Formulação MTZ para o Problema da Árvore Geradora Mínima sob Restrições de Conflito

Yuri Barbosa Bittencourt
Curso de Ciência da Computação, Universidade Federal do Ceará
Campus do Pici, Bloco 910, Pici, Fortaleza, Ceará, CEP 60440-900
yuribitten@gmail.com

Manoel Campêlo
Departamento de Estatística e Matemática Aplicada, Universidade Federal do Ceará
Campus do Pici, Bloco 910, Pici, Fortaleza, Ceará, CEP 60440-900
mcampelo@lia.ufc.br

Fábio Carlos Sousa Dias
Curso de Ciência da Computação, Universidade Federal do Ceará
Campus de Quixadá, Quixadá, Ceará, CEP 63900-000
fabiocsd@lia.ufc.br

RESUMO

Dados um grafo $G = (V, E)$ ponderado nas arestas e um conjunto C, subconjunto de pares (não ordenados) de elementos de E, pares de arestas conflitantes, busca-se uma árvore geradora de G livre de conflitos de custo mínimo. Tal árvore pode incluir no máximo uma aresta de cada par em C. O problema é NP-difícil no caso geral. Neste trabalho introduzimos uma formulação de programação inteira para o problema que define uma rotulação dos vértices para evitar ciclos. Usando um conjunto de instâncias da literatura, obtivemos bons resultados computacionais em comparação aos melhores existentes.

PALAVRAS CHAVE. Árvore geradora mínima sob restrição de conflito, Formulação matemática, Programação inteira.

Área Principal: Otimização Combinatória, Programação Matemática

ABSTRACT

Given a edge-weigthed graph $G = (V, E)$ and a set C, a subset of pairs of elements of E, conflicting edges, we aim at finding a spanning tree of G with at most one edge of each conflicting pair and minimum cost. The problem is NP-Hard in general. We introduce ILP formulation that defines a labeling of the vertices in order to avoid cycles. Using test instances from the literature, we could obtain good computational results in comparison with the best ones available.

KEYWORDS. Minimum spanning trees under conflict constraints, ILP formulation, Integer programming.

Main Area: Combinatorial Optimization, Mathematical Programming

1. Introdução

O problema de Árvore Geradora Mínima (AGM) é um clássico em ciência da computação, de grande importância devido à sua aplicabilidade em diversos problemas práticos, relacionados principalmente a projetos de redes (de comunicação, de computadores, de transportes etc). O problema é definido em um grafo não direcionado $G = (V, E)$ com custo $c_e \geq 0$ associado a cada aresta $e \in E$. Deseja-se encontrar um subgrafo gerador $H = (V, E')$, $E' \subseteq E$ de G tal que H possua custo mínimo e que seja uma árvore, isto é, um grafo conexo e acíclico. O custo da árvore é a soma dos pesos de suas arestas.

Na literatura, têm-se estudado vários problemas relacionados ao AGM, onde normalmente se impõe uma restrição adicional às de gerar uma árvore (conectividade e ausência de ciclos). Dentre esses problemas, destacamos o Problema da Árvore Geradora Mínima sob Restrições de Conflito (AGMRC), que consiste em, dados um grafo $G = (V, E)$ com custo $c_e \geq 0$ associado a cada aresta $e \in E$ e um conjunto C, subconjunto de pares (não ordenados) de elementos de E, de pares de arestas que definem os conflitos, encontrar uma árvore geradora mínima de G, dentre aqueles que têm a seguinte propriedade: no máximo uma aresta de $\{e, e'\} \in C$ pode estar na árvore.

Um conceito que usualmente é utilizado para definir esse problema é o de grafo de conflitos. O grafo de conflitos $\bar{G} = (E, C)$ é obtido a partir de G e C: seus vértices são dados pelo conjunto E de aresta de G, enquanto suas arestas são descritas pelo conjunto C de pares conflitantes. Com isso, uma solução viável para AGMRC, que é um subconjunto de arestas, corresponde simultaneamente a uma árvore geradora de G e a um conjunto independente de $\bar{G}$.

