

O Problema da Floresta Geradora k -Rotulada

Pedro Jorge de Abreu Figueredo

Mestrado e Doutorado em Ciência da Computação – Universidade Federal do Ceará
Campus do Pici, Bl. 910, 60.440-900 – Fortaleza – CE – Brasil
pedrojorge@lia.ufc.br

Manoel Campêlo

Mestrado e Doutorado em Ciência da Computação – Universidade Federal do Ceará
Campus do Pici, Bl. 910, 60.440-900 – Fortaleza – CE – Brasil
mcampelo@lia.ufc.br

RESUMO

Neste trabalho, estudamos o problema da Floresta Geradora k -Rotulada. Dado um grafo com um rótulo atribuído a cada aresta, o objetivo é encontrar um subgrafo gerador que é uma floresta com a menor quantidade de componentes e cujas arestas têm no máximo k rótulos distintos. O problema é NP-Difícil, e a maioria dos métodos propostos na literatura são baseados em metaheurísticas. Analisamos características do problema com o intuito de propor dois modelos matemáticos de programação inteira mista binária e inteira binária. Propomos algoritmos *Branch & Cut* para resolver os modelos, explorando desigualdades válidas como cortes, regras de *branching*, poda, limites e decomposição de Benders. Além disso, melhoramos o único método exato disponível na literatura, usando paralelismo. Realizamos testes computacionais usando grafos gerados de forma pseudoaleatória, mostrando as melhores configurações para os modelos e *Cplex* encontradas.

PALAVRAS CHAVE. Floresta Geradora k -Rotulada. *Branch & Cut*. Programação Paralela.

Tópicos: Otimização Combinatória, Programação Inteira, Grafos.

ABSTRACT

In this work, we study the k -Labeled Spanning Forest Problem. Given a graph with a label assigned to each edge, the goal is to find a spanning subgraph that is a forest with the smallest amount of components and whose edges have at most k different labels. The problem is NP-Hard, and most of the methods proposed in the literature are based on metaheuristics. We study the characteristics of the problem to propose two mathematical models in 0-1 integer and mixed-integer programming. We propose Branch & Cut algorithms to solve the problem by exploring valid inequalities as cuts, branching rules, pruning, bounds, and Benders decomposition. Moreover, we improve the only exact method available in the literature by using parallelism. We carry out computational tests on pseudorandomly generated graphs, showing the best found configurations for the models and Cplex.

KEYWORDS. k -Labeled Spanning Forest. Branch & Cut. Parallel Programming.

Paper topics: Combinatorial Optimization, Integer Programming, Graphs.

1. O problema

Um grafo simples e não direcionado é dito colorido em arestas (GCA) quando possui um rótulo (cor) associado a cada aresta, podendo ser definido por uma tupla $G = (V, E, L, \ell)$, onde V é o conjunto de vértices, E o conjunto de arestas, L um conjunto de rótulos e $\ell : E \rightarrow L$ é uma função de coloração das arestas de G . Diferente de pesos atribuídos às arestas, as cores mensuram agora qualitativamente as relações entre os vértices extremidade e marcam diferenças entre eles. Uma quantidade considerável de trabalhos foi e ainda é dedicada a problemas em grafos coloridos, segundo [Granata et al., 2013]. Usualmente, problemas em grafos coloridos procuram encontrar subestruturas homogêneas ou heterogêneas segundo esses rótulos. Eles são importantes para campos de pesquisa em redes de comunicação, redes de transporte multimodal, dentre outros.

Dado um inteiro positivo k , seja $\mathcal{F}_G(k)$ o conjunto de todas as florestas geradoras de G com no máximo k rótulos distintos em suas arestas. Seja ainda $\mathcal{W}(G)$ o número de componentes conexas de G , onde as componentes de um grafo são os subgrafos disjuntos conexos maximais (em vértices e arestas). Assim, define-se formalmente o problema abordado, da seguinte forma:

Definição 1.1. *Dados um GCA $G = (V, E, L, \ell)$ e $k \in \mathbb{Z}^+$, o problema da Floresta Geradora k -Rotulada (FGkR) consiste em encontrar uma floresta geradora $F^* \in \mathcal{F}_G(k)$ tal que $\mathcal{W}(F^*) \leq \mathcal{W}(F)$, $\forall F \in \mathcal{F}_G(k)$.*

Seja $G = (V, E, L, \ell)$ um GCA. Dado $E' \subseteq E$, denota-se por $\ell(E')$ o subconjunto de rótulos das arestas em E' , ou seja, $\ell(E') = \{\ell(e) \mid e \in E'\}$. Ao se referir a um subgrafo $H \subseteq G$, com conjunto de vértices $V' \subseteq V$ e conjunto de arestas $E' \subseteq E \cap (V' \times V')$, está referindo-se ao GCA $H = (V', E', \ell(E'), \ell')$, onde ℓ' é a função ℓ restrita a E' . Por simplicidade, usa-se ℓ em lugar de ℓ' na descrição de H , ficando entendido que a função se restringe às arestas do subgrafo. Por outro lado, dado um subconjunto de rótulos $L' \subseteq L$, denota-se por $G[L']$ o subgrafo gerador de G induzido pelos rótulos em L' , ou melhor, pelas arestas $E(L') = \{e \in E \mid \ell(e) \in L'\}$. Em outros termos, $G[L'] = (V, E(L'), L', \ell)$.

Veja que o FGkR pode ser relaxado, sem perda de otimalidade, pedindo determinar apenas um subgrafo gerador (não necessariamente floresta), induzido por até k cores, com a menor quantidade de componentes. Assim, pode-se redefinir o problema como a seguir [Cerulli et al., 2014].

