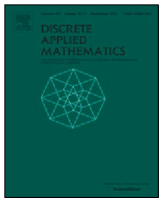
Matemática Aplicada Discreta 295 (2021) 85-93

Listas de conteúdo disponíveis no ScienceDirect

Matemática Aplicada Discreta

Página inicial da revista: www.elsevier.com/locate/dam



Um algoritmo simples e uma fórmula min-max para o problema inverso de arborescência☆

András Frank*, Gergely Hajdu

Grupo de Pesquisa MTA-ELTE Egerváry, Departamento de Pesquisa Operacional, Universidade Eötvös Loránd, Pázmány P. s. 1/c, Budapeste, H-1117, Hungria

artigo info

Histórico do artigo:
Recebido em 5 de outubro de 2020
Recebido em formato revisado em 10 de fevereiro de 2021 Aceito em 12 de fevereiro de 2021
Disponível on-line em 2 de março de 2021

Palavras-chave:
Otimização combinatória inversa
Arborescência
Teorema do mínimo-máximo Algoritmo simples

abstract

Em 1998, Hu e Liu desenvolveram um algoritmo fortemente polinomial para resolver o problema da arborescência inversa que visa modificar minimamente uma determinada função de custo no conjunto de arestas de um digrafo D, de modo que uma arborescência de entrada de D se torne a mais barata. Nesta nota, desenvolvemos um algoritmo conceitualmente mais simples, juntamente com uma nova fórmula mínima e máxima para a modificação mínima da função de custo. A abordagem se baseia em uma ligação com um teorema de mín-máx e um algoritmo simples (guloso de duas fases) do primeiro autor, de 1979, referente ao problema de otimização primário de encontrar um subgrafo mais barato de um digrafo que cubra uma família de interseção, juntamente com o problema de otimização dupla correspondente.

1. Introdução

Uma **arborescência** é uma árvore direcionada na qual o grau de entrada de todos os nós, exceto um, é 1. O nó excepcional é chamado de **raiz** e seu grau de entrada é 0. Seja $D = (V, A)$ um digrafo sem loop com n nós e m arestas, e seja $r_0 \in V$ um nó específico chamado de nó-raiz de D. Por r_0 **-arborescência** F de D, entendemos um subgrafo de D que é uma arborescência de raiz r_0 contendo todos os nós de D (ou seja, F é uma arborescência de D com raiz r_0). Sem perda de generalidade, assumiremos que nenhuma aresta de D entra em r_0 . Em 1965, Chu e Liu [5] desenvolveram um algoritmo simples e fortemente polinomial

para calcular uma arborização mais barata (= custo mínimo) r_0 de D com relação a uma determinada função de custo em A. Em 1967,
Edmonds [7] descreveu um algoritmo e um teorema de mínimo-máximo para o problema intimamente relacionado ao peso máximo
ramificações.

No **problema de arborescência inversa**, recebemos uma arborescência de extensão F_0 de D com raiz r_0 (ou seja, uma r_0 -arborescência) e uma função de custo $w_0 : A \to \mathbf{R}_+$. O objetivo é modificar w_0 para que F_0 se torne uma arborescência r_0 mais barata em relação à função de custo revisada w, e o desvio de w em relação a w_0 seja o menor possível. O **desvio** $\sum |w - w_0|$
de w (de w_0) é definido por $(|w(a) - w_0(a)| : a \in A)$, e usaremos ao longo do artigo essa norma ℓ_1 para medir a otimização de w.

Em 1998, Hu e Liu [16] descreveram um algoritmo fortemente polinomial para esse problema inverso. Tanto o algoritmo quanto a prova de sua correção eram bastante complexos. O objetivo do presente trabalho é desenvolver um algoritmo conceitualmente mais simples.

que se baseia em uma nova fórmula min-max (Teorema 5.2) para o desvio mínimo μ^* do custo revisado-
função w para a qual a entrada r_0 -arborescência é a mais barata de D. A abordagem é baseada em um link para um artigo

do

☆A pesquisa foi parcialmente apoiada pelo Fundo Nacional de Pesquisa, Desenvolvimento e Inovação da Hungria (FK-18) - No. NKFI-128673.

*Autor correspondente.
Endereços eletrônicos: frank@cs.elte.hu (A. Frank), hgergely91@gmail.com (G. Hajdu).

https://doi.org/10.1016/j.dam.2021.02.027
0166-218X/

primeiro autor [10], de 1979, que inclui um algoritmo particularmente simples para resolver uma extensão natural da solução mais barata de
r_0 -problema de arborescência. Não apenas o algoritmo é simples, mas também a prova de sua correção.

Cai e Li [3,4] mostraram como o problema de interseção de matroides inversas pode ser reduzido a um problema de circulação de custo mínimo e, portanto, um algoritmo puramente combinatório e fortemente polinomial para circulação (por exemplo, o primeiro de Tardos [18]) pode ser aplicado. Como r_0 -arborescências formam as bases comuns de dois matroides especiais, a solução de Cai e Li pode ser especializada para r_0 -arborescências. Nosso principal objetivo é apresentar uma abordagem que dê origem a um algoritmo mais simples e mais eficiente, bem como a uma fórmula de mínimo e máximo. A situação é análoga àquela em que o algoritmo geral de interseção de matroides ponderadas de Edmonds [9] não tornou supérfluo o algoritmo direto mais simples e mais eficiente de Chu e Liu [5] referente às r_0 -arborescências mais baratas. Entretanto, deve-se observar que a abordagem de fluxo de custo mínimo de
Cai e Li fornecem uma solução para o problema significativamente mais geral quando há uma restrição de limite superior $g(a)$ em $|w(a) - w_0(a)|$ para cada borda a, e o objetivo é minimizar $\sum[c(a)|w(a) - w_0(a)| : a \in A]$ onde $c : A \to \mathbf{R}_+$ é uma determinada função de custo.

Há uma rica história de problemas de otimização combinatória inversa. Um dos primeiros trabalhos é de Burton e Toint [2] e foi publicado em 1992. Ele trata de problemas inversos de caminhos mais curtos e foi fortemente motivado por aplicações práticas (provenientes das ciências geofísicas). Desde então, surgiu um grande número de artigos que foram maravilhosamente analisados em dois excelentes artigos de pesquisa de Heuberger [15] e Demange e Monnot [6].

Em um trabalho recente, Frank e Murota [13] desenvolveram uma fórmula geral min-max para uma ampla classe de problemas de otimização combinatória inversa. Por exemplo, sua estrutura abrange a generalização do presente problema para o caso em que os limites superiores são impostos à alteração de w_0 e/ou a norma ℓ_1 - ou ℓ_2 - ponderada ou algumas outras normas são usadas. No entanto, deve-se enfatizar que as provas em [13] não são algorítmicas, e o desenvolvimento de algoritmos polinomiais continua sendo um grande desafio para investigações futuras.