1.1. Trabalhos Relacionados

O problema AGMRC foi introduzido em [Darmann et al., 2009, 2011], onde os autores mostram ser NP-Difícil mesmo o caso onde o grafo de conflito é a união de caminhos de tamanho 2. Outros resultados de complexidade aparecem em [Zhang et al., 2011]. Mostra-se que AGMRC é fortemente NP-Difícil e mesmo decidir se existe solução viável é um problema NP-Completo. Em particular, quando o grafo de entrada é um cactus (grafo onde quaisquer dois ciclos não possuem aresta em comum) o problema de decisão torna-se polinomial, mas o de otimização continua NP-Difícil. Outro caso polinomial mostrado no artigo ocorre quando o grafo de conflito é uma união de cliques disjuntas.

Também em [Zhang et al., 2011], são apresentadas algumas heurísticas para o problema e experimentos computacionais preliminares. Métodos de solução mais elaborados foram proposto em [Samer and Urrutia, 2013] e [Samer and Urrutia, 2015]. Os autores usam uma abordagem com *branch and cut* aplicada a duas formulações, uma delas baseadas nas restrições de eliminação de sub-rotas (*Subtour Elimination Constraints*, SECs), introduzidas por Dantzig et al. [Dantzig et al., 1954], e a segunda utilizando restrições de cortes direcionados (*Directed Cutset Constraints*, DCUTs). Os melhores resultados foram obtidos por [Samer and Urrutia, 2015] utilizando a formulação com restrições de SECs.

Para AGM, formulações baseadas em SECs e DCUTs possuem a propriedade de integralidade (a relaxação linear fornece o ótimo inteiro), sendo portanto uma opção natural para modelar problemas relacionados. Porém ambos esses grupos de restrições são potencialmente em números exponencial e demandam a aplicação de algoritmos de separação que, mesmo polinomial, pode tornar bastante dispendioso, quando aplicados muitas vezes.

2. Formulação com Restrição de Eliminação de Ciclos via Rótulos

Neste trabalho, iremos utilizar uma conhecida forma de modelar a eliminação de ciclos, usando apenas um número linear de variáveis, em adição às tradicionais variáveis que marcam as arestas escolhidas.

A formulação para árvore, apresentada por Gouveia [Gouveia, 1995], se baseia nas restrições Miller-Tucker-Zemlin [Miller et al., 1960], comumente usadas na modelagem do Problema do Caixeiro Viajante e problemas relacionados. Basicamente, a ideia pode ser empregada em quaisquer

problemas onde ciclos não são permitidos. A estratégia é determinar rótulos para os vértices e escolher um arco (i, j) apenas se o rótulo de j for maior que o rótulo de i. Isto vai assegurar que, em qualquer caminho formado, os vértices estão dispostos em ordem crescente de rótulos, o que evita a formação de ciclos.

Nesse sentido, vamos transformar o grafo não-direcionado $G = (V, E)$ em um grafo direcionado $D = (V, A)$. Substituímos cada aresta $e = \{i, j\} \in E$ pelos arcos (i, j) e (j, i) em A, com custo igual c_e em ambas as direções. Definida uma raiz $r \in V(G)$, os arcos escolhidos para formar a árvore são tais que o caminho entre r e cada outro vértice $v \in V \setminus \{r\}$ é orientado, com origem em r e destino em v. Para isso, para cada $(i, j) \in A$, seja y_{ij} uma variável binária que será igual a 1 se o arco (i, j) estiver na solução e 0 caso contrário.

Vamos definir também uma variável não-negativa u_i, para cada $i \in V$, que representará o rótulo de i, sendo que o rótulo da raiz r é $u_r = 0$. Para o Problema do Caixeiro Viajante, onde existe um único caminho na solução, um rótulo será atribuído de forma exclusiva para um vértice. Já no caso de AGM, um mesmo rótulo pode ser usado em diferentes vértices.

