Minimizando Ramificações em Árvores Geradoras

Adalton S. Almeida
Instituto de Computação – Universidade Federal Fluminense
adalton@ic.uff.br

Loana T. Nogueira
Instituto de Computação – Universidade Federal Fluminense
loana@ic.uff.br

Vinícius G. Pereira de Sá
Instituto de Matemática – Universidade Federal do Rio de Janeiro
vigusmao@dcc.uff.br

RESUMO

Embora não se conheçam algoritmos eficientes para resolvê-los, problemas NP-difíceis estão com frequência presentes em situações reais que demandam solução, justificando o interesse por abordagens heurísticas e algoritmos aproximativos. Uma dessas situações é a da alocação de switches WDM (*Wavelength Division Multiplexing*) em redes óticas. Tais switches são dispositivos sofisticados e caros, portanto minimizar sua quantidade em uma rede torna-se desejável. O problema em que estamos interessados pode ser modelado formalmente como se segue: dado um grafo conexo G, encontrar uma árvore geradora T de G que possua a menor quantidade possível de ramificações, ou seja, que minimize o número de vértices v com grau $d_T(v) \geq 3$. Apresentamos novas heurísticas para o problema, conseguindo resultados superiores aos obtidos por aquelas anteriormente propostas na literatura.

PALAVRAS CHAVE. Grafos, Árvore Geradora, Ramificações, Heurística.

ABSTRACT

Although there are no known efficient algorithms for solving NP-hard problems, such problems frequently arise in practical situations which demand best-effort solutions, hence the interest in approximation algorithms and heuristic-based approaches. One of these situations is the allocation of WDM (*Wavelength Division Multiplexing*) switches in optical networks. Such devices are quite expensive, therefore it is an important task to minimize their number in the network. The problem we are interested in can be formally described as follows: given a connected graph G, find a spanning tree T of G which presents the smallest number of branches, i.e., vertices v with degree $d_T(v) \geq 3$. We introduce novel heuristics for the problem, obtaining better results when compared to those previously proposed in the literature.

KEYWORDS. Graph, Spanning Tree, Branch Vertices, Heuristics.

1. Introdução

O Problema da Árvore Geradora com Número Mínimo de Ramificações (PAGNMR) foi apresentado em [Gargano et al. 2002], e pode ser definido da seguinte forma: dado um grafo G conexo, não direcionado e sem pesos em suas arestas, encontrar uma árvore geradora T de G que possua a menor quantidade possível de vértices com grau maior ou igual a três em T. Tais vértives são chamados de *ramificações* de T.

A motivação prática para o PAGNMR surgiu da necessidade da alocação de *switches* especiais chamados WDM (*Wavelength Division Multiplexing*) em redes óticas multicast. Tais redes utilizam um mecanismo de multiplexação por divisão de comprimento de onda, o WDM, pelo qual é possível ter múltiplos feixes de luz trafegando em um mesmo cabo de fibra ótica, desde que os feixes utilizem comprimentos de onda diferentes. Um *light path* é um caminho percorrido por um feixe ótico, isto é, uma sequência de enlaces de fibra conectando nós consecutivos em uma rede. A tecnologia dos switches óticos WDM permite a divisão do feixe de luz, estendendo o conceito de light paths para *light trees* [Gargano et al. 2004]. Utilizando uma light tree é possível realizar a comunicação multicast em uma rede ótica de um nó origem para múltiplos nós destinos. Em outras palavras, é possível que switches WDM copiem dados diretamente no domínio ótico (pela divisão do feixe de luz). Isto se traduz em uma grande vantagem em relação ao multicast eletrônico, que, ao invés de dividir o feixe de luz, copia pacotes de dados de um ponto a outro utilizando buffers de armazenamento temporário [Gargano et al. 2002]. No entanto, esse sofisticado tipo de switch é bastante caro, de forma que minimizar sua quantidade em uma rede torna-se tarefa importante.

Esse problema de ordem prática foi formulado no âmbito da otimização combinatória. A minimização de switches WDM em uma rede se traduz na minimização de vértices com grau alto (maior ou igual a três) em uma árvore que conecte todo o grafo correspondente à rede em questão. Note que tais vértices indicam exatamente a necessidade de se dividir o feixe de luz, ao invés de apenas retransmiti-lo adiante.

