HEURÍSTICA ITERATED LOCAL SEARCH PARA RESOLVER O PROBLEMA MINIMUM BRANCH VERTICES

Jorge Moreno Ramírez, Alexandre Plastino,
Simone de L. Martins
Universidade Federal Fluminense
Niterói – RJ – Brasil
{jmoreno, plastino, simone}@ic.uff.br

RESUMO

O problema de determinar a árvore geradora com menor quantidade de "branch vertices" (Minimum Branch Vertices Problem ou MBV) consiste em encontrar uma árvore geradora num grafo não dirigido que contenha a menor quantidade de vértices com grau maior do que dois ("branch vertices"). Foi provado que este problema é NP-completo e na literatura aparecem diferentes abordagens não exatas para resolvê-lo baseadas em heurísticas ou meta-heurísticas. As meta-heurísticas representam uma importante ferramenta para obtenção de soluções de qualidade, em tempo viável, para problemas de otimização computacionalmente intratáveis. Neste trabalho se propõe a aplicação da meta-heurística Iterated Local Search (ILS) para resolver o problema MBV. Foram realizados experimentos nas instâncias mais utilizadas da literatura para este problema e os resultados obtidos mostram a efetividade do método proposto.

PALAVRAS CHAVE. Meta-heurística, Iterated Local Search, Branch Vertices.

Tópicos (Metaheurísticas, Otimização Combinatória, Teoria e Algoritmos em Grafos)

ABSTRACT

The problem of determining the spanning tree with minimum number of branch vertices (Minimum Branch Vertices Problem or MBV) consists in finding a spanning tree in a non-directed graph containing minimum number of vertices with degree greater than two (branch vertices). This problem has been proven to be NP-complete and in literature there are different non-exact approaches to solve it based on heuristics or metaheuristics. Metaheuristics are among the most important tools for obtaining good solutions, in reasonable time, for computationally intractable optimization problems. This paper proposes the application of Iterated Local Search (ILS) metaheuristic to solve the MBV problem. Experiments were carried out in the most used instances in literature on this problem and the results show the effectiveness of the proposed method.

KEYWORDS. Metaheuristic, Iterated Local Search, Branch Vertices.

Paper topics (Metaheuristics, Combinatorial Optimization, Theory and Algorithms in Graphs)

[1] Trabalho parcialmente financiado por Capes e CNPq

1. Introdução

Os problemas relacionados com árvores geradoras aparecem com frequência na literatura. Um destes problemas consiste em determinar a árvore geradora de um grafo não dirigido com a menor quantidade de vértices de grau maior do que dois ("branch vertices"), conhecido na literatura como "Minimum Branch Vertices problem" (MBV). Este problema foi introduzido inicialmente por [Gargano et al. 2002] a partir do problema de projetar redes óticas com número mínimo de switches. Posteriormente, [Cerrulli et al. 2009] realizaram uma modelagem inteira mista para este problema e apresentaram três heurísticas: *Edge Weighting Strategy* (EWS), *Node Coloring Heuristic* (NCH) e *Combined Strategy* (CS), uma combinação das duas anteriores.

Outras abordagens na literatura, além de propor modelos matemáticos, apresentaram heurísticas ou meta-heurísticas [Sundar et al. 2012] para resolver este problema. Em [Silva et al. 2011] é apresentada uma heurística baseada em uma troca de arestas, substituindo arestas de maior peso por arestas de menor peso. O critério para determinar os pesos das arestas era baseado nos graus dos vértices adjacentes. Um refinamento dessa heurística é proposto em [Silva et al. 2014] com os melhores resultados para o conjunto de instâncias propostas nesse ano. Em [Carrabas et al. 2013] foram propostas outras duas heurísticas, também testadas sobre um amplo conjunto de instâncias.

Em [Marín 2015] é apresentado um novo modelo, além de uma relaxação e uma heurística com os melhores resultados médios para as instâncias utilizadas em [Carrabas et al. 2013]. As instâncias utilizadas em [Silva et al. 2014] também são comparadas no trabalho de [Marín 2015], melhorando significativamente os resultados de trabalhos anteriores.

Neste trabalho é desenvolvida uma heurística Iterated Local Search (ILS) para resolver o problema MBV. Esta meta-heurística tem sido aplicada com sucesso em vários problemas de otimização e sua principal característica reside na possibilidade de escapar de ótimos locais mediante o emprego de uma perturbação na estrutura das soluções encontradas na etapa de busca local.