1.1. Terminologia e notação

Para uma aresta (ou arco) direcionada $a = uv$, v é chamada de **cabeça** de a, enquanto u é sua **cauda**. Dizemos que uv **entra** (**deixa**) o subconjunto Z de nós se $v \in Z$ e $u \notin Z$ ($v \notin Z$ e $u \in Z$). Em um digrafo $D = (V, A)$, o número de arestas que entram em Z é denotado por $\varrho_D(Z) = \varrho_A(Z)$, enquanto o número de arestas que saem de Z é denotado por $\delta_D(Z) = \delta_A(Z)$. Diz-se que um subconjunto L de bordas **entra em** Z se L contiver uma borda que entra em Z , ou seja, se $\varrho_L(Z) \geq 1$. Para uma família $\mathcal{F}$ de subconjuntos, dizemos que L **cobre** $\mathcal{F}$ se L entra em cada membro de $\mathcal{F}$. Para dois elementos s e t, um conjunto Z é chamado de *conjunto* **ts** se $t \in Z$ e $s \notin Z$.

Um digrafo D é chamado de **conectado à raiz** com relação a um nó-raiz r_0 se $\varrho_D(Z) \geq 1$ for válido para todo subconjunto não vazio
$Z \subseteq V - r_0$. Claramente, a conectividade da raiz é equivalente a exigir que cada nó de D seja acessível a partir de r_0 (ao longo de um dipath),
ou que D inclui uma r_0 -arborescência. Uma propriedade fácil e bem conhecida é que um subgráfico de D conectado à raiz mínima por inclusão é uma r_0 -arborescência. Em termos mais gerais, D é *conectado por k* **arestas com raiz** se $\varrho_D(Z) \geq k$ for válido para todo subconjunto não vazio $Z \subseteq V - r_0$. Também é uma propriedade bem conhecida que, em um digrafo conectado *por* k *arestas com* raiz mínima e inclusão
o grau de entrada de cada nó $v \in V - r_0$ é k.

Uma função $x : S \to \mathbf{R}$ em S pode ser estendida a uma função de conjunto $\tilde{x}$ por $\tilde{x}(Z) := \sum[x(s) : s \in Z]$ $(Z \subseteq S)$. Analogamente, para uma função de conjunto y em S e para uma família $\mathcal{F}$ de subconjuntos de S, usamos a notação $\tilde{y}(\mathcal{F}) := \sum[y(Z) : Z \in \mathcal{F}]$.

Dois conjuntos X e Y são chamados de **interseção** se $X \cap Y \neq \emptyset$. Se, além disso, $X - Y$ e $Y - X$ não forem vazios, então X e Y são **devidamente interceptados**. Uma família $\mathcal{F}$ de conjuntos é **laminar** se não tiver dois membros que se cruzem adequadamente. $\mathcal{F}$ é **interseccional** se tanto $X \cap Y$ quanto $X \cup Y$ pertencerem a $\mathcal{F}$ sempre que X e Y forem membros interseccionais de $\mathcal{F}$. Dado um digrafo $D = (V, A)$, dizemos que uma família interseccional $\mathcal{F}$ de subconjuntos distintos de V é um **sistema de kernel** [10] se $\varrho_D(Z) > 0$ para cada $Z \in \mathcal{F}$. Todos os outros
As noções e notações podem ser encontradas em [12].

2. Arborescências e sistemas de núcleo

2.1. Mais barato r_0 -arborescências

Seja $D = (V, A)$ um digrafo conectado à raiz com um nó-raiz r_0 e seja $c : A \to \mathbf{R}_+$ uma função de custo não negativa no conjunto de bordas. O problema primário consiste em determinar uma arborescência r_0 mais barata. Dizemos que uma função $y : \mathcal{F} \to \mathbf{R}$ definida em um sistema de conjuntos $\mathcal{F} \subseteq 2^V$ é **c-fácil** se $y \geq 0$ e

$$\sum[y(Z) : Z \in \mathcal{F}, Z \text{ é inserido por } a] \leq c(a) \text{ para toda aresta } a \in A. \tag{2.1}$$

Quando $\mathcal{F} := \{X : \emptyset \neq X \subseteq V - r_0\}$, uma função c-fácil y será chamada de **solução dupla** para o problema de arborescência mais barata r_0 -. Chamamos uma aresta a de D de **c-tight** (ou apenas **tight**) (com relação a y) se $\sum[y(Z) : Z \in \mathcal{F}, Z \text{ é inserido por } a] = c(a)$. Bock [1] e Fulkerson [14] provaram a seguinte fórmula min-max.

Teorema 2.1 (*Bock, Fulkerson*)**.** *Seja c uma função de custo não negativa no conjunto de bordas de um digrafo conectado à raiz $D = (V, A)$.*
O custo mínimo de uma r_0 -arborescência de D é igual a

$$\max\{\sum [y(Z) : Z \subseteq V - r_0] : y\ c\text{ - feasible}\}. \tag{2.2}$$

Existe uma solução dual ótima y para a qual $\{Z : y(Z) > 0\}$ *é laminar. Se c tiver valor inteiro, a solução ótima y também pode ser escolhida com valor inteiro.* ■

Observe que Fulkerson [14] desenvolveu um algoritmo simples para calcular o vetor dual ideal y que ocorre em o teorema. O teorema implica imediatamente os seguintes critérios de otimização.

Corolário 2.2. *Seja y^* uma função c-fácil na família de subconjuntos não vazios de $V - r_0$ e seja F_0 uma r_0 -arborescência de D para a qual os seguintes critérios de otimização são válidos:*

F_0 consiste em bordas apertadas,

$y^*(Z) > 0$ implica que $\varrho_{F_0}(Z) = 1$.

Então, F_0 é uma arborescência de custo mínimo r_0 de D para a qual $\tilde{c}(F_0) = \sum [y^*(Z) : Z \subseteq V - r_0]$. ■

O livro de Schrijver [17] oferece um rico histórico e uma bibliografia de algoritmos e resultados mínimos e máximos sobre arborescências e objetos relacionados, como ramificações.

2.2. Teorema conhecido de min-max para sistemas de kernel

Em 1979, Frank [10] estendeu o problema das r_0 -arborescências mais baratas para os sistemas de kernel, quando se está interessado em encontrar um subconjunto L de bordas mais barato que entre em todos os membros de uma família de subconjuntos $\mathcal{F}$ de interseção (resumidamente, L cobre $\mathcal{F}$). Um caso especial é quando $\mathcal{F}$ consiste em todos os subconjuntos não vazios de $V - r_0$. Nesse caso, o mínimo de inclusão subconjuntos de bordas que cobrem $\mathcal{F}$ são exatamente as r_0 -arborescências.

Outro caso especial é quando $\mathcal{F}$ consiste em todos os *conjuntos ts*: nesse caso, os conjuntos de arestas mínimas de inclusão que cobrem $\mathcal{F}$ são precisamente os *st-dipaths*. Precisaremos de um terceiro caso especial em que $\mathcal{F}$ consiste nos subconjuntos Z de $V - r_0$ que são inseridos exatamente uma vez por uma r_0 *-arborescência* F_0 especificada (ou seja, $\varrho_{F_0}(Z) = 1$).