Proposição 1.2. *Dados um $G = (V, E, L, \ell)$ e $k \in \mathbb{Z}^+$, o FGkR equivale a encontrar um subconjunto de rótulos $L^* \subseteq L$ tal que $|L^*| \leq k$ e $\mathcal{W}(G[L^*]) \leq \mathcal{W}(G[L'])$, $\forall L' \subseteq L, |L'| \leq k$.*

A prova de complexidade assintótica e os métodos propostos até agora para resolver o FGkR estão diretamente relacionados ao problema da Árvore Geradora Minimamente Rotulada (*Minimum Spanning Tree Problem* - MLSTP, em inglês), cujo objetivo é encontrar uma árvore geradora com a menor quantidade de cores de um GCA. Em [Chang e Shing-Jiuan, 1997] prova-se que a versão de decisão do MLSTP é um problema NP-Completo, e esse resultado pode ser estendido para uma versão de decisão do FGkR, através de uma redução direta apresentada por [Cerulli et al., 2014]. Logo, o FGkR é NP-Difícil.

[Cerulli et al., 2014], [Consoli e Moreno Pérez, 2015] e [Consoli et al., 2017] apresentam trabalhos com enfoque em métodos meta-heurísticos para o FGkR, adaptando algoritmos similares desenvolvidos para o MLSTP. Dentre as meta-heurísticas utilizadas nesses três trabalhos encontram-se as bem conhecidas Arrefecimento Simulado, Algoritmo Genético, GRASP e Busca em Vizinhança Variável (*Variable Neighborhood Search* - VNS, em inglês), além das variantes do VNS chamadas Intel-VNS e Co-VNS (propostas em [Consoli et al., 2013] para o MLSTP) e do Método Piloto (ver descrição dessa meta-heurística em [Voß et al., 2005]).

Com respeito a métodos exatos, ao nosso conhecimento, o único trabalho existente na literatura também se deve a [Cerulli et al., 2014], que propõem um algoritmo enumerativo, usando *backtracking*.

Neste trabalho propomos duas formulações de programação inteira para o problema, juntamente com os métodos usados para resolvê-las. A primeira usa a ideia de cortes coloridos introduzida em [Silva, 2018] para o MLSTP, enquanto a segunda baseia-se num modelo de fluxo. Além disso, aplicamos paralelismo ao algoritmo de *backtracking* proposto por [Cerulli et al., 2014]. Comparamos os métodos, aplicando-os a um conjunto de instâncias baseadas na literatura, mostrando aqui os resultados com um subconjunto das instâncias mais interessantes. Destes métodos, mostra-se que aquele baseado no modelo por cortes coloridos possui um melhor desempenho geral.

2. Revisão Bibliográfica

O problema da Floresta Geradora k -Rotulada foi definido em [Cerulli et al., 2014] e a literatura associada ao problema ainda é bastante restrita. Nesta seção, são explorados métodos da literatura que são aplicáveis ao problema e estão relacionados ao escopo deste trabalho.

2.1. Heurística k MVCA

O k MVCA consiste de uma adaptação simples da heurística gulosa MVCA, proposta para o MLSTP. A MVCA, inicialmente sugerida por [Chang e Shing-Jiuan, 1997] e depois revisada por [Krumke e Wirth, 1998], escolhe iterativamente rótulos até que o subgrafo por eles induzido seja conexo. Na versão revisada, a escolha do rótulo $\ell^* \in L - L^*$ a ser acrescido a L^* (solução atual) é tal que minimiza o número de componentes de $G[L^*]$ com a inclusão das arestas de rótulo ℓ^* . A complexidade de tempo desse algoritmo é $O(|E| + |L| \cdot \alpha(|E|, |V|) \cdot \min\{|L| \cdot |V|, |E|\})$, onde α é a função inversa de **Ackermann**, que para fins práticos é no máximo 5. O k MVCA, proposto por [Cerulli et al., 2014], consiste simplesmente em aplicar MVCA até atingir k rótulos usados.

2.2. Algoritmo Enumerativo

O algoritmo exato proposto por [Cerulli et al., 2014] aplica um procedimento de *backtracking*, que se baseia no fato de FGkR, conforme Proposição 1.2, consistir em determinar

$$\min\{\mathcal{W}(G[L']) : L' \subseteq L, |L'| \leq k\}.$$

Com essa perspectiva, o procedimento enumera implicitamente todos os subconjuntos de rótulos $L' \subseteq L$, com $|L'| \leq k$, e guarda o que fornece o menor valor de $\mathcal{W}(G[L'])$. Esse processo é organizado em uma árvore de enumeração, como no método *Branch & Bound*, onde a ramificação é feita escolhendo-se um rótulo para ser usado ou proibido na solução, e as podas empregam o limitante obtido com a seguinte proposição, como é explicado a seguir.

Proposição 2.1. [Cerulli et al., 2014] *Seja um GCA $G = (V, E, L, \ell)$ e dois subconjuntos não vazios $M, M' \subseteq L$ tais que $M \cap M' = \emptyset$. A seguinte desigualdade é válida:*

$$\mathcal{W}(G[M \cup M']) \geq \mathcal{W}(G[M]) - \sum_{c \in M'} (\mathcal{W}(G[M]) - \mathcal{W}(G[M \cup \{c\}])).$$

Cada nó da árvore de enumeração é representado por uma solução parcial, definida por um conjunto de rótulos já escolhidos e outros ainda candidatos. Na verdade, é estabelecida uma ordem inicial dos rótulos $L = \{c_1, c_2, \dots, c_{|L|}\}$, tal que $|E(c_i)| \geq |E(c_{i+1})|$ para todo $i \in \{1, \dots, |L| - 1\}$. Então, se d é a profundidade do nó, os rótulos escolhidos em sua solução parcial formam um subconjunto de $\{c_1, \dots, c_d\}$, enquanto os candidatos são os rótulos $c_{d+1}, \dots, c_{|L|}$. A partir desse nó são gerados dois subproblemas: em um o rótulo c_{d+1} é proibido; em outro, ele é obrigatório.