O artigo está organizado como explicado a seguir. Na Seção 2 é definido o problema MBV. Na Seção 3 é apresentado o esquema geral da meta-heurística ILS e as estratégias desenvolvidas para gerar a solução inicial, busca local e perturbação. Os resultados alcançados nos experimentos computacionais são apresentados na Seção 4. Finalmente, conclusões do trabalho são expostas na Seção 5.

2. O Problema MBV

Dado um grafo $G = (V_G, E_G)$ não dirigido, onde V_G representa o conjunto de vértices do grafo e E_G denota o conjunto de arestas. O grau do vértice $v \in V_G$ será denotado como $d_G(v)$. Se $T = (V_T, E_T)$, com $V_T \subseteq V_G, E_T \subseteq E_G$, é uma árvore de G, então $B_T = \{v \in V_T \mid d_T(v) \geq 2\}$ denota o conjunto de branch vertices de T. O problema consiste em encontrar um a árvore geradora T^*, tal que:

$$|B_{T^*}| = \min\{|B_{T'}| \mid T' \text{é uma árvore geradora de } G\}$$

Neste trabalho a vizinhança de um vértice v num grafo $G = (V_G, E_G)$ será denotada como: $Adj_G(v) = \{u \mid (v,u) \in E_G\}$. Se T é uma árvore geradora de um grafo G, então $T(u,v)$ denota a sub-árvore resultante ao eliminar aresta (u,v) e que contém o vértice u. A outra sub-árvore será denotada como $T(v,u)$ e contém ao vértice v. Deve se notar que nesta notação é importante a ordem em que os vértices são colocados.

As Figuras 2.1, 2.2 e 2.3 ilustram as definições anteriores. Na Figura 2.1 representa-se uma árvore T, enquanto nas Figuras 2.2 e 2.3 representam-se as sub-árvores $T(u,v)$ e $T(v,u)$

respectivamente, obtidas ao eliminar a aresta (u, v).

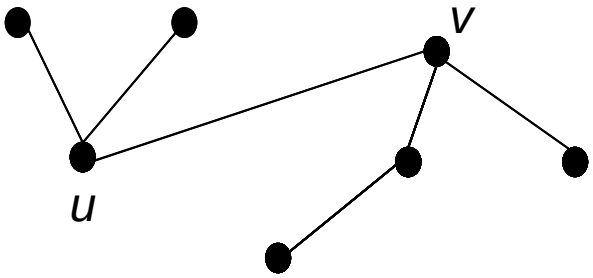

Fig 2.1. Árvore *T*

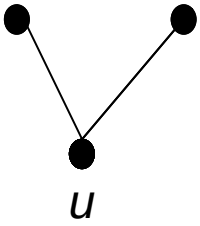

Fig 2.2. Sub-árvore *T(u,v)* de *T*

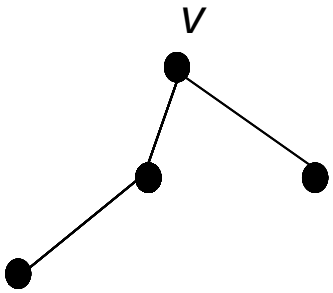

Fig 2.3. Sub-árvore *T(v,u)* de *T*

3. Heurística ILS para o Problema MBV

3.1. Iterated Local Search (ILS)

A meta-heurística Iterated Local Search (ILS) [Lorenço et al. 2010] foi escolhida para desenvolver este trabalho. A mesma tem sido usada em vários problemas de otimização e obtido bons resultados [Penna et al. 2013]. O pseudocódigo para o método ILS é apresentado na Figura 3.1. Inicialmente uma solução inicial é gerada para o problema (linha 1) e uma busca local é aplicada sobre esta solução com o objetivo de melhorar a qualidade da solução construída (linha 2). Entre as linhas 3 e 7 executam-se iterações até se atingir um critério de parada. Em cada iteração realizam-se uma perturbação na solução corrente para tentar escapar dos ótimos locais e logo após uma busca local. Na linha 6 utiliza-se um critério para decidir se a solução corrente será substituída pela nova solução gerada.

procedimento ILS()
1. $x_0 := Solucao_{Inicial}(\square)$
2. $x^* := Busca_{Local}(x_0)$
3. **repita**
4. $x' := Perturbacao(x^*)$;
5. $x'' := Busca_{Local}(x')$;
6. $x^* := Criterio_{de\_Aceitacao}(x^*, x'', \text{history})$;
7. **até** Criterio_de_Parada();
8. **retorne** x^*;

Figura 3.1: Pseudocódigo da meta-heurística ILS

Para o problema abordado, foram desenvolvidos os procedimentos para geração da solução inicial, busca local e perturbação. O critério de aceitação utilizado atualiza a solução corrente escolhendo a árvore que tenha menor número de branch vértices entre a melhor solução encontrada e o ótimo obtido na busca local.