O problema primário consiste em encontrar um subconjunto de bordas mais barato que cubra $\mathcal{F}$. O problema duplo consiste em encontrar um função c-fácil $y : \mathcal{F} \to \mathbf{R}$ para a qual

$$y(\mathcal{F}) = \sum [y(Z) : Z \in \mathcal{F}]$$

é tão grande quanto possível. Em [10], o algoritmo de Fulkerson [14] foi estendido de forma natural para sistemas de kernel gerais. Além disso, [10] descreveu uma segunda fase do algoritmo que calcula, de forma gulosa, o subconjunto mais barato L de bordas que cobrem $\mathcal{F}$.

Esse algoritmo guloso de duas fases provou a seguinte extensão do Teorema 2.1.

Teorema 2.3 (*[10]*)**.** *Seja $\mathcal{F}$ um sistema de conjuntos de interseção no conjunto de nós do digrafo $D = (V, A)$ para o qual A cobre $\mathcal{F}$, e seja $c : A \to \mathbf{R}_+$ uma função de custo. Então*

$$\min\{\tilde{c}(L) : L \subseteq A, L \text{ cobre } \mathcal{F}\} = \max\{y(\mathcal{F}) : y \ c\text{- factível}\}. \tag{2.3}$$

Se c tiver valor inteiro, a solução dual ótima y também pode ser escolhida com valor inteiro. Além disso, existe uma solução ótima y para a qual o sistema de conjuntos $\{Z : y(Z) > 0\}$ é laminar. ■

A direção trivial max ≤ min do teorema implica que, para uma solução dupla y e para um subconjunto L de bordas que cobrem $\mathcal{F}$ se eles atenderem aos seguintes **critérios de otimização**:

toda borda em L é estreita (com relação a y),

$y(Z) > 0$ implica que $\varrho_L(Z) = 1$,

então y é uma solução dual ótima e L é uma cobertura c-cheapest de $\mathcal{F}$. A direção não trivial max ≥ min do teorema é equivalente a afirmar que existe uma solução dual y^* e uma cobertura $L^* \subseteq A$ de $\mathcal{F}$ que atende aos critérios de otimização.

No caso especial de $\mathcal{F} := \{Z : \emptyset \neq Z \subseteq V - r_0\}$, os subconjuntos de bordas que cobrem $\mathcal{F}$ são exatamente os conjuntos de bordas de subgrafos de D conectados à raiz. Como os membros mínimos dos subgrafos conectados à raiz são as r_0 -arborescências de D, isso Nesse caso, as coberturas mais econômicas de $\mathcal{F}$ são as r_0 -arborescências mais econômicas e, portanto, o Teorema 2.3 geral se reduz ao Teorema 2.1.

Observamos que [10] na verdade inclui um teorema min-max (referente a uma cobertura mais barata de uma função supermodular de interseção) que é significativamente mais geral do que o Teorema 2.3. Aqui não precisamos desse resultado geral e apenas observamos que sua prova em [10] não é construtiva, ao contrário da prova algorítmica simples do Teorema 2.3.

Também observamos que o teorema do mínimo-máximo e o algoritmo guloso de duas fases em [10] foram generalizados em [11] para o problema de encontrar a cobertura mais barata de um sistema de bi-conjunto de interseção. Consulte também o livro de Frank [12], página 395.

3. Algoritmo para sistemas de kernel

Em [10], o Teorema 2.3 foi comprovado com a ajuda de um algoritmo guloso de duas fases. A primeira fase para calcular uma solução dual ótima c-fácil $y^\star$ é uma extensão direta do algoritmo de Fulkerson [14] para o dual da solução mais barata r_0 -problema de arborescência. Como precisamos dessa primeira fase para a solução sugerida para o problema de arborescência inversa, nós a descrevemos com todos os detalhes. Descreveremos também a segunda fase, que calcula, também de forma gulosa, uma cobertura mais barata L de F. Esse algoritmo será necessário para calcular a estrutura dupla ideal que aparece na fórmula min-max do Teorema 5.2 referente ao desvio mínimo da função de custo desejada no problema de arborescência inversa.

3.1. Algoritmo conhecido para sistemas de kernel gerais

Começamos delineando o algoritmo guloso de duas fases descrito em [10]. Dada uma função de custo $c' : A \to \mathbf{R}_+$, chamamos um subconjunto $Z \subseteq V$ c'**-positivo** se o custo c' de cada aresta que entra em Z for positivo. Seja $\mathcal{F}$ uma família de subconjuntos de V com interseção para a qual $\varrho_A(Z) \geq 1$ se mantém para todo $Z \in \mathcal{F}$.

Primeira fase Se $\mathcal{F}$ não tiver nenhum membro c *positivo*, então $y^* \equiv 0$ é uma solução dual ideal (e o conjunto de bordas com zero
c-cost é uma cobertura mais barata de $\mathcal{F}$), caso em que o algoritmo é encerrado. Portanto, supomos que $\mathcal{F}$ admite uma cobertura c-positiva
membro.

O algoritmo determina, um a um, os membros $c_1 := c, c_2, c_3, \ldots$ de uma sequência de funções de custo de valor real não negativo em A e uma sequência $Z_1, Z_2, Z_3, \ldots$ de membros de $\mathcal{F}$ juntamente com variáveis duais positivas $y^*(Z_i)$ atribuídas a esses
conjuntos Z_i, que são de valor inteiro quando c é de valor inteiro.

Na Etapa $i = 1$, deixe Z_1 ser um membro mínimo c_1 -positivo de $\mathcal{F}$. Deixe

$$y^*(Z_1) := min\{c_1(f) : f \in A, f \text{ entra em } Z\}_1$$

e definir c_2, como segue.

$$c_2(f) := \begin{cases} c_1(f) & \text{se } f \text{ não entrar em } Z_1 \\ c_1(f) - y^*(Z)_1 & \text{se } f \textit{ entrar} \text{ em } Z_1. \end{cases} \tag{3.1}$$

Para o caso geral $i \geq 2$, suponha que c_i já tenha sido computado. Se cada membro de $\mathcal{F}$ for inserido por uma borda de custo 0 c_i, a primeira fase será encerrada. Se esse não for o caso, então deixe Z_i ser um membro mínimo c_i positivo de inclusão
de $\mathcal{F}$. Seja

$$y^*(Z_i) := min\{c_i(f) : f \in A, f \text{ entra em } Z\}_i$$

e definir c_{i+1}, como segue.

$$c_{i+1}(f) := \begin{cases} c_i(f) & \text{se } f \text{ não entrar em } Z_i\, c_i \\ (f) - y^*(Z)_i & \text{se } f \textit{ entrar} \text{ em } Z_i. \end{cases} \tag{3.2}$$

O algoritmo é ganancioso no sentido de que, uma vez determinada uma variável dual positiva $y^*(Z_i)$, ela não será mais alterada nas etapas posteriores. O algoritmo precisa da seguinte sub-rotina.

