde a um conjunto de 25 grafos. Já as instâncias de

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

tamanho grande foram agrupadas pela quantidade de vértices e arestas, assim gerando um conjunto de 5 grafos para cada grupo formado.

O desempenho dos algoritmos construtivos foi comparado com o algoritmo construtivo do trabalho de Moreno et al. [2018], enquanto as versões do GRASP foram comparadas com a meta-heurística ILS, também desenvolvida por Moreno et al. [2018]. O ILS será a referência dentre as meta-heurísticas pois atualmente é o estado da arte. Todos eles também foram comparados com os valores ótimos, que são conhecidos. Vale a pena relembrar que o algoritmo R-BEP, do autor Beltrão et al. [2023], foi reimplementado neste trabalho e os resultados obtidos por esse algoritmo foram similares aos apresentados em seu artigo. O R-BEP será utilizado para comparação com a proposta de variação α-BEP.

A Tabela 3 apresenta os resultados médios considerando a melhor das 100 execuções para cada agrupamento de instâncias, onde as duas primeiras colunas, $|V|$ e $|E|$, indicam a quantidade média de vértices e arestas de cada grupo, respectivamente, enquanto a coluna OPT mostra a média dos valores ótimos. As próximas seis colunas exibem os resultados dos algoritmos construtivos. Nelas, a abreviação **M**, se refere ao algoritmo construtivo de Moreno et al. [2018]. Em cada uma das linhas foi marcado em negrito o algoritmo, dentre os implementados neste trabalho, que obteve o menor resultado. Os algoritmos que tiveram desempenho igual ou superior a coluna **M** foram sublinhados.

Em seguida, as próximas seis colunas apresentam os resultados das meta-heurísticas. Nelas, a abreviação **ILS** é o algoritmo ILS de Moreno et al. [2018], enquanto **G1** e **G2** se referem aos algoritmos GRASP-FIXED-α-BEP-TTVS e GRASP-R-BEP-TTVS, respectivamente. Em cada uma das linhas foi marcado em negrito o algoritmo, dentre os implementados neste artigo, que obteve o menor resultado. Os algoritmos que tiveram desempenho igual ou superior a coluna **ILS** foram sublinhados.

Tanto nos algoritmos construtivos quanto nas meta-heurísticas, o sufixo *gap* nas colunas indica os *gaps* médios entre a solução obtida pelo algoritmo e o ótimo conhecido para cada instância. O *gap* de uma solução S em relação ao ótimo S^* indica uma medida de distância relativa entre ambas e é calculado por meio da equação $gap(S, S^*) = 100 \cdot \left(\frac{f(S)-f(S^*)}{f(S^*)}\right)$.

4.1. Análise dos resultados dos algoritmos construtivos

Todos os algoritmos construtivos tiveram desempenho superior ao construtivo de Moreno et al. [2018], tanto para as instâncias médias quanto para as grandes. O fato do α-BEP e R-BEP possuírem características aleatórias pode ter contribuído para isso, pois o determinismo do algoritmo construtivo de referência pode estar restringindo a exploração do espaço de soluções.

Para as instâncias médias, o α-BEP obteve, de forma unânime, as melhores médias para todos os agrupamentos de instâncias, com *gaps* inferiores a 8% em 14 dos 16 agrupamentos. Já para as instâncias grandes, o R-BEP performou melhor, sendo superior em 17 dos 25 agrupamentos, com *gaps* inferiores a 3% em 18 dos 25 grupos. Entretanto, apesar dessas diferenças, os dois algoritmos obtiveram números similares.

4.2. Análise dos resultados do GRASP

Os resultados da comparação entre os algoritmos implementados, se mostraram bem similares aos analisados na subseção anterior. O algoritmo GRASP-FIXED-α-BEP-TTVS (G1, na Tabela 3) foi melhor que o GRASP-R-BEP-TTVS (G2, na Tabela 3) para todos os agrupamentos de instâncias médias, com *gaps* inferiores a 7% em 14 dos 16 grupos. Além disso, o GRASP-R-BEP-TTVS (G2, na tabela 3) foi superior ao GRASP-FIXED-α-BEP-TTVS (G1, na Tabela 3) em 23 dos 25 agrupamentos de instâncias grandes, com *gaps* inferiores a 3% em 24 desses 25 grupos. Em comparação com os resultados de Moreno et al. [2018], nenhuma das versões do GRASP conseguiu superar resultados da meta-heurística ILS, apesar dos números ficarem próximos. O *gap* médio

