Formulações para o Problema da k-Floresta Geradora Mínima*

José Robertty de Freitas Costa[1], **Gabriel Hellen de Sousa**[1], **Manoel Campêlo**[2]

[1]Departamento de Computação – Universidade Federal do Ceará (UFC)

[2]Dep. de Estatística e Matemática Aplicada – Universidade Federal do Ceará (UFC)

robertty@alu.ufc.br, gabrielsousa.9957@gmail.com, mcampelo@ufc.br

Resumo. *Introduzimos duas formulações de programação inteira para o problema da k-Floresta Geradora Mínima. Avaliamos seu desempenho computacional com instâncias de teste. Comparamos a quantidade de soluções ótimas encontradas e o tempo médio demandado em função do parâmetro k.*

Abstract. *We introduce two integer programming formulations for the Minimum Spanning k-Forest problem. We evaluate their computational performance with test instances. We compare the number of obtained optimal solutions and computing time as a function of parameter k.*

1. Introdução

O problema da k-Floresta Geradora Mínima (k-FGM) consiste em, dados um grafo G não direcionado com pesos/custos nas arestas e um inteiro k, encontrar uma floresta geradora de G com k árvores, de forma a minimizar o peso da árvore mais pesada. Para $k = 1$, resume-se a encontrar árvore geradora mínima do grafo G. O problema foi introduzido em [Madkour et al. 2019] motivado por aplicação em topologia computacional.

Para exemplificar, considere a instância dada pelo grafo da Figura 1 e $k = 2$. Particionamos, então, o grafo em dois subgrafos conexos, identificados pelas cores azul e amarelo na figura. Destacamos em seguida uma árvore geradora mínima de cada subgrafo, de peso 17 e 16, respectivamente. Essa solução tem, portanto, valor 17, sendo a ótima.

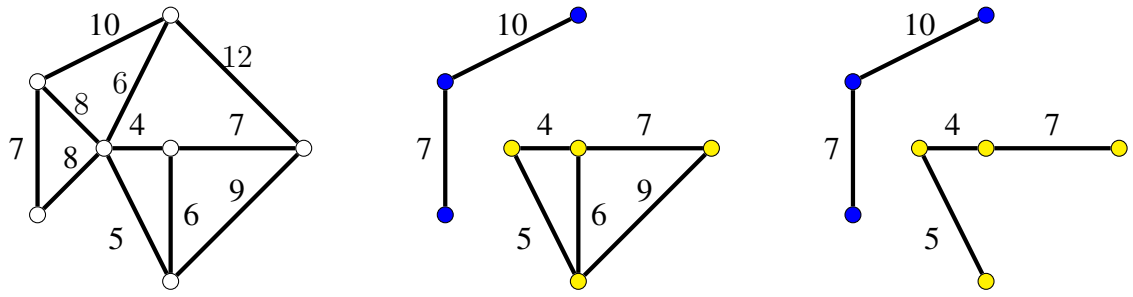

Figura 1. Exemplo de Instância e Solução para k-FGM.

O k-FGM é NP-Difícil já para $k = 2$ [Madkour et al. 2019] ou mesmo que os pesos sejam unitários [Vaishali et al. 2018]. Para o problema, duas heurísticas podem ser encontradas em [Madkour et al. 2019], mas não existe, até o momento, uma abordagem de programação matemática. Neste trabalho, propomos duas formulações de Programação Linear Inteira Mista nas Seções 2 e 3. Apresentamos resultados de experimentos computacionais com as mesmas na Seção 4.

*Financiado por CAPES (J.R. Costa, G. Sousa) e parcialmente financiado por CNPq (M. Campêlo)

2. Formulação por rotulação de árvores (F_TREE)

Ao longo do texto, usamos o conjunto $A(G) = \{uv, vu : uv \in E(G)\}$ para representar a orientação simétrica de G. Inicialmente, apresentamos uma formulação baseada na rotulação (idêntica) dos vértices e arestas de cada árvore. Para cada $t \in [k] := \{1, \ldots, k\}$, definimos as variáveis binárias x_u^t e y_{uv}^t para indicar, respectivamente, se o vértice $u \in V(G)$ e o arco $uv \in A(G)$ pertencem à árvore t. A variável π_u registra a distância entre $u \in V(G)$ e a raiz de sua árvore, e z é o maior peso de uma árvore da floresta.