Subrotina (K): *Determina, para uma determinada função de custo não negativa c', se $\mathcal{F}$ tem um membro c' positivo ou não e, se tiver, a sub-rotina calcula um membro mínimo c positivo de $\mathcal{F}$.*

Na Seção 4, descreveremos como a sub-rotina **(K)** pode ser realizada de forma eficiente para uma família especial de interseção $\mathcal{F}_0$ que será definida em (4.1) e usada para a solução algorítmica do problema de arborescência inversa.

Foi destacado em [10] que a seguinte propriedade da solução dupla y^* é uma consequência facilmente comprovável da algoritmo guloso acima.

Afirmação 3.1. *A solução dual ótima y^* fornecida pelo algoritmo tem a propriedade de que o sistema de conjuntos $\mathcal{F}^* := \{Z : y^*(Z) > 0\}$ é laminar.* ■

A segunda fase do algoritmo em [10] para calcular uma cobertura mais barata de $\mathcal{F}$ é a seguinte.

Segunda fase Deixe c' denotar a função de custo obtida no final da Fase 1 e deixe $A_0 := \{a \in A : c'(a) = 0\}$. Pela regra de término da Fase 1, A_0 cobre $\mathcal{F}$. Observe que as bordas em A_0 são c-tight com relação à solução dupla y^* obtida pela primeira fase. (De fato, uma indução simples em i mostra que, no momento em que $y^*(Z_i)$ se torna positivo e c_{i+1} é definido, $c_{i+1}(f) = c(f) - \sum [y^*(Z) : Z$ é inserido por $f]$ é válido e, portanto, se $c_i(f) > 0 = c_{i+1}(f)$, então f é restrito).

Para construir uma cobertura mais barata L de $\mathcal{F}$, pegamos as bordas de A_0 uma a uma, como segue. No início, L é vazio. Na etapa geral, verificamos se o L atual cobre $\mathcal{F}$ ou não. Em caso afirmativo, a Fase 2 (e todo o algoritmo) é encerrada. Se L ainda não cobrir $\mathcal{F}$, selecionamos um membro máximo Z de $\mathcal{F}$ não inserido por L e selecionamos uma aresta a de A_0 entrando em Z cujo custo se tornou zero no mínimo durante a Fase 1 (ou, em outras palavras, a se tornou apertado no mínimo). Deixe L^* denotar a cobertura de $\mathcal{F}$ obtida pelo término da Fase 2. O seguinte lema de [10] implica imediatamente o Teorema 2.3. Não incluímos sua prova aqui, apenas observamos que é uma consequência fácil da afirmação 3.1 e da regra da escolha mais cedo (usada para selecionar o elemento subsequente de L).

Lema 3.2 (*[10]*)**.** *A solução dupla y^* fornecida pela Fase 1 e a cobertura L^* de $\mathcal{F}$ obtida na Fase 2 atendem aos critérios de otimização.* ■

4. Algoritmo específico para o sistema de kernel F_0

Seja F_0 uma r_0 -arborescência de $D = (V, A)$. A ideia principal da nossa abordagem para o problema de arborescência inversa é que aplicamos a fórmula min-max do Teorema 2.3 e o algoritmo da Seção 3.1 referente a sistemas de kernel gerais a um

sistema de kernel específico $\mathcal{F}_0$ atribuído a F_0. Ou seja, deixe $\mathcal{F}_0$ denotar o sistema de subconjuntos $Z \subseteq V - r_0$ para o qual $\varrho_{F_0}(Z) = 1$, isto é,

$$\mathcal{F}_0 := \{Z \subseteq V - r_0 : \varrho_{F_0}(Z) = 1\}. \tag{4.1}$$

A seguir, uma observação padrão.

Afirmação 4.1. *O sistema de conjuntos $\mathcal{F}_0$ é interseção.*

Prova. *Sejam* X e Y dois membros de interseção de $\mathcal{F}_0$. Então, $\varrho_{F_0}(X \cap Y) \geq 1$ e $\varrho_{F_0}(X \cup Y) \geq 1$

implicam $1 + 1 = \varrho_{F_0}(X) + \varrho_{F_0}(Y) \geq \varrho_{F_0}(X \cap Y) + \varrho_{F_0}(X \cup Y) \geq 1 + 1$

e, portanto, $\varrho_{F_0}(X \cap Y) = 1$ e $\varrho_{F_0}(X \cup Y) = 1$, ou seja, tanto $X \cap Y$ quanto $X \cup Y$ estão em $\mathcal{F}_0$. ■

Pela afirmação 4.1, o Teorema 2.3 pode de fato ser aplicado a $\mathcal{F}_0$. Nosso objetivo é mostrar como a sub-rotina **(K)** solicitada para o algoritmo geral pode ser implementada para esse sistema de kernel concreto $\mathcal{F}_0$.

4.1. Abordagem simples

Calculamos o ótimo y^* aplicando a primeira fase do algoritmo descrito na Seção 2 para sistemas de kernel gerais ao sistema de kernel específico $\mathcal{F}_0$ definido por (4.1). Para isso, mostramos como a sub-rotina **(K)** necessária para a
O algoritmo geral pode ser implementado no caso especial de $\mathcal{F} := \mathcal{F}_0$. Lembre-se de que a sub-rotina **(K)** na Seção 3.1 decide, para uma função de custo de entrada c', se existe um membro c' positivo de $\mathcal{F}$ (ou seja, um membro para o qual cada borda de entrada tem custo c' positivo) e, se existir, a sub-rotina deve determinar um membro que seja mínimo em termos de inclusão.

Para construir essa subrotina, descrevemos a Subrotina $\mathbf{(K)}_f$ que decide, para qualquer elemento dado $f = uv$ da entrada r_0-*arborescência* F_0 se há um membro c'-positivo de $\mathcal{F}^f_0 := \{Z \in \mathcal{F}_0 : f \text{ entra em } Z\}$ e, se houver, a sub-rotina calcula o menor membro (único) de $\mathcal{F}^f_0$.

Deixe c'_f denotar a função de custo resultante de c' de tal forma que o c'-custo de cada elemento de $F_0 - f$ seja reduzido a
0, isto é,

$$c'_f(e) := \begin{cases} c'(e) & \text{se } e = f, \\ 0 & \text{se } e \in F_0 - f, \\ c'(e) & \text{se } e \in A - F_0. \end{cases} \tag{4.2}$$

Seja $A_f := \{a \in A : c'(a) = 0\}$ e seja $D_f = (V, A_f)$. A sub-rotina $\mathbf{(K)}_f$ calcula o conjunto S_f dos nós a partir dos quais v é alcançável em D_f. (Isso pode ser feito, por exemplo, por um BFS em tempo linear). Se r_0 estiver em S_f, a sub-rotina será encerrada
com a conclusão de que $\mathcal{F}^f_0$ não tem nenhum membro c' positivo. Se r_0 não estiver em S_f, o subprograma produzirá S_f como o membro mínimo solicitado de $\mathcal{F}_0$. A correção do subprograma decorre da afirmação a seguir.