O PAGNMR claramente estende o clássico problema de se encontrar um *caminho hamiltoniano* em um grafo: dado um grafo G com n vértices, deseja-se encontrar um caminho que percorra todos os n vértices do grafo G passando uma única vez por cada vértice. De fato, uma vez que um caminho hamiltoniano é uma árvore geradora sem qualquer ramificação, o número de ramificações na solução ótima do PAGNMR para a instância G será zero se, e somente se, G possui caminho hamiltoniano. Uma vez que decidir se um grafo admite caminho hamiltoniano é problema NP-completo [Garey e Johnson 1979], fica estabelecida a NP-dificuldade do PAGNMR. Sendo assim, não apenas são desconhecidas quaisquer maneiras de se encontrar solução ótima para o PAGNMR em tempo polinomial, como também a existência de uma tal maneira implicaria P = NP, resolvendo um dos problemas matemáticos em aberto mais importantes de nossos tempos. Dessa forma, abordagens pragmáticas para solucionar o PAGNMR compreendem a utilização de métodos não-exatos como heurísticas e algoritmos aproximativos.

Métodos baseados em heurísticas são maneiras de se obter soluções boas, na prática, ainda que não sejam necessariamente as melhores possíveis. Por serem computadas em tempo polinomial no tamanho da entrada e possuirem qualidade aceitável, são por vezes o melhor caminho a se seguir. Algoritmos aproximativos, de forma semelhante, buscam soluções que estejam *comprovadamente* dentro de certa janela de proximidade da solução ótima, ainda que esta última não seja conhecida.

Neste trabalho, apresentamos uma nova heurística para o PAGNMR. Trata-se, em verdade, de uma série de modificações em relação ao algoritmo proposto em [Silva 2011]. Tais modificações lograram obter, nos testes realizados, resultados superiores àqueles conseguidos pela versão original do referido algoritmo, que por sua vez obtivera resultados melhores que os de [Cerulli, Gentili e Iossa 2009], onde o PAGNMR havia sido abordado por meio de heurísticas com ponderação de arestas e coloração de vértices.

Empregaremos ao longo do texto a notação $d_G(v)$ para indicar o grau do vértice v no grafo G, e usaremos $V(G)$ e $E(G)$, respectivamente, para indicar os conjuntos de vértices e arestas

Algoritmo 1 Algoritmo de Refinamento Iterativo

Entrada: um grafo G conexo
Saída: uma árvore geradora T de G com poucas ramificações

1: atribua pesos aleatórios às arestas de G
2: obtenha pelo algoritmo de Kruskal uma árvore geradora T de G com peso mínimo
3: $T_{melhor} \leftarrow T$
4: **repita**
5: `substituição_feita` $\leftarrow$ `falso`
6: determine a lista L_c com todas as possíveis arestas de corte
7: **enquanto** `substituição_feita` $=$ `falso` **e** $L_c \neq \emptyset$ **faça**
8: escolha a melhor aresta de corte $u_c v_c$ dentre as arestas em L_c
9: $T \leftarrow T - u_c v_c$
10: remova $u_c v_c$ de L_c
11: determine a lista L_r com todas as possíveis arestas de reposição
12: **enquanto** $L_r \neq \emptyset$ **faça**
13: escolha a melhor aresta de reposição u_r, v_r dentre as arestas em L_r
14: $T \leftarrow T + u_r v_r$
15: remova $u_r v_r$ de L_r
16: **se** a troca de $u_c v_c$ por $u_r v_r$ é vantajosa **então**
17: $T_{melhor} \leftarrow T$
18: `substituição_feita` $\leftarrow$ `verdadeiro`
19: **senão**
20: $T \leftarrow T - u_r v_r$
21: **fim se**
22: **fim enquanto**
23: **se** `substituição_feita` $=$ `falso` **então**
24: $T \leftarrow T + u_c v_c$
25: **fim se**
26: **fim enquanto**
27: **até** `substituição_feita` $=$ `falso`
28: **retorne** T_{melhor}

de G, com $n = |V(G)|$ e $m = |E(G)|$. Seja $e = uv$ uma aresta do grafo G, e seja S um subgrafo de G. Denotaremos por $S + e$ o grafo cujo conjunto de vértices é $V(S)$ e cujo conjunto de arestas é $E(S) \cup \{e\}$, e denotaremos por $S - e$ o grafo com o mesmo conjunto de vértices e com conjunto de arestas $E(S) \setminus \{e\}$.