Durante o processo de chamadas recursivas, ao encontrar uma solução parcial L' com k cores, essa será comparada com a melhor solução até o momento, armazenada em L^* , e, caso melhor, atualiza-se a melhor solução encontrada. Quando a solução parcial tem uma única componente, ela é ótima, e o processo termina. Caso $|L'| < k$ e o conjunto de rótulos candidatos não é suficiente para criar uma solução com k cores, uma poda por inviabilidade é aplicada.

Para podar soluções parciais que não podem gerar soluções com valor melhor que o limite superior $\mathcal{W}(G[L^*])$, [Cerulli et al., 2014] usam o limite inferior derivado da Proposição 2.1. Tomando $M = L'$, determina-se $M' \subseteq \{c_{d+1}, c_{d+2}, \dots, c_{|L|}\}$ com $|M'| = k - |M|$ que geraria a maior redução possível para $\mathcal{W}(G[M])$, ou seja, tal que $\sum_{c \in M'} (\mathcal{W}(G[M]) - \mathcal{W}(G[M \cup \{c\}]))$ é máximo.

3. Conceitos e propriedades

Nesta seção são apresentados conceitos e propriedades necessários para a descrição dos modelos, como a definição de um problema equivalente, cortes coloridos e dominância entre cores.

3.1. Problema Equivalente

Para determinar o número de componentes através de uma função linear, converte-se o FGkR, definido sobre um GCA G , em um MLSTP equivalente, definido sobre um grafo estendido G' , como segue.

Definição 3.1. Dado um GCA $G = (V, E, L, \ell)$, chamamos de GCA estendido o grafo conexo $G' = (V', E', L', \ell')$, onde $V' = V \cup \{s\}$ tal que s é um novo vértice, $E' = E \cup \{(s, v) : v \in V\}$, $L' = L \cup C$, sendo $C = \{c_v : v \in V\}$ um conjunto de novos rótulos distintos ($C \cap L = \emptyset$), e ℓ' é tal que:

$$\ell'(e) = \begin{cases} \ell(e), & e \in E, \\ c_v, & e = (s, v), v \in V. \end{cases} \quad (1)$$

A Figura 1 apresenta o GCA original e o GCA estendido a ele associado.

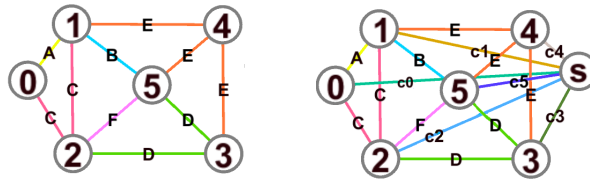


Figura 1: Grafo G e grafo estendido G'

Proposição 3.2. O problema FGkR em G é equivalente a encontrar um subgrafo gerador conexo $G'[L^* \cup C^*]$, com $L^* \subseteq L$ e $C^* \subseteq C$, tal que $|L^*| \leq k$ e $|C^*|$ seja mínimo.

Demonstração. Seja $L^* \subseteq L$ um subconjunto de rótulos. Sejam $U \subseteq V$ um conjunto com exatamente um vértice de cada componente conexa de $G[L^*]$ e $C^* = \{c_v : v \in U\}$. Temos que $G'[L^* \cup C^*]$ é um subgrafo conexo de G' e $|C^*| = \mathcal{W}(G[L^*])$. Em particular, se L^* induz uma solução ótima para FGkR em G , então $(L^* \cup C^*)$ induz uma solução viável para o novo problema em G' , e ambas as soluções têm o mesmo valor.

Reciprocamente, suponha que $G'[L^* \cup C^*]$ seja um subgrafo conexo. Então, o número de componentes de $G[L^*]$ é no máximo $|C^*|$. Em particular, se $(L^* \cup C^*)$ induz uma solução ótima para o novo problema em G' com valor w , então L^* induz uma solução viável para FGkR em G com valor menor ou igual a w .

Portanto, os dois problemas têm soluções ótimas correspondentes e de mesmo valor. $\square$

3.2. Cortes Coloridos e Dominância

Com respeito a cortes em grafos coloridos, [Silva et al., 2019] definem os cortes coloridos. Eles fornecem propriedades que fundamentam os modelos que serão definidos.

Definição 3.3. *Sejam $G = (V, E, L, \ell)$ um GCA e S um subconjunto próprio não vazio de V . Denomina-se $K_G(S) = \{\ell(uv) : u \in S, v \in V - S\}$ um corte colorido.*

Se grafo G está claro pelo contexto, é possível remover o sobrescrito G da notação $K_G(S)$. A Definição 3.3 conecta-se a uma propriedade muito importante neste trabalho.

Proposição 3.4. [Silva, 2018] *Sejam $G = (V, E, L, \ell)$ um GCA e $H = (V, E', \ell(E'), \ell)$ um subgrafo gerador de G . Então H é conexo se, e somente se, $\ell(E')$ contém pelo menos uma cor de todo corte colorido não-vazio de G .*

Outro conceito relevante, apresentado a seguir, é o de dominância, utilizado para descartar um subconjunto de cores que não podem melhorar uma solução parcial válida.