Fixada a raiz r, a referida formulação para AgM é dada por:

$$(\text{AGM}) \quad \min \left\{ \sum_{(i,j)\in A} c_{ij} y_{ij} : \mathbf{y} \in \mathcal{Y}^r \right\}$$

onde $\mathcal{Y}^r$ é o conjunto dos $y \in R^{|A|}$ que satisfazem as restrições abaixo:

$$\sum_{i:(i,j)\in A} y_{ij} = 1, \forall j \in V \backslash \{r\} \tag{1}$$

$$u_i - u_j + n y_{ij} \leq n - 1, \forall (i, j) \in A, j \neq r \tag{2}$$

$$1 \leq u_i \leq n - 1, \forall i \in V \backslash \{r\} \tag{3}$$

$$u_r = 0 \tag{4}$$

$$y_{ij} \in \{0, 1\}, \forall (i, j) \in A \tag{5}$$

A restrição (1) garante que, para todo vértice diferente da raiz, apenas um arco de entrada será selecionado na solução. A restrição (2) garante que cada arco incluído na árvore é direcionado de um vértice com menor rótulo para um de maior. A restrição (4) atribui o rótulo 0 à raiz da árvore. As restrições (3) definem os limites dos rótulos. Assim, a atribuição de rótulos aos vértices verifica-as seguintes condições. Para cada aresta $\{i, j\}$, se:

- $y_{ij} = 1$ (e $y_{ji} = 0$), então $1 \leq u_j - u_i \leq n - 1$.
- $y_{ji} = 1$ (e $y_{ij} = 0$), então $1 \leq u_i - u_j \leq n - 1$.
- $y_{ij} = 0$ e $y_{ji} = 0$, então $-(n - 1) \leq u_i - u_j \leq n - 1$.

Para um mesmo y que verifique (1), qualquer atribuição de rótulos que satisfaçam as condições acima retorna uma solução viável.

Desroches e Laporte [Desrochers and Laporte, 1991] observaram que a desigualdade (2) pode ser fortalecida com o seguinte *lifting*:

$$u_i - u_j + n y_{ij} + (n - 2) y_{ji} \leq n - 1, \forall (i, j) \in A, j \neq r \tag{6}$$

Ele obriga também que, se $y_{ij} = 1$, então $u_j = u_i + 1$.

Gouveia [Gouveia, 1995] apresentou outras desigualdades válidas para o modelo:

$$\sum_{k\neq i:(k,j)\in A} y_{kj} + u_i - u_j + ny_{ij} \leq n-1, \forall (i,j) \in A, j \neq r \tag{7}$$

$$\sum_{k\neq i:(k,j)\in A} y_{kj} + u_i - u_j + ny_{ij} + (n-3)y_{ji} \leq n-1, \forall (i,j) \in A, j \neq r \tag{8}$$

A restrição (2) pode ser substituída por uma das desigualdades (6), (7) e (8). Mais ainda, podemos reescrever a restrição (8) ao observar que $\sum_{(k,j)\in A} y_{kj} = \sum_{k\neq i:(k,j)\in A} y_{kj} + y_{ij}$ para todo vértice $j \neq r$ e para toda aresta $(i,j) \in A$. Pela restrição (1) temos que $\sum_{k\neq i:(k,j)\in A} y_{kj} + y_{ij} = 1$, logo $\sum_{k\neq i:(k,j)\in A} y_{kj} = 1 - y_{ij}$. Substituindo a última expressão na restrição (8), chegamos a desigualdade abaixo.

$$u_i - u_j + (n-1)y_{ij} + (n-3)y_{ji} \leq n-2, \forall (i,j) \in A, j \neq r \tag{9}$$

Observe que o mesmo pode ser feito com a restrição (7).