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

$\|V\|$	$\|E\|$	OPT	M	M_{gap}	$\alpha-BEP$	$\alpha-BEP_{gap}$	$R-BEP$	$R-BEP_{gap}$	ILS	ILS_{gap}	$G1$	$G1_{gap}$	$G2$	$G2_{gap}$
20	41,8	0,80	1,04	30,00	**0,80**	**0,00**	0,84	5,00	0,80	0,00	**0,80**	**0,00**	0,84	5,00
40	70,8	2,80	3,76	34,30	**3,28**	**17,14**	**3,28**	**17,14**	3,00	7,10	**3,12**	**11,43**	**3,12**	**11,43**
60	95,0	6,30	8,16	29,50	**7,00**	**11,11**	7,12	13,02	6,70	6,30	**6,84**	**8,57**	6,96	10,48
80	119,8	9,20	11,68	27,00	**9,88**	**7,39**	10,36	12,61	9,60	4,30	**9,76**	**6,09**	10,12	10,00
100	144,0	13,30	16,24	22,10	**14,32**	**7,67**	14,76	10,98	13,80	3,80	**14,16**	**6,47**	14,56	9,47
120	168,8	17,50	20,88	19,30	**18,76**	**7,20**	18,92	8,11	18,20	4,00	**18,64**	**6,51**	18,80	7,43
140	193,0	20,90	24,52	17,30	**22,40**	**7,18**	22,92	9,67	21,60	3,30	**22,12**	**5,84**	22,40	7,18
160	217,8	25,00	29,84	19,40	**26,92**	**7,68**	27,36	9,44	25,90	3,60	**26,72**	**6,88**	26,76	7,04
180	242,0	29,10	33,44	14,90	**31,12**	**6,94**	31,40	7,90	30,20	3,80	**30,64**	**5,29**	30,92	6,25
200	266,8	32,60	37,56	15,20	**34,72**	**6,50**	35,32	8,34	33,80	3,70	**34,44**	**5,64**	34,60	6,13
250	321,0	44,60	50,72	13,70	**46,96**	**5,29**	47,52	6,55	45,70	2,50	**46,56**	**4,39**	47,12	5,65
300	380,0	57,40	63,16	10,00	**60,28**	**5,02**	60,80	5,92	58,70	2,30	**59,60**	**3,83**	59,96	4,46
350	434,8	68,60	76,12	11,00	**72,16**	**5,19**	72,52	5,71	70,10	2,20	**71,32**	**3,97**	71,60	4,37
400	489,0	81,80	90,84	11,10	**85,52**	**4,55**	86,04	5,18	83,50	2,10	**85,00**	**3,91**	85,24	4,21
450	548,0	93,40	102,04	9,30	**97,76**	**4,67**	98,08	5,01	95,30	2,00	**96,88**	**3,73**	97,16	4,03
500	602,8	106,70	116,64	9,30	**111,04**	**4,07**	111,72	4,70	108,60	1,80	**110,24**	**3,32**	110,56	3,62
600	637	183,80	187,60	2,10	**185,60**	**0,98**	**185,60**	**0,98**	185,00	0,70	**185,00**	**0,70**	**185,00**	**0,70**
600	674	167,20	173,40	3,70	170,40	1,91	**169,80**	**1,56**	168,60	0,80	170,60	2,03	**170,00**	**1,67**
600	712	150,60	159,40	5,80	156,00	3,59	**154,80**	**2,79**	153,00	1,60	154,40	2,52	**153,80**	**2,12**
600	749	138,80	145,40	4,80	143,80	3,60	**143,20**	**3,17**	140,40	1,20	143,00	3,03	**142,00**	**2,31**
600	787	125,80	135,40	7,60	**131,60**	**4,61**	131,80	4,77	128,80	2,40	131,00	4,13	**130,60**	**3,82**
700	740	214,40	218,80	2,10	216,80	1,12	**216,40**	**0,93**	215,40	0,50	216,80	1,12	**216,60**	**1,03**
700	780	198,00	204,40	3,20	201,80	1,92	**201,20**	**1,62**	199,80	0,90	**201,00**	**1,52**	**201,00**	**1,52**
700	821	180,00	189,00	5,00	184,80	2,67	**184,20**	**2,33**	182,40	1,30	184,60	2,56	**184,00**	**2,22**
700	861	164,00	174,20	6,20	169,60	3,41	**168,60**	**2,80**	167,40	2,10	169,00	3,05	**168,20**	**2,56**
700	902	154,20	165,20	7,10	**159,00**	**3,11**	159,60	3,50	157,00	1,80	**158,60**	**2,85**	158,80	2,98
800	843	245,60	250,80	2,10	248,80	1,30	**248,20**	**1,06**	246,60	0,40	248,60	1,22	**248,00**	**0,98**
800	886	227,60	235,00	3,30	231,40	1,67	**231,00**	**1,49**	229,80	1,00	231,00	1,49	**230,80**	**1,41**
800	930	208,40	216,20	3,70	213,60	2,50	**213,00**	**2,21**	211,40	1,40	212,40	1,92	**212,00**	**1,73**
800	973	194,20	205,00	5,60	**199,40**	**2,68**	**199,40**	**2,68**	197,20	1,50	198,60	2,27	**198,40**	**2,16**
800	1017	176,20	189,00	7,30	183,20	3,97	**183,00**	**3,86**	179,60	1,90	181,40	2,95	**181,00**	**2,72**
900	944	279,60	285,20	2,00	**282,80**	**1,14**	**282,80**	**1,14**	280,60	0,40	282,80	1,14	**282,60**	**1,07**
900	989	259,20	268,80	3,70	264,00	1,85	**263,40**	**1,62**	261,00	0,70	263,00	1,47	**262,80**	**1,39**
900	1034	240,60	250,60	4,20	246,00	2,24	**245,60**	**2,08**	243,60	1,20	246,00	2,24	**245,40**	**2,00**
900	1079	223,20	235,80	5,60	230,60	3,32	**230,00**	**3,05**	226,60	1,50	229,60	2,87	**228,40**	**2,33**
900	1124	206,00	219,00	6,30	212,60	3,20	**212,40**	**3,11**	209,40	1,70	211,80	2,82	**211,20**	**2,52**
1000	1047	312,00	317,40	1,70	315,00	0,96	**314,80**	**0,90**	313,20	0,40	314,60	0,83	**314,20**	**0,71**
1000	1095	290,00	299,60	3,30	295,40	1,86	**294,40**	**1,52**	292,20	0,80	295,00	1,72	**294,20**	**1,45**
1000	1143	271,20	283,00	4,40	277,00	2,14	**276,00**	**1,77**	275,00	1,40	276,40	1,92	**275,60**	**1,62**
1000	1191	251,00	264,00	5,20	258,60	3,03	**258,20**	**2,87**	254,80	1,50	257,40	2,55	**257,00**	**2,39**
1000	1239	235,20	249,60	6,10	242,80	3,23	**242,40**	**3,06**	238,60	1,40	**240,80**	**2,38**	241,00	2,47