Afirmação 4.2. *Se a cabeça v da aresta $f \in F_0$ for alcançável a partir de r_0 (em D_f), então $\mathcal{F}_0$ não admite nenhum membro c' positivo inserido por f. Se v não é acessível a partir de r_0, então S_f é o (único) membro mínimo c' positivo de $\mathcal{F}^f_0$.*

Prova. Se v é acessível a partir de r_0, então todos os nós de D_f são acessíveis a partir de r_0, pois o custo c' de cada aresta em $F_0 - f$ é
0. Mas, nesse caso, para cada subconjunto $Z \subseteq V - r_0$, há uma borda de entrada com custo 0 c'. Como o c'-cost e o c'_f-cost
das bordas que entram em um membro Z de $\mathcal{F}^f_0$ coincidem, $\mathcal{F}^f_0$ não tem nenhum membro c'-positivo.
Se v não for acessível a partir de r_0, então também não é acessível a partir de u e, portanto, f entra em S_f. Como o c'_f-custo de cada
outro elemento de F_0 é 0 e S_f é c'_f-positivo, f é a única aresta em F_0 que entra em S_f e, portanto, $S_f \in \mathcal{F}_0$. Além disso, há
é um dipath de custo 0 c'_f de cada nó em S_f para v e, portanto, S_f não pode ter um subconjunto c'_f positivo adequado
contendo v. ■

Ao aplicar a sub-rotina $\mathbf{(K)}_f$ separadamente a cada elemento f de F_0, a sub-rotina **(K)** solicitada para o sistema de kernel especial $\mathcal{F}_0$ está de fato disponível.

O algoritmo acima é fortemente polinomial e conceitualmente simples. Entretanto, pode-se achar que ele não é particularmente eficiente, pois o cálculo das variáveis duais positivas subsequentes $y^*(Z)$ requer a aplicação da sub-rotina $\mathbf{(K)}_f$
para cada membro f de F_0. Essa desvantagem é superada pela seguinte abordagem mais compacta.

O algoritmo na Seção 4.1 forneceu uma realização da sub-rotina **(K)** desenvolvendo a sub-rotina $\mathbf{(K)}_f$ e aplicando-a separadamente a cada elemento f da r_0 -arborescência de entrada F_0 . Aqui apresentamos uma realização mais eficiente da sub-rotina

(K) considerando os elementos de F_0 em uma ordem específica $f_1, \ldots, f_{n-1}$ com Propriedade **(O)**: a borda $f \in F_0$ precede a borda $e \in F_0$

nessa ordenação se F_0 admitir um dipath da cabeça de $e \in F_0$ para a cauda de f . Observe que essa ordem pode ser facilmente

calculado em tempo linear: considere uma construção de F_0 que começa em r_0 e adiciona novas bordas, uma a uma, de tal forma

que cada borda recém-adicionada deixa a sub-arborescência já construída. (Esse procedimento de construção de F_0 também é realizável em tempo linear.) Segue-se imediatamente que, ao inverter a ordem de construção, obtemos $f_1, \ldots, f_{n-1}$ com a Propriedade **(O)**. Observamos que a Propriedade **(O)** será usada somente na prova do Lema 4.3 referente à correção do algoritmo.

O algoritmo considera os elementos de F_0 um a um na ordem determinada. Ele consiste em $n - 1$ estágios subsequentes em que o estágio j está relacionado à borda f_j. Durante um estágio, calculamos uma sequência (possivelmente vazia) de subconjuntos (formando uma cadeia crescente) para a qual f_j é a única borda em F_0 que entra nesses conjuntos e para a qual sua variável dupla será positiva. De acordo com a regra da Fase 1 do algoritmo geral, quando essa variável dupla é definida, reduzimos a função de custo atual, denotada por c'. Deve-se enfatizar que pode ser que no Estágio j nenhum novo conjunto receba uma variável dupla positiva e também é possível que mais de um desses conjuntos receba uma variável dupla positiva.

No início do Estágio 1, $c' := c$. Considere agora o Estágio j referente à borda f_j e deixe c' denotar a função de custo atual. Com a ajuda da Subrotina **(K)**$_{f_j}$ descrita acima, decida se há um conjunto positivo para c' para o qual f_j é o único membro de F_0 que entra nesse conjunto. Se esse conjunto não existir, a Etapa j será encerrada. Nesse caso, se $j = n - 1$, todo o algoritmo termina, enquanto se $j < n - 1$, passamos para o Estágio $j + 1$ referente à borda f_{j+1}.

Suponha agora que o conjunto c' positivo em questão exista e considere o menor desses conjuntos Z' calculado por Sub-rotina **(K)**$_{f_j}$. Copiando a (primeira fase do) algoritmo geral para sistemas de kernel, vamos

$$y^*(Z') := min\{c'(f) : f \in A, f \text{ entra em } Z'\}$$

e revisar c' da seguinte forma.

$$c'(f) := \begin{cases} c'(f) & \text{se } f \text{ não entrar em } Z' \\ c'(f) - y^*(Z') & \text{se } f \text{ entrar em } Z'. \end{cases} \tag{4.3}$$

(Observe que a notação atual está em harmonia com a usada na descrição do algoritmo geral, com a única diferença de que aqui usamos Z' no lugar de Z_i usado na descrição do algoritmo geral e, também, usamos aqui c' em no lugar de c_i ou c_{i+1}).

Se $c'(f_j) = 0$ for válido para c' definido em (4.3), o Estágio j será encerrado. Nesse caso, se $j = n - 1$, todo o algoritmo termina, enquanto se $j < n - 1$, passamos para o Estágio $j + 1$ referente à borda f_{j+1}. Se $c'(f_j) > 0$, continue a execução do Estágio j com a mesma f_j e com a função de custo c' revisada em (4.3).

Nosso objetivo final é verificar a correção do algoritmo, ou seja, provar que o procedimento produz uma solução dupla ideal. O algoritmo da Seção 3.1 para calcular uma solução dual ótima para um sistema de núcleo geral era genérico no sentido de que o conjunto Z_i escolhido atualmente deveria ser um membro mínimo c_i positivo de $\mathcal{F}$, mas, dentro desses requisitos, não importava qual desses conjuntos foi realmente selecionado para ser Z_i.

Para provar a correção do presente algoritmo com relação a $\mathcal{F}_0$, mostramos que a escolha de Z' pode ser interpretado como uma escolha específica de Z_i que ocorre no algoritmo geral. Esse é exatamente o conteúdo do próximo lema e, portanto, o lema, juntamente com a exatidão do algoritmo geral para sistemas de kernel, implica a exatidão do algoritmo presente .