3.2. Solução Inicial

Para gerar uma solução, inicialmente encontram-se as pontes do grafo G como proposto em [Marín 2015]. Uma ponte em um grafo é uma aresta cuja remoção aumenta o número de componentes conexos do grafo e, portanto, toda ponte tem que pertencer a qualquer árvore geradora. Aqueles vértices que possuem grau maior do que dois e têm duas ou mais arestas incidentes que são pontes serão obrigatoriamente branch vertices. Baseado neste fato, determina-se o conjunto daqueles vértices que obrigatoriamente serão branch vértices (denotado por O_G) e o conjunto daqueles vértices para os quais todas suas arestas incidentes são pontes (denotado por P_G).

Para gerar uma solução inicial foi desenvolvida a heurística apresentada na Figura 3.2, baseada na seleção de arestas do grafo G que ainda não estiverem na árvore T . O peso de uma aresta (u,v) será $weight(u,v) = d_G(u) + d_G(v)$. Inicialmente, é construída uma floresta com as m_0 arestas que foram classificadas como pontes. Então são adicionadas arestas até completar a árvore. A adição de arestas é realizada de forma a tentar inserir as arestas de menor custo que não tornem os vértices branch vertices.

procedimento Solucao_Inicial()
1. $T = \{\}$;
2. Incluir em T todas as m_0 pontes do Grafo G ;
3. $m := m_0$;
4. **repita**
5. Incluir em T a aresta de menor peso $(u,v) \in E_G$, tal que $d_T(u) + d_T(v) \leq 1$;
6. $m := m + 1$;
7. **até que não exista aresta** $(u,v) \in E_G$, **tal que** $d_T(u) + d_T(v) \leq 1$;
8. **repita**
9. Incluir em T a aresta de menor peso $(u,v) \in E_G$, e sem formar ciclo tal que $d_T(u) = 1$ e $d_T(v) = 1$;
10. $m := m + 1$;
11. **até que não exista aresta** $(u,v) \in E_G$, **e sem formar ciclo tal que** $d_T(u) = 1$ e $d_T(v) = 1$;
12. **repita**
13. Escolher a aresta $(u,v) \in (E_G \setminus E_T)$ de maior peso tomando a seguinte precedência e sem formar um ciclo:
14. **a)** Aresta tal que $d_T(u) \leq 1$ e $d_T(v) > 2$;
15. **b)** Aresta tal que $d_T(u) > 2$ e $d_T(v) > 2$;
16. **c)** Aresta tal que $d_T(u) \leq 1$ e $d_T(v) = 2$;
17. **d)** Aresta tal que $d_T(u) = 2$ e $d_T(v) > 2$;
18. **e)** Aresta tal que $d_T(u) = 2$ e $d_T(v) = 2$;
19. $m := m + 1$;
20. **até** $m = |V_G| - 1$;
21. **retorne** *T*;

Figura 3.2: Pseudocódigo para geração da solução inicial

3.3. Busca Local

Esta etapa tem como objetivo melhorar a qualidade da solução atual, analisando soluções vizinhas à solução encontrada. Neste trabalho, a busca local é realizada conforme mostrado na Figura 3.3. Na linha 1, a melhor solução corrente $Tbest$ é inicializada com a solução inicial T e o procedimento First_best_neighbor descrito na Figura 3.4 é executado enquanto houver melhoria na solução corrente.

O procedimento First_best_neighbor obtém soluções vizinhas à solução corrente gerando árvores a partir da árvore atual T conforme descrito a seguir. Procura-se um nó de grau 3 (linha 3) e retira-se uma de suas arestas incidentes, digamos (u, v) (linha 5). Então, se existe uma aresta que não pertence à árvore e que liga um nó de grau diferente de 2 na sub-árvore $T(u, v)$ a um nó com grau também diferente de 2 na sub-árvore $T(v, u)$, essa aresta é adicionada, obtendo-se uma árvore geradora T' com menor número branch vértices que T .

```
procedimento Busca_Local( T )
1.      Tbest := T;
2.      repita
3.          melhora := 0;
4.          T' := First_best_neighbor( Tbest );
5.          se |B'_T| < |B_Tbest| então
6.              Tbest := T';
7.              melhora := 1;
8.          fim-se
9.      até melhora = 0;
10.     retorne Tbest;
```