2. Algoritmo de Refinamento Iterativo

O algoritmo de refinamento iterativo (RI) apresentado em [Silva 2011] para o PAGNMR está baseado na pesquisa de [Deo e Kumar 1997], que propõe um método geral para encontrar soluções para problemas de otimização em árvores geradoras. Descrevemos a seguir o algoritmo RI, que é dado em pseudocódigo como Algoritmo 1.

Dado um grafo G conexo, não direcionado e sem pesos nas arestas, o primeiro passo do algoritmo RI para o PAGNMR é atribuir pesos aleatórios às arestas de G. A seguir, é computada uma solução inicial para o problema na forma de uma árvore geradora de peso mínimo T de G, sendo para isso utilizado o algoritmo clássico de [Kruskal 1956]. De posse dessa solução inicial, o algoritmo entra em seu laço principal, fazendo a cada iteração uma troca entre arestas: sai uma aresta da árvore T, dita *aresta de corte*, e entra em seu lugar uma aresta de $G \setminus T$, dita *aresta de reposição*. Isto é feito desde que as duas condições seguintes sejam atendidas:

- a troca preserve a conexidade e a aciclicidade de T, isto é, o novo subgrafo T obtido continue a ser uma árvore geradora de G; e

- a troca seja *vantajosa* (segundo algumas heurísticas gulosas, como veremos a seguir).

A cada iteração, é gerada uma lista L_c com as possíveis arestas de corte, isto é, com todas as arestas que são naquele momento candidatas a serem removidas "vantajosamente" de T. A lista L_c será então percorrida para se determinar a aresta $e_c = u_c v_c$ que possibilitar uma maior melhoria em T segundo determinados critérios. Após a remoção de e_c, teremos momentaneamente uma floresta $T - e_c$ com duas árvores T_1 e T_2. Para a escolha da aresta de reposição, que reconectará T_1 a T_2, é selecionada (de uma lista L_r com todas as possíveis candidatas) a aresta $e_r = u_r v_r$, caso exista, que apresente a menor desvantagem (segundo aqueles mesmos critérios) ao ser incluída na árvore geradora. Caso não seja possível reconectar T_1 a T_2, ou se a troca de e_c por e_r não for vantajosa, a aresta e_c não será removida, dando vez à outra possível aresta de corte. O algoritmo continua até que não haja mais trocas vantajosas, retornando então a árvore geradora corrente T e encerrando sua execução.

As escolhas da melhor aresta de corte (linha 8 do Algoritmo 1) e da melhor aresta de reposição (linha 13) — assim como a comparação entre a arestas de corte e de reposição escolhidas para verificar se é vantajoso trocar a primeira pela segunda (linha 16) — são feitas segundo um critério heurístico que se baseia em parâmetros bem definidos. É precisamente a definição desses parâmetros que irá orientar, portanto, toda a execução do algoritmo e a qualidade da solução por ele retornada.

Seja T uma árvore geradora de G, e seja $e = uv$ uma aresta de T. O critério originalmente utilizado pelo algoritmo RI baseia-se nos dois parâmetros seguintes, associados a e:

- *alfa* – $\alpha_T(e)$: é a quantidade de ramificações de T em que a aresta e incide, podendo ser 0, 1 ou 2, segundo a fórmula abaixo:

$$\alpha_T(e) = \sum_{x \in \{u,v\}} b_T(x),$$

onde

$$b_T(x) = \begin{cases} 0, & \text{se} \quad d_T(x) < 3, \\ 1, & \text{se} \quad d_T(x) \geq 3; \end{cases}$$

- *sigma* – $\sigma_T(e)$: é o somatório dos graus dos vértices x e y em T, desconsiderando-se a própria aresta e, segundo a fórmula abaixo:

$$\sigma_T(e) = \left(\sum_{x \in \{u,v\}} d_T(x) \right) - 2.$$

Utilizando esses dois parâmetros, as arestas são avaliadas da seguinte maneira:

- **escolha da aresta de corte**: dentre todas as arestas da lista L_c, é escolhida aquela que possuir o maior α_T, com empates sendo resolvidos pelo critério de possuir o maior σ_T;

- **escolha da aresta de reposição**: dentre todas as arestas da lista L_r, é escolhida aquela que possuir o menor α_T, com empates sendo resolvidos pelo critério de possuir o menor σ_T.