Definição 3.5. *Dados um subconjunto de rótulos não vazio $L' \subset L$ e um rótulo $c \in L \setminus L'$, c é dito dominado por L' , se $\mathcal{W}(G[L']) = \mathcal{W}(G[L' \cup \{c\}])$.*

4. Formulações

Apresentamos dois modelos de programação inteira para o problema FGkR, a serem referenciados como modelo por cortes coloridos e modelo por fluxo. Eles tomam por base as caracterizações do problema derivadas nas proposições 1.2 e 3.2. Os dois usam o grafo estendido G' . As variáveis de decisão utilizadas para definição dos modelos são:

- $x_{uv} \in \{0, 1\}$ recebe 1 se a aresta $uv \in E(G)$ é usada na solução e, caso contrário, 0.
- $z_\ell \in \{0, 1\}$ recebe 1 se a cor $\ell \in L \cup C$ está sendo utilizada e, caso contrário, 0.
- $f_{ij} \geq 0$ é quantidade de fluxo de $i \in V(G')$ para $j \in V(G')$.

4.1. Modelo por cortes coloridos (FGkR_CCut)

O modelo (FGkR_CCut) segue a formulação CCut proposta por [Silva et al., 2019] para o MLSTP. Essa formulação aplica a ideia de cortes coloridos (Definição 3.3). Para o FGkR, usamos sua equivalência dada na Proposição 3.2 e a caracterização de subgrafo conexo expressa na Proposição 3.4. Com isso, obtemos o seguinte modelo, expresso em termos do grafo estendido $G' = (V', E', L \cup C, \ell')$:

$$\min \sum_{\ell \in C} z_\ell \quad (2a)$$

$$\text{s.a.} \quad \sum_{\ell \in K(S)} z_\ell \geq 1 \quad \forall S \subset V', S \neq \emptyset, \quad (2b)$$

$$\sum_{\ell \in L} z_\ell \leq k, \quad (2c)$$

$$z_\ell \in \{0, 1\} \quad \forall \ell \in L \cup C. \quad (2d)$$

A função objetivo (2a) se encarrega de minimizar o número de cores de C usadas na solução, ou seja, o número de componentes definidas em G . A desigualdade (2b) assegura que pelo menos um rótulo de cada corte colorido seja escolhido, garantindo assim que a solução represente um subgrafo conexo de G' . A desigualdade (2c) é utilizada para atender o limitante de cores. Por último, em (2d), definem-se as variáveis z_ℓ como binárias.

4.2. Modelo por fluxo (FGkR_Fluxo)

Esse modelo é similar ao de cortes coloridos, com a diferença de que agora a conectividade é garantida enviando um fluxo a partir do vértice universal $s \in V(G')$ para todo outro vértice. Novamente, a Proposição 3.2 é base para a corretude do modelo.

$$\min \sum_{\ell \in C} z_{\ell} \quad (3a)$$

$$\text{s.a.} \quad f_{sj} + \sum_{i:(i,j) \in E} f_{ij} - \sum_{i:(i,j) \in E} f_{ji} = 1 \quad \forall j \in V, \quad (3b)$$

$$0 \leq f_{ij}, f_{ji} \leq M z_{\ell(i,j)} \quad \forall (i,j) \in E, \quad (3c)$$

$$0 \leq f_{sj} \leq M z_{\ell(s,j)} \quad \forall j \in V, \quad (3d)$$

$$\sum_{\ell \in L} z_{\ell} \leq k, \quad (3e)$$

$$z_{\ell} \in \{0, 1\} \quad \forall \ell \in L \cup C. \quad (3f)$$

A função objetivo (3a) e restrições (3e)-(3f) são idênticas às do modelo por cortes coloridos. As igualdades (3b) estabelecem um fluxo que inicia no vértice fonte s e é distribuído para todos os vértices de G' ; logo se um fluxo positivo chega a todo vértice, garante-se que existe um caminho de arestas entre s e todo vértice. Veja que há duas variáveis, f_{ij} e f_{ji} , para cada aresta (i, j) do grafo original G , que passa a ser representada por dois arcos antiparalelos (um indo de i para j e outro de j para i). Por outro lado, há apenas uma variável, f_{sj} , para cada aresta (s, j) acrescida quando da formação do grafo estendido G' . Note ainda que as restrições (3c)-(3d) garantem, através de um termo Big-M, que o fluxo só pode atravessar arestas cujos rótulos foram escolhidos. Durante os experimentos foi usado $M = |V|$.

5. Implementação Modelos

Nesta seção são descritas algumas características de como os modelos serão aplicados ao problema.

5.1. Modelos Iniciais

Os modelos iniciais passados para o *solver Cplex* não são exatamente aqueles descritos na seção anterior. Todas ou quase todas as restrições (2b), que são em número exponencial, serão omitidas do modelo (FGkR_CCut) inicial e incluídas a posteriori por um processo de geração de restrições, quando violadas. Por exemplo, [Silva, 2018] sugere utilizar apenas as restrições (2b) definidas por conjuntos unitários, isto é,

$$\sum_{\ell \in K(\{v\})} z_{\ell} \geq 1, \forall v \in V'. \quad (4)$$

Por outro lado, o modelo (FGkR_CCut) inicial pode conter a seguinte desigualdade sugerida por [Silva, 2018] para o MLSTP:

$$\sum_{\ell \in L \cup C} |E'(\{\ell\})| \cdot z_{\ell} \geq |V'| - 1. \quad (5)$$

Como o número de arestas em qualquer subgrafo gerador conexo de G' tem pelo menos $|V'| - 1$ arestas e cada rótulo $\ell \in L \cup C$ escolhido contribui com $|E'(\{\ell\})|$ dessas arestas, a desigualdade (5) é válida para uma solução do FGkR no grafo estendido. As restrições (4)-(5) também são válidas no modelo inicial (FGkR_Fluxo).