Figura 3.3: Pseudocódigo da Busca Local

```
procedimento First_best_neighbor ( T )
1.      T' := T;
2.      Tbest := T';
3.      para cada v ∈ (B_T − O_G) tais que d_T(v) = 3
4.          para cada u ∈ Adj_T(v) tais que (u, v) não é uma ponte
5.              remover (u, v) de T;
6.              para cada w ∈ (V_T'(v,u) − P_G) tais que
                d_T'(v,u)(w) ≠ 2
7.                  se existe em G a aresta (w, u) então
8.                      adicionar (w, u) a T' e retorne Tbest;
9.                  senão
10.                     se existe em G a aresta (w, x) com
11.                     x ∈ (V_T(u,v) − O_G) e d_T(u,v)(x) ≠ 2  então
12.                         adicionar (w, u) a T' e retorne Tbest;
13.                     fim-se
14.             fim-para
15.             adicionar (u, v) a T';
16.         fim-para
17.     fim-para
18.     retorne Tbest;
```

Figura 3.4: Pseudocódigo do procedimento First_best_neighbor

A busca local desenvolvida neste trabalho foi inspirada nas vizinhanças desenvolvidas em [Marín 2015] e pode diminuir ainda mais os branch vértices nas árvores obtidas por [Marín 2015].

O movimento de melhora IMP1 desenvolvido por [Marín 2015] gera uma solução vizinha localizando uma aresta fora da árvore corrente, cujos vértices pertençam à árvore corrente e sejam de grau diferente de 2. Esta aresta é inserida na árvore corrente e percorre-se o ciclo formado, verificando-se se a substituição de uma aresta da árvore corrente pela aresta inserida diminui o grau de um vértice de grau 3 da árvore corrente. Caso isso ocorra, a nova solução corrente é atualizada substituindo-se as arestas.

Considere a árvore com 4 branch vértices apresentada na Figura 3.5. A linha descontínua indica uma aresta que não pertence à árvore, mas que existe no grafo original. Aplicando-se o movimento IMP1, quatro árvores diferentes poderiam ser obtidas, sendo que em todos os casos a nova árvore teria 3 branch vértices, independentemente da ordem em que os vértices fossem analisados.

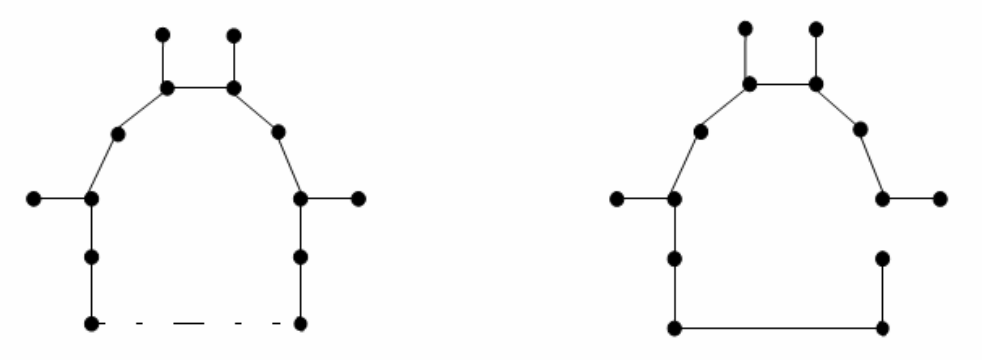

Figura 3.5. Resultado de aplicar IMP1 proposto em [Marín 2015]

Por outro lado, na busca local proposta neste trabalho, além de se obterem essas árvores, a árvore de 2 branch vértices mostrada na Figura 3.6 poderia ser obtida, caso o primeiro vértice analisado seja o vértice v e o primeiro vértice adjacente considerado seja o vértice u .