Tabela 3: Resultados obidos pelos algoritmos construtivos e versões do GRASP implementados.

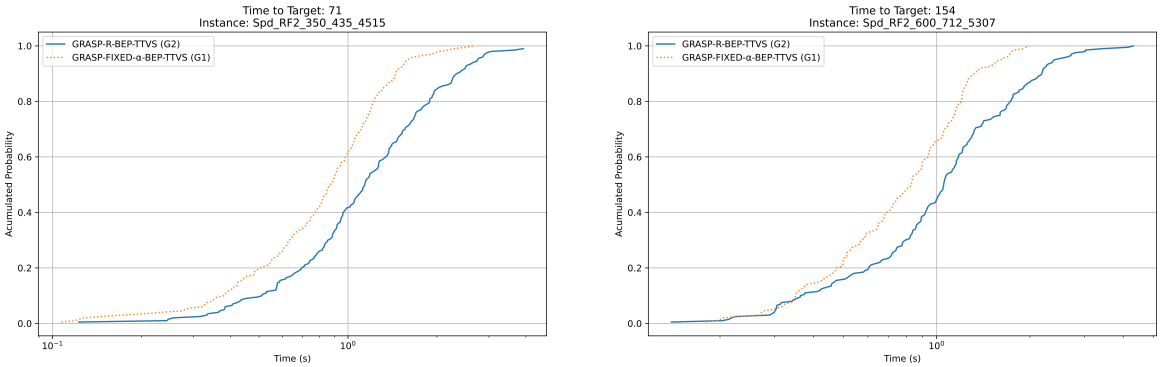
entre o maior dos resultados do GRASP-FIXED-α-BEP-TTVS (G1, na Tabela 3) e GRASP-R-BEP-TTVS (G2, na Tabela 3) em relação ao ILS foi de $1,19\%$, considerando todos os agrupamentos de instâncias, o que evidencia essa proximidade.