Nos testes aqui reportados, ambas as desigualdades (4) e (5) foram usadas nos modelos iniciais, pois possivelmente geram melhores relaxações lineares [Silva, 2018].

5.2. Lazy Constraints

As restrições (2b) são acrescentadas ao modelo por um procedimento *LazyConstraintCallback* do *Cplex*. Ele verifica se a solução inteira (0-1) obtida induz um subgrafo conexo do grafo estendido. Para isso, as variáveis z com valor 1 na solução são determinadas e são codificadas em um *bitmap*. A partir desse *bitmap*, usa-se uma interface que reduz o grafo base do modelo, mantendo somente as arestas cujas cores estão ativas (variáveis z com valor 1), ou seja, obtém-se um grafo reduzido (induzido pelas cores ativas). Por fim, aplica-se um DFS, que classifica os vértices por componente e retorna o número de componentes do grafo reduzido.

Em seguida, é verificado se o número de componentes é igual a 1, ou seja, se o grafo reduzido é conexo. Caso seja conexo, temos uma solução inteira válida para o (FGkR_CCut). Do contrário, o grafo reduzido não é conexo e alguma restrição de corte colorido está sendo violada. Cada componente $W_i, i = 0, 1, 2, \dots, p$, desse grafo define um corte colorido $K(W_i) = \{\ell(e) : e = (u, v) \in E', u \in W_i, v \notin W_i\}$ da seguinte forma:

$$\sum_{\ell \in K(W_i)} z_\ell \geq 1, \quad i = 0, 1, 2, \dots, p. \quad (6)$$

Adicionando esse conjunto de restrições, o problema relaxado seria fortalecido, invalidando a solução inteira passada para o *LazyConstraintCallback*. Com esse método, o problema de separação das desigualdades (2b) é resolvido. Após testes preliminares, indica-se que sejam adicionados ao modelo do *Cplex* como cortes locais, de modo a serem considerados apenas para o nó corrente e seus descendentes na árvore de *Branch & Cut*.

5.3. Planos de Cortes

Como comentado, as Equações (6) vão sendo iterativamente adicionadas ao modelo relaxado inicial de (FGkR_CCut) e, de uma certa forma, ajudam a selecionar quais cores são necessárias para uma solução conexa. Por exemplo, se um corte colorido é monocromático, conclui-se que esta cor faz parte da solução ótima, para esse modelo. Com essa intuição, um procedimento de planos de corte usando as Equações (6) como cortes é desenvolvido.

Para tal, usa-se o método *UserCutCallback* provido pelo *Cplex*. Diferentemente dos procedimentos do tipo *LazyConstraint* apresentados na subseção anterior, aplicados quando se encontra uma solução inteira, agora são usadas as soluções fracionárias das relaxações lineares dos nós do *Branch&Cut* processados. Dada uma tal solução, os valores das variáveis são “arredondados” para 1, caso maiores que zero, ou para 0, caso contrário. Cria-se assim uma solução inteira. Perceba, porém, que o grafo reduzido por ela gerado não necessariamente possui k rótulos. De qualquer modo, a partir dessa solução, aplica-se o mesmo procedimento que foi usado para o *LazyConstraintCallback* de (FGkR_CCut) com uma diferença: os cortes adicionados serão globais, valendo a partir daí durante todo o processo enumerativo.

Veja porém que existe a possibilidade de se identificar vários outros cortes coloridos violados, “agrupando” componentes desse grafo com a “ativação” de outras cores. Mais precisamente, se L_z é o conjunto de cores ativas na solução z , $\ell \in (L \cup C) \setminus L_z$ e $G'[L_z \cup \{\ell\}]$ é desconexo, pode-se gerar uma desigualdade violada da forma (6) para cada uma das p_ℓ componentes de $G'[L_z \cup \{\ell\}]$. Com isso, o número de desigualdades geradas passa a ser no máximo igual ao anterior, pois $p_\ell \leq G'[L_z]$, e o número de variáveis z nelas envolvidas torna-se estritamente menor, já que z_ℓ não aparece mais. Isto vai ao encontro do fato de que, quanto menor o número de cores que definem o lado esquerdo da desigualdade (6), mais forte ela se torna.

Nessa perspectiva, o seguinte procedimento de perturbação é proposto. Iterativamente, procura-se ativar mais cores no grafo reduzido de forma que ele ainda continue desconexo. Em

outras palavras, procura-se uma cor ainda não presente nesse grafo, tal que ele se mantenha desconexo, mesmo com a introdução de todas as arestas dessa cor. Esse procedimento pode ser realizado, por exemplo, por uma heurística gulosa baseada no k MVCA, onde cada passo tem como objetivo encontrar uma cor que, quando ativa, diminui o número de componentes do menor valor possível. Com essa estratégia gulosa, procura-se ativar o maior número de cores antes de o grafo se tornar conexo. Observe que, a cada iteração desse processo de perturbação, pode-se gerar um conjunto de desigualdades de cortes coloridos violadas, possivelmente diferente daquele da iteração anterior, mas com alguma interseção com este. O número de desigualdades geradas vai decrescendo a cada iteração e pode chegar a ser unitário, quando o grafo a ela associado tiver apenas 2 componentes.

A estratégia utilizada neste trabalho consiste em limitar o escopo do processo de perturbação ao *UserCutCallback*, em conjunto com um critério de parada adicional, limitando a k o número de cores do grafo original ativadas no grafo reduzido. Além disso, depois de testes, utiliza-se na geração de planos de corte somente o grafo final do processo de perturbação.