Lema 4.3. *No momento em que o algoritmo encontra um conjunto mínimo c'-positivo Z' para o qual f_j é o único elemento de F_0 que entra em Z', o conjunto Z' é um membro mínimo c'-positivo de $\mathcal{F}_0$.*

Prova. Suponha, indiretamente, que no momento de encontrar Z', $\mathcal{F}_0$ tenha um membro c' *positivo* Z'' para o qual $Z'' \subset Z'$. Deixe f_h denotar a única borda em F_0 que entra em Z''. A escolha mínima de Z' mostra que $h \neq j$, o que implica que f_h deve estar completamente em Z'. Portanto, o único dipath em F_0 da raiz r_0 para f_h deve passar por f_j e, portanto, a Propriedade **(O)** implica que $h < j$, ou seja, o Estágio h precedeu o Estágio j. No final do Estágio h, havia uma aresta $a \in A$ entrando em Z'' cujo custo atual naquele momento era 0. Mas então $c'(a) = 0$, em uma contradição com a suposição indireta de que Z'' era *w-positivo* no momento da definição de Z'. ■

Teorema 4.4. *A complexidade do algoritmo acima é $O(mn)$, em que m e n denotam o número de arestas e nós de D, respectivamente.*

Prova. O algoritmo consiste em $|F_0| = n - 1$ estágios. No início de cada estágio, calculamos o conjunto S_{f_j} de nós a partir dos quais a cabeça de f_j é alcançável em D_{f_j}. Por meio de um BFS, isso pode ser feito em tempo $O(m)$ e, portanto, esses conjuntos podem ser calculados em tempo $O(mn)$.

As outras etapas dos estágios são usadas para determinar as variáveis duais positivas. Em vez de estimar o número dessas etapas separadamente por estágio, fornecemos um limite superior $O(n)$ para o número total de conjuntos que obtêm variáveis duais positivas. De fato, a segunda metade do Teorema 2.3 implica que a família de conjuntos com variáveis duais positivas é laminar e, portanto, sua cardinalidade é de no máximo $2n$. Em resumo, o número total de

etapas é de fato $O(mn)$. ■

5. Solução para o problema inverso da arborescência

Vamos nos voltar para o problema de arborescência inversa, no qual queremos fazer uma determinada (chamada entrada) r_0 *-arborescência* F_0 para ser a mais barata, revisando uma determinada função de custo não negativa w_0 de forma mínima. Ou seja, o objetivo é encontrar uma nova função de custo $w : A \to \mathbf{R}$ para a qual F_0 seja uma r_0 -arborescência mais barata e o desvio $|w - w_0| := (|w(a) - w_0(a)| : a \in A)$ de w em relação a w_0 seja o menor possível. Esse mínimo será denotado por μ^* .

Uma observação essencial e natural é que o conjunto de funções de custo w para as quais F_0 é uma arborescência mais barata r_0 forma um poliedro. Isso implica que o mínimo de desvio de fato existe e o conjunto de funções de custo minimizadoras de desvio (ou seja, aquelas com desvio μ^*) também é um poliedro.

5.1. Teorema e algoritmo do mínimo-máximo

Como preparação, precisamos da seguinte observação simples.

Proposição 5.1. *Seja* $D' = (V, F_0 \cup L')$ *um digrafo no qual* F_0 *é uma* r_0 *-arborescência de* D' *que é disjunta de* L' *. Seja* $\mathcal{F}_0 := \{Z \subseteq V - r_0 : \varrho_{F_0}(Z) = 1\}$*. Então,* D' *é enraizado e conectado por 2 arestas se e somente se* L' *cobre* $\mathcal{F}_0$ *. Além disso, se* L' *é um conjunto mínimo de inclusão de arestas que cobre* $\mathcal{F}_0$ *, então* $\varrho_{L'}(v) = 1$ *(ou equivalentemente* $\varrho_{D'}(v) = 2$*) para cada nó* $v \in V - r_0$ *.*

Prova. A primeira afirmação é uma consequência imediata da definição de conectividade enraizada de 2 arestas. Para ver a segunda, observe que, se Z_1 e Z_2 são dois subconjuntos de interseção de $V - r_0$ para os quais $\varrho_{D'}(Z_i) = 2$, então, $2 + 2 = \varrho_{D'}(Z_1) + \varrho_{D'}(Z_2) \geq \varrho_{D'}(Z_1 \cap Z_2) + \varrho_{D'}(Z_1 \cup Z_2) \geq 2 + 2$, de onde se conclui que $\varrho_{D'}(Z_1 \cap Z_2) = 2$. Isso implica que todo nó $v \in V - r_0$ está contido em um único menor subconjunto Z_v para o qual $\varrho_{D'}(Z_v) = 2$. Isso e a minimalidade de L' implicam que cada aresta $f \in L'$ entra em v entra em Z_v . Mas, então, $2 = \varrho_{D'}(Z_v) = \varrho_{F_0}(Z_v) + \varrho_{L'}(v) \geq 1 + \varrho_{L'}(v)$, donde $\varrho_{L'}(v) \leq 1$. Por outro lado, $\{v\} \in \mathcal{F}_0$ implica que $\varrho_{L'}(v) \geq 1$ e, portanto, $\varrho_{L'}(v) = 1$. ■

Chamamos uma função de custo $w : A \to \mathbf{R}$ w_0 **-adequada** ou apenas **adequada** se F_0 for uma arborescência mais barata r_0 com relação a w e

$$\begin{cases} w(f) \leq w_0(f) \text{ para cada } f \in F_0 \\ w(e) \geq w_0(e) \text{ para cada } e \in A - F_0 \end{cases} \tag{5.1}$$

.

Se, além disso, $w \geq 0$ e $w(e) = w_0(e)$ para cada $e \in A - F_0$, então dizemos que w é **fortemente adequado**. Um novo resultado importante do presente trabalho é a seguinte fórmula min-max, que serve como base estrutural do novo algoritmo para o problema inverso de arborescência.

Teorema 5.2. *Seja* $w_0 \geq 0$ *uma função de custo no conjunto de bordas do digrafo* $D = (V, A)$ *e seja* F_0 *uma* r_0 *-arborescência de* D*. Seja* $\mathcal{F}_0 := \{Z \subseteq V - r_0 : \varrho_{F_0}(Z) = 1\}$*. Então*

$$\begin{aligned} \mu^* := \min\{|w - w_0| : w \text{ uma função de custo para a qual } F_0 \text{ é a mais barata } r_0\text{ - arborescência}\} \\ = \max\{w(F) - w(L) : L \subseteq A, L \text{ cobre } \mathcal{F}_0\}. \end{aligned} \tag{5.2}$$

Além disso, existe uma solução ótima primal fortemente adequada $w = w^*$ *para* (5.2) *que, além disso, tem valor inteiro quando* w_0 *é de valor inteiro.*

Prova. Se w é uma função de custo para a qual F_0 é uma arborescência r_0 mais barata e seu desvio $|w - w_0|$ é mínimo, então w é obviamente adequado. Portanto, para provar que max ≤ min, basta mostrar que

$$|w - w_0| \geq w(F) - w(L) \tag{5.3}$$

é válido para toda cobertura $L \subseteq A$ de $\mathcal{F}_0$ e para toda função de custo adequada w. Como w_0 é não-negativo, basta provar (5.3) somente quando L for uma cobertura mínima de $\mathcal{F}_0$ com inclusão.