$$\text{(F\_TREE)} \min \; z \tag{1}$$

$$\text{s.t. } z \geq \sum_{uv \in E(G)} c_{uv}(y_{uv}^t + y_{vu}^t) \qquad \forall t \in [k] \tag{2}$$

$$\sum_{u \in V(G)} x_u^t \geq 1 \qquad \forall t \in [k] \tag{3}$$

$$\sum_{t=1}^{k} x_u^t = 1 \qquad \forall u \in V(G) \tag{4}$$

$$y_{uv}^t + y_{vu}^t \leq x_u^t \, , y_{uv}^t + y_{vu}^t \leq x_v^t \qquad \forall uv \in E(G), \forall t \in [k] \tag{5}$$

$$\sum_{uv \in E(G)} (y_{uv}^t + y_{vu}^t) = \sum_{u \in V(G)} x_u^t - 1 \qquad \forall t \in [k] \tag{6}$$

$$\sum_{u:uv \in E(G)} y_{uv}^t \leq x_v^t \qquad \forall v \in V(G), \forall t \in [k] \tag{7}$$

$$\begin{aligned}(M-1)y_{uv}^t - (M+1)(1-y_{vu}^t) + 1 \leq \pi_u - \pi_v \\ \leq 1 + (M-1)(1-y_{vu}^t) - (M+1)y_{uv}^t \qquad \forall uv \in E(G), t \in [t]\end{aligned} \tag{8}$$

$$\pi_u \geq 0, 0 \leq x_u^t \leq 1 \qquad \forall u \in V(G), \forall t \in [k] \tag{9}$$

$$y_{uv}^t \in \mathbb{B} \qquad \forall uv \in A(G), \forall t \in [k] \tag{10}$$

A restrição (2) junto com a função objetivo (1) asseguram que z seja o maior peso de árvore. Pelas restrições (3)-(4), há pelo menos um vértice em cada árvore construída e todo vértice está em exatamente uma delas. A restrição (5) garante que os arcos uv e vu não podem ser selecionados simultaneamente e que só podem pertencer a uma árvore se ambos u e v pertencem a ela. A restrição (6) assegura que o número de arcos em cada árvore é uma unidade menor que sua quantidade de vértices. A clássica restrição MTZ (8) estabelece $\pi_v = \pi_u + 1$ se o arco uv está numa árvore. Junto com (7), ela garante que os arcos escolhidos não formam ciclo [Desrochers and Laporte 1991]. Dado que o grafo induzido por y é uma floresta e que a restrição (5) faz $x_u^t = x_v^t = 1$ se o arco uv pertence a árvore t, então a restrição (6) garante a integralidade das variáveis x. Em (F_TREE), consideramos $M = \lceil \frac{|V|-k}{2} \rceil$, visto que a raiz de cada árvore não é escolhida a priori.

Para eliminar soluções simétricas, consideramos uma ordenação do conjunto V e adicionamos as seguintes restrições, definindo a formulação (F_TREE$^+$):

$$x_u^t = 0, \forall u \in V, \forall t \in [k] : t > u \quad \text{e} \quad \sum_{j=1}^{t-1} x_w^j \geq x_u^t - \sum_{v<u} x_v^t, \forall u, w \in V : w < u, \forall t \in [u].$$

Tais restrições garantem que a matriz $[x_{ut}]$ tem forma escada, de modo que $u_{t+1} > u_t$, onde $u_t = \min\{u : x_u^t = 1\}$ é o menor vértice na árvore t.

3. Formulação por vértices representantes (F_REP)

A segunda formulação segue a ideia de representantes de classe [Campêlo et al. 2008]. A partir de uma ordenação do conjunto $V(G)$, definimos x_v^u como a variável que determina se o vértice u representa o vértice $v \geq u$, enquanto y_{vw}^u indica se o arco $vw \in A(G)$ é utilizado na árvore representada por $u \leq v, w$. Com isso, o representante de cada árvore é seu vértice de menor índice. As variáveis π_u e z são as mesmas do modelo anterior.

$$\text{(F\_REP)} \min \; z \tag{11}$$

$$\text{s.t. } z \geq \sum_{vw \in E: v,w \geq u} c_{vw}(y_{vw}^u + y_{wv}^u) \qquad \forall u \in V \tag{12}$$

$$\sum_{u \in V} x_u^u = k \tag{13}$$

$$\sum_{u \in V: u \leq v} x_v^u = 1 \qquad \forall v \in V \tag{14}$$

$$y_{vw}^u + y_{wv}^u \leq x_v^u \,, y_{vw}^u + y_{wv}^u \leq x_w^u \qquad \forall u \in V, vw \in E : v,w \geq u \tag{15}$$

$$\sum_{u \in V} \sum_{v > u} y_{vu}^u = 0 \tag{16}$$

$$\sum_{vw \in E: v \geq u} y_{vw}^u = x_w^u \qquad \forall u, w \in V : w > u \tag{17}$$

$$\begin{aligned}(M-1)y_{wv}^u - (M+1)(1-y_{vw}^u) + 1 \leq \pi_w - \pi_v \\ \leq 1 + (M-1)(1-y_{vw}^u) - (M+1)y_{wv}^u \; \forall u \in V, vw \in E : u \leq v < w\end{aligned} \tag{18}$$

$$0 \leq \pi_u \leq M(1 - x_u^u) \qquad \forall u \in V \tag{19}$$

$$0 \leq x_v^u \leq 1 \qquad \forall u, v \in : u \leq v \tag{20}$$

$$y_{vw}^u, y_{wv}^u \in \mathbb{B} \qquad \forall u \in V, \forall vw \in E : v, w \geq u \tag{21}$$