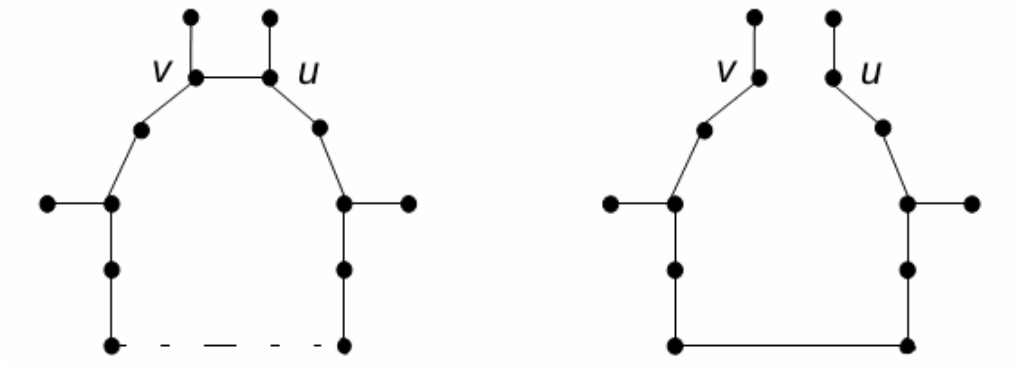

Figura 3.6. Possível resultado ao aplicar First_best_neighbor

3.4. Perturbação

A perturbação é uma etapa de vital importância na meta-heurística ILS, pois seu objetivo é escapar dos ótimos locais alcançados na busca local. A perturbação desenvolvida para o problema abordado consta de duas etapas para modificar a estrutura da árvore como mostrado na Figura 3.7.

A primeira etapa é similar à busca local, sendo que os vértices de grau maior que 3 da árvore corrente são analisados. A segunda etapa é realizada caso o grau de nenhum branch vértice for diminuído na primeira etapa e também é similar à busca local. Neste caso, os vértices a serem considerados são aqueles pertencentes à $T(v,u)$ com grau maior ou igual a 3, e na sub-árvore $T(u,v)$ serão considerados os vértices de grau 2. Claramente a segunda etapa pode incrementar o número de branch vértices na árvore, piorando a qualidade da solução. O objetivo desta etapa é inserir diversidade no procedimento, gerando soluções que possam auxiliar a escapada dos "ótimos locais".

Os movimentos feitos na etapa de permutação não podem ser desfeitos enquanto a Busca Local não encontrar uma solução melhor. Isto é estabelecido com o propósito de não criar ciclos de movimentos ao adicionar e eliminar a mesma aresta sem ter uma melhora. O critério de parada definido consiste em parar as iterações quando não houver mais movimentos possíveis de perturbação.

```
procedimento Perturbacao ( T )
1.      //Primeira etapa
2.      para cada v ∈ (B_T − O_G) tais que d_T(v) ≥ 3
3.          para cada u ∈ Adj_T(v) tais que (u, v) não é uma ponte
4.              remover (u, v) de T;
5.              para cada w ∈ (V_T(v,u) − P_G) tais que
                d_T(v,u)(w) ≠ 2
6.                  se existe em G a aresta (w, u) então
7.                      adicionar (w, u) a T e sair;
8.                  senão
9.                      se existe em G a aresta (w, x) com
10.                     x ∈ (V_T(u,v) − O_G) e d_T(u,v)(x) ≠ 2  então
11.                         adicionar (w, u) a T' e sair;
12.                     fim-se
13.             fim-para
14.             adicionar (u, v) a T';
15.         fim-para
16.     fim-para
17.     //Segunda etapa
18.     para cada v ∈ (B_T − O_G)
19.         para cada u ∈ Adj_T(v) tais que (u, v) não é uma ponte
20.             remover (u, v) de T;
21.             para cada w ∈ (V_T(v,u) − P_G) tais que
                d_T(v,u)(w) ≠ 2
22.                 se existe em G a aresta (w, x) com
23.                 x ∈ V_T(u,v) e d_T(u,v)(x) ≠ 2  então
24.                     adicionar (w, u) a T' e sair;
25.                 fim-se
26.             fim-para
27.             adicionar (u, v) a T';
28.         fim-para
29.     fim-para
```

Figura 3.7: Pseudocódigo da Perturbação utilizada

4. Experimentos Computacionais

Para validar a efetividade do método proposto, foram desenvolvidos experimentos sobre as instâncias mais usadas na literatura neste problema. Nesse sentido foram usadas 500 das instâncias propostas por [Carrabas et al. 2013]. Estas instâncias (Instâncias I-1) contêm grafos esparsos que possuem diferentes densidades. Também foram usadas 31 das instâncias propostas por [Silva et al. 2014]. Estas instâncias (Instâncias I-2) estão baseadas em grafos que possuem um caminho hamiltoniano e portanto é possível achar árvores geradoras sem branch vertices.