5.4. Regra de Branching

Dada a importância da seleção da variável fracionária a ser ramificada no método *Branch & Cut*, uma sequência de prioridade das variáveis pode ser informada ao *Cplex*. As diferentes formas de organizar essa sequência afetam o desempenho do método. No sentido de se aproximar das características do Algoritmo Enumerativo, sugerir-se-ia, a princípio, dar preferência à ramificação em variáveis que representem cores com maior número de arestas.

Todavia, com o intuito de melhorar o processo de ramificação, como alternativa a escolher uma ordem de prioridades inicial fixa, tenta-se utilizar informações da solução atual para gerar uma ordem mais dinâmica. Por exemplo, veja que, quando uma solução parcial está a um nível de profundidade de uma solução com k cores, ou seja, falta somente uma cor a ser incluída nessa para completar o máximo possível de cores, esse subproblema resume-se a fazer uma busca linear pela cor não usada (não dominada) que, ao ser adicionada à solução parcial, induz o menor número de componentes. Estendendo esse raciocínio, o seguinte critério de ramificação é definido:

1. Construir a solução parcial viável $\bar{z} \in \{0, 1\}^{|L|}$ (variáveis de cor do grafo original), baseada no nó atual, fixando em 1 as variáveis que são limitadas inferiormente por 1 ($z_i \geq 1$) no processo de ramificação. Logo, as demais variáveis são fixadas em 0.
2. Encontrar $I \subseteq L$ o conjunto de variáveis de cor do grafo original G as quais possuem valores de relaxação linear fracionários no nó atual e determinar $\bar{L} = \{c \in L : \bar{z}_c = 1\}$.
3. Aplicar ramificação ($z_i \geq 1$ ou $z_i \leq 0$) na cor $i \in I$ tal que $\mathcal{W}(G[\bar{L} \cup \{i\}]) \leq \mathcal{W}(G[\bar{L} \cup \{j\}])$, $\forall j \in I$, ou equivalentemente, $i \in \arg \max_{j \in I} (\mathcal{W}(G[\bar{L}]) - \mathcal{W}(G[\bar{L} \cup \{j\}]))$.
4. Caso não possa fixar mais cores em 1 (por causa de k) ou $I = \emptyset$, deixar o *Cplex* escolher a variável de cor adicional em C a ser ramificada.

Vale observar a estreita relação entre os cálculos necessários para aplicação do item 3 acima e de outros ingredientes já discutidos anteriormente. Por exemplo, o procedimento para fazer poda por limite inferior, derivado da Proposição 2.1, já calcula a diferença $(\mathcal{W}(G[\bar{L}]) - \mathcal{W}(G[\bar{L} \cup \{j\}]))$, que determina a redução no número de componentes de uma solução parcial $\bar{L} \subseteq L$ com a inclusão de uma cor $j \in L \setminus \bar{L}$. Além disso, quando $\mathcal{W}(G[\bar{L}]) - \mathcal{W}(G[\bar{L} \cup \{j\}]) = 0$, a cor j é dominada por $\bar{L}$ (Definição 3.5) e pode ser descartada. Dessa forma, a partir desses cálculos comuns, adiciona-se um *BranchCallback*, usando o método de *prune*, para aplicar poda por limite inferior, e um *makeBranch*, para fixar cores dominadas pela solução parcial atual em zero, ao mesmo tempo em que aplica ramificação na variável que induz o menor número de componentes.

6. Experimentos

Nesta seção são vistos os parâmetros utilizados para geração de instâncias. Em seguida, são explorados como os métodos de resolução são aplicados.

6.1. Ambiente Computacional dos testes

Os testes foram executados usando máquina com processador i7-3770 3.40Ghz no modo *Turbo Boost*, 8GB de memória RAM DDR3 e disco rígido de 1TB. As implementações foram feitas em C++ usando Cplex 12.8 em modo oportunístico com 8 *threads*, as bibliotecas providas pela *Boost C++ Libraries* e a biblioteca *Threading Building Blocks* (TBB) para paralelizar o Algoritmo Enumerativo. Vale ressaltar que, por padrão, o *Cplex* desativa multiprocessamento quando é usado *Control Callbacks* (*LazyConstraintCallback* e *UserCutCallback*), porém esse pode ser forçado quando sua implementação é segura em *threads*.

Usa-se como limite superior (*UpperCutoff*) o valor objetivo provido pela solução heurística do *k*MVCA e como limite inferior inicial o valor trivial, ou seja, referente a um grafo conexo (uma componente). O *Cplex* é executado com maior ênfase para busca por soluções inteiras, e o método de *presolve* é desligado em todos os modelos. O tempo limite para resolução de cada instância foi estipulado em 20 minutos (1200s) para cada método.

6.2. Geração de instâncias

Os testes são baseados em [Cerulli et al., 2014], onde se sugere o uso de instâncias obtidas da seguinte maneira. Cada instância, um grafo $G = (V, E, L, \ell)$ rotulado em arestas e um inteiro k , é gerada de forma pseudo-aleatória, usando o gerador *Mersenne twister* (na biblioteca *boost* como *mt19937*) e aplicando uma distribuição uniforme para geração de arestas e rótulos, garantindo que há pelo menos uma aresta com cada rótulo. O grafo é gerado com os seguintes parâmetros: $n(G) \in \{100, 200, 300, 400, 500, 1000\}$, $m(G) = d(G) \frac{n(G) \cdot (n(G)-1)}{2}$, $d(G) = \{0.2, 0.4, 0.6\}$, $|L| \in \{\lfloor n(G)/4 \rfloor, n(G)/2, n, \lfloor 1.25 \cdot n(G) \rfloor\}$ e um identificador de instância $id \in \{1, 2, \dots, 10\}$. [Cerulli et al., 2014] propõem encontrar o k de forma empírica, para que o problema não seja tão fácil de resolver. Para tanto, usa-se $k = \frac{n(G)}{2^i}$, onde i é o menor valor para o qual o *k*MVCA não reporta uma solução trivial (com uma única componente). A instância assim obtida é identificada por $n(G)-m(G)-|L|-k-id$. Dessa forma, é construído um banco de testes (600 instâncias) com grafos com densidade pelo menos 0.2, diferente de [Cerulli et al., 2014] em que ela é fixa em 0.2.