2.1. Formulação Matemática para o AGMRC

Para um dada raiz r definimos P_c^r como:

$$P_c^r = \left\{ \begin{array}{rll} y_{ji} + y_{ij} + y_{kq} + y_{qk} & \leq 1, & \forall (u,v) \in C, u = (i,j), v = (k,q), i,j,k,q \neq r \\ y_{ji} + y_{ij} + y_{rq} & \leq 1, & \forall (u,v) \in C, u = (i,j), v = (r,q), i,j,q \neq r \\ y_{rj} + y_{rq} & \leq 1, & \forall (u,v) \in C, u = (r,j), v = (r,q), j,q \neq r \end{array} \right\}$$

$P_c^r \cap \mathbb{B}^{|A|}$ é o conjunto de vetores de incidência que satisfazem as restrições de conflito.

Dessa forma, o AGMRC pode ser formulado como:

$$(\texttt{AGMRC}) \quad \min \left\{ \sum_{(i,j)\in A} c_{ij} y_{ij} : \mathbf{y} \in \mathcal{Y}^r \cap P_c^r \right\}$$

2.2. Restrições de Cliques

Uma clique no grafo de conflito $\bar{G}$ representa os conflitos mútuos existentes entre as arestas associadas aos vértices da clique. Por exemplo, suponha que a Figura 1 representa uma clique em $\bar{G}$, sendo e_1, e_2 e e_3 arestas no grafo original G.

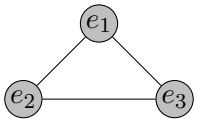

Figura 1: Exemplo Clique

Por simplicidade, iremos representar aqui uma restrição de conflito usando o identificador da aresta em G, ou seja, do vértice de $\bar{G}$. Então, a estrutura da Figura 1 gera as seguintes restrições de conflito: $e_1 + e_2 \leq 1$, $e_1 + e_3 \leq 1$ e $e_2 + e_3 \leq 1$. As três restrições podem ser substituídas no modelo por $e_1 + e_2 + e_3 \leq 1$. Logo, para toda clique H de $\bar{G}$, podemos substituir todas as restrições de conflito relativas às arestas de H por $\sum_{e_i \in H} e_i \leq 1$. Essa estratégia torna-se mais efetiva quando consideramos as cliques maximais do grafo. Em outras palavras, podemos substituir todas as restrições de conflito (relativas a cada aresta de $\bar{G}$) por qualquer família de restrições de clique associadas a uma cobertura das arestas de $\bar{G}$ por cliques maximais. Por cobertura queremos dizer que cada aresta de $\bar{G}$ deve fazer parte de pelo menos uma das cliques maximais escolhidas.

Utilizamos o procedimento guloso descrito pelo Algoritmo 1 para determinar uma cobertura por cliques maximais.

Algoritmo 1: Pseudo-Código: Restrições de Cliques

```
1  C̄ ← C;
2  enquanto C̄ ≠ ∅ faça
3  |  Escolha uma aresta (u, v) ∈ C̄;
4  |  C̄ = C̄\(u, v);
5  |  H = {u, v};
6  |  Seja N(H) os vizinhos simultâneos dos vértices em H;
7  |  enquanto N(H) ≠ ∅ faça
8  |  |  Escolha um vértice k ∈ N(H);
9  |  |  H = H ∪ {k};
10 |  fim enqto
11 |  Adiciona a restrição relacionada a clique H ao modelo;
12 fim enqto
```

3. Resultados Computacionais

Com a finalidade de testarmos o desempenho desta formulação, nós a implementamos no Solver Matemático CPLEX, segundo quatro versões: duas utilizando P_c^r e outras duas utilizando as restrições de cliques. Para cada uma dessas opções, utilizamos as restrições (2) ou o seu lifting (9).

A formulação foi implementada e executada na linguagem C++, usando o sistema operacional Linux em um computador Intel Core i5, com 2,53 GHz e 4 GBytes de memória RAM. Utilizamos o Solver Matemático CPLEX na versão 12.6.1, através da biblioteca Concert Technology. Em todas as implementações, o tempo máximo pré-estabelecido para resolução de cada instâncias foi de 7200 segundos (2 horas) de processamento. Desativamos todas as opções de pré-processamento, heurísticas e geração de cortes do CPLEX.