A avaliação de uma possível troca é feita de maneira análoga. Seja T' a árvore geradora de G que é obtida de T pela troca de uma aresta $e_c \in T$ por uma aresta $e_r \in G \setminus T$. Tal troca é considerada vantajosa se:

- $\alpha_{T'}(e_r) < \alpha_T(e_c)$, ou
- $\alpha_{T'}(e_r) = \alpha_T(e_c)$ e $\sigma_{T'}(e_r) < \sigma_T(e_c)$.

A justificativa dos critérios acima é baseada no "nível de infração" dos vértices incidentes às arestas avaliadas: se um vértice é ramificação, ele é mais infrator do que outro que não o seja; como critério de desempate, quanto mais alto o grau de um vértice, mais infrator considera-se aquele vértice.

3. Novas heurísticas para o algoritmo de Refinamento Iterativo

Mantendo a mesma estrutura básica do algoritmo RI proposto em [Silva 2011] com mudanças apenas pontuais, mas empregando novos critérios heurísticos para as avaliações das trocas de arestas, conseguimos obter resultados experimentais significativamente melhores. Detalharemos a seguir as modificações propostas.

3.1. Alteração da solução inicial

Como ponto de partida, o algoritmo RI original emprega o algoritmo de Kruskal, de complexidade $O(m \log n)$. Para executá-lo, são atribuídos pesos aleatórios às arestas do grafo. Não vemos, no entanto, justificativa para partir de uma árvore geradora inicial de peso mínimo, se os pesos em questão são, de todo modo, escolhidos aleatoriamente. Alteramos, portanto, o primeiro passo do algoritmo, que passa a efetuar uma simples busca em largura, de complexidade linear $O(n + m)$, determinando como solução inicial a árvore de largura obtida. Não observamos qualquer prejuízo na qualidade das soluções encontradas.

3.2. Novos critérios de refinamento

Para a implementação de novos critérios de escolha das arestas de corte e reposição, definimos os seguintes parâmetros, onde T é novamente uma árvore geradora do grafo de entrada G, e $e = uv$ é uma aresta de T:

- *beta* – $\beta_T(e)$: indica a *diferença* entre a quantidade de ramificações que existem no conjunto $\{u, v\}$ com e sem a presença da aresta e, isto é, o número de vértices incidentes a e (em G) que são ramificações em $T - e$, menos o número de vértices incidentes a e (em G) que são ramificações em $T + e$;
- *gama* – $\gamma_T(e)$: assume o valor $d_T(u)$ ou $d_T(v)$, o que estiver mais à esquerda na sequência $3, 4, 5, ..., n - 1, 2, 1$.

Valendo-se desses novos parâmetros, as arestas passam a ser avaliadas como descrevemos a seguir (note que os parâmetros são considerados lexicograficamente, isto é, para $k \geq 2$, o k-ésimo parâmetro da lista só é considerado se houver empate do primeiro ao $(k-1)$-ésimo parâmetro, definindo-se o resultado da comparação pelo primeiro parâmetro, na ordem dada, segundo o qual não houver empate):

- **escolha da aresta de corte**: dentre todas as arestas da lista L_c, é escolhida aquela que possuir o menor β_T, depois γ_T mais à esquerda na sequência $3, 4, 5, n - 1, 2, 1$, depois menor σ_T, e finalmente menor σ_G;
- **escolha da aresta de reposição**: dentre todas as arestas da lista L_c, é escolhida aquela que possuir o maior β_T, depois γ_T mais à direita na sequência $3, 4, 5, n - 1, 2, 1$, depois maior σ_T, e finalmente maior σ_G.