Em lugar rotular cada árvore com um número $t \in [k]$, essa formulação identifica cada árvore por seu vértice de menor índice, não permitindo assim soluções simétricas. As restrições (12), (14), (15) e (18) tem significado similar a (2), (4), (5) e (8), respectivamente. Pela restrição (13), há exatamente k vértices representantes, ou melhor, k árvores. A restrição (16) define u como raiz da árvore que ele representa. Já a restrição (17), que é a contraparte de (7), define exatamente um pai para todo vértice de cada árvore, exceto sua raiz. A integralidade de x deve-se às restrições (13), (17) e (21). Em (F_REP), usamos $M = |V| - k$, já que, escolhida uma árvore, sua raiz está previamente definida.

4. Experimentos Computacionais

Inicialmente foi gerado aleatoriamente um conjunto de grafos para serem base das instâncias de teste. Estes grafos foram criadas a partir de dois parâmetros: número de vértices e densidade d de arestas. Consideramos $|V| \in \{10, 15, 20, 30, 40, 50, 60, 80\}$ e $d \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$. A partir de um mesmo grafo, foram geradas diferentes instâncias, apenas variando o parâmetro de k, com $k = 2^r$, $r = 0, 1, 2 \ldots$, tal que $k \leq |V|$. O fato de k assumir valores de potência de dois deve-se às aplicações práticas do problema citadas na literatura. No total foram geradas 215 instâncias. Porém a instância de teste com $|V| = 10$, $d = 0.3$ e $k = 1$ se mostrou inviável, pois o grafo gerado possui mais de uma componente conexa.

Os modelos foram implementados na linguagem C++ e executados em um computador Intel Xeon, com 2.30 GHz, 8 GBytes de memória RAM e usando o CentOS 7.0. Para a execução dos modelos foi utilizado o CPLEX em sua versão 12.7. A Tabela 1 apresenta o resumo dos resultados dos experimentos computacionais com as instâncias de teste. Para cada uma das formulações (F_REP, F_TREE e F_TREE$^+$), exibimos o tempo médio em segundos demandado para todas as instâncias (Tempo) e a porcentagem de soluções ótimas encontradas dentro do limite de tempo de 30min (% OPT). Cada linha da tabela refere-se às instâncias com o mesmo valor do parâmetro k.

k	**F_REP**		**F_TREE**		**F_TREE$^+$**	
	% OPT	**Tempo**	**% OPT**	**Tempo**	**% OPT**	**Tempo**
1	20,51	1466,59	**94,87**	**140,74**	92,30	164,66
2	27,50	1360,33	67,50	702,05	**72,50**	**567,50**
4	37,50	1131,42	40,00	1077,24	**45,00**	**922,86**
8	**50,00**	**907,96**	25,00	998,69	40,00	1000,28
16	**63,33**	**709,91**	23,33	1297,66	33,33	1099,30
32	**90,00**	**438,27**	25,00	1084,94	40,00	1084,47
64	**100,00**	**423,30**	20,00	481,47	80,00	433,57

Tabela 1. Resumo dos resultados em relação ao parâmetro k

Podemos notar que as restrições de eliminação de simetrias aplicadas a F_TREE foram bem úteis para a melhorar o desempenho da formulação, levando F_TREE$^+$ a resolver otimamente um maior número de instâncias e em menor tempo médio, em todos os grupos com $k \geq 2$. Já na comparação com F_REP, é possível observar que, em relação a porcentagem de soluções ótimas encontradas, F_TREE$^+$ se mostra mais eficiente apenas para $k \leq 4$; nos demais grupos, em média F_REP se apresenta superior. Porém, de modo geral F_TREE$^+$ encontrou o ótimo em 56,54% e F_REP em 44,86% das instâncias. Em relação ao tempo médio de execução, aquele da formulação F_REP tende a diminuir com o aumento do parâmetro k enquanto F_TREE$^+$ é mais rápida quando k assume valores extremos (1, 2, 4 e 64). Esse comportamento pode ser explicado em parte pelo fato de F_TREE ter menos variáveis que F_REP quando k é menor, mas tal diferença diminui quando k tende ao número de vértices. Nesse extremo, porém, o problema se torna fácil.

Como trabalhos futuros propomos a investigação de formulações mais elaboradas, a busca por desigualdades válidas e métodos de solução que melhor explorem propriedades do problema.

Referências

Campêlo, M., Campos, V., and Corrêa, R. (2008). On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156(7):1097–1111.

Desrochers, M. and Laporte, G. (1991). Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Res. Letters*, 10:27 – 36.

Madkour, A.-R., Nadolny, P., and Wright, M. (2019). Finding minimum spanning forests in a graph. In *Proc. of The Midwest Instruction and Computing Symposium (MICS)*.

Vaishali, S., Atulya, M., and Purohit, N. (2018). Efficient algorithms for a graph partitioning problem. In *International Workshop on Frontiers in Algorithmics*, pages 29–42.