Nossos resultados foram comparados com aqueles obtidos pelo algoritmo branch-and-cut de [Samer and Urrutia, 2015], que usa a formulação baseada em SECs. O código foi gentilmente cedido pelos autores e executado na mesma máquina. Realizamos uma análise dos tempos computacionais das 4 versões da formulação.

Usamos instâncias do benchmark padrão do problema, proposto por [Zhang et al., 2011]. Essas instâncias são divididos em dois grupos, tipo 1 e tipo 2; Aquelas do tipo 1 são consideradas mais difíceis, para algumas das quais não se sabe dizer se existe solução viável. Quanto as do tipo 2 são instâncias mais fáceis, com soluções ótimas já conhecidas. Todas as instâncias possuem custos inteiros. Iremos exibir os resultados para as instâncias do tipo 1.

Destacamos que em [Samer and Urrutia, 2013], os autores relatam que as formulações por eles apresentados tanto aquelas baseada nas restrições DCUTs quanto nas restrições SECs, não foram capazes de resolver instâncias do tipo 1 em menos que dezenas de horas de execução em uma máquina com processador Intel Core i7, com 3.33 GHz e 16 GBytes de memória RAM.

A Tabela 1 apresenta um resumo indicando a quantidade de instâncias para os quais os algoritmos conseguiram encontrar solução ótima, ou provar a inviabilidade.

Tabela 1: Quantidade de Instâncias que tiveram sua viabilidade/otimalidade provada pelos algoritmos

Algoritmo	Quantidade
Samer and Urrutia [2015] com SECs	10
P_c^r + Restrição 2	12
P_c^r + Restrição 9	13
Clique + Restrição 2	11
Clique + Restrição 9	11

Já na Tabela 2 encontramos os resultados detalhados. As três primeiras colunas indicam a quantidade de vértices, arestas e restrições de conflitos das instâncias, respectivamente. Para cada

algoritmo, temos duas colunas: a primeira indica o status final e a segunda, o tempo computacional. A coluna de status contem o valor da solução ótima, o intervalo em que ela se encontra ou uma das siglas SI, OOM ou SNF, indicando solução inviável, falta de memória, limite máximo de tempo alcançado, respectivamente. Para o algoritmo de [Samer and Urrutia, 2015], a coluna de status exibe o valor da solução ótima, quando o algoritmo a encontra, ou um limite inferior e superior, obtendo com isso um gap. Note que, em algumas instâncias, o algoritmo não encontra uma solução viável, apenas um limite inferior. A última linha contém a média dos tempos computacionais referente às 10 instâncias para as quais todos os algoritmos conseguiram encontrar o ótimo ou provar inviabilidade.

Tabela 2: Resultados computacionais.