Para a avaliação de uma possível troca, seja mais uma vez T' a árvore geradora de G que é obtida de T pela troca de uma aresta $e_c \in T$ por uma aresta $e_r \in G \setminus T$. Tal troca é considerada vantajosa se:

- $\sum_{x \in V(G)} b_{T'}(x) < \sum_{x \in V(G)} b_T(x)$, ou seja, a troca implicou em redução do número total de ramificações; ou
- houve empate no critério anterior, e $\gamma_{T'}(e_r)$ está mais à direita do que $\gamma_T(e_c)$ na sequência $3, 4, 5, ..., n-1, 2, 1$; ou
- houve empate nos critérios anteriores, e $\sigma_{T'}(e_r) > \sigma_T(e_c)$; ou
- houve empate nos critérios anteriores, e $\sigma_G(e_r) > \sigma_G(e_c)$.

Justificamos a seguir a escolha dos novos critérios.

O critério *beta* indica precisamente uma melhora na função-objetivo. Note que um valor negativo para $\beta_T(e)$ indica que o número de ramificações em T diminui com a remoção de e. Considere a seguinte situação envolvendo a escolha da aresta de corte dentre as candidatas

- $e_1 = u_1v_1$, com $d_T(u_1) = 3$ e $d_T(v_1) = 6$; e
- $e_2 = u_2v_2$, com $d_T(u_2) = d_T(v_2) = 5$.

Pela heurística original, e avaliando primeiramente o critério *alfa* por ela utilizado, as arestas estariam empatadas, pois $\alpha_T(e_1) = \alpha_T(e_2) = 2$. O segundo critério, *sigma*, desempataria a favor da segunda aresta, pois $\sigma_T(e_2) > \sigma_T(e_1)$. No entanto, a remoção da aresta e_2 não reduziria o número de ramificações ($\beta(e_2) = 0$), ao passo que a remoção da aresta e_1 traria ganho imediato para a árvore, reduzindo em uma unidade o número de ramificações ($\beta(e_1) = -1$). Perceba que só foi possível medir esse ganho avaliando a *variação* da quantidade de ramificações da árvore, o que é dado precisamente pelo parâmetro *beta*. Segundo nosso critério, seria e_1 a aresta escolhida.

De forma intuitiva, o parâmetro *gama* permite priorizar arestas de corte e reposição que façam a árvore se aproximar mais rapidamente de uma situação em que possa se ver livre de ramificações que ainda possua. Quanto mais à esquerda estiver o grau $d_T(v)$ de um vértice v na sequência $3, 4, 5, ..., n-1, 2, 1$, mais próximo está v de *deixar de ser* uma ramificação de T. O critério proposto observa, portanto, dentre os vértices incidentes a uma aresta, qual deles está mais próximo de deixar de ser uma ramificação, e utiliza apenas o grau *desse vértice* (e não de ambos os vértices) como parâmetro para aquela aresta.

Quanto ao parâmetro *sigma* com subscrito T, que dá a soma dos graus em T dos vértices incidentes a uma aresta, nosso critério prioriza a saída de arestas com σ_T *baixo* e a entrada de arestas com σ_T *alto*. Este critério opera, portanto, de forma exatamente antagônica ao critério utilizado no algoritmo original de [Silva 2011]. Se o objetivo é minimizar a quantidade de ramificações, entendemos que seja mais intuitivo escolhermos uma aresta que, ao sair da árvore, mais aproxime os vértices que a ela incidiam de uma situação em que deixem de ser ramificações. Nossa percepção é a de que tende a ser ineficaz remover arestas cujos extremos tenham graus muito altos (e que dificilmente deixarão de ser ramificações com o desenrolar do algoritmo, de qualquer maneira), mais valendo beneficiar aqueles vértices que tem maiores chances de deixarem de ser ramificações por já apresentarem graus relativamente baixos. Este é, em suma, o mesmo raciocínio utilizado no critério *gama*, mas considerando agora ambos os vértices incidentes à aresta em questão. Note que o parâmetro *sigma* só será observado se tiver havido empate no parâmetro *gama*, em que apenas o vértice de cada aresta que está mais à esquerda na sequência (mais próximo de deixar de ser ramificação) é considerado.