Seja w uma função de custo adequada e L uma cobertura mínima de inclusão de $\mathcal{F}_0$. Para cada $e \in L \cap F_0$, seja e' uma nova aresta paralela a e e seja $w(e') := w(e)$. Seja N' o conjunto de novas arestas, $L' := (L - F_0) \cup N'$, e $A^+ := A \cup N'$. Obviamente, $|L'| = |L|$ e $w(L') = w(L)$. Também decorre dessas definições que F é uma arborescência r mais barata (com relação à custo estendido $\tilde{w}$ em A^+) no digrafo (V, A^+) e, portanto, F_0 é uma barata r_0 -arborescência em $D' := (V, F_0 \cup L')$, também.

Pela Proposição 5.1, temos $\varrho_{F_0}(v) + \varrho_{L'}(v) = 2$ para cada $v \in V - r_0$. Isso é o teorema de Edmonds [8] sobre r_0-arborescências disjuntas implicam que D' é a união de duas r_0-arborescências disjuntas F_1 e F_2. Como F_0 é uma r_0-arborescência mais barata em D' com relação a w, temos

$$\tilde{w}(L) + \tilde{w}(F_0) = \tilde{w}(L') + \tilde{w}(F_0) = \tilde{w}(L' \cup F_0) = \tilde{w}(F_1) + \tilde{w}(F_2) \geq \tilde{w}(F_0) + \tilde{w}(F_0),$$

from which

$$\tilde{w}(L) \geq \tilde{w}(F_0) \tag{5.4}$$

segue. Como w é adequado, para $\overline{F_0} := A - F_0$, temos: ($F$

$$\begin{aligned} \tilde{w}(\overline{F_0}) - w(F_0) &= \\ &= [\tilde{w}(L) - \tilde{w}_0(L)] + [w(\overline{F_0} - L) - w_0(\overline{F_0} - L)] - [\tilde{w}(L \cap L) - w_0(F_0 \cap L)] \\ &\geq w(L) - w_0(L) + 0 + \end{aligned} \tag{5.5}$$

Isso e (5.4) implicam: 0.

$$\begin{aligned} |w - w_0| &= \\ &= [w(F_0) - w_0(F_0)] + [w_0(F_0) - w(F_0)] \\ &\geq [\tilde{w}(F_0) - w(\tilde{F}_0)] + [w(\tilde{L}) - \tilde{w}(L)] \\ &= [w_0(F_0) - w_0(L)] + [\tilde{w}(L) - w_0(F_0)] \\ &\geq \tilde{w}(F_0) - w^0(L), \end{aligned} \tag{5.6}$$

conforme exigido em (5.3) e, portanto, segue-se max ≤ min.

Para ver a direção inversa max ≥ min, basta provar que há uma função de custo adequada w^* (com valor inteiro quando w_0 é assim) e uma cobertura L^* de F_0 para a qual (5.3) é válida com igualdade. Por (5.6), temos igualdade em (5.3) precisamente se
ambas as desigualdades em (5.6) são válidas com igualdade. Além disso, temos igualdade em (5.6) precisamente se ambos (5.4) e (5.5) forem válidos com igualdade, o que é equivalente aos seguintes critérios de otimização.

(OC.1) $w^*(L^*) = w^*(F_0)$,

(OC.2) $\tilde{w}^*(\overline{F_0} - L^*) = \tilde{w}^*_0(\overline{F_0} - L^*)$,

(OC.3) $\tilde{w}^*(F_0 \cap L^*) = \tilde{w}^*_0(F_0 \cap L^*)$.

Aplique o Teorema 2.3 ao sistema de núcleo especial F_0 no lugar de F e à função de custo $c := w_0$. Deixe L denotar a solução primária ideal, ou seja, L^* é uma cobertura mais barata de F_0, e deixe y^* ser a solução dupla ideal em (2.3). Pela
Teorema, $\tilde{y}^*(F_0) = \tilde{w}^*_0(L^*)$. Defina w da seguinte forma.

$$w^*(a) := \begin{cases} w_0(a) & \text{se } a \in A - F_0 \\ \sum [y^*(Z) : a \text{ entra em } Z] & \text{se } a \in F_0. \end{cases} \tag{5.7}$$

Esse w^* é não-negativo e $w^*(a) = w_0(a)$ para cada $a \in A - F_0$, além disso, y^* é claramente w^*-fácil (para a definição de viabilidade, consulte (2.1)) e F_0 consiste em bordas apertadas com relação a w^*. Aplicando o Corolário 2.2 a w^* no lugar de c,
Obtemos que F_0 é uma r_0-arborescência mais barata de D com relação a w^*. Portanto, w^* é fortemente w_0-adequado e, por
Teorema 2.3, temos que $\tilde{w}^*(F_0) = \tilde{y}(F_0) = \tilde{w}^*_0(L^*) = \tilde{w}(L)$, ou seja, **(OC.1)** é válido.

A forte adequação de w^* implica imediatamente **(OC.2)**. Para $a \in F_0$, temos $w^*(a) = \sum [y^*(Z) : Z \in F_0, a \text{ entra em } Z]$ enquanto que para $a \in L^*$ temos $w_0(a) = \sum [y^*(Z) : Z \in F_0, a \text{ entra em } Z]$. Portanto, $w^*(a) = w_0(a)$ é válido para $a \in F_0 \cap L^*$ e, portanto
(OC.3) é o seguinte.

Por fim, observe que, se w_0 tiver valor inteiro, então a solução dual ótima y^* também pode ser escolhida com valor inteiro pelo Teorema 2.3 e, portanto, w^* definido em (5.7) também tem valor inteiro. ■

Observação 5.3. Pode-se achar que o uso do teorema profundo de Edmonds na prova da desigualdade "trivial" max ≤ min não é realmente apropriado. De fato, isso pode ser evitado se nos basearmos na descrição poliédrica do politopo de extensão
r_0-arborescências e usar um argumento um pouco mais técnico. Ou seja, o casco convexo de r_0-arborescências de D é dado por: $\{x : x \geq 0, \varrho_x(Z) \geq 1$ para todo subconjunto não vazio $Z \subseteq V - r_0$, e $\varrho_x(v) = 1$ para todo nó $v \in V - r_0\}$. (Essa é uma consequência imediata, por exemplo, do Teorema 2.1.) Lembre-se de que o digrafo $D' := (V, A')$ (onde $A' = F_0 \cup L'$) era
demonstrou ser enraizada com 2 arestas conectadas, para as quais $\varrho_{D'}(v) = 2$ para cada $v \in V - r_0$. Seja $z := \chi(A')/2$ a função identicamente 1/2
em A'. Usando a descrição poliédrica das r_0-arborescências de D, obtemos que z pode ser expresso como o combinação convexa de r_0-arborescências: $z = \sum [\lambda_i \chi(F_i) : i = 1, \ldots, q]$ onde $\lambda_i > 0$ para cada i e $\sum \lambda_i = 1$. Então, temos $w(F') + w(L') = w(A') = 2wz = \sum [\lambda_i w(F_i) : i = 1, \ldots, q] \geq \sum [\lambda_i w_0(F_i) : i = 1, \ldots, q] = \tilde{w}(L) = \tilde{w}_0(L)$, conforme necessário para max ≤ min.