			Samer and Urrutia [2015] com SECs		P_c^r + Restrição 2		P_c^r + Restrição 9		Clique + Restrição 2		Clique + Restrição 9	
V	E	C	w	T(s)	w	T(s)	w	T(s)	w	T(s)	w	T(s)
50	200	199	708	0,10	708	0,29	708	0,22	708	0,26	708	0,14
50	200	398	770	0,16	770	0,14	770	0,18	770	0,16	770	0,27
50	200	597	917	1,12	917	0,67	917	0,95	917	0,77	917	0,66
50	200	995	1324	16,83	1324	21,79	1324	8,33	1324	11,78	1324	22,05
100	300	448	4041	1,82	4041	0,15	4041	0,12	4041	0,17	4041	0,17
100	300	897	5658	704,93	5658	49,93	5658	24,74	5658	77,79	5658	55,47
100	300	1344	[6583,33 -]	7200,01	SI	3951,50	SI	512,93	SI	4225,05	SI	2996,11
100	500	1247	4275	0,60	4275	0,86	4275	0,37	4275	1,11	4275	0,40
100	500	2495	5997	3425,49	5997	743,27	5997	560,13	5997	525,62	5997	515,00
100	500	3741	[6538,51 - 9207]	7200,02	OOM	2034,18	OOM	7267,00	OOM	2667,92	OOM	2408,17
100	500	6237	[7583,26 -]	7200,01	SNF	7247,00	SNF	7239,60	SNF	7262,50	SNF	7256,70
100	500	12474	[9847,39 -]	7200,07	SNF	7254,90	SI	527,65	SNF	7231,90	SNF	7225,90
200	600	1797	[13263,97 - 14161]	7200,01	OOM	5586,91	SNF	7294,70	SNF	7256,80	SF	7244,00
200	600	3594	[17490,99 -]	7200,01	SNF	7228,80	SNF	7269,89	SNF	7261,20	SNF	7248,40
200	600	5391	SI	41,20	SI	545,98	SI	33,23	SI	133,52	SI	192,12
200	800	3196	[20679,28 - 22743]	7200,01	OOM	4361,56	OOM	4154,48	SNF	7280,03	OOM	5587,44
200	800	6392	[26196,36 -]	7200,80	SNF	7258,00	SNF	7270,70	SNF	7272,20	SNF	7263,70
200	800	9588	[29986,73 -]	7200,01	SNF	7251,10	SNF	7270,00	SNF	7258,90	SNF	7252,00
200	800	15980	[33068,87 -]	7200,01	SNF	7225,60	SNF	7250,50	SNF	7235,90	SNF	7235,00
300	800	3196	SI	1313,06	SI	147,34	SI	82,20	SI	109,59	SI	491,06
300	1000	4995	[51449,89 -]	7200,01	SNF	7252,50	SNF	7270,80	SNF	7279,10	SNF	7272,00
300	1000	9990	[60867,88-]	7200,01	SNF	7244,80	SNF	7260,50	SNF	7267,20	SNF	7256,90
300	1000	14985	[69162,45 -]	7200,01	SI	6031,10	SI	3481,94	SNF	7241,70	SNF	7223,30
Média Tempo				550,53		151,04		71,05		86,08		127,73

Com os resultados contidos nas tabelas 1 e 2, percebemos que as formulações com as restrições MTZ se mostram mais eficazes para o AGMRC com o uso do Solver CPLEX definindo o status de 3 instâncias a mais, em comparação ao algoritmo branch-and-cut de [Samer and Urrutia, 2015], mais precisamente, ela consegue mostrar que as instâncias 100 - 300 - 1344, 300 - 1000 - 14985 e 100 - 500 - 12474 são inviáveis. Além do mais, considerando os tempos computacionais para as 10 instâncias de que todos os algoritmos obtiveram o status, as formulações MTZ são bem mais eficientes; Mais uma vez, a formulação P_c^r com restrição (9) obtém o melhor tempo computacional na média.

Comparando as quatro versões da formulação MTZ, observamos as duas que utilizam P_c^r, foram capazes de resolver duas instâncias a mais que as versões que empregam as restrições de clique: em uma instância encontraram o ótimo e na outra provaram inviabilidade. Em geral, a formulação P_c^r com restrição 9 é bem mais eficiente, seja por definir o status de uma instâncias a mais, seja por ter o melhor média de tempo computacional. De fato, a Tabela 3 apresenta uma comparação dos tempos computacionais para as quatro versões da formulação. Calculamos a média dos tempos para as instâncias que todas as versões definiram o seu status, seja com a solução ótima encontrada ou seja provando a inviabilidade, 11 ao todo.

A Tabela 4 apresenta uma comparação das quantidades de nós gerados no *branch-and-bound* pelas quatros versões paras as 9 instâncias que todas encontraram a solução ótima. Podemos verificar que as formulações que empregam restrições de clique geram, em média, mais nós do que

Tabela 3: Média dos tempos para 11 instâncias resolvidas.

Algoritmo	Média Tempos
P_c^r + Restrição 2	498,54
P_c^r + Restrição 9	111,22
Clique + Restrição 2	462,35
Clique + Restrição 9	388,50

as formulações com P_c^r.