Por fim, a Figura 1 exibe gráficos de tempo até o alvo típico (*time to target* [Aiex et al., 2007]) para as instâncias Spd_RF2_350_435_4515, com *target* 71 e Spd_RF2_600_712_5307, com *target* 154. Tais valores para o alvo foram escolhidos por serem os menores que os algoritmos atingiram para essas instâncias. É possível observar nas Figuras 1a e 1b que o algoritmo GRASP-FIXED-α-BEP-TTVS levou menos tempo que o GRASP-R-BEP-TTVS para alcançar 100% de probabilidade para alcançar o *target*. Apesar da Tabela 3 apontar que o GRASP-R-BEP-TTVS obteve resultados melhores para instâncias grandes, o GRASP-FIXED-α-BEP-TTVS se mostrou mais rápido para a instância de tamanho 600 vértices, que pertence a essa categoria.

4.3. Análise dos resultados dos conjuntos de instâncias Like Merabet e Leighton

Em relação ao conjunto Leighton [Marín, 2015], os algoritmos α-BEP e GRASP-FIXED-α-BEP-TTVS apresentaram desempenho igual ou superior, respectivamente, ao R-BEP e GRASP-R-BEP-TTVS em todas as instâncias, encontrando o valor ótimo para as instâncias mais densas. Já no conjunto Like Merabet [Merabet e Molnár, 2016], o desempenho dos algoritmos foi similar independentemente do tamanho da instância, com exceção da comparação entre o GRASP-R-BEP-TTVS e o GRASP-FIXED-α-BEP-TTVS, na qual o primeiro obteve resultados médios inferiores

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br



(a) Instância Spd_RF2_350_435_4515.

(b) Instância Spd_RF2_600_712_5307.

Figura 1: Gráficos Time to Target para o GRASP-FIXED-α-BEP-TTVS e GRASP-R-BEP-TTVS.

ao segundo na maioria dos agrupamentos de instâncias grandes.

5. Conclusões e trabalhos futuros

Este artigo apresentou heurísticas com o objetivo de encontrar soluções para o Problema MBV. Foram apresentados dois algoritmos construtivos, onde o primeiro deles, R-BEP, consiste numa das propostas de Beltrão et al. [2023], enquanto o segundo, α-BEP, é uma variação do R-BEP proposta neste trabalho, que se inspira na heurística construtiva do GRASP. De acordo com os experimentos realizados, observamos que ambas obtiveram resultados melhores do que o algoritmo construtivo de Moreno et al. [2018].

Além disso, a busca local *Turbo Treevial Search*, também proposta neste trabalho, serviu de apoio para as duas versões do GRASP: GRASP-R-BEP-TTVS e GRASP-FIXED-α-BEP-TTVS. A primeira delas se mostrou melhor para as instâncias grandes, enquanto a segunda obteve os melhores resultados para as instâncias médias. Infelizmente, as versões do GRASP propostas não conseguiram superar os resultados do ILS de Moreno et al. [2018]. Apesar disso, *gap* médio entre o maior dos resultados do GRASP-FIXED-α-BEP-TTVS e GRASP-R-BEP-TTVS em relação ao ILS foi de $1,19\%$, considerando todos os agrupamentos de instâncias.