Os experimentos foram desenvolvidos sobre um Intel(R) Core(TM) i5-650, 3.20 GHz, com 4Mb de cache e 8 Gb de RAM sobre o sistema operacional Fedora 21 e os métodos foram programados na linguagem C++ usando o compilador gcc. Para cada uma das instâncias, a meta-heurística foi executada 100 vezes e foram obtidos os valores mínimo, médio e desvio padrão.

4.1. Resultados para Instâncias I-1

As Tabelas 4.1 e 4.2 mostram os resultados obtidos sobre as instâncias propostas em [Carrabas et al. 2013] classificadas como Medium Instances (Tabela 4.1) e Large Instances (Tabela 4.2). Cada linha indica um conjunto de 25 grafos para a Tabela 4.1 e 5 grafos para a Tabela 4.2. As duas primeiras colunas representam a quantidade média de vértices e a quantidade média de arestas de cada conjunto. A coluna **Ótimo** representa o valor ótimo para as instâncias analisadas. A coluna **Temp. Oti.** representa o tempo necessário para obter o valor ótimo usando o método proposto por [Marín 2015] que utiliza a sua heurística como limite superior. A coluna **UB** representa os resultados obtidos pela heurística de [Marín 2015]. A coluna **Gap** apresenta a diferença percentual do valor obtido por **UB** em relação ao ótimo calculado utilizando a seguinte equação:

$$\text{Gap} = \frac{UB - Ótimo}{Ótimo} \times 100$$

Finalmente, as três últimas colunas representam o valor mínimo médio e o gap em relação ao ótimo alcançado pelo método proposto, e o tempo médio para processar as instâncias respectivamente.

n'	m'	Ótimo	Temp. Oti.	UB	Gap	ILS	Gap	Temp.Med.
20	41,8	0,8	0	0,8	0,0	0,8	0,0	0
40	70,8	2,8	0,1	2,9	3,6	**2,8**	**0,0**	0,00072
60	95	6,3	1,4	6,6	4,8	**6,5**	**3,2**	0,00232
80	119,8	9,2	2,2	9,5	3,3	**9,4**	**2,2**	0,00508
100	144	13,3	2,9	13,9	4,5	**13,4**	**0,8**	0,00924
120	168,8	17,5	3,7	18	2,9	**17,6**	**0,6**	0,01752
140	193	20,9	4,9	21,8	4,3	**21,2**	**1,4**	0,02896
160	217,8	25	6,1	25,8	3,2	**25,5**	**2,0**	0,04628
180	242	29,1	6,8	30,3	4,1	**29,6**	**1,7**	0,07404
200	266,8	32,6	7,8	33,8	3,7	**33**	**1,2**	0,08364
250	321	44,6	11,3	46	3,1	**45,2**	**1,3**	0,18904
300	380	57,4	13,6	59	2,8	**57,9**	**0,9**	0,32796
350	434,8	68,6	20,3	70,3	2,5	**69,4**	**1,2**	0,55544
400	489	81,8	24,2	83,8	2,4	**82,9**	**1,3**	0,82208
450	548	93,4	29,7	95,7	2,5	**94,5**	**1,2**	1,34588
500	602,8	106,7	35,3	109,4	2,5	**108**	**1,2**	1,9402

Tabela 4.1. Resultados para Instâncias I-1(Medium Instances).

n'	m'	Ótimo	Temp. Oti.	UB	Gap	ILS	Gap	Temp.Med.
600	637	183,8	5,1	184	0,1	**183,8**	0,0	0,4
600	674	167,2	10,9	168,8	1,0	**168,2**	**0,6**	1,1
600	712	150,6	19,9	154,8	2,8	**152,4**	**1,2**	1,9
600	749	138,8	26,7	144	3,7	**140**	**0,9**	2,3
600	787	125,8	39,2	132,8	5,6	**128,2**	**1,9**	2,9
700	740	214,4	6,5	215	0,3	215	**0,3**	0,7
700	780	198	16,8	199,6	0,8	**198,8**	**0,4**	1,9
700	821	180	37,8	184	2,2	**182**	**1,1**	3,4
700	861	164	39,7	169,2	3,2	**166**	**1,2**	3,8
700	902	154,2	57,8	161,8	4,9	**156,6**	**1,6**	4,6
800	843	245,6	6,7	246,6	0,4	**246,4**	**0,3**	0,9
800	886	227,6	19,1	229,6	0,9	**229**	**0,6**	2,7
800	930	208,4	85	213	2,2	**210,4**	**1,0**	4,7
800	973	194,2	65,6	200,2	3,1	**196,6**	**1,2**	6,8
800	1017	176,2	167,2	184	4,4	**179,2**	**1,7**	8,8
900	944	279,6	10	280,6	0,4	280,6	**0,4**	1,3
900	989	259,2	23,4	261,6	0,9	**260,8**	**0,6**	4
900	1034	240,6	44,5	245,4	2,0	**243,2**	**1,1**	6,6
900	1079	223,2	94	229,8	3,0	**226**	**1,3**	9,6
900	1124	206	81,2	214,8	4,3	**208,6**	**1,3**	10
1000	1047	312	10,6	313,4	0,4	**313**	**0,3**	1,6
1000	1095	290	71,1	292,4	0,8	**291,8**	**0,6**	6,2
1000	1143	271,2	112,4	275,8	1,7	**274,2**	**1,1**	11,1
1000	1191	251	150,2	257,8	2,7	**253,6**	**1,0**	14,5
1000	1239	235,2	642,9	244,6	4,0	**238,4**	**1,4**	18,2