Desse banco, 43 instâncias são escolhidas, as que possuem parâmetros de entrada mais interessantes (por exemplo, $k \geq 2$). Vale destacar que esse grupo selecionado mantém a diversidade dos parâmetros usados na geração.

6.3. Paralelizando o Algoritmo Enumerativo

A paralelização do Algoritmo Enumerativo foi feita usando C++ com a biblioteca TBB da Intel, onde o nó inicial é alocado em uma lista inicial de tarefas e passado para o método genérico da biblioteca *parallel.do*. Esse método irá processar cada nó da lista, alocando apropriadamente recursos e sincronizando *threads*.

Cada nó é processado por um método que funciona de forma similar ao procedimento de *backtracking*, porém, para estabelecer sincronização entre os nós, a melhor solução encontrada é mantida em uma variável global atômica, acessada por um método não-bloqueante *compare_and_swap*, que verifica de forma atômica se o valor da variável é consistente, antes de mudar seu valor; caso contrário, o fluxo lógico é reiniciado, na forma de um laço, para um valor atualizado. O processo de ramificação é estabelecido pela criação de dois nós, que alimentam o método *parallel.feeder*, responsável por alocar os nós na fila de tarefas. O algoritmo termina quando todos os elementos da fila de tarefas são processados.

6.4. Modo oportunístico

Uma das estratégias para melhorar os tempos de computação do *Cplex* é ativar o modo oportunístico. Diferentemente do modo determinístico padrão, no modo oportunístico a sincronização é relaxada para que haja maior aproveitamento do paralelismo. Vale ressaltar que testes foram feitos em modo determinístico, porém o desempenho médio no modo oportunístico foi melhor em geral em todos os testes feitos.

6.5. Configurações dos métodos

Nesta subseção são descritos os modelos usados, as configurações do *Branch & Cut* e do *Cplex*, definindo as seguintes abordagens de solução, identificadas (em parênteses) conforme apresentação nas tabelas de resultados:

- Algoritmo paralelo de *backtracking* (Paralelo): Este usa o mesmo processo algorítmico da Subseção 6.3, com alguns ajustes de programação e com um melhor limite superior, fornecido pelo *kMVCA*.
- Modelo de fluxo usando Benders (*Benders*): Baseado na formulação da Subseção 4.2 para o grafo estendido, adicionando as desigualdades da Subseção 5.1 ao problema mestre. Usa a Decomposição de Benders automatizada do *Cplex* e prioridade de ramificação para variáveis de cores com maior número de arestas (essa estratégia sem essa priorização piorou o desempenho médio nos testes).
- Modelo baseado em cortes coloridos (*Ccut*): Abordagem baseada em (*FGkR_Ccut*), com modelo inicial seguindo Subseção 5.1, em conjunto com os procedimentos propostos nas Subseções 5.2 e 5.3 e usando o processo de ramificação dinâmico proposto na Subseção 5.3.

7. Resultados

Nesta seção são avaliados os desempenhos dos três métodos de solução para o FGkR, a partir de parâmetros como tempo de computação e número de instâncias resolvidas. Além disso, é possível analisar a distância relativa entre os valores da relaxação linear de cada modelo e das soluções viáveis por ele obtidas através do *GAP* percentual, calculado da seguinte forma:

$$GAP = 100 \frac{(OBJ - LI)}{OBJ}, \quad (7)$$

onde *LI* e *OBJ* são, respectivamente, o limite inferior e o melhor valor objetivo de solução inteira encontrados pela formulação dentro do tempo limite de 1200s.

Na Tabela 1 são apresentados os valores médios dos seguintes parâmetros: tempo de computação em segundos (Tempo(s)), melhor valor de solução viável (*OBJ*), *gap* percentual (*GAP*) e número de nós explorados na árvore de *Branch & Cut* (#Nós). Esses dois últimos valores, que não se aplicam ao método paralelo, são substituídos pela profundidade média (*d*) em que o algoritmo enumerativo encontra a melhor solução (*d*=0 quando a melhor solução é a encontrada pelo *kMVCA*). Também é apresentado o número de instâncias resolvidas à otimalidade (#Ótimos).

Em geral, *Ccut* produziu os melhores resultados em todos os parâmetros analisados. Com exceção de um, todos os limites superiores (*OBJ*) encontrados pelo *Ccut* são menores ou iguais aos encontradas pelos outros dois métodos. A única exceção ocorre na instância onde o algoritmo paralelo é melhor que ambos *Benders* e *Ccut*, nos quais os limites inferiores é a solução trivial conexa. Vale destacar que, das 43 instâncias, o algoritmo de *backtracking*, o modelo de fluxo usando Benders e o *Ccut* encontram soluções ótimas para 29, 25 e 31 delas, respectivamente. Não

somente o *Ccut* encontrou mais soluções ótimas, como também o fez no menor tempo médio e explorando menos nós que o *Benders* (Tabela 1).