Como trabalhos futuros, propomos o desenvolvimento de outras meta-heurísticas, como a *Variable Neighborhood Search* (VNS) e o estudo de novos critérios de aceitação na busca local a fim de aumentar a diversificação dos algoritmos. Outra vertente seria utililizar a conjectura proposta por de Lima et al. [2024], que estima a quantidade de vértices *branch* de uma solução baseada na densidade do grafo, no contexto de um novo algoritmo construtivo ou busca local. Por fim, um estudo interessante seria analisar o impacto de uma escolha semi-gulosa aleatorizada para os vértices candidatos a serem ramificações no algoritmo α-BEP (Algoritmo 2).

Referências

Aiex, R. M., Resende, M. G., e Ribeiro, C. C. (2007). Ttt plots: a perl program to create time-to-target plots. *Optimization Letters*, 1(4):355–366.

Beltrão, A., Mafort, R., e Protti, F. (2023). Explorando os limites de heurísticas construtivas para o problema da Árvore geradora com número mínimo de ramificações. In *Anais do LV Simpósio Brasileiro de Pesquisa Operacional (SBPO 2023)*, São José dos Campos, SP, Brasil. Galoá.

Boeres, M. C. S., Plastino, A., e de Lima Martins, S. (2023). Identifying elements of solutions of combinatorial optimization problems using measures of graph centrality. In *Anais do Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, volume 55.

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br

Carrabs, F., Cerulli, R., Gaudioso, M., e Gentili, M. (2013). Lower and upper bounds for the spanning tree with minimum branch vertices. *Computational Optimization and Applications*, 56 (2):405–438.

Cerrone, C., Cerulli, R., e Raiconi, A. (2014). Relations, models and a memetic approach for three degree-dependent spanning tree problems. *European Journal of Operational Research*, 232(3): 442–453. ISSN 0377-2217.

Cerulli, R., Gentili, M., e Iossa, A. (2009). Bounded-degree spanning tree problems: models and new algorithms. *Computational Optimization and Applications*, 42(3):353–370. ISSN 1573-2894.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2009). *Introduction to Algorithms*. The MIT Press, 3rd edition. ISBN 978-0-262-53305-8.

De Almeida, A., Martins, P., e Souza, M. (2012). Min-degree constrained minimum spanning tree problem: Complexity, properties, and formulations. *International Transactions in Operational Research*, 19:323 – 352.

de Lima, V. P., de Lima Martins, S., e Boeres, M. C. S. (2024). Utilizando centralidades de grafos e um algoritmo iterated local search para resolver o problema da árvore geradora com número mínimo de vértices d-branch. In *Anais do LVI Simpósio Brasileiro de Pesquisa Operacional*, Fortaleza. Galoá.

Deo, N. e Kumar, N. (1997). Computation of constrained spanning trees: A unified approach. In *Network Optimization*, volume 450 of *Lecture Notes in Economics and Mathematical Systems*, p. 194–220. Springer-Verlag, New York.

FEO, T. A. e RESENDE, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Gargano, L., Hell, P., Stacho, L., e Vaccaro, U. (2002). Spanning trees with bounded number of branch vertices. In Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., e Hennessy, M., editors, *Automata, Languages and Programming*, p. 355–365, Berlin, Heidelberg. Springer Berlin Heidelberg. ISBN 978-3-540-45465-6.

Marín, A. (2015). Exact and heuristic solutions for the minimum number of branch vertices spanning tree problem. *European Journal of Operational Research*, 245(3):680–689. ISSN 0377-2217.

Merabet, M. e Molnár, M. (2016). Generalization of the Minimum Branch Vertices Spanning Tree Problem. Research report, Nanyang Technological University, Singapore. URL `https://hal.science/hal-01403864`.

Moreno, J., Frota, Y., e Martins, S. (2018). An exact and heuristic approach for the d-minimum branch vertices problem. *Computational Optimization and Applications*, 71(3):829 – 855. Cited by: 4.

Silva, D. M., Silva, R. M. A., Mateus, G. R., Gonçalves, J. F., Resende, M. G. C., e Festa, P. (2011). An iterative refinement algorithm for the minimum branch vertices problem. *Lecture Notes in Computer Science*, 6630 LNCS:421 – 433. Cited by: 6.

Sundar, S., Singh, A., e Rossi, A. (2012). New heuristics for two bounded-degree spanning tree problems. *Information Sciences*, 195:226 – 240. Cited by: 20.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. John Wiley & Sons.

DOI: 10.59254/sbpo-2025-212262
https://proceedings.science/p/212262?lang=pt-br