Tabela 4.2. Resultados para Instâncias I-1(Large Instances).

Os valores em negrito são aqueles obtidos por este trabalho que melhoraram os valores obtidos pela heurística **UB** que apresentava os melhores valores para essas instâncias. Em 28 dos 31 grupos de instâncias das tabelas, foram conseguidos valores estritamente menores que os melhores resultados para métodos não exatos conhecidos na literatura consultada e em 3 grupos de instâncias foram atingidos os mesmos valores.

4.2. Resultados para Instâncias I-2

As Tabelas 4.3 a 4.6 mostram os resultados obtidos sobre 31 das instâncias propostas em [Silva et al. 2014]. Cada linha indica um grafo. As primeiras três colunas das tabelas representam o nome do arquivo, o número de vértices ***n*** e o número de arestas ***m*** do grafo. As colunas 4, 5 e 6 mostram, respectivamente, os resultados para estas instâncias obtidos pelas heurísticas de [Silva et al. 2014] (**ES**), [Marín 2015] (**UB**) e o tempo para obter a solução **UB** (**Tempo UB**) que representa o melhor tempo conhecido para estas instâncias. As colunas restantes mostram os resultados mínimos, médios e o desvio padrão alcançados pela meta-heurística proposta. No caso da Tabela 4.6, a coluna **Tempo ES** corresponde ao tempo obtido em [Silva et al. 2014] e representa o melhor tempo conhecido na literatura para essas instâncias pois não há resultados para estas heurísticas em [Marín 2015].

Instância	n	m	ES	UB	Tempo UB	ILS			
						Min	Média	Desv	Tempo
le450_5a	450	5714	1	0	0,015	0	0,0	0,0	0,099
le450_5b	450	5734	1	0	0,007	0	0,0	0,0	0,100
le450_5c	450	9803	0	0	0,072	0	0,0	0,0	0,154
le450_5d	450	9757	0	0	0,025	0	0,0	0,0	0,152
le450_15a	450	8186	4	0	0,004	0	0,0	0,0	0,150
le450_15b	450	8169	3	0	0,055	0	0,0	0,0	0,149
le450_15c	450	16680	0	0	0,004	0	0,0	0,0	0,263
le450_15d	450	16750	0	0	0,004	0	0,0	0,0	0,266
le450_25a	450	8160	8	0	0,002	0	0,0	0,0	0,163
le450_25b	450	8263	4	0	0,002	0	0,0	0,0	0,159
le450_25c	450	17343	0	0	0,005	0	0,0	0,0	0,288
le450_25d	450	17425	0	0	0,004	0	0,0	0,0	0,287

Tabela 4.3. Resultados para Instâncias I-2 (*Leighton* Instances).

Instância	n	m	ES	UB	Tempo UB	ILS			
						Min	Média	Desv	Tempo
steind11	1000	5000	33	4	0,451	**0**	0,0	0,0	0,352
steind12	1000	5000	26	4	0,478	**0**	0,0	0,0	0,354
steind13	1000	5000	28	4	0,450	**0**	0,0	0,0	0,353
steind14	1000	5000	28	4	0,420	**0**	0,0	0,0	0,385
steind15	1000	5000	27	4	0,487	**0**	0,0	0,0	0,373

Tabela 4.4. Resultados para Instâncias I-2 (*Beasley* Instances).

Instância	n	m	ES	UB	Tempo UB	ILS			
						Min	Média	Desv	Tempo
alb1000	1000	1998	54	9	0,840	**1**	3,5	0,5	1,194
alb2000	2000	3996	121	19	6,970	**1**	5,8	0,2	9,85
alb3000a	3000	5999	191	29	24,670	**3**	8,1	0,9	42,51
alb4000	4000	7997	247	39	87,830	**4**	9,7	1,7	107,34

Tabela 4.5. Resultados para Instâncias I-2 (*TSPLIB* Instances).