Essa prova é tecnicamente um pouco mais complicada do que a que usa o teorema de Edmonds. Entretanto, ela pode ser usada em outros problemas de otimização inversa em que o análogo do teorema de Edmonds (ou seja, a propriedade discreta de Carathéodory

de r_0 -arborescências) não se mantém enquanto a descrição poliédrica dos objetos em questão estiver disponível.

Uma consequência imediata da prova acima é a seguinte.

Corolário 5.4. *Seja y^* uma solução dual ótima para o problema primário de encontrar uma cobertura de custo w_0 mínima do sistema de kernel $F_0 := \{Z \subseteq V - r_0 : \varrho_{F_0} = 1\}$. Então, a função de custo w^* definida por (5.7) é uma solução ótima para o problema de arborescência inversa.* ■

Algoritmo. O Corolário 5.4 implica que o algoritmo desenvolvido na Seção 4 para o sistema de kernel especial $F_0 := \{Z \subseteq V - r_0 : \varrho_{F_0}(Z) = 1\}$, quando aplicado a $c := w_0$, calcula em tempo $O(mn)$ tanto a solução primária ótima L^* quanto a

solução dual ótima y^* em (2.3). Provamos que w^*, conforme definido em (5.7) por y^*, é uma solução primária ótima (fortemente adequada) em (5.2) para o problema de arborescência inversa, enquanto L^* é uma solução dupla ótima em (5.2).

Observação 5.5. A afirmação do Teorema 5.2 de que a função de custo ótima desejada w^* pode ser escolhida de tal forma que $w^*(e) = w_0(e)$ seja válida para cada aresta $e \in A - F_0$ já foi provada por Hu e Liu [16]. É interessante observar que a propriedade correspondente não se mantém na contraparte não direcionada do problema de arborescência inversa, em que o objetivo
é fazer com que uma árvore de abrangência de entrada F_0 de um gráfico não direcionado G seja uma árvore mais barata. Para ver isso, deixe G ser um triângulo com bordas f, g, h cujos custos são 1, 1, 0 e deixe $F_0 = \{f, g\}$ ser uma árvore de abrangência. Se não for permitido alterar os custos fora de F_0, então o custo de f e g deve ser reduzido a 0 para tornar F_0 uma árvore mais barata e, portanto, a alteração total
(desvio) é 2. Por outro lado, se aumentarmos o custo de h em 1, então F_0 se torna uma árvore mais barata, ou seja, o desvio nesse caso é de apenas 1. O mesmo exemplo mostra que a propriedade também não se aplica a matroides em que uma base de entrada B_0 deve ser transformada em uma base mais barata.

Observação 5.6. Um dos nossos principais objetivos era desenvolver um algoritmo polinomial para o problema da arborescência inversa que fosse definitivamente mais simples do que o algoritmo existente de Hu e Liu [16]. Essa foi a razão pela qual enfatizamos, ao longo da descrição acima, que o algoritmo "de fundo" solicitado em relação aos sistemas de kernel é um algoritmo guloso de duas fases conceitualmente simples. Por outro lado, a presente abordagem pode ser adaptada a outros problemas inversos para os quais nenhum algoritmo polinomial foi desenvolvido até o momento. Nesse caso, não se espera que o algoritmo de fundo análogo seja particularmente simples, a única exigência é que ele seja polinomial.

Agradecimentos

Os autores são gratos aos revisores anônimos por seu trabalho cuidadoso e conselhos úteis.

Referências

[1] F. Bock, An algorithm to construct a minimum directed spanning tree in a directed network, em: B. Avi-Itzhak (Ed.), Developments of Operations Research, Vol. 1, Tel Aviv, 1969, em: Proceedings of the Third Annual Israel Conference on Operations Research, Gordon and Breach, Nova York, 1971, pp. 29-44.
[2] D. Burton, Ph.L. Toint, On an instance of the inverse shortest paths problem, Math. Program. 53 (1992) 45-61.
[3] M.-C. Cai, Inverse problems of matroid intersections (Problemas inversos de interseções de matróides), J. Combin. Optim. 3 (4) (1999) 465-474.
[4] M.-C. Cai, Y.-J. Li, Inverse matroid intersection problem, Math. Methods Oper. Res. 45 (2) (1997) 235-243.
[5] Y.-J. Chu, T.-H. Liu, On the shortest arborescence of a directed graph, Sci. Sinica (Peking) 14 (1965) 1397-1400.
[6] M. Demange, J. Monnot, An introduction to inverse combinatorial problems, em: V.Th. Paschos (Ed.), Paradigms of Combinatorial Optimization: Problems and New Approaches, segunda edição, ISTE LTd e John Wiley and Sons, 2014, pp. 547-586, Capítulo 17.
[7] J. Edmonds, Optimum branchings, J. Res. Nat. Bur. Stand. B 71 (1967) 233-240.
[8] J. Edmonds, Edge-disjoint branchings, em: B. Rustin (Ed.), Combinatorial Algorithms, Acad. Press, Nova York, 1973, pp. 91-96.
[9] J. Edmonds, Matroid intersection, Ann. Discrete Math. 4 (1979) 39-49.
[10] A. Frank, Kernel systems of directed graphs, Acta Sci. Math. (Szeged) 41 (1-2) (1979) 63-76.
[11] A. Frank, Increasing the rooted connectivity of a digraph by one (Aumentando a conectividade enraizada de um digrafo em um), em: A. Frank (Ed.), Connectivity Augmentation of Networks: Structures and Algorithms, Mathematical Programming, Ser. B, Vol. 84, No. 3, 1999, pp. 565-576.
[12] A. Frank, Connections in Combinatorial Optimization (Conexões na otimização combinatória), em: Oxford Lecture Series in Mathematics and its Applications, vol. 38, Oxford University Press, ISBN: 978-0-19-920527-1, 2011.
[13] A. Frank, K. Murota, A discrete convex min-max formula for box-TDI polyhedra, Math. Oper. Res. (2021) no prelo.
[14] D.R. Fulkerson, Packing rooted directed cuts in a weighted directed graph, Math. Program. 6 (1974) 1-13.
[15] C. Heuberger, Inverse combinatorial optimization: A survey on problems, methods, and results (Uma pesquisa sobre problemas, métodos e resultados), J. Combin. Optim. 8 (2004) 329-361.
[16] Z. Hu, Z. Liu, A strongly polynomial algorithm for the inverse shortest arborescence problem, Discrete Appl. Math. 82 (1998) 135-154.
[17] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, em: Algorithms and Combinatorics, vol. 24, Springer, 2003.
[18] É. Tardos, A strongly polynomial minimum cost circulation algorithm, Combinatorica 5 (1985) 247-255.