Tanto em testes anteriores quanto nesses, pode-se perceber que, quando os modelos, usando a mesma prioridade de ramificação das variáveis de cores, não conseguem gerar limites inferiores bons (pelo menos maiores que 1), o paralelismo do *backtracking* é mais propício a gerar bons resultados.

	<i>Ccut</i>	<i>Benders</i>	Paralelo
Tempo (s)	377.33	552.64	501.79
#Ótimos	31	25	29
OBJ	19.14	19.74	19.72
GAPs	22%	32%	-
#Nós	81618	270398	-
d	-	-	28

Tabela 1: Comparativo Final

Vale mencionar que a decomposição de Benders incrementa o desempenho do modelo por fluxo. Entretanto, ela não conseguiu ser suficiente para torná-lo superior ao modelo por cortes coloridos. Talvez por motivos de limitações impostas pela decomposição automatizada pelo *Cplex* ou por, principalmente, ao longo deste trabalho, não terem sido encontradas desigualdades válidas relacionando variáveis de cores e de fluxo “boas” o suficiente para gerar planos de cortes profundos. Mesmo assim, foi observado que, em comparação ao enumerativo, o tempo médio para o Benders (552s) é bem próximo, com uma diferença de aproximadamente 51s. Posto que Benders resolve otimamente 4 instâncias a menos, nas quais atinge o tempo limite, essa pequena diferença nos tempos médios mostra que a decomposição consegue um desempenho bem próximo nas instâncias em que ambos encontram o ótimo.

Finalmente, deve-se observar que o desempenho superior do *Ccut* deve-se fortemente à combinação entre o processo de *branching* através de uma escolha dinâmica e a geração dos planos de cortes usando perturbação limitada, que levam a *GAP's* menores.

8. Conclusões

Neste trabalho apresentamos duas formulações matemáticas para o FGkR, fundamentadas em uma caracterização derivada para o problema, relacionando-o ao MLSTP. A partir dessa conexão, puderam-se empregar, diretamente ou com ligeiras adaptações, vários resultados da literatura do MLSTP. Em particular, é usado o conceito de cortes coloridos, onde se representa uma solução pelas cores escolhidas e não diretamente pelas arestas selecionadas. Isso leva à necessidade de menos variáveis na definição dos modelos matemáticos.

Objetivando a resolução eficiente das formulações pelo método *Branch & Cut*, foram consideradas e avaliadas diferentes estratégias, como uso de *lazy constraints*, adição de desigualdades válidas, interferência nas regras de ramificação e testes de poda, identificação de cores descartáveis ao longo do processo de busca ou priorização de escolha daquelas cores com maior evidência de participarem de uma solução ótima.

Combinando as abordagens sugeridas para esses vários elementos de um *Branch & Cut*, foram apresentados três métodos de resolução exata para o problema, sobressaindo-se o método *Ccut*, que apresentou em média os melhores resultados. Dentre os ingredientes do método, destaca-se a geração de limites inferiores melhores que o trivial, uma das principais dificuldades para a

resolução do problema, observadas tanto nos métodos aqui apresentados, como em várias outros testes adicionais realizados. A propósito, quando não é possível gerar bons limites inferiores, o algoritmo enumerativo, em sua implementação paralela, pode ser o mais indicado, na disponibilidade de ambiente computacional com melhores recursos para paralelismo.

Referências

- Cerulli, R., Fink, A., Gentili, M., e Raiconi, A. (2014). The k-labeled spanning forest problem. *Procedia - Social and Behavioral Sciences*, 108:153 – 163. ISSN 1877-0428.
- Chang, R.-S. e Shing-Jiuan, L. (1997). The minimum labeling spanning trees. *Information Processing Letters*, 63(5):277 – 282. ISSN 0020-0190.
- Consoli, S., Pérez, J. M., e Mladenović, N. (2013). Intelligent variable neighbourhood search for the minimum labelling spanning tree problem. *Electronic Notes in Discrete Mathematics*, 41:399 – 406. ISSN 1571-0653.
- Consoli, S. e Moreno Pérez, J. A. (2015). Variable neighbourhood search for the k-labelled spanning forest problem. *Electronic Notes in Discrete Mathematics*, 47:29 – 36. ISSN 1571-0653. The 3rd International Conference on Variable Neighborhood Search (VNS'14).
- Consoli, S., Moreno Pérez, J. A., e Mladenović, N. (2017). Comparison of metaheuristics for the k-labeled spanning forest problem. *International Transactions in Operational Research*, 24(3): 559–582.
- Granata, D., Cerulli, R., Scutellà, M. G., e Raiconi, A. (2013). Maximum flow problems and an NP-complete variant on edge-labeled graphs. In Pardalos, P. M., Du, D.-Z., e Graham, R. L., editors, *Handbook of Combinatorial Optimization*, p. 1913–1948. Springer New York.
- Krumke, S. O. e Wirth, H.-C. (1998). On the minimum label spanning tree problem. *Information Processing Letters*, 66(2):81 – 85. ISSN 0020-0190.
- Silva, T. G. d. (2018). *The Minimum Labeling Spanning Tree and Related Problems*. PhD thesis, Universidade Federal Fluminense, Université d'Avignon et des Pays de Vaucluse, Nitéroï.
- Silva, T. G. d., Gueye, S., Michelon, P., Ochi, L. S., e Cabral, L. d. A. F. (2019). A polyhedral approach to the generalized minimum labeling spanning tree problem. *EURO Journal on Computational Optimization*, 7(1):47–77. ISSN 2192-4414.
- Voßs, S., Fink, A., e Duin, C. (2005). Looking ahead with the pilot method. *Annals of Operations Research*, 136(1):285–302. ISSN 1572-9338.