Instância	n	m	ES	UB	Tempo ES	ILS			
						Min	Média	Desv	Tempo
p – 1	200	1300	4	-	0,46	**0**	0,0	0,0	0,013
p – 2	200	1500	2	-	0,50	**0**	0,0	0,0	0,014
p – 3	200	2000	2	-	0,57	**0**	0,0	0,0	0,017
p – 4	200	2200	1	-	0,54	**0**	0,0	0,0	0,018
p – 5	200	2900	1	-	0,69	**0**	0,0	0,0	0,022
p – 6	300	3150	1	-	1,58	**0**	0,0	0,0	0,040
p – 7	300	4500	2	-	2,12	**0**	0,0	0,0	0,055
p – 8	300	5155	1	-	2,04	**0**	0,0	0,0	0,060
p – 9	300	6075	0	-	2,05	**0**	0,0	0,0	0,073
p – 10	300	6300	0	-	3,51	**0**	0,0	0,0	0,072

Tabela 4.6. Resultados para Instâncias I-2 (*Klingman* Instances).

Os valores em negrito são aqueles obtidos neste trabalho que melhoraram os resultados anteriores. No caso das Tabelas 4.3 e 4.4, o método proposto alcançou todos os 17 valores ótimos enquanto que as heurísticas **ES** e **UB** obtiveram 6 e 12 valores ótimos, respectivamente. A heurística proposta obteve melhores resultados, tanto mínimos quanto médios, que as heurísticas **ES** e **UB** para todas as instâncias da Tabela 4.5. Os resultados apresentados na Tabela 4.6 mostram que todos os 10 valores ótimos foram obtidos pelo método proposto enquanto que a heurística **ES** só encontrou 2 valores ótimos. Uma das boas propriedades apresentada pela heurística implementada é que as soluções geradas em cada execução são muito próximas como indica o desvio padrão nas tabelas.

5. Conclusões

O trabalho desenvolvido propõe uma heurística baseada em ILS para abordar o problema de encontrar a árvore geradora de um grafo com número mínimo de branch vértices (MBV). Os experimentos realizados mostraram que, tanto para grafos esparsos (Instâncias I-1), como para grafos que possuam um caminho hamiltoniano (Instâncias I-2), o algoritmo proposto obtém os melhores resultados da literatura consultada.

A variante proposta obteve resultados de melhor qualidade em 95% das instâncias (ou grupo de instâncias) que ainda poderiam ser melhoradas, pois as heurísticas da literatura não obtiveram soluções ótimas para estas instâncias. Para as restantes 5%, a variante proposta obteve o mesmo valor que o melhor resultado não exato conhecido na literatura consultada. Por outro lado, a variante proposta gera soluções muito próximas, sendo esta propriedade muito importante tomando em consideração a qualidade das mesmas.

Um novo desafio será analisar o comportamento da meta-heurística implementada sobre um conjunto de instâncias com outra topografia, tomando em consideração que na literatura aparecem para este problema somente grafos esparsos (com muitas pontes) ou grafos com um ciclo hamiltoniano (sem pontes).

Referências

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

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

Gargano, L., Hell, P., Stacho, L., & Vaccaro, U. (2002). Spanning trees with bounded number of branch vertices. In *Automata, Languages and Programming* (pp. 355-365). Springer Berlin Heidelberg.

Lorenço, H. R., Martin, O. C., & Stützle, T. (2010), Iterated Local Search: Framework and Applications, em Gendreau, M., & Potvin, J. Y. (2010). *Handbook of Metaheuristics* (Vol. 2)., 363-397, New York: Springer.

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.

Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, *19*(2), 201-232.

Silva, D. M. (2011). *Abordagem de refinamento iterativo para o problema da árvore geradora com número mínimo de vértices branch* (Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte (MG), Brazil, 2011.

Silva, R. M., Silva, D. M., Resende, M. G., Mateus, G. R., Gonçalves, J. F., & Festa, P. (2014). An edge-swap heuristic for generating spanning trees with minimum number of branch vertices. *Optimization Letters*, *8*(4), 1225-1243.

Sundar, S., Singh, A., & Rossi, A. (2012). New heuristics for two bounded-degree spanning tree problems. *Information Sciences*, *195*, 226-240.