Finalmente, o parâmetro *sigma* com subscrito G é calculado de forma análoga ao anterior, com a diferença de que em $\sigma_G(e)$ consideramos os graus dos vértices incidentes a e no grafo G, ao

invés dos graus daqueles vértices na árvore T, como em $\sigma_T(e)$. Para a escolha da aresta de corte, selecionamos em L_c aquela que tiver o menor σ_G. Isto se justifica pelo fato de que um vértice v com grau baixo em G incide provavelmente em um número pequeno de arestas de $G \setminus T$. Removendo uma aresta incidente a v, aumentamos o número de arestas "livres" incidentes a v, isto é, arestas que poderão mais à frente ser usadas como aresta de reposição. Se escolhêssemos, por outro lado, remover arestas incidentes a vértices com grau alto em G, o aumento de uma unidade no número de arestas de $G \setminus T$ incidentes àqueles vértices seria menos significativo, uma vez que já havia um bom número de tais arestas. Para a escolha da aresta de reposição, escolhemos em L_r aquela de menor σ_G, por raciocícnio análogo.

4. Resultados experimentais

Nesta seção, apresentamos os resultados experimentais das novas heurísticas propostas na Seção 3, comparando-os com os resultados obtidos pelo algoritmo RI original de [Silva 2011].

O primeiro passo da etapa experimental de nosso trabalho foi a definição dos grafos de entrada. Para isso, implementamos um gerador de grafos aleatórios do modelo $G_{n,p}$ de Erdos-Renyi, em que n é o número de vértices do grafo e p é a densidade esperada do grafo, isto é, a probabilidade de que dois vértices v, w sejam vizinhos em G, para todo par $v, w \in V(G)$. Os grafos gerados foram armazenados em arquivos-texto para serem processados pelas heurísticas.

Em seguida, implementamos o algoritmo IR apresentado na Seção 2, tanto em sua versão original quanto na versão que contempla todas as alterações citadas no inicio da Seção 3. Submetemos, então, cada grafo de entrada aos dois programas, e comparamos a quantidade de ramificações presentes nas soluções retornadas.

Nossos grafos de entrada compreenderam:

1. 500 grafos com 100 vértices e densidades 5%, 10%, 20% e 30%, totalizando 2.000 grafos;
2. 1000 grafos com 50 vértices e densidades 5%, 10%, 20% e 30%, totalizando 4.000 grafos;
3. 5.000 grafos com 35 vértices e densidades 5%, 10%, 20% e 30%, totalizando 20.000 grafos;
4. 10.000 grafos com 25 vértices e densidades 5%, 10%, 20% e 30%, totalizando 40.000 grafos.

Conforme descrito nos modelos acima, percebemos que há uma variação entre o número de grafos, vértices e densidade. A densidade é o principal parâmetro que pode afetar significativamente os resultados, em função de contribuir ou não para a presença caminhos hamiltonianos no grafo G como demonstrado em [Dirac 1952]. Considerando esse aspecto e evitar que sejamos repetitivos na apresentação das análises, apresentaremos nas seções seguintes os resultados obtidos em cada modelo, comparando as densidades extremas dos modelos, ou seja de 5% e 30%.

Analisando as Tabelas 1 a 4, podemos perceber que a heurística proposta foi superior em todos os modelos testados. Os melhores resultados foram obtidos para grafos do modelo com densidade de 5%, e o maior número de empates, por outro lado, foi observado para grafos do modelo com densidade de 30%. Omitimos os resultados obtidos para grafos dos modelos com densidades de 10% e 20%.

5. Conclusão e trabalhos futuros

A partir do algoritmo mais recente encontrado na literatura para o Problema da Árvore Geradora com Número Mínimo de Ramificações, propusemos novas heurísticas, logrando obter melhores resultados.

Verificamos, através de experimentação, que a heurística proposta foi mais eficaz em todos os modelos de grafos aleatórios utilizados, sobretudo nos grafos menos densos, em que caminhos hamiltonianos (árvores com zero ramificações) são encontrados com menos frequência.

Tabela 1: Heurística original × Heurística proposta — 500 grafos com 100 vértices

Densidade 5%			**Densidade 30%**		
Vitórias da heurística original	12	2,4%	Vitórias da heurística original	1	0,2%
Vitórias da heurística proposta	422	84,4%	Vitórias da heurística proposta	4	0,8%
Empates	66	13,2%	Empates	495	99,0%
Total	500	100%	Total	500	100%

Tabela 2: Heurística original × Heurística proposta — 1000 grafos com 50 vértices