Tabela 4: Quantidades de nós gerado pelo branch-and-bound do Cplex.

V	E	C	P_c^r + Restrição 2	P_c^r + Restrição 9	Clique + Restrição 2	Clique + Restrição 9
50	200	199	813	268	592	259
50	200	398	107	269	131	182
50	200	597	297	253	369	271
50	200	995	14356	13660	7237	14001
100	300	448	5	5	5	5
100	300	897	17106	14149	21815	16149
100	300	1344	240725	304401	264576	191764
100	500	1247	58	76	96	16
100	500	2495	144311	111302	92458	89956

4. Conclusão e Perspectivas

Neste trabalho apresentamos uma formulação para o Problema da Árvore Geradora Mínima sob Restrições de Conflito (AGMRC) baseada nas restrições Miller-Tucker-Zemlin [Miller et al., 1960]. Essas restrições permitem definir uma formulação compacta para árvore geradora. Apresentamos 4 versões da formulação para o AGMRC, combinando dois conjuntos de restrições MTZ, (2) ou (9), com dois conjuntos de restrições de conflitos, por aresta ou clique maximal do grafo de conflito.

Apresentamos uma avaliação computacional destas 4 versões entre si e com o algoritmo branch-and-cut de [Samer and Urrutia, 2015], que é baseado em uma formulação com restrições tipo SECs. Usando instâncias da literatura, a formulação MTZ mostrou ser mais eficaz, resolvendo otimamente 13 instâncias em comparação a 10 solucionadas por [Samer and Urrutia, 2015]. Além disso, o tempo requerido pela nova formulação foi bem inferior.

Em relação as quatro versões da formulação MTZ, aquela que usa P_c^r com restrição 9 se mostra mais eficiente por definir o status de uma instância a mais e por possuir o menor tempo computacional.

Ressaltamos que nossos experimentos são preliminares, mas já mostram o potencial da formulação MTZ. Percebemos, por outro lado, que a inclusão de muitas restrições de conflitos de clique, em substituição a restrição de aresta, tornaram a matriz mais densa e tiveram um efeito indesejado de aumento do tempo computacional, como podemos observar na Tabela 4, onde o Solver CPLEX resolve, em média, menos nós em mais tempo. Como trabalho futuro, pretendemos estudar mais a fundo estratégias de inclusão de restrições para fortalecer a relaxação do modelo sem torná-lo excessivamente pesado.

Referências

Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M. (1954). Solution of a large scale traveling salesman problem. *Operations Research*, 2:393–410.

Darmann, A., Pferschy, U., and Schauer, J. (2009). *Algorithmic Decision Theory: First International Conference, ADT 2009, Venice, Italy, October 20-23, 2009. Proceedings*, chapter Determining a Minimum Spanning Tree with Disjunctive Constraints, pages 414–423. Springer Berlin Heidelberg, Berlin, Heidelberg.

Darmann, A., Pferschy, U., Schauer, J., and Woeginger, G. J. (2011). Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726 – 1735. 8th Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2009).

Desrochers, M. and Laporte, G. (1991). Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36.

Gouveia, L. (1995). Using the Miller-Tucker-Zemlin constraints to formulate a minimal spanning tree problem with hop constraints. *Computers & Operations Research*, 22(9):959–970.

Miller, C., Tucker, A., and Zemlin, R. (1960). Integer programming formulation of traveling salesman problems. *Journal of ACM*, 7:326–329.

Samer, P. and Urrutia, S. (2013). Um algoritmo de branch and cut para árvores geradoras mínimas sob restrições de conflito. In *Anais do XLV SBPO*, pages 2521–2532, Rio de Janeiro. SOBRAPO.

Samer, P. and Urrutia, S. (2015). A branch and cut algorithm for minimum spanning trees under conflict constraints. *Optimization Letters*, 9(1):41–55.

Zhang, R., Kabadi, S. N., and Punnen, A. P. (2011). The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization*, 8(2):191 – 205.