Densidade 5%			**Densidade 30%**		
Vitórias da heurística original	15	1,5%	Vitórias da heurística original	2	0,2%
Vitórias da heurística proposta	916	91,6%	Vitórias da heurística proposta	15	1,5%
Empates	69	6,9%	Empates	983	98,3%
Total	1.000	100%	Total	1.000	100%

Tabela 3: Heurística original × Heurística proposta — 5000 grafos com 35 vértices

Densidade 5%			**Densidade 30%**		
Vitórias da heurística original	85	1,7%	Vitórias da heurística original	6	0,1%
Vitórias da heurística proposta	4.375	87,5%	Vitórias da heurística proposta	93	1,9%
Empates	540	10,8%	Empates	4.901	98%
Total	5.000	100%	Total	5.000	100%

Tabela 4: Heurística original × Heurística proposta — 10.000 grafos com 25 vértices

Densidade 5%			**Densidade 30%**		
Vitórias da heurística original	201	2,0%	Vitórias da heurística original	11	0,1%
Vitórias da heurística proposta	7.748	77,5%	Vitórias da heurística proposta	269	2,7%
Empates	2.051	20,5%	Empates	9.720	97,2%
Total	10.000	100%	Total	10.000	100%

Como trabalhos futuros, heurísticas ainda mais eficazes para o algoritmo de Refinamento Iterativo são certamente passíveis de investigação. Uma possibilidade interessante é observar como se comporta o algoritmo quando, ao invés de escolhermos (segundo critérios heurísticos) primeiro a melhor aresta de corte, e em seguida a melhor aresta de reposição, escolhermos a *melhor troca possível*, avaliando todos os possíveis pares (aresta de corte, aresta de reposição). É evidente que tal abordagem será mais cara em termos computacionais, mas demandará ainda tempo polinomial, podendo ser, portanto, uma boa alternativa, caso consiga em geral soluções mais satisfatórias.

Finalmente, existem, na literatura, diversos trabalhos que utilizam algoritmos aproximativos com busca local para resolver problemas em árvores geradoras, dentre os quais podemos citar [Fürer e Raghavachari 1992], [Lu e Ravi 1992], [Khuller, Bhatia e Pless 2003] e [Knauer e Spoerhase 2009]. Em [Chimani e Spoerhase 2013], foi apresentado um algoritmo aproximativo para o PAGMNR, representando a primeira melhoria com relação a um fator de aproximação que pode ser obtido trivialmente. Parece bastante promissor envidar esforços nessa direção, objetivando melhores fatores de aproximação ou complexidades mais baixas para o mesmo fator.

Referências

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

Chimani, M., Spoerhase, J. (2013). Approximating spanning trees with few branches. *Approximation and Online Algorithms*, *Lecture Notes in Computer Science* 7846:30–41.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2001). *Introduction to algorithms*. MIT press.

Deo, N., Kumar, N. (1997). Computation of constrained spanning trees: A unified approach. *Network Optimization*, *Lecture Notes in Economics and Mathematical Systems* 450:196–220.

Dirac, G. A. (1952). Some theorems on abstract graphs. *Proc. London Math. Soc.* 2(3):69–81.

Fürer, M. Raghavachari, B. (1992). Approximating the minimum degree spanning tree to within one from the optimal degree. In *Proceedings of the third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 317–324.

Garey, M. R., Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Co.

Gargano, L., Hammar, M., Hell, P., Stacho, L., Vaccaro, U. (2004). Spanning spiders and light-splitting switches. *Discrete Mathematics* 285(1):83–95.

Gargano, L., Hell, P., Stacho, L., Vaccaro, U. (2002). Spanning trees with bounded number of branch vertices. *Automata, Languages and Programming*, *Lecture Notes in Computer Science* 2380:355–365.

Khuller, S., Bhatia, R., Pless, R. (2003). On local search and placement of meters in networks. *SIAM Journal on Computing* 32(2):470–487.

Knauer, M., Spoerhase, J. (2009). Better approximation algorithms for the maximum internal spanning tree problem. In *Algorithms and Data Structures*, pp. 459–470. Springer.

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

Lu, H.-I., Ravi, R. (1992). The power of local optimization: approximation algorithms for maximum-leaf spanning tree. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 30, p. 533.

Silva, D. M. (2011). Aboradagem de refinamento iterativo para o problema da árvore geradora com número mínimo de vértices branch. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil.