Minimização de funções submodulares

Juliana Barby Simão

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIAS

Programa: Ciência da Computação
Orientador: Prof. Dr. José Coelho de Pina Jr.

Durante o desenvolvimento deste trabalho a autora recebeu auxílio financeiro da FAPESP
através do processo 04/11340-8

São Paulo, junho de 2009

Minimização de funções submodulares

Este exemplar corresponde à redação final
da dissertação devidamente corrigida e
defendida por Juliana Barby Simão e
aprovada pela Comissão Julgadora.

Banca Examinadora:

Prof. Dr. José Coelho de Pina Jr. (orientador)	IME-USP
Prof. Dr. Arnaldo Mandel	IME-USP
Prof. Dr. Orlando Lee	IC-UNICAMP

Agradecimentos

Agradeço

ao Coelho, pela excelente orientação, pelos ''insights" que fizeram toda a diferença e por me fazer acreditar sempre que o trabalho daria certo;

aos Professores Arnaldo Mandel e Orlando Lee, pela participação na banca e sugestões. Novamente ao Orlando, pela cuidadosa leitura do trabalho;

aos Professores Yoshiko Wakabayashi e Paulo Feofiloff, pelos valorosos ensinamentos durante a graduação e o mestrado, assim como por terem participado da defesa;

aos colegas do BCC 2001 (Ellen, Marcelo, Marcel, Cardonha, Domingos), aos Imescos e ao pessoal da sala VIP, pelas discussões e agradáveis momentos. Novamente ao Leo, por sua querida companhia durante o desenvolvimento do trabalho;

ao Luis Gabriel, por seu carinho e apoio na reta final;

à tia Marli, por sua presença constante (e por ter providenciado o bolo), e ao vô Pedro, que permaneceu com os olhos abertos até o último minuto da defesa;

aos meus irmãos: Magoozito, pela companhia nas tardes de estudo, e Guilherme, pelas partidas de ''Super Mário" nas férias;

aos meus pais, Carlos e Luzia, pelos constantes suporte e carinho e a quem devo grande parte de minhas conquistas.

Resumo

Funções submodulares aparecem naturalmente em diversas áreas, tais como probabilidade, geometria e otimização combinatória. Pode-se dizer que o papel desempenhado por essas funções em otimização discreta é similar ao desempenhado por convexidade em otimização contínua. Com efeito, muitos problemas em otimização combinatória podem ser formulados como um problema de minimizar uma função submodular sobre um conjunto apropriado. Além disso, submodularidade está presente em vários teoremas ou problemas combinatórios e freqüentemente desempenha um papel essencial em uma demonstração ou na eficiência de um algoritmo.

Nesta dissertação, estudamos aspectos estruturais e algorítmicos de funções submodulares, com ênfase nos recentes avanços em algoritmos combinatórios para minimização dessas funções. Descrevemos com detalhes os primeiros algoritmos combinatórios e fortemente polinomiais para esse propósito, devidos a Schrijver e Iwata, Fleischer e Fujishige, além de algumas outras extensões. Aplicações de submodularidade em otimização combinatória também estão presentes neste trabalho.

Palavras-chave: funções submodulares, algoritmos combinatórios, otimização combinatória.

Abstract

Submodular functions arise naturally in various fields, including probability, geometry and combinatorial optimization. The role assumed by these functions in discrete optimization is similar to that played by convexity in continuous optimization. Indeed, we can state many problems in combinatorial optimization as a problem of minimizing a submodular function over an appropriate set. Moreover, submodularity appears in many combinatorial theorems or problems and frequently plays an essencial role in a proof or an algorithm.

In this dissertation, we study structural and algorithmic aspects of submodular functions. In particular, we focus on the recent advances in combinatorial algorithms for submodular function minimization. We describe in detail the first combinatorial strongly polynomial-time algorithms for this purpose, due to Schrijver and Iwata, Fleischer, and Fujishige, as well as some extensions. Some applications of submodularity in combinatorial optimization are also included in this work.

Keywords: submodular functions, combinatorial algorithms, combinatorial optimization.

Sumário

Introdução

Em 1935, Whitney [Whi35] definiu o conceito de matróide com o intuito de capturar propriedades fundamentais de dependência linear. Como as funções posto de matróides distintos são distintas, então pode-se obter informações sobre a estrutura de um matróide através da investigação de sua função posto. A função posto de um matróide é submodular e talvez seja a mais notória dentre as muitas funções submodulares.

Pesquisadores perceberam que, dentre as propriedades da função posto de um matróide, submodularidade era a mais importante e passaram a investigá-la. De fato, como mais tarde ficou evidente, o interesse em submodularidade vai além da teoria dos matróides. Funções submodulares aparecem naturalmente em uma variedade de áreas, incluindo otimização combinatória, probabilidade e geometria.

Edmonds [Edm70] iniciou um estudo sistemático de funções submodulares e de certos poliedros associados a estas; os polimatróides. Edmonds mostrou que otimizar uma função linear sobre um polimatróide estendido é uma tarefa fácil e pode ser feita através de uma extensão natural do método guloso. Posteriormente, outros modelos poliédricos associados a funções submodulares foram definidos. Dentre eles, citamos os fluxos submodulares, de Edmonds e Giles [EG77], e os *kernel systems* e *generalized polymatroids*, ambos de Frank [Fra79b, Fra84a, Fra84b].

Talvez o primeiro trabalho onde se tenha visto um arcabouço geral para grafos utilizando submodularidade tenha sido o de Lovász [Lov70], onde uma conjectura de Paul Erdős é demonstrada. Frank [Fra93b], usando técnicas básicas em funções submodulares, demonstrou teoremas clássicos de Menger, Mader, Tutte e Nash-Williams. Essas técnicas se mostraram ainda ingredientes chaves na solução de problemas de caminhos disjuntos e de aumento de conexidade em grafos [Fra90, Fra94].

Em otimização contínua, funções convexas desempenham um papel central. Funções submodulares desempenham, de certa forma, um papel similar em otimização discreta, como mostrou Lovász [Lov83]. Com efeito, muitos problemas em otimização combinatória podem ser formulados como um problema de minimizar uma função submodular sobre um conjunto apropriado. Ilustrações desse fenômeno incluem o problema do T-corte ímpar mínimo e o problema da intersecção de dois matróides. A minimização de funções submodulares é também um ingrediente fundamental na resolução de problemas em fluxos submodulares [FI00b], que modelam uma série de problemas em aumento de conexidade em grafos. Assim, não é surpresa que um problema central em teoria algorít-

mica de funções submodulares seja o de minimizá-las.

Grötschel, Lovász e Schrijver [GLS81, GLS88] apresentaram um algoritmo fortemente polinomial para minimizar funções submodulares. Entretanto, esse algoritmo não é plenamente satisfatório, já que envolve o método dos elipsóides e, portanto, usa todo o arcabouço pesado de divisão, arredondamento e aproximação; além de ser sabidamente lento do ponto de vista prático. Assim, encontrar um algoritmo polinomial puramente combinatório foi considerado por muitos como o problema em aberto mais desafiador na área [GLS81, Lov83, Fra84b].

Um dos recentes avanços mais excitantes em submodularidade foi o surgimento de dois algoritmos combinatórios fortemente polinomiais para minimizar funções submodulares, desenvolvidos independentemente em 1999 por Iwata, Fleischer e Fujishige [IFF01] e Schrijver [Sch00]. Em 2003, os artigos desses pesquisadores receberam da *Mathematical Programming Society* e *American Mathematical Society* o *Fulkerson Prize*. Seus algoritmos foram considerados como um passo importante na história dos algoritmos fortemente polinomiais para problemas em otimização discreta e, sob esse ponto de vista, foram comparados ao algoritmo de Edmonds e Karp [EK72] para o problema do fluxo máximo e ao algoritmo de Tardos [Tar85] para o problema do fluxo máximo de custo mínimo. Ao desenvolvimento algorítmico alcançado por Iwata, Fleischer, Fujishige e Schrijver em submodularidade, seguiram-se outros avanços [IMS00, FIM02, Iwa02, Iwa03, FI03, IMS05, Orl07, Orl09, IO09].

Este mestrado tem como objetivo o estudo de aspectos estruturais e algorítmicos de funções submodulares, com ênfase nos recentes algoritmos combinatórios e fortemente polinomiais para minimização dessas funções.

Estrutura do texto

A dissertação foi desenvolvida como uma espécie de histórico sobre minimização de funções submodulares, com foco nas idéias que levaram ao desenvolvimento dos primeiros algoritmos combinatórios e fortemente polinomiais.

Nos capítulos 1 e 2, apresentamos os pré-requisitos teóricos para os algoritmos. No capítulo 1, além de definição e exemplos, algumas operações e construções envolvendo funções submodulares são apresentadas. No capítulo 2, exploramos os primeiros modelos em combinatória poliédrica associados a funções submodulares: os polimatróides e os polimatróides estendidos. Apresentamos o método guloso para otimização de funções lineares sobre tais poliedros, bem como uma caracterização de seus pontos extremos. Os resultados presentes nesses capítulos são recorrentes ao longo do texto.

No capítulo 3, introduzimos o problema de minimização de funções submodulares além das idéias que levaram Grötschel, Lovász e Schrijver ao estabelecimento de sua polinomialidade. Algumas considerações sobre complexidade de algoritmos, bem como sobre a representação das funções submodulares para efeitos algorítmicos, também são apresentadas.

No capítulo 4, tratamos das primeiras idéias combinatórias associadas à minimização

de funções submodulares, desenvolvidas principalmente por Cunningham no contexto do problema da pertinência no politopo dos conjuntos independentes de um matróide. Nosso objetivo foi apresentar as idéias de Cunningham como um método básico para minimizar funções submodulares, o qual foi gradualmente incrementado até que se chegasse em algoritmos fortemente polinomiais.

Nos capítulos 5 e 6, apresentamos respectivamente os algoritmos combinatórios e fortemente polinomiais devidos a Schrijver e Iwata, Fleischer e Fujishige. Ambos os algoritmos são descritos como extensões do método de Cunningham, de maneira a auxiliar o leitor na compreensão de como se chegou ao seu desenvolvimento. Avanços obtidos a partir desses primeiros algoritmos também são apresentados nesses capítulos.

Por fim, no capítulo 7, apresentamos algumas aplicações de submodularidade em otimização combinatória. Começamos com uma visão geral dos modelos poliédricos com restrições submodulares. Encerramos com a prova de alguns resultados em grafos que se utilizam da submodularidade de certas funções como ferramenta. O capítulo é uma espécie de apêndice, com algumas motivações para o estudo de submodularidade.

Notação e considerações gerais

Funções submodulares, supermodulares e modulares são funções reais definidas sobre as partes de um certo conjunto base finito. No decorrer do texto, sempre que não especificado, esse conjunto base será denotado por S. Sendo assim, freqüentemente escrevemos trechos da forma: "seja f uma função submodular", como abreviatura de: "seja f uma função submodular das partes de S em $\mathbb{R}$". O conjunto das partes de S será denotado por $\mathcal{P}(S)$.

Além disso, não fazemos distinção entre funções reais definidas sobre determinado conjunto e vetores reais indexados pelos elementos desse mesmo conjunto. Se w é um vetor indexado por S, denotamos $w(U) := \sum_{u \in U} w(u)$, para cada parte U de S.

Para qualquer parte U de S, o **vetor característico** ou **vetor de incidência** de U é o vetor χ^U, definido por $\chi^U(u) := 1$, se u está em U, e $\chi^U(u) := 0$, se u não está em U. Para qualquer elemento s de S, denotamos ainda $\chi^s := \chi^{\{s\}}$. O restante da notação utilizada é baseada principalmente no livro de Schrijver [Sch03].

A maioria dos conceitos necessários são definidos ao longo do texto. Entretanto, supomos que o leitor tenha alguma familiaridade com conceitos básicos de álgebra linear, teoria e algoritmos em grafos, programação linear, análise de algoritmos e complexidade computacional. Uma revisão desses conceitos pode ser feita também através do livro de Schrijver [Sch03].

Capítulo 1

Funções submodulares

Neste capítulo, apresentamos funções submodulares, uma caracterização alternativa e exemplos dessas funções. Além disso, descrevemos algumas operações úteis que preservam a submodularidade e serão exploradas ao longo do texto. Funções supermodulares e modulares também são apresentadas.

1.1 Funções submodulares, supermodulares e modulares

Seja S um conjunto finito e seja f uma função das partes de S em $\mathbb{R}$. A função f é **submodular** se

$$f(T) + f(U) \geq f(T \cap U) + f(T \cup U) \tag{1.1}$$

para quaisquer partes T e U de S. De forma análoga, f é uma função **supermodular** se satisfaz (1.1) com $\leq$ no lugar de $\geq$. Por fim, f é **modular** se satisfaz (1.1) com igualdade. Segue que se f é uma função submodular, então $-f$ é uma função supermodular. Além disso, temos que uma função é modular se, e somente se, é ao mesmo tempo submodular e supermodular. De forma simplificada, dizemos que f é uma função submodular (supermodular ou modular) **sobre** S, para representar que f é uma função submodular (supermodular ou modular) definida sobre as partes de S.

Uma função f sobre as partes de S é **não-decrescente** se $f(T) \leq f(U)$, sempre que $T \subseteq U$, e é **não-crescente** se $f(T) \geq f(U)$, sempre que $T \subseteq U$. Ademais, se f é não-decrescente ou não-crescente, então dizemos que f é uma função **monótona**.

Funções modulares são muito simples. Qualquer função modular fica completamente determinada se forem especificados um parâmetro μ e um valor associado a cada elemento de S. Mais especificamente, uma função f sobre as partes de S é modular se, e somente se, existem um vetor real w indexado por S e um número real μ tais que

$$f(U) = w(U) + \mu, \text{ para todo } U \subseteq S.$$

Dessa descrição, segue que funções modulares são estruturalmente idênticas a vetores indexados por S.

Por outro lado, funções submodulares e supermodulares não possuem descrições tão simples. A proposição seguinte é uma caracterização alternativa para funções submodulares. Muitas vezes, ela é uma ferramenta para verificar se determinada função é submodular.

Seja f uma função definida sobre as partes de S. Para cada s em S, defina a função f_s como

$$f_s(U) := f(U \cup \{s\}) - f(U), \text{ para cada } U \subseteq S \setminus \{s\}.$$

Proposição 1.1: *Uma função f sobre as partes de S é submodular se, e somente se, f_s é uma função não-crescente, para todo s em S.*

Demonstração: Seja f uma função submodular. Fixe s em S e partes T e U de $S \setminus \{s\}$ tais que $T \subseteq U$. Pela submodularidade de f, vale que

$$f(T \cup \{s\}) + f(U) \geq f(T) + f(U \cup \{s\}),$$

de onde concluímos $f_s(T) = f(T \cup \{s\}) - f(T) \geq f(U \cup \{s\}) - f(U) = f_s(U)$. Segue que f_s é uma função não-crescente, para todo s em S.

Suponha agora que f_s seja uma função não-crescente, para todo s em S. Sejam T e U partes quaisquer de S. Queremos verificar que f satisfaz (1.1) para T e U. Faremos isso por indução em $|T \triangle U|$, em que $T \triangle U$ representa a diferença simétrica $(T \setminus U) \cup (U \setminus T)$ entre T e U.

Se $T \subseteq U$ ou $U \subseteq T$, o resultado vale trivialmente. Podemos supor, portanto, que existe t em $T \setminus U$ e u em $U \setminus T$. Se $|T \triangle U| = 2$, então, como em particular f_t é não-crescente, vale que

$$\begin{aligned} f(T) - f(T \cap U) &= f\big((T \cap U) \cup \{t\}\big) - f(T \cap U) \\ &= f_t(T \cap U) \\ &\geq f_t\big((T \cap U) \cup \{u\}\big) \\ &= f\big((T \cap U) \cup \{t, u\}\big) - f\big((T \cap U) \cup \{u\}\big) \\ &= f(T \cup U) - f(U), \end{aligned}$$

de onde segue o resultado.

Suponha então que $|T \triangle U| \geq 3$. Por simetria, podemos supor que $|T \setminus U| \geq 2$. Seja t em $T \setminus U$. Como $|T \setminus \{t\} \triangle U| < |T \triangle U|$ e $|T \triangle (T \cup U) \setminus \{t\}| < |T \triangle U|$, então, pela hipótese de indução, temos

$$f(U) - f(T \cap U) \geq f\big((T \cup U) \setminus \{t\}\big) - f(T \setminus \{t\}),$$

como também que

$$f\big((T \cup U) \setminus \{t\}\big) - f(T \setminus \{t\}) \geq f(T \cup U) - f(T).$$

Das duas desigualdades acima, segue a validade de (1.1) para T e U. ∎

A proposição 1.1 nos permite concluir que funções submodulares modelam retornos marginais decrescentes, desempenhando um papel análogo ao desempenhado pelas funções côncavas em modelos econômicos, conforme observou Orlin [Orl07]. Por outro lado, Lovász [Lov83] mostrou que funções submodulares têm um comportamento algorítmico mais próximo ao das funções convexas, algo que será explicitado no capítulo 3.

1.2 Exemplos de funções submodulares

Uma grande variedade de funções recorrentes em problemas envolvendo matemática discreta e otimização combinatória são submodulares. Além disso, muitos problemas conhecidos podem ser formulados como um problema de minimizar uma função submodular. Alguns exemplos são apresentados a seguir.

Posto de submatrizes. Seja S o conjunto das colunas de uma matriz A. Para cada parte U de S, defina $r(U)$ como o posto da matriz formada pelas colunas em U. A função r é submodular, o que pode ser verificado com a utilização da proposição 1.1 ou através de técnicas básicas de álgebra linear.

Componentes e subflorestas maximais. Seja (V, E) um grafo. Para cada parte F de E, seja $c(F)$ o número de componentes conexos do subgrafo (V, F). Temos que c é uma função supermodular, e que, portanto, $-c$ é uma função submodular. Tal afirmação também pode ser verificada, por exemplo, através da proposição 1.1. Conseqüentemente, a função c', definida por $c'(F) := |V| - c(F)$, para todo $F \subseteq E$, também é submodular. Para cada parte F de E, o valor $|V| - c(F)$ é o número de arestas em qualquer floresta maximal do subgrafo (V, F).

Matróides. O posto de submatrizes e o número de arcos em florestas maximais de subgrafos geradores são casos particulares de um tipo mais geral de função: a função posto de um matróide. O conceito de matróides foi introduzido por Whitney [Whi35], com o objetivo de descrever de forma abstrata e combinatória propriedades de dependência linear. Seja S um conjunto finito e seja $\mathcal{I}$ uma família de subconjuntos de S. Um par $(S, \mathcal{I})$ é dito um **matróide** se satisfaz:

(i) $\emptyset \in \mathcal{I}$;

(ii) se $I \subseteq J$ e se $J \in \mathcal{I}$, então $I \in \mathcal{I}$;

(iii) se $I, J \in \mathcal{I}$ e são tais que $|I| > |J|$, então existe $i \in I \setminus J$ tal que $J \cup \{i\} \in \mathcal{I}$.

A propriedade (iii) acima é equivalente ao fato de que todos os conjuntos maximais em $\mathcal{I}$ têm o mesmo tamanho. Os conjuntos em $\mathcal{I}$ são ditos **conjuntos independentes** do matróide. A função posto r de um matróide $(S, \mathcal{I})$ é definida como

$$r(U) := \max\{|I| \colon I \subseteq U, I \in \mathcal{I}\}, \text{ para cada } U \subseteq S.$$

Toda função posto r de um matróide é uma função inteira, não-negativa, não-decrescente, submodular e tal que $r(U) \leq |U|$, para toda parte U de S. Mais especificamente, qualquer função r que satisfaça essas propriedades é a função posto do matróide sobre S cujo conjunto de independentes é dado por $\mathcal{I} := \{I \subseteq S : r(I) = |I|\}$. O livro de Schrijver [Sch03] detalha aspectos teóricos e algorítmicos sobre matróides, além de apresentar uma coletânea de referências no assunto.

Edmonds [Edm70] mostrou que o fecho convexo dos vetores de incidência dos conjuntos independentes de um matróide é o politopo em $\mathbb{R}^S$ descrito por:

$$P := \{x \in \mathbb{R}^S : x \geq \mathbf{0},\ x(U) \leq r(U),\ \text{para todo } U \subseteq S\},$$

em que r é a função posto do matróide. O problema de verificar se um determinado vetor x em $\mathbb{R}^S$ está em P é equivalente a minimizar a função submodular $f(U) := r(U) - x(U)$. Conforme veremos no capítulo 4, Cunningham [Cun84] apresentou um algoritmo combinatório para esse propósito, o qual serviu como base para muitos dos demais avanços em algoritmos combinatórios para o problema de minimização de funções submodulares.

Capacidade de cortes. Seja (V, E) um grafo e seja U uma parte de V. Denote por $\delta(U)$ o conjunto das arestas com uma ponta em U e a outra em $V \setminus U$. O conjunto $\delta(U)$ é dito um **corte**. Defina $d(U) := |\delta(U)|$. A função d é submodular. Um simples argumento de contagem é suficiente para verificar esse fato. Podemos também considerar uma função 'capacidade' c que associa um número real não-negativo $c(e)$ a cada aresta e do grafo. Nesse caso, temos que a função f, definida por $f(U) := c(\delta(U)) := \sum_{e \in \delta(U)} c(e)$, para cada $U \subseteq V$, também é submodular.

De forma análoga, se (V, A) é um grafo orientado, denotamos respectivamente por $\delta^{\text{in}}(U)$ e $\delta^{\text{out}}(U)$ o conjunto dos arcos que entram e saem de U, para cada $U \subseteq V$. Temos também que $d^{\text{in}}(U) := |\delta^{\text{in}}(U)|$ e $d^{\text{out}}(U) := |\delta^{\text{out}}(U)|$ são submodulares. Além disso, se c é uma função capacidade sobre os arcos em A, então $c(\delta^{\text{in}}(U))$ e $c(\delta^{\text{out}}(U))$ também são submodulares.

Dados dois vértices distintos s e t em V, podemos definir também a função $f(U) := c\big(\delta^{\text{out}}(U \cup \{s\})\big)$, para todo $U \subseteq V \setminus \{s, t\}$. A função f representa a **capacidade dos st-cortes** no grafo orientado e é também uma função submodular. Ford e Fulkerson [FF56] mostraram que a intensidade de um st-fluxo máximo que respeite as capacidades c é dada justamente pela capacidade mínima de um st-corte, representada por

$$\min\{f(U) : U \subseteq V \setminus \{s, t\}\}.$$

Portanto, os algoritmos para o problema do st-fluxo máximo [EK72, Din70, GT88] também resolvem o problema de minimização de funções submodulares para um caso particular. As idéias para problemas em fluxos máximos também estão bastante presentes nos algoritmos combinatórios para minimização de funções submodulares.

A submodularidade das funções associadas aos cortes em grafos tem sido crucial em

estudos relacionados a aumento de conexidade e caminhos disjuntos em grafos [Fra90, Fra94]. Esse fato está ilustrado no capítulo 7.

Vizinhança em grafos bipartidos. Seja G um grafo bipartido, com bipartição $\{V, W\}$. Seja U uma parte de V. Denotamos por $N(U)$ o conjunto dos vizinhos de U em G e definimos $b(U) := |N(U)|$. A função b é submodular. De fato, note que, para quaisquer partes T e U de V, temos

$$b(T) + b(U) = |N(T) \cup N(U)| + |N(T) \cap N(U)|.$$

Também é claro que $N(T) \cup N(U) = N(T \cup U)$ e que $N(T) \cap N(U) \supseteq N(T \cap U)$. Ou seja, $|N(T) \cup N(U)| + |N(T) \cap N(U)| \geq b(T \cap U) + b(T \cup U)$, de onde segue que b é submodular. Conseqüentemente, a função

$$f(U) := |V \setminus U| + b(U), \text{ para cada } U \subseteq V,$$

também é submodular. O valor mínimo de f é o tamanho de uma cobertura mínima de G e, pelo teorema de Kőnig [Kön31], é também o tamanho de um emparelhamento máximo em G.

Diâmetro de subárvores. Seja (V, E) uma floresta. Para cada subgrafo K de (V, E), defina diâmetro(K) como o comprimento de um caminho máximo em K. Para cada parte F de E, defina

$$f(F) := \sum_K \text{diâmetro}(K),$$

onde K varia sobre os componentes de (V, F). Temos também que a função f é submodular [Sch03].

Probabilidade. Sejam $A_1, \ldots, A_n$ eventos num espaço de probabilidades e seja $S := \{A_1, \ldots, A_n\}$. Para cada parte U de S, defina $f(U)$ como a probabilidade de todos os eventos em U ocorrerem simultaneamente. Então f é supermodular [Lov83].

1.3 Operações sobre funções submodulares

Muitas operações naturais preservam a submodularidade. Existem também construções que, quando aplicadas sobre funções submodulares, produzem novas funções submodulares, com algumas características desejadas. Tais fatos são convenientes quando se deseja estender um teorema, por exemplo. A seguir, listamos algumas dessas operações.

Operações elementares. Se f é uma função submodular, então $f'(U) := f(S \setminus U)$ e $f''(U) := f(U) + \mu$, para qualquer número real μ, também são submodulares. Se f e g são submodulares, então $f + g$ é submodular.

Soma direta. Seja f_i uma função sobre as partes de S_i, para cada $i = 1, \ldots, k$, onde os S_i's são conjuntos mutuamente disjuntos. A **soma direta** de $f_1, \ldots, f_k$ é a função f sobre as partes de $S := S_1 \cup \cdots \cup S_k$, definida por

$$f(U) := f_1(S_1 \cap U) + \cdots + f_k(S_k \cap U), \text{ para cada } U \subseteq S.$$

A soma direta de funções submodulares é uma função submodular.

Monotonicidade. Em algumas situações é conveniente trabalhar com funções submodulares monótonas. Se f é uma função definida sobre as partes de S, então

$$f_{\text{mon}}(U) := \min\{f(T) \colon U \subseteq T\}, \text{ para cada } U \subseteq S, \tag{1.2}$$

é uma função não-decrescente. Ademais, se f é submodular, então f_{mon} é submodular. De forma análoga, pode-se obter uma função submodular não-crescente a partir de uma função submodular f. Basta definir

$$f_{\text{mon-d}}(U) := \min\{f(T) \colon T \subseteq U\}, \text{ para cada } U \subseteq S.$$

Mínimo e máximo entre funções. Nem sempre o mínimo ou o máximo entre duas funções submodulares é uma função submodular. Entretanto, vale o seguinte fato.

Proposição 1.2: *Sejam f e g funções submodulares tais que $f - g$ é monótona. Então $\min(f, g)$ é também uma função submodular.* ∎

Transformação de Dilworth. Seja f uma função definida sobre as partes de um conjunto S. A **transformação de Dilworth de** f (*Dilworth truncation of* f) é a função f^*, também definida sobre as partes de S, dada por:

$$f^*(U) := \begin{cases} 0, & \text{se } U = \emptyset; \\ \min\left\{\sum_{i=1}^{k} f(U_i) \colon \{U_1, \ldots, U_k\} \text{ partição de } U\right\}, & \text{se } U \neq \emptyset. \end{cases} \tag{1.3}$$

Tal construção é devida a Dilworth [Dil44].

Se f é submodular e $f(\emptyset) \geq 0$, então f^* é idêntica a f em todas as partes não-vazias de S. Em geral, podemos estabelecer a seguinte relação entre f^* e f.

Teorema 1.3: *Seja f uma função submodular. Seja $\mathcal{G}$ o conjunto das funções g definidas sobre as partes de S com as seguintes propriedades:*

(a) g é submodular,

(b) $g(\emptyset) = 0$,

(c) $g(U) \leq f(U)$, para toda parte não-vazia U de S.

Então $f^* \in \mathcal{G}$ e $f^* \geq g$, para todo $g \in \mathcal{G}$. ∎

Podemos definir também a **transformação de Dilworth superior de** f (*upper Dilworth truncation of* f), trocando "min" na definição (1.3) por "max". Nesse caso, temos que tal operação preserva a supermodularidade.

Convolução. Sejam f e g funções definidas sobre as partes de um conjunto S. A **convolução de** f **e** g é a função h definida por

$$h(U) := \min\{f(T) + g(U \setminus T) \colon T \subseteq U\}, \text{ para cada } U \subseteq S. \tag{1.4}$$

Pelo teorema da intersecção de matróides [Edm70], temos que se f e g são funções postos de matróides, então h é a função posto de sua intersecção. Como a intersecção de dois matróides nem sempre é um matróide, então a convolução nem sempre preserva a submodularidade. Entretanto, vale a seguinte relação.

Teorema 1.4: *A convolução de uma função submodular e uma função modular é uma função submodular.* ∎

Vale também o seguinte resultado.

Teorema 1.5: *Sejam f uma função submodular e g uma função modular. Suponha $f(\emptyset) = g(\emptyset) = 0$. Seja $\mathcal{H}$ o conjunto das funções h' definidas sobre as partes de S tais que:*

(a) h' é submodular,

(b) $h'(\emptyset) = 0$,

(c) $h' \leq f$ e $h' \leq g$.

Então a convolução de f e g é a função h tal que $h \in \mathcal{H}$ e $h \geq h'$, para todo $h' \in \mathcal{H}$. ∎

Capítulo 2

Polimatróides e método guloso

Apresentamos aqui os polimatróides e os polimatróides estendidos, que estão entre os primeiros modelos poliédricos associados a funções submodulares. A descrição do método guloso para otimização sobre polimatróides, bem como algumas de suas conseqüências, também estão presentes neste capítulo.

Edmonds [Edm70] observou que uma função objetivo linear pode ser otimizada sobre um polimatróide (estendido) através do método guloso. Edmonds utilizou ainda o método guloso para caracterizar tais poliedros. Os pontos extremos de um polimatróide também podem ser caracterizados polinomialmente com o auxílio do método guloso [BCT85, Top82]. Tais resultados foram fortemente explorados no desenvolvimento dos algoritmos combinatórios para minimização de funções submodulares apresentados nos capítulos seguintes.

2.1 Poliedros

Fixe um natural $n > 0$. Um **poliedro** é um subconjunto de $\mathbb{R}^n$ que pode ser descrito por

$$\{x \in \mathbb{R}^n : a_i x \leq b_i,\ i \in M\},$$

em que M é um conjunto finito de índices, cada a_i é um vetor em $\mathbb{R}^n$ e cada b_i é um escalar. Uma inequação do tipo $a_i x \leq b_i$ é dita uma **restrição** do poliedro. Duas restrições são **linearmente independentes** se os vetores a_i e a_j que as definem são linearmente independentes. Se um vetor x satisfaz uma restrição $a_i x \leq b_i$ com igualdade, dizemos que tal restrição é **ativa em** x. Um subconjunto F de um poliedro P é uma **face** de P se existe uma restrição $a_i x \leq b_i$ de P tal que $F = P \cap \{x \in \mathbb{R}^n : a_i x = b_i\}$.

Sejam $x_1, \ldots, x_k$ vetores em $\mathbb{R}^n$ e sejam $\lambda_1, \ldots, \lambda_k$ números reais não-negativos tais que $\sum_{i=1}^k \lambda_i = 1$. O vetor $\sum_{i=1}^k \lambda_i x_i$ é dito uma **combinação convexa** dos vetores $x_1, \ldots, x_k$. O **fecho convexo** dos vetores $x_1, \ldots, x_k$ é o conjunto de todas as combinações convexas desses vetores.

Seja P um poliedro. Um vetor x em P é um **ponto extremo** de P se não existem vetores y

e z em P, ambos diferentes de x, e um escalar λ, com $0 \leq \lambda \leq 1$, tais que $x = \lambda y + (1-\lambda)z$. Pode-se provar que um vetor x em P é um ponto extremo se, e somente se, existem n restrições linearmente independentes ativas em x, dentre as restrições que descrevem P.

Dois pontos extremos x e y de um poliedro P são **adjacentes** se existem, dentre as inequações que descrevem P, $n-1$ restrições linearmente independentes que são simultaneamente ativas em x e em y.

Um **politopo** é o fecho convexo de um número finito de vetores. Pode-se mostrar que um poliedro é um politopo se, e somente se, é limitado. Isto é, um poliedro P é um politopo se, e somente se, existe um escalar não-negativo u tal que $|x_i| \leq u$, $1 \leq i \leq n$, para todo $x = (x_1, \ldots, x_n)$ em P. Vale também que um politopo é o fecho convexo de seus pontos extremos.

Dizemos que os vetores $x_1, \ldots, x_k$ em $\mathbb{R}^n$ são **afim independentes** se não existem escalares $\lambda_1, \ldots, \lambda_k$ tais que $\lambda_1 x_1 + \cdots + \lambda_k x_k = 0$, $\lambda_1 + \cdots + \lambda_k = 0$ e nem todos os λ_i são iguais a zero. O teorema de Carathéodory [Car11] estabelece o seguinte resultado.

Teorema 2.1 (Teorema de Carathéodory): *Seja X um conjunto de vetores em $\mathbb{R}^n$. Se x está no fecho convexo dos vetores em X, então existem vetores afim independentes $x_1, \ldots, x_k$ em X tais que x está no fecho convexo de $x_1, \ldots, x_k$.*

Segue da definição de politopo, bem como do teorema de Carathéodory, que qualquer vetor em um politopo em $\mathbb{R}^n$ pode ser representado como combinação convexa de no máximo $n+1$ de seus pontos extremos.

2.2 Polimatróides e polimatróides estendidos

Seja f uma função submodular. Considere os seguintes poliedros, associados à função f:

$$P(f) := \{x \in \mathbb{R}^S \colon x \geq \mathbf{0},\ x(U) \leq f(U), \text{ para todo } U \subseteq S\};$$
$$EP(f) := \{x \in \mathbb{R}^S \colon x(U) \leq f(U), \text{ para todo } U \subseteq S\}.$$

Observe que $P(f)$ é não-vazio se, e somente se, $f \geq \mathbf{0}$, e que $EP(f)$ é não-vazio se, e somente se, $f(\emptyset) \geq 0$. O poliedro $P(f)$ é chamado **polimatróide de** f, e o poliedro $EP(f)$ é o **polimatróide estendido de** f. Mais geralmente, um poliedro é um polimatróide se é o polimatróide de alguma função submodular. Analogamente, um poliedro é um polimatróide estendido se é o polimatróide estendido de alguma função submodular.

O polimatróide de qualquer função submodular f é limitado, já que $0 \leq x(s) \leq f(\{s\})$, para todo s em S. Portanto, todo polimatróide é um politopo.

O **politopo dos conjuntos independentes de um matróide** é o fecho convexo dos vetores de incidência de seus conjuntos independentes. Conforme mencionamos na seção 1.2, se o matróide estiver definido sobre S, Edmonds [Edm70] mostrou que tal politopo pode

ser descrito por

$$\{x \in \mathbb{R}^S : x \geq \mathbf{0},\ x(U) \leq r(U),\ \text{para todo } U \subseteq S\},$$

onde r é a função posto do matróide. Polimatróides são, portanto, uma generalização desse conceito.

Seja f uma função submodular e seja x um vetor em $EP(f)$. Defina

$$\mathcal{T}(f,x) := \{U \subseteq S : x(U) = f(U)\}, \tag{2.1}$$

como a **família dos conjuntos x-justos em relação a f**. Se U é uma parte de S que está em $\mathcal{T}(f,x)$, dizemos simplesmente que U é **x-justo**.

O seguinte fato é recorrente em demonstrações que utilizam submodularidade.

Teorema 2.2 (União e intersecção de conjuntos justos): *Seja f uma função submodular e seja x um vetor em $EP(f)$. A família dos conjuntos x-justos em relação a f é fechada por união e intersecção.*

Demonstração: Sejam T e U elementos de $\mathcal{T}(f,x)$. Temos

$$\begin{aligned} f(T) + f(U) &\geq f(T \cap U) + f(T \cup U) && (2.2)\\ &\geq x(T \cap U) + x(T \cup U) && (2.3)\\ &= x(T) + x(U) \\ &= f(T) + f(U), && (2.4) \end{aligned}$$

onde (2.2) é devido à submodularidade de f, (2.3) se verifica pois x está em $EP(f)$ e (2.4) vale pois T e U estão em $\mathcal{T}(f,x)$. Segue que todas as relações acima valem com igualdade.

Portanto, $T \cap U$ e $T \cup U$ também estão em $\mathcal{T}(x,f)$. ∎

Seja f uma função submodular e seja w um vetor indexado por S. Defina a função $f|w$ como a convolução de f e w, conforme (1.4). Isto é,

$$(f|w)(U) := \min\{f(T) + w(U \setminus T) : T \subseteq U\},\ \text{ para cada } U \subseteq S.$$

Pelo teorema 1.4, $(f|w)$ é uma função submodular. Também é fácil verificar que

$$EP(f|w) = \{x \in EP(f) : x \leq w\} \quad \text{e} \quad P(f|w) = \{x \in P(f) : x \leq w\}.$$

Segue que se P é um polimatróide (ou polimatróide estendido), então

$$P \cap \{x \in \mathbb{R}^S : x \leq w\},$$

também é um polimatróide (estendido), para qualquer vetor w indexado por S.

Por fim, fazemos algumas considerações sobre o poliedro:

$$P' := \{x \in \mathbb{R}^S : x(U) \leq f(U), \text{ para todo } U \subseteq S,\ U \neq \emptyset\}.$$

Conforme já discutido, temos que se f é uma função submodular tal que $f(\emptyset) < 0$, então seu polimatróide estendido é vazio. Ignorando a condição $x(\emptyset) \leq f(\emptyset)$ do conjunto de restrições que definem $EP(f)$, obtemos exatamente o poliedro P'.

É interessante observar que P' é ainda um polimatróide estendido, mas associado a outra função submodular – a transformação de Dilworth f^* de f, definida em (1.3). Tal afirmação pode ser facilmente verificada a partir das definições de P' e $EP(f^*)$.

2.3 Método guloso

Edmonds [Edm70] observou que uma função objetivo linear pode ser otimizada em tempo polinomial sobre um polimatróide estendido através de um algoritmo guloso. O método guloso para otimização sobre polimatróides é essencialmente o método definido para otimização sobre o politopo dos conjuntos independentes de um matróide.

Quando o método guloso é aplicado sobre o politopo dos conjuntos independentes de um matróide, a otimalidade da solução gulosa obtida segue do lema da dualidade de programação linear. Edmonds percebeu que a característica da função posto que garante sua viabilidade é a submodularidade. Tal observação permitiu a extensão natural do método para otimização sobre polimatróides.

Estamos supondo que uma função submodular f é dada por um **oráculo** tal que, dada uma parte U de S, devolve $f(U)$. Mais considerações sobre a representação de funções submodulares para efeitos algorítmicos são apresentadas no capítulo 3.

Descrição

Se x e y são vetores indexados por um conjunto finito S, denotamos por xy o **produto interno** $\sum_{s \in S} x(s)y(s)$ entre x e y.

Seja f uma função submodular e seja w um vetor indexado por S. Queremos resolver o problema:

$$\max \{wx : x \in EP(f)\}. \tag{2.5}$$

Se $f(\emptyset) < 0$, então $EP(f)$ é vazio e nada temos a fazer. Em particular, podemos supor que $f(\emptyset) = 0$. De fato, se $f(\emptyset) > 0$, redefinindo $f(\emptyset) := 0$ estamos tomando a transformação de Dilworth f^* no lugar de f, conforme (1.3), que também é uma função submodular. Além disso, a alteração pode ser feita já que $EP(f) = EP(f^*)$ nesse caso.

Podemos supor também que $w \geq \mathbf{0}$, uma vez que se algum componente de w fosse negativo, então o problema (2.5) seria ilimitado.

Segue que o problema a ser resolvido é um problema de programação linear que possui

uma solução ótima limitada e cujo problema dual é:

$$\min \left\{ yf : y \in \mathbb{R}_+^{\mathcal{P}(S)}, \sum_{U \subseteq S} y(U)\chi^U = w \right\}. \tag{2.6}$$

Intuitivamente, podemos adotar a seguinte estratégia "gulosa" para resolução do problema (2.5). Começamos caminhando no poliedro $EP(f)$ o máximo possível na direção cujo componente de w é o maior. Essa seria a direção a nos trazer maiores ganhos na função objetivo. Fixado o valor da coordenada nessa direção, repetimos o procedimento, caminhando agora o máximo possível na segunda direção que nos traria melhores resultados – direção cujo componente de w é o segundo maior –, sem sair do poliedro. Seguimos repetindo sucessivamente o procedimento para as demais direções, ordenadas pelos respectivos valores dos componentes de w.

Mais formalmente, essa estratégia gulosa pode ser descrita da seguinte maneira: renomeie os elementos de S como $s_1, \ldots, s_n$, de maneira que $w(s_1) \geq \cdots \geq w(s_n)$. Defina:

$$\begin{aligned} x^*(s_1) &:= \max \{ x(s_1) : x \in EP(f) \} \\ x^*(s_2) &:= \max \{ x(s_2) : x \in EP(f),\ x(s_1) = x^*(s_1) \} \\ &\vdots \\ x^*(s_n) &:= \max \{ x(s_n) : x \in EP(f), x(s_1) = x^*(s_1), \ldots, x(s_{n-1}) = x^*(s_{n-1}) \}. \end{aligned} \tag{2.7}$$

Note que $x^* = \big(x^*(s_1), \ldots, x^*(s_n)\big)$ é um vetor em $EP(f)$, caso exista. Edmonds mostrou que x^* é de fato uma solução ótima do problema (2.5) quando o poliedro $EP(f)$ é um polimatróide estendido. Mais ainda, mostrou que é possível calcular as coordenadas ótimas como em (2.7) através de simples equações, conforme mostramos a seguir. A descrição do método guloso para polimatróides estendidos está abaixo.

Método GULOSO(S, f, w): *Dados um conjunto finito S, uma função submodular f sobre S tal que $f(\emptyset) = 0$ e um vetor w não-negativo indexado por S, devolve um vetor x em $EP(f)$ que é solução de (2.5).*

O método GULOSO(S, f, w) consiste no seguinte: renomeie os elementos de S como $s_1, \ldots, s_n$, de maneira que $w(s_1) \geq \cdots \geq w(s_n)$. Considere as partes de S abaixo:

$$U_i := \{s_1, \ldots, s_i\}, \quad \text{para cada } i = 0, \ldots, n.$$

Devolva o seguinte vetor x:

$$x(s_i) := f(U_i) - f(U_{i-1}), \quad \text{para cada } i = 1, \ldots, n. \tag{2.8}$$

Correção

Mostramos agora que o vetor x devolvido pelo método guloso é de fato uma solução ótima para o problema (2.5).

Teorema 2.3 (Método guloso): *Seja f uma função submodular tal que $f(\emptyset) = 0$. Seja w um vetor não-negativo indexado por S. O vetor x devolvido por* GULOSO(S, f, w) *é solução ótima do problema* (2.5).

Demonstração: Começamos verificando a viabilidade de x em $EP(f)$. Para tanto, vamos mostrar, por indução em $|U|$, que $x(U) \leq f(U)$, para toda parte U de S.

Se $|U| = 0$, então $U = \emptyset$ e o resultado vale, já que $f(\emptyset) = 0$. Suponha, então, $|U| > 0$ e seja k o maior índice tal que s_k está em U. Temos

$$\begin{aligned} x(U) &= x(U \setminus \{s_k\}) + x(\{s_k\}) \\ &\leq f(U \setminus \{s_k\}) + x(\{s_k\}) && (2.9) \\ &= f(U \setminus \{s_k\}) + \big(f(U_k) - f(U_{k-1})\big) && (2.10) \\ &\leq f(U), && (2.11) \end{aligned}$$

onde (2.9) segue da hipótese de indução, (2.10) é devido à definição de x e (2.11) vale pela submodularidade de f. Portanto, x é viável em $EP(f)$.

Para mostrar a otimalidade de x, definimos o seguinte vetor y, indexado pelas partes de S, que, como veremos, é solução ótima do problema dual (2.6).

$$\begin{array}{ll} y(U_i) := w(s_i) - w(s_{i+1}), & \text{para cada } i = 1, \ldots, n-1; \\ y(S) := w(s_n); & \\ y(U) := 0, & \text{para } U \neq U_i,\ i = 1, \ldots, n, \end{array} \tag{2.12}$$

em que $U_i := \{s_1, \ldots, s_i\}$, para cada $i = 1, \ldots, n$.

É fácil ver que y é viável no problema dual (2.6). De fato, o vetor y é claramente não-negativo por definição. Ademais, para cada i, vale que

$$\sum_{U\,:\,s_i \in U} y(U) = \sum_{j \geq i} y(U_j) = w(s_i),$$

de onde segue a viabilidade de y no problema (2.6).

Para concluir, basta observarmos que $wx = yf$. Seguirá, então, pelo lema da dualidade de programação linear, que x e y são ótimos. O vetor y será um certificado da otimalidade de x. Temos

$$\begin{aligned} wx &= \sum_{s \in S} w(s)x(s) = \sum_{i=1}^{n} w(s_i)(f(U_i) - f(U_{i-1})) \\ &= \sum_{i=1}^{n-1} f(U_i)\big(w(s_i) - w(s_{i+1})\big) + f(S)w(s_n) = \sum_{U \subseteq S} y(U)f(U) = yf, \end{aligned}$$

onde a terceira igualdade segue de uma simples reordenação dos termos da soma e também do fato de que $f(\emptyset) = 0$.

Portanto, x e y são respectivamente soluções ótimas dos problemas (2.5) e (2.6). ∎

É evidente que o método pode ser facilmente adaptado para fornecer simultaneamente um par de vetores x e y que são soluções dos problemas duais (2.5) e (2.6), respectivamente. No entanto, para os nossos propósitos, será conveniente que o método devolva apenas um vetor em $EP(f)$.

Observe que se f é uma função não-decrescente, então o vetor x devolvido pelo método guloso é não-negativo. Assim, nesse caso temos que x devolvido por GULOSO(S, f, w) é também solução ótima do problema

$$\max\{wx\colon x \in P(f)\}, \tag{2.13}$$

de maneira que podemos estabelecer o seguinte corolário.

Corolário 2.4: *Seja f uma função submodular não-decrescente tal que $f(\emptyset) = 0$. Seja w um vetor não-negativo indexado por S. O método* GULOSO(S, f, w) *devolve uma solução ótima de* (2.13). ∎

2.4 Algumas caracterizações

Dizemos que um poliedro P é **inteiro** se, para todo vetor racional w, existe uma solução inteira x para o problema $\max\{wx\colon x \in P\}$, supondo que o ótimo exista. Muitos problemas de programação linear inteira obtidos a partir de problemas em otimização combinatória são formulados sobre poliedros inteiros. Para vários desses problemas, temos a validade de uma versão inteira do teorema da dualidade de programação linear e, portanto, uma **relação min-max**. Edmonds e Giles [EG77] definiram o poderoso conceito de total dual integralidade, visando a generalizar relações min-max em otimização combinatória.

Um sistema racional $Ax \leq b$ é dito **totalmente dual integral** (TDI) se o mínimo na equação de dualidade em programação linear

$$\max\{wx\colon Ax \leq b\} = \min\{yb\colon yA = w,\ y \geq 0\}$$

pode ser alcançado por um vetor inteiro, para todo vetor inteiro w para o qual o ótimo exista.

Edmonds e Giles também caracterizaram poliedros inteiros utilizando o conceito de total dual integralidade.

Teorema 2.5: *Se $Ax \leq b$ é um sistema TDI e b é um vetor inteiro, então $Ax \leq b$ determina um poliedro inteiro.* ∎

A correção do método guloso traz conseqüências importantes no que diz respeito a integralidade de polimatróides (estendidos). Note que se f é uma função inteira, então o

vetor x devolvido pelo método guloso, definido em (2.8), é inteiro. Além disso, se w é um vetor inteiro, então (2.12) é uma solução inteira para o problema dual (2.6). Disso segue o seguinte teorema.

Teorema 2.6: *Se f é uma função submodular inteira, então $EP(f)$ é um poliedro inteiro. Ademais, o sistema que define $EP(f)$ é totalmente dual integral.* ∎

Um poliedro P em $\mathbb{R}^n$ é **monótono** se, para cada vetor y em P, todos os vetores x em $\mathbb{R}^n$ com $x \leq y$ estão em P. Mencionamos que a correção do método guloso caracteriza os polimatróides estendidos. O seguinte resultado é devido a Edmonds [Edm70].

Teorema 2.7: *Seja $P \neq \emptyset$ um poliedro monótono em $\mathbb{R}^n$. Para todo vetor não-negativo w em $\mathbb{Q}^n$, é possível maximizar wx sobre P através do método guloso se, e somente se, P é um polimatróide estendido.* ∎

No mesmo trabalho de Edmonds, um resultado equivalente utiliza o método guloso para caracterizar polimatróides dentre os poliedros em $\mathbb{R}^n_+$.

O método guloso também nos permite concluir que existe uma correspondência biunívoca entre polimatróides estendidos e funções submodulares f que satisfazem $f(\emptyset) = 0$. Dada uma função submodular f com $f(\emptyset) = 0$, segue da forma como as soluções ótimas são definidas em (2.8) e (2.12) que

$$f(U) = \max\{x(U) \colon x \in EP(f)\}, \text{ para todo } U \subseteq S.$$

Para verificar isso, basta tomar w como o vetor característico de U e aplicar o método guloso. Note que essa correspondência relaciona polimatróides estendidos inteiros a funções submodulares inteiras.

Existe uma correspondência similar entre polimatróides e funções submodulares não-decrescentes f, com $f(\emptyset) = 0$. Isto é, se f é uma função submodular não-decrescente tal que $f(\emptyset) = 0$, então

$$f(U) = \max\{x(U) \colon x \in P(f)\}, \text{ para todo } U \subseteq S. \tag{2.14}$$

Ademais, se f é uma função submodular não-negativa qualquer, é fácil verificar que $P(f) = P(\overline{f})$, em que $\overline{f}$ é dada por:

$$\overline{f}(U) := \begin{cases} 0, & \text{se } U = \emptyset; \\ \min\{f(T) \colon U \subseteq T\}, & \text{se } U \neq \emptyset,\ U \subseteq S. \end{cases}$$

Observe que $\overline{f}$ é claramente não-decrescente. Ademais, como f é não-negativa, temos que $\overline{f}$ é a transformação de Dilworth da função f_{mon}, como definido em (1.2) e em (1.3). Disso segue que $\overline{f}$ é submodular.

Por (2.14) e como $P(\overline{f}) = P(f)$, então, para qualquer função submodular não-negativa f

definida sobre as partes de S, temos

$$\overline{f}(U) = \max\{x(U) \colon x \in P(f)\}, \text{ para todo } U \subseteq S.$$

2.5 Intersecção de polimatróides

Alguns resultados obtidos para polimatróides também se estendem para a intersecção de dois polimatróides, o que aumenta o interesse nesses poliedros como ferramentas em otimização combinatória. Um fenômeno semelhante ocorre entre matróides e sua intersecção.

Com auxílio do método guloso, Edmonds [Edm70] obteve o seguinte teorema, que enunciamos aqui utilizando o conceito de total dual integralidade, apresentado por Edmonds e Giles [EG77], definido na seção anterior.

Teorema 2.8: *Sejam f e g funções submodulares. Suponha $f(\emptyset) = g(\emptyset) = 0$. O sistema*

$$x(U) \leq \min\{f(U), g(U)\}, \text{ para todo } U \subseteq S,$$

que descreve $EP(f) \cap EP(g)$, é totalmente dual integral. Ademais, se f e g são funções inteiras, então $EP(f) \cap EP(g)$ é um poliedro inteiro. ∎

Uma conseqüência da demonstração de Edmonds para o teorema acima é a seguinte relação min-max, que generaliza o teorema da intersecção de matróides.

Teorema 2.9: *Sejam f e g funções submodulares tais que $f(\emptyset) = g(\emptyset) = 0$. Então*

$$\max\{x(U) \colon x \in EP(f) \cap EP(g)\} = \min_{T \subseteq U}\{f(T) + g(U \setminus T)\}, \tag{2.15}$$

para cada parte U de S. ∎

O problema de encontrar um vetor de peso máximo sobre a intersecção de dois polimatróides (estendidos) também pode ser resolvido em tempo fortemente polinomial. Entretanto, há até pouco tempo, os algoritmos para esse propósito eram baseados na existência de subrotinas capazes de minimizar funções submodulares [Fra84b, Sch03].

Mais recentemente, as idéias presentes nos algoritmos polinomiais combinatórios para minimização de funções submodulares permitiram o desenvolvimento de algoritmos polinomiais independentes para problemas em fluxos submodulares, dos quais a otimização sobre a intersecção de polimatróides é um caso particular [FI00a, FIM02, IMS05]. O capítulo 7 traz uma breve introdução sobre fluxos submodulares.

2.6 Ordens e pontos extremos

Ordens parciais e lineares

Uma **ordem parcial** $\prec$ sobre um conjunto S é uma relação binária definida sobre S que satisfaz às seguintes propriedades:

(i) Reflexiva: $s \prec s$, para todo s em S;

(ii) Anti-simétrica: $s \prec t$ e $t \prec s$ implicam $s = t$, para quaisquer s, t em S;

(iii) Transitiva: $s \prec t$ e $t \prec u$ implicam $s \prec u$, para quaisquer s, t, u em S.

Dizemos que uma ordem parcial $\prec$ sobre S é uma **ordem linear** se a relação $\prec$ estiver definida para todo par s e t de elementos em S. Ou seja, a ordem parcial $\prec$ é uma ordem linear se, e somente se, $s \prec t$ ou $t \prec s$ para todo par s e t de elementos em S.

Se $\prec$ é uma ordem parcial de S e s é um elemento de S, denotamos por $[s]_\prec$ o conjunto de todos os elementos s' tais que $s' \prec s$. Isto é,

$$[s]_\prec := \{s' \in S : s' \prec s\}. \tag{2.16}$$

Para cada s em S, o conjunto $[s]_\prec$ é dito um **ideal** de $\prec$.

Caracterização de pontos extremos de polimatróides

Seja f uma função submodular e seja $\prec$ uma ordem linear de S. Suponha ainda que $S = \{s_1, \ldots, s_n\}$, de maneira que $s_1 \prec \ldots \prec s_n$.

Considere um vetor w não-negativo, indexado por S e tal que $w(s_1) \geq \ldots \geq w(s_n)$. Ao invocarmos GULOSO($S$, f, w), obtemos um vetor x que está no polimatróide estendido $EP(f)$. Note que esse mesmo vetor x pode ser obtido através de GULOSO(S, f, w'), para todo vetor w' não-negativo tal que $w'(s_1) \geq \ldots \geq w'(s_n)$. Assim, vemos que o papel de w no método guloso é essencialmente fornecer uma ordem linear $\prec$ dos elementos de S a partir da qual o vetor x será definido.

Podemos então apresentar a seguinte descrição alternativa do método guloso, que recebe uma ordem linear $\prec$ de S ao invés de um vetor w.

> **Método** GULOSO(S, f, $\prec$): *Dados um conjunto finito S, uma função submodular f sobre S tal que $f(\emptyset) = 0$ e uma ordem linear $\prec$ de S, devolve um vetor $x^\prec$ em $EP(f)$.*

A definição do vetor $x^\prec$, que é a saída do método guloso, continua exatamente como em (2.8). Entretanto, começamos o método redefinindo os elementos do conjunto base S como $s_1, \ldots, s_n$ de maneira que $s_1 \prec \ldots \prec s_n$. Em particular, utilizando a notação de ordens lineares, temos que o vetor $x^\prec$ é definido pelo método guloso da seguinte maneira:

$$x^\prec(s_i) := f\big([s_i]_\prec\big) - f\big([s_{i-1}]_\prec\big), \text{ para cada } i = 1, \ldots, n. \tag{2.17}$$

Dizemos que o vetor $x^\prec$ devolvido por GULOSO(S, f, $\prec$) é **gerado pela ordem** $\prec$ **atra-**

vés do método guloso. Dizemos ainda simplesmente que $x^{\prec}$ é **gerado pelo método guloso**. Observe também que quando escrevemos $x^{\prec}$ estamos nos referindo ao vetor de $EP(f)$ devolvido por GULOSO$(S, f, \prec)$.

O seguinte teorema, relacionado ao método guloso e que trata de pontos extremos de polimatróides estendidos, é bastante referenciado nas descrições e demonstrações dos algoritmos para minimização de funções submodulares.

Teorema 2.10: *Seja f uma função submodular e seja $\prec$ uma ordem linear de S. Seja $x^{\prec}$ o vetor gerado por $\prec$ através do método guloso. Então,*

(i) Para cada elemento s de S, o ideal $[s]_{\prec}$ de $\prec$ é $x^{\prec}$-justo. Isto é,

$$x^{\prec}\big([s]_{\prec}\big) = f\big([s]_{\prec}\big), \quad \textit{para cada } s \in S;$$

(ii) O vetor $x^{\prec}$ é um ponto extremo de $EP(f)$.

Demonstração: A parte (i) do teorema segue diretamente da maneira como o vetor $x^{\prec}$ é definido em (2.17) pelo método guloso. Para a parte (ii), basta observarmos que

$$x^{\prec}\big([s]_{\prec}\big) \leq f\big([s]_{\prec}\big), \quad \text{para cada } s \in S,$$

formam um conjunto de $|S|$ restrições de $EP(f)$ linearmente independentes que são ativas em $x^{\prec}$. Como $x^{\prec}$ está em $EP(f)$ e como $EP(f)$ é um poliedro em $\mathbb{R}^S$, segue que $x^{\prec}$ é um ponto extremo de EP_f. ∎

Concluímos pelo teorema 2.10 que todos os vetores gerados pelo método guloso são pontos extremos de polimatróides estendidos. Na verdade, temos que qualquer ponto extremo de determinado polimatróide estendido pode ser gerado pelo método guloso. Apresentamos a seguir alguns resultados de Bixby, Cunningham e Topkis [BCT85] que descrevem quais são as ordens lineares de S que geram certo ponto extremo de $EP(f)$.

Seja x um ponto extremo de $EP(f)$. Para cada s em S, defina

$$T^x(s) := \bigcap \big\{T \in \mathcal{T}(f, x) \colon s \in T\big\},$$

como o subconjunto x-justo minimal de S que contém s. Bixby, Cunningham e Topkis [BCT85] estabeleceram o seguinte resultado.

Teorema 2.11 (Ordem parcial associada ao ponto extremo): *Se x é um ponto extremo de $EP(f)$, então a relação binária $\prec_x$ sobre S, definida por*

$$s \prec_x t \iff s \in T^x(t), \tag{2.18}$$

*para todo s, t em S, é uma ordem parcial. Dizemos que $\prec_x$ é a **ordem parcial de** x **em relação a** f.*

Demonstração: Como x é um ponto extremo de $EP(f)$, x é solução única de um sistema de $n := |S|$ equações linearmente independentes, da forma

$$x(U_i) = f(U_i), \text{ para cada } i = 1, \ldots, n. \tag{2.19}$$

Segue que cada elemento s de S está em algum U_i. De fato, caso contrário, poderíamos alterar a componente $x(s)$ de forma a obtermos outra solução para o sistema (2.19). Portanto, cada s está em algum conjunto x-justo, de maneira que $T^x(s)$ é não-vazio, para todo s em S. Disso segue a validade da propriedade reflexiva para $\prec_x$.

De forma análoga, para cada par s, t de elementos em S, com $s \neq t$, deve existir um U_i em (2.19) que contém um deles e não contém o outro. Caso contrário, poderíamos alterar as componentes $x(s)$ e $x(t)$ de forma a obtermos uma outra solução para o sistema (2.19), o que seria uma nova contradição. Assim, $s \in T^x(t)$ e $t \in T^x(s)$ se, e somente se, $s = t$, de onde segue a validade da propriedade anti-simétrica para $\prec_x$.

Por fim, se s está em $T^x(t)$, então s está em todo conjunto x-justo que contém t. Logo, se t está em $T^x(u)$, que é um conjunto x-justo, então s também está nesse conjunto, de maneira que vale também a propriedade transitiva para $\prec_x$.

Segue que a relação binária $\prec_x$, definida por (2.18), é uma ordem parcial sobre S. ∎

Uma ordem parcial $\prec_p$ sobre S e uma ordem linear $\prec_l$ desse mesmo conjunto são ditas **compatíveis** se, dados s, t distintos em S, se $s \prec_p t$, então $s \prec_l t$. O próximo resultado estabelece quais são as ordens lineares de S que geram determinado ponto extremo de $EP(f)$.

Teorema 2.12: *Seja x um ponto extremo de $EP(f)$, $\prec_x$ sua ordem parcial e $\prec$ uma ordem linear de S. O vetor x é gerado por $\prec$ através do método guloso se, e somente se, $\prec_x$ e $\prec$ são compatíveis.*

Demonstração: Suponha que a ordem linear $\prec$ de S gere x. Pelo teorema 2.10, o ideal $[s]_\prec$ é x-justo, para cada elemento s em S. Logo, $T^x(s)$ está contido em $[s]_\prec$, para cada s em S. Segue que se t não está em $[s]_\prec$, e, portanto, se $t \not\prec s$, então $t \not\prec_x s$. Logo, $\prec_x$ e $\prec$ são compatíveis.

Suponha agora que a ordem linear $\prec$ de S seja compatível com a ordem parcial $\prec_x$. Seja s um elemento de S e considere o ideal $[s]_\prec$. Para todo t em $[s]_\prec$, devemos ter $T^x(t)$ como uma parte de $[s]_\prec$. De fato, como as ordens são compatíveis, qualquer elemento em $T^x(t)$ deve preceder t na ordem linear $\prec$. Concluímos, assim, que $[s]_\prec$ é a união de conjunto x-justos e, portanto, que $[s]_\prec$ é também um conjunto x-justo. Segue que x é solução do sistema

$$x\big([s]_\prec\big) = f\big([s]_\prec\big), \text{ para cada } s \in S.$$

Como o vetor gerado pelo método guloso através de $\prec$ também é solução do sistema acima e tal sistema admite solução única, segue que x é gerado por $\prec$ através do método guloso. ∎

Bixby, Cunningham e Topkis descreveram no mesmo trabalho [BCT85] um algoritmo que devolve uma representação da ordem parcial associada a um ponto extremo x de um polimatróide estendido $EP(f)$ em tempo polinomial no tamanho do conjunto base S. A partir desse algoritmo, é possível descrever todas as ordens lineares de S que geram determinado ponto extremo de $EP(f)$.

Por fim, definimos o poliedro

$$B(f) := \{x \in \mathbb{R}^S : x(S) = f(S),\ x(U) \leq f(U),\ \text{para todo } U \subseteq S\},$$

que é uma face de $EP(f)$. O poliedro $B(f)$ é o **polimatróide das bases** de f.

Concluímos com o teorema 2.12 que todo ponto extremo de $EP(f)$ pode ser gerado pelo método guloso. Assim, segue do teorema 2.10 que todo ponto extremo x de f satisfaz $x(S) = f(S)$, de maneira que $B(f)$ é a face de $EP(f)$ que contém todos os seus pontos extremos. Note também que o polimatróide das bases é um politopo, uma vez que $f(\{s\}) \geq x(s) = x(S) - x(S \setminus \{s\}) \geq f(S) - f(S \setminus \{s\})$, para todo s em S.

O politopo $B(f)$ é bastante explorado como ferramenta nos algoritmos combinatórios para minimização de funções submodulares.

2.7 Adjacência

Seja $\prec$ uma ordem linear de $S = \{s_1, \ldots, s_n\}$ e sejam $s := s_k$ e $t := s_{k+1}$ elementos de S, de maneira que $s_1 \prec \ldots \prec s_k = s \prec s_{k+1} = t \prec \ldots \prec s_n$. Seja $\prec'$ a ordem linear que se obtém trocando s e t de posição em $\prec$. Considere ainda uma função submodular f sobre S. Vale o seguinte resultado.

Teorema 2.13: *O ponto extremo $x^{\prec'}$ de $EP(f)$, gerado por $\prec'$, satisfaz:*

$$x^{\prec'} = x^{\prec} + \alpha(\chi^t - \chi^s), \tag{2.20}$$

em que $\alpha = f\big([t]_{\prec} \setminus \{s\}\big) - x^{\prec}\big([t]_{\prec} \setminus \{s\}\big)$.

Demonstração: Pela maneira como $x^{\prec}$ e $x^{\prec'}$ são definidos em (2.17) pelo método guloso, segue que os dois pontos extremos devem diferir somente nos componentes associados a t e a s. Além disso, concluímos

$$\begin{aligned}
x^{\prec}(s) - x^{\prec'}(s) &= x^{\prec'}(t) - x^{\prec}(t) \\
&= \Big(f\big([s_{k+1}]_{\prec} \setminus \{s_k\}\big) - f\big([s_{k-1}]_{\prec}\big)\Big) - \Big(f\big([s_{k+1}]_{\prec}\big) - f\big([s_k]_{\prec}\big)\Big) \\
&= f\big([s_{k+1}]_{\prec} \setminus \{s_k\}\big) - \Big(f\big([s_{k-1}]_{\prec}\big) + \Big(f\big([s_{k+1}]_{\prec}\big) - f\big([s_k]_{\prec}\big)\Big)\Big) \\
&= f\big([s_{k+1}]_{\prec} \setminus \{s_k\}\big) - \Big(x^{\prec}\big([s_{k-1}]_{\prec}\big) + x^{\prec}(s_{k+1})\Big) \\
&= f\big([s_{k+1}]_{\prec} \setminus \{s_k\}\big) - x^{\prec}\big([s_{k+1}]_{\prec} \setminus \{s_k\}\big) \\
&= f\big([t]_{\prec} \setminus \{s\}\big) - x^{\prec}\big([t]_{\prec} \setminus \{s\}\big),
\end{aligned}$$

de onde segue o resultado. ∎

Quando $\alpha > 0$ no teorema 2.13, temos que $x^{\prec}$ e $x^{\prec'}$ são pontos extremos adjacentes em $B(f)$. Topkis [Top82] caracterizou adjacência em polimatróides e mostrou que todos os pontos extremos adjacentes a $x^{\prec}$ no polimatróide das bases $B(f)$ são obtidos como no teorema 2.13, para algum par de elementos s e t de S. Operações envolvendo pontos extremos adjacentes no polimatróide das bases são recorrentes nos algoritmos combinatórios para minimização de funções submodulares.

Capítulo 3

Minimização de funções submodulares

Muitos problemas em otimização combinatória podem ser formulados como um problema de minimizar uma função submodular sobre um conjunto apropriado. Alguns exemplos desse tipo foram apresentados na seção 1.2. Além disso, os algoritmos conhecidos para otimização sobre a intersecção de polimatróides, bem como sobre outros modelos poliédricos com restrições submodulares, são fortemente baseados em idéias associadas a minimização de funções submodulares [Fra84b, FI00b, FI00a, FIM02, IMS05]. Assim, um problema chave em teoria algorítmica de funções submodulares é o de minimizar tais funções.

Neste capítulo, definimos o problema de minimização de funções submodulares, que é também a questão central discutida neste trabalho. Ademais, apresentamos uma relação entre submodularidade e convexidade que foi utilizada por Grötschel, Lovász e Schrijver [GLS81, GLS88] para estabelecer a polinomialidade do problema. Os algoritmos combinatórios para minimizar funções submodulares são discutidos a partir do próximo capítulo.

3.1 Definição do problema e representação das funções

O problema de minimização de funções submodulares é definido como:

Problema MIN-FUNÇÕES-SUBMODULARES(S,f): *Dada uma função submodular f sobre S tal que $f(\emptyset) = 0$, encontrar uma parte W de S tal que $f(W) \leq f(U)$, para todo $U \subseteq S$.*

O conjunto W é dito um **minimizador** da função submodular f. Em geral, supõe-se também que a função a ser minimizada satisfaz $f(\emptyset) = 0$. Note que se $f(\emptyset) \neq 0$, a função f' definida por $f'(U) := f(U) - f(\emptyset)$ é também uma função submodular. Além disso, os problemas de minimizar as funções f e f' são equivalentes. Todos os algoritmos

que descrevemos ao longo do texto, bem como o método guloso apresentado no capítulo anterior, fazem a suposição de que a função submodular f satisfaz $f(\emptyset) = 0$. Desse modo, assumiremos, a partir deste ponto e sem perda de generalidade, que qualquer função submodular f satisfaz $f(\emptyset) = 0$.

Para tratar o problema algorítmico de minimização de funções submodulares, supomos que uma função submodular f é dada por um **oráculo** tal que, dada uma parte U de S, devolve $f(U)$. Um oráculo pode ser imaginado como uma espécie de rotina "caixa-preta" que calcula o valor da função para determinada parte do conjunto base S.

Supomos também que o oráculo devolve $f(U)$ em tempo $O(\gamma)$, em que γ é alguma função de $n := |S|$. Em particular, para muitas das funções submodulares conhecidas, podemos imaginar implementações dos oráculos que consumam tempo polinomial no tamanho das estruturas envolvidas no problema. Por exemplo, para as funções capacidades de cortes (seção 1.2), é possível construir um oráculo que devolva o valor da função em certa parte de S em tempo proporcional à soma do número de vértices com o número de arestas (ou arcos) do grafo considerado. Observe que se f fosse dada explicitamente através de uma tabela contendo todas as partes de S, encontrar um minimizador de f em tempo polinomial no tamanho da entrada seria um problema trivial.

Assim, dizemos que um algoritmo para minimizar uma função submodular é **fortemente polinomial** se for capaz de realizar a tarefa com consumo de tempo limitado por uma função polinomial em n e em γ. Em outras palavras, exigimos que o número de chamadas ao oráculo, bem como o número de outras operações efetuadas pelo algoritmo seja polinomial no tamanho do conjunto base da função submodular a ser minimizada. Um algoritmo será dito **pseudo-polinomial** quando seu consumo de tempo depende também polinomialmente do valor $M := \max\{|f(U)| : U \subseteq S\}$, além de n e de γ. Ademais, se o consumo de tempo do algoritmo puder ser limitado por uma função polinomial em n, γ e $\log M$, então tal algoritmo será dito **fracamente polinomial**. De forma análoga, referimos-nos aos tempos consumidos pelos algoritmos como fortemente, fracamente e pseudo-polinomial.

Retomando agora o método guloso detalhado no capítulo 2, podemos estabelecer os seguintes resultados, no que se refere ao seu consumo de tempo.

Teorema 3.1: *Dados uma função submodular f e um vetor racional w indexado por S, um vetor x em $EP(f)$ que maximiza wx pode ser encontrado em tempo fortemente polinomial. Ademais, se f é também uma função não-decrescente, então wx pode ser maximizado sobre $P(f)$ em tempo fortemente polinomial.* ∎

Teorema 3.2: *O método* GULOSO*(S, f, $\prec$) devolve um vetor em $B(f)$ em tempo fortemente polinomial $O(n\gamma)$.* ∎

Para encerrar a seção, fazemos uma observação sobre o problema de maximizar uma função submodular. Apesar de parecer em princípio bastante análogo ao problema de minimização, tal problema se revela muito mais difícil. Note que ele contém como caso particular um problema **NP**-completo, que é o problema de encontrar um corte de capa-

cidade máxima em um grafo [Kar72]. Uma revisão sobre conceitos de complexidade de algoritmos pode ser feita através do capítulo 4 do livro de Schrijver [Sch03].

3.2 Submodularidade e convexidade

Seja S um conjunto finito. Qualquer vetor não-negativo e não-nulo w indexado pelos elementos de S pode ser escrito de forma única como:

$$w = \lambda_1 \chi^{U_1} + \cdots + \lambda_k \chi^{U_k}, \tag{3.1}$$

em que $\lambda_i > 0$, para todo i, e $\emptyset \neq U_1 \subset \cdots \subset U_k \subseteq S$.

Seja f uma função sobre as partes de S. Seja $w = \lambda_1 \chi^{U_1} + \cdots + \lambda_k \chi^{U^k}$ um vetor não-negativo e não-nulo indexado por S, em que w, $\lambda_1, \ldots, \lambda_k$ e $U_1, \ldots, U_k$ satisfazem (3.1). Defina

$$\hat{f}(w) := \lambda_1 f(U_1) + \cdots + \lambda_k f(U_k), \tag{3.2}$$

bem como $\hat{f}(\mathbf{0}) = 0$.

É fácil ver que $\hat{f}(\chi^U) = f(U)$, para toda parte não-vazia U de S. Desse modo, a função $\hat{f}$ pode ser considerada uma extensão de f, originalmente definida apenas sobre vetores de incidência de partes de S, para todos os vetores não-negativos indexados por S.

Além disso, temos que se f é submodular, então

$$\hat{f}(w) = \max\{wx \colon x \in EP(f)\}, \tag{3.3}$$

para todo vetor não-negativo w indexado por S.

Para verificar a relação acima, começamos escrevendo o problema dual de

$$\max\{wx \colon x \in EP(f)\},$$

como feito anteriormente em (2.6):

$$\min\left\{yf \colon y \in \mathbb{R}_+^{\mathcal{P}(S)}, \sum_{U \subseteq S} y(U)\chi^U = w\right\}. \tag{3.4}$$

Utilizando o método guloso, podemos obter uma solução ótima y^* (como em (2.12)) para o problema (3.4) em que as partes U de S tais que $y^*(U) > 0$ formam uma família $\{U_1, \ldots, U_k\}$ satisfazendo $\emptyset \neq U_1 \subset \ldots \subset U_k \subseteq S$.

Mas então, pela viabilidade de y^* no problema (3.4), temos

$$y^*(U_1)\chi^{U_1} + \cdots + y^*(U_k)\chi^{U_k} = w. \tag{3.5}$$

Ademais, pela definição (3.2) de $\hat{f}$, pela relação (3.5) e pela maneira como a solução

ótima y^* do problema (3.4) foi definida, vale que

$$\hat{f}(w) = y^*(U_1)f(U_1) + \cdots + y^*(U_k)f(U_k) = y^*f = \max\{wx\colon x \in EP(f)\},$$

de onde concluímos a relação (3.3).

Uma função real f é **convexa** se satisfaz, para quaisquer reais a e b,

$$f\big(\lambda a + (1-\lambda)b\big) \leq \lambda f(a) + (1-\lambda)f(b), \text{ para todo } 0 \leq \lambda \leq 1. \tag{3.6}$$

Ademais, uma função real é **côncava** se satisfaz (3.6) com $\geq$ no lugar de $\leq$. O próximo teorema, devido a Lovász [Lov83], é uma conexão importante entre convexidade e submodularidade.

Teorema 3.3: *Se f é uma função definida sobre as partes de um conjunto finito S e $\hat{f}$ é a extensão de f aos vetores não-negativos indexados por S, então $\hat{f}$ é convexa (côncava) se, e somente se, f é submodular (supermodular).*

Demonstração: Por (3.3), se f é uma função submodular, então $\hat{f}$ pode ser definida através de um programa linear. Disso segue que $\hat{f}$ é uma função convexa.

Por outro lado, se $\hat{f}$ é uma função convexa, então

$$\hat{f}(\chi^U + \chi^T) \leq \hat{f}(\chi^U) + \hat{f}(\chi^T) = f(U) + f(T).$$

Além disso, pela definição (3.2) da função $\hat{f}$, dadas partes U e T quaisquer de S, vale que

$$\hat{f}(\chi^U + \chi^T) = \hat{f}(\chi^{U\cap T}) + \hat{f}(\chi^{U\cup T}) = f(U\cap T) + f(U\cup T).$$

Portanto, f é uma função submodular. ∎

Uma outra relação entre submodularidade e convexidade está no próximo teorema, devido a Frank [Fra82].

Teorema 3.4 (*Frank's discrete sandwich theorem*): *Sejam f uma função submodular e g uma função supermodular tais que $g \leq f$. Então existe uma função modular h tal que $g \leq h \leq f$. Ademais, se f e g são funções inteiras, a função h pode ser escolhida inteira.* ∎

O teorema 3.4 é análogo ao *'continuous sandwich theorem'*, que estabelece a existência de uma função linear entre uma função convexa e outra côncava. Sua demonstração pode ser feita utilizando-se a extensão convexa de uma função submodular, como definido acima, ou através de resultados sobre intersecção de polimatróides [Sch03].

3.3 Polinomialidade do problema

Em seu conhecido trabalho, Grötschel, Lovász e Schrijver [GLS81, GLS88] estabeleceram a equivalência entre separação e otimização sobre poliedros a partir do uso do método dos elipsóides.

Seja f uma função submodular tal que $f(\emptyset) = 0$. O problema da separação que consiste em decidir se $\mathbf{0}$ está no polimatróide estendido $EP(f^{\mu})$, em que f^{μ} é a função submodular obtida quando subtraímos de f uma constante $\mu \leq 0$, é equivalente a determinar se o mínimo da função submodular f é pelo menos μ. Como o problema de otimização sobre polimatróides pode ser resolvido em tempo polinomial através do método guloso, a equivalência entre separação e otimização garante que o problema descrito acima também pode ser resolvido em tempo polinomial. Além disso, o valor máximo μ^* tal que $\mathbf{0}$ está em $EP(f^{\mu^*})$ é justamente o mínimo da função f. Assim, se f assume valores racionais, um minimizador para f pode ser encontrado utilizando essa estratégia combinada com uma busca binária.

Essas idéias estabelecem a polinomialidade do problema de minimização de funções submodulares e originam um algoritmo fracamente polinomial, cujo consumo de tempo depende polinomialmente não só de n e de γ, como também de $\log M$, em que $M := \max\{|f(U)| : U \subseteq S\}$.

Ainda baseados no poderoso arcabouço que desenvolveram, Grötschel, Lovász e Schrijver [GLS88] mostraram que o problema de minimização de funções submodulares pode ser resolvido em tempo fortemente polinomial. Para tanto, utilizaram-se da extensão convexa $\hat{f}$ de f, conforme definido em (3.2), bem como da seguinte relação:

$$\min\{\hat{f}(x) : x \in [0,1]^S\} = \min\{f(U) : U \subseteq S\}. \tag{3.7}$$

Observe que uma das desigualdades em (3.7) é direta. Com efeito, como $f(U) = \hat{f}(\chi^U)$, para toda parte U de S, então

$$\min\{\hat{f}(x) : x \in [0,1]^S\} \leq \min\{f(U) : U \subseteq S\}.$$

Por outro lado, suponha que x^* minimize $\hat{f}$ em $[0,1]^S$, com $x^* = \lambda_1\chi^{U_1} + \cdots + \lambda_k\chi^{U_k}$, como em (3.1). Tomando $\mu := \min\{f(U) : U \subseteq S\}$, temos

$$\hat{f}(x^*) = \lambda_1 f(U_1) + \cdots + \lambda_k f(U_k) \geq (\lambda_1 + \cdots + \lambda_k)\mu \geq \mu.$$

Observe que $\mu \leq 0$, uma vez que $f(\emptyset) = 0$ por hipótese, e que $0 \leq \lambda_1 + \cdots + \lambda_k \leq 1$, já que x^* pertence ao cubo $[0,1]^S$. Disso segue a igualdade (3.7).

Note também que se $x^* = \lambda_1\chi^{U_1} + \cdots + \lambda_k\chi^{U_k}$ minimiza $\hat{f}$ em $[0,1]^S$, então cada U_i minimiza f. Assim, minimizar f sobre as partes de S e encontrar uma parte U de S que atinja o valor mínimo de f é, de certa forma, equivalente a minimizar $\hat{f}$ sobre $[0,1]^S$.

Se f é uma função submodular, a relação (3.3), juntamente com o fato de podermos otimizar sobre polimatróides estendidos em tempo polinomial, garante que o valor da

função $\hat{f}$ pode ser obtido em tempo polinomial, para todo vetor racional não-negativo pertencente ao cubo $[0,1]^S$. Além disso, o teorema 3.3 nos diz que $\hat{f}$ é uma função convexa nesse caso. Esses resultados permitiram a Grötschel, Lovász e Schrijver [GLS88] utilizar seu arcabouço baseado no método dos elipsóides para desenvolver o primeiro algoritmo fortemente polinomial para minimizar funções submodulares. Segue, portanto, o seguinte teorema.

Teorema 3.5 (GLS): *O problema* MIN-FUNÇÕES-SUBMODULARES(S, f) *pode ser resolvido em tempo fortemente polinomial.* ∎

O algoritmo de Grötschel, Lovász e Schrijver foi bastante importante pois permitiu classificar o problema de minimização de funções submodulares com relação à sua complexidade. Entretanto, tal algoritmo não é plenamente satisfatório: além de ser pouco prático, uma vez que se utiliza de todo o arcabouço pesado de divisão, arredondamento e aproximação proveniente do método dos elipsóides, revela muito pouco sobre a estrutura combinatória do problema. Em função disso, encontrar um algoritmo polinomial puramente combinatório foi considerado por muitos como o problema em aberto mais desafiador na área [GLS81, Lov83, Fra84b] e que só pôde ser resolvido cerca de 20 anos após o estabelecimento da polinomialidade da questão. As idéias combinatórias associadas a minimização de funções submodulares, bem como os principais algoritmos combinatórios, estão descritos nos próximos capítulos.

Capítulo 4

Método de Cunningham

Grötschel, Lovász e Schrijver estabeleceram a polinomialidade do problema de minimização de funções submodulares, utilizando seu arcabouço baseado no método dos elipsóides, conforme apresentamos no capítulo 3. Entretanto, tal algoritmo não é satisfatório do ponto de vista prático. Faltava também explorar as propriedades combinatórias do problema.

Pode-se dizer que as primeiras idéias combinatórias associadas à minimização de funções submodulares estão presentes no trabalho de Cunningham [Cun84]. Nesse trabalho, Cunningham apresenta um algoritmo fortemente polinomial para o problema da pertinência no politopo dos conjuntos independentes de um matróide. Esse algoritmo, por sua vez, parece uma combinação interessante de técnicas em

- fluxos em redes (Ford e Fulkerson [FF56], Dinic [Din70], Edmonds e Karp [EK72]);
- partição de grafos/matróides (Tutte [Tut61], Nash-Williams [NW61, NW64], Edmonds [Edm65]);
- intersecção de polimatróides (Schönsleben [Sch80], Lawler e Martel [LM82]).

Bixby, Cunningham e Topkis [BCT85] estenderam para funções submodulares as idéias desenvolvidas por Cunningham para o problema sobre matróides e, com isso, apresentaram o primeiro algoritmo combinatório para minimização de funções submodulares.

Nos referimos a essas primeiras idéias combinatórias associadas à minimização de funções submodulares como **método de Cunningham**. O método de Cunningham, descrito neste capítulo, é a base para os algoritmos combinatórios e fortemente polinomiais que descrevemos nos capítulos seguintes.

4.1 Pertinência no politopo dos matróides

Nesta seção, descrevemos em linhas gerais como Cunningham resolveu o problema da pertinência no politopo dos conjuntos independentes de um matróide. No restante do

capítulo, mostramos com mais detalhes como as idéias para o problema sobre matróides foram estendidas para funções submodulares.

Conforme mencionamos no capítulo 2, o politopo dos conjuntos independentes de um matróide M é o fecho convexo dos vetores de incidência de seus conjuntos independentes. Edmonds [Edm70] mostrou que tal politopo pode ser descrito por

$$P_M := \{x \in \mathbb{R}^S : x \geq \mathbf{0},\ x(U) \leq r(U),\ \text{para todo } U \subseteq S\},$$

em que S é o conjunto sobre o qual o matróide está definido e r é a sua função posto. O problema da pertinência no politopo dos conjuntos independentes de um matróide, ou, simplesmente, problema da pertinência em matróides, é definido da seguinte maneira:

Problema PERTINÊNCIA-MATRÓIDES(S, M, y): *Dados um matróide M sobre S e um vetor y indexado por S, decidir se y está em P_M.*

Cunningham [Cun84] resolveu o problema da pertinência em matróides e forneceu uma prova algorítmica fortemente polinomial para a seguinte importante relação min-max, devida também a Edmonds [Edm70]:

$$\max\{x(S) : x \leq y,\ x \in P_M\} = \min\{r(U) + y(S \setminus U) : U \subseteq S\}, \tag{4.1}$$

em que r é a função posto do matróide M sobre S e y é um vetor qualquer indexado por S.

O algoritmo de Cunningham recebe um matróide M sobre S, dado por um oráculo (de independência, por exemplo), e um vetor y indexado por S. Se y está no politopo P_M, devolve uma descrição de y como combinação convexa de vetores de incidência de conjuntos independentes de M. Caso contrário, devolve uma parte W de S tal que $y(W) > r(W)$, certificando que y não está em P_M.

Cunningham adota em seu algoritmo uma abordagem de caminhos de aumento semelhante ao que se conhece para problemas de fluxo máximo [FF56, Din70, EK72], utilizando a relação min-max (4.1) como análoga ao teorema do fluxo máximo e corte mínimo [FF56]. Durante toda a sua execução, o algoritmo mantém um vetor $x \leq y$ viável para o problema de maximização em (4.1) como combinação convexa de vetores de incidência de conjuntos independentes do matróide. A cada iteração, busca melhorar a solução mantida através de um caminho de aumento em determinado grafo auxiliar.

De forma análoga ao que acontece no algoritmo de Edmonds para o problema da partição de matróides [Edm65], os caminhos do grafo auxiliar representam, de alguma forma, as possíveis "trocas vantajosas" entre conjuntos independentes que compõem a combinação convexa que descreve x. A partir dessas trocas, pode-se obter a cada iteração um vetor x' ainda em P_M tal que $x'(S) > x(S)$. Além disso, quando não existem mais caminhos de aumento, o grafo auxiliar fornece um certificado da maximalidade da solução x mantida: uma parte W de S para a qual $r(W) + y(S \setminus W) = x(S)$.

Quando um vetor x máximo para o problema (4.1) é encontrado, se $x = y$, o vetor y está no politopo P_M. Caso contrário, utilizando propriedades do grafo auxiliar, é fácil verificar que a parte W obtida, que atinge o mínimo no problema, satisfaz também a desigualdade

$y(W) > r(W)$, que separa y do politopo P_M.

Como a função posto de qualquer matróide é submodular, a função

$$g(U) := r(U) + y(S \setminus U), \text{ para cada } U \subseteq S,$$

minimizada pelo algoritmo, é também uma função submodular. Cabe observar, portanto, que o algoritmo de Cunningham resolve o problema de minimização de funções submodulares para um caso bem particular.

Para garantir que o algoritmo termina após um número fortemente polinomial de aumentos, Cunningham baseia-se em técnicas devidas a Lawler e Martel [LM82] e Schönsleben [Sch80], presentes em trabalhos sobre intersecção de polimatróides. Tais técnicas consistem, dentre outras coisas, em realizar aumentos através de caminhos resultantes de busca em largura no grafo auxiliar e lexicograficamente menores segundo ordenação linear pré-estabelecida dos vértices desse grafo. A técnica é chamada por Cunningham de *consistent breadth-first search*. Schrijver [Sch00] utiliza uma adaptação dessa idéia para garantir que seu algoritmo para minimizar funções submodulares seja fortemente polinomial, conforme veremos no capítulo 5.

4.2 Relações min-max sobre polimatróides

Percebeu-se que a relação min-max (4.1) também vale quando a função posto r é substituída por uma função submodular f qualquer, com $f(\emptyset) = 0$. Em particular, tomando o vetor $y := \mathbf{0}$, podemos estabelecer o seguinte resultado.

Teorema 4.1: *Seja f uma função submodular sobre S tal que $f(\emptyset) = 0$. Então*

$$\max\{x(S) \colon x \leq \mathbf{0},\ x \in EP(f)\} = \min\{f(U) \colon U \subseteq S\}. \tag{4.2}$$

Demonstração: Dado x em $EP(f)$ tal que $x \leq 0$, temos $x(S) \leq x(U) \leq f(U)$, para toda parte U de S, de maneira que max $\leq$ min vale trivialmente. Para a relação inversa, considere um vetor z que atinge o máximo em (4.2). Se $z(S) = 0$, então $z(S) = f(\emptyset)$ e estamos feitos. Caso contrário, para todo s em S tal que $z(s) < 0$, deve existir um conjunto z-justo U_s que contenha s. De fato, se existe s tal que $z(s) < 0$ e que não esteja em nenhum conjunto z-justo, podemos aumentar a componente $z(s)$ de forma a aumentar o máximo em (4.2), o que seria uma contradição. Definimos, então,

$$U := \bigcup_{\{s : z(s) < 0\}} U_s,$$

que também é um conjunto z-justo, pelo teorema 2.2. Assim,

$$z(S) = z(U) + z(S \setminus U) = f(U) + z(S \setminus U) = f(U),$$

uma vez que U é z-justo e que $S \setminus U$ só contém elementos s de S tais que $z(s) = 0$. Segue a validade da relação min-max. ∎

Conseqüentemente, uma estratégia natural para o desenvolvimento de um algoritmo combinatório para minimizar funções submodulares consistiu em estender para polimatróides estendidos a abordagem de Cunningham para o politopo dos conjuntos independentes de um matróide. Conforme veremos ao longo do capítulo, o primeiro algoritmo combinatório para minimizar funções submodulares, devido a Bixby, Cunningham e Topkis [BCT85], baseia-se justamente nessa estratégia.

Neste texto, entretanto, seguimos a maioria dos trabalhos na área [Fle00, Sch00, IFF01, Iwa02, Fuj03, Fuj05] e descrevemos os algoritmos combinatórios para minimização de funções submodulares baseados em uma variação da relação min-max (4.2). Esta outra relação min-max, apresentada no teorema abaixo, não exige a restrição $x \leq \mathbf{0}$ e é baseada no polimatróide das bases

$$B(f) := \{x \in \mathbb{R}^S : x(S) = f(S),\ x(U) \leq f(U),\ \text{para todo } U \subseteq S\},$$

que é a face do polimatróide estendido $EP(f)$ que contém todos os seus pontos extremos, conforme definido na seção 2.6.

Dado x em $\mathbb{R}^S$, defina $x^-(s) := \min\{x(s), 0\}$.

Teorema 4.2: *Seja f uma função submodular sobre S tal que $f(\emptyset) = 0$. Então*

$$\max\{x^-(S) : x \in B(f)\} = \min\{f(U) : U \subseteq S\}. \tag{4.3}$$

Demonstração: Seja x um vetor em $B(f)$. Para qualquer parte U de S, temos que $x^-(S) \leq x(U) \leq f(U)$, uma vez que, no pior caso, U contém todos os elementos s de S tais que $x(s) < 0$ e nenhum elemento s de S tal que $x(s) > 0$. Portanto, vale a relação max $\leq$ min. Provamos agora que vale a igualdade. Seja z um vetor que é solução ótima do problema de maximização em (4.3). Se $z(s) \geq 0$ para todo s em S, então $z^-(S) = 0 = f(\emptyset)$. Além disso, se $z(s) \leq 0$ para todo s em S, então $z^-(S) = z(S) = f(S)$, uma vez que z está em $B(f)$. Temos, portanto, que vale a relação max $=$ min nesses dois primeiros casos. Quando não estamos nesses casos, dados quaisquer s e t em S tais que $z(s) > 0$ e $z(t) < 0$, temos que existe um conjunto z-justo $U_{s,t}$ que contém t e não contém s. De fato, caso contrário, poderíamos aumentar o valor de $z(t)$ e diminuir o valor de $z(s)$ de forma a manter a viabilidade de z em $B(f)$ e aumentar o valor da solução ótima, o que seria uma contradição. Consideremos, então, o conjunto

$$U := \bigcup_{\{t:\, z(t)<0\}} \bigcap_{\{s:\, z(s)>0\}} U_{s,t}.$$

Note que U é também um conjunto z-justo (teorema 2.2). Além disso, todos os elementos de S que definem os componentes negativos de z estão em U e todos os elementos de S

que definem os componentes positivos de z estão fora de U, de maneira que

$$z^-(S) = z(U) = f(U),$$

de onde segue a validade da relação min-max. ∎

4.3 Elementos do método

Os algoritmos para minimização de funções submodulares baseados no método de Cunningham, assim como o algoritmo de Cunningham para o problema da pertinência no politopo dos conjuntos independentes de um matróide, imitam a abordagem de caminhos de aumento aplicada aos problemas duais do fluxo máximo e do corte mínimo. Para tanto, utilizam a relação min-max (4.3) como análoga ao teorema do fluxo máximo e corte mínimo.

Nos algoritmos para o problema do fluxo máximo, um fluxo viável é mantido e tem seu valor aumentado em cada iteração através de caminhos em determinado grafo auxiliar. Quando não existem mais caminhos de aumento, um corte de mesma capacidade do fluxo mantido pode ser facilmente encontrado.

De forma análoga, os algoritmos para o problema de minimização de funções submodulares mantêm um vetor x no polimatróide das bases $B(f)$ que é um vetor viável para o problema de maximização em (4.3). A cada iteração, buscam aumentar o valor de $x^-(S)$ também através de caminhos em um grafo auxiliar. Quando não existem mais caminhos de aumento, propriedades do grafo permitem identificar uma parte W de S tal que $x^-(S) = f(W)$, de maneira que W minimiza f.

Representação de vetores em $B(f)$

Encontramos aqui um primeiro obstáculo. Garantir a viabilidade de um vetor x em $B(f)$ não é uma tarefa trivial, dada a clássica descrição exponencial desse poliedro. Com o intuito de solucionar tal problema, Cunningham sugeriu representar o vetor x como combinação convexa de pontos extremos de $B(f)$. Isto é, os algoritmos mantêm

$$x = \sum_{i \in I} \lambda_i x^{\prec_i}, \tag{4.4}$$

em que $x^{\prec_i}$ é um ponto extremo de $B(f)$, para cada i em I, $\lambda_i > 0$ e $\sum_{i \in I} \lambda_i = 1$.

Pelo teorema de Carathéodory (teorema 2.1), cada vetor em $B(f)$ pode ser representado como combinação convexa de no máximo $|S| + 1$ de seus pontos extremos. Além disso, os pontos extremos de $B(f)$ são caracterizados polinomialmente através do método guloso, conforme descrito na seção 2.6. Para cada $x^{\prec_i}$ na combinação convexa que descreve x, os algoritmos mantêm uma ordem linear $\prec_i$ de S que o gera através do método guloso e, portanto, certifica que esse vetor é um ponto extremo de $B(f)$. Assim, a idéia de Cunningham fornece uma caracterização polinomial para os vetores em $B(f)$.

De forma mais específica, os algoritmos que descrevemos ao longo deste texto manipulam **representações** de vetores x em $B(f)$, compostas por

- um conjunto de índices I_x;
- para cada i em I_x, um número real positivo λ_i;
- para cada i em I_x, uma ordem linear $\prec_i$ de S

tais que

$$x = \sum_{i \in I_x} \lambda_i x^{\prec_i} \quad \text{e} \quad \sum_{i \in I_x} \lambda_i = 1. \tag{4.5}$$

Sempre que nos referirmos a representações de vetores x em $B(f)$, estamos nos referindo a representações desse tipo. Vale ressaltar também que, em geral, procura-se fazer com que os algoritmos mantenham representações de x tais que $|I_x| \leq |S| + 1$, de maneira que se consiga demonstrar que tenham consumo de tempo polinomial em $|S|$.

Pivotações e st-potenciais

A cada iteração, os algoritmos baseados no método de Cunningham buscam aumentar o valor de $x^-(S)$, transformando o vetor x em um vetor x' tal que $x'^-(S) \geq x^-(S)$, mas sempre de forma a manter sua viabilidade em $B(f)$. Para tanto, se utilizam de um passo básico, o qual chamamos **pivotação**.

Em uma pivotação, aumenta-se o valor de um componente de x e diminui-se o valor de outro componente em uma mesma quantidade, de forma a se obter o vetor x'. Mais especificamente, temos

$$x' := x + \alpha(\chi^t - \chi^s), \tag{4.6}$$

para algum par de elementos distintos s e t em S e algum $\alpha \geq 0$. Numa operação desse tipo, dizemos que estamos **pivotando** s **e** t **em** x.

Como precisamos garantir que o vetor x' resultante continue em $B(f)$ e como qualquer vetor x em $B(f)$ satisfaz $x(S) = f(S)$, a idéia de aumentar um componente e diminuir outro em uma mesma quantidade surge naturalmente. Além disso, como queremos obter $x'^-(S) \geq x^-(S)$, escolhem-se elementos s e t tais que $x(s)$ é positivo e $x(t)$ é negativo para aplicar a pivotação. Dessa forma, com algum controle sobre a constante α, é possível fazer com que x' permaneça viável, bem como com que, idealmente, o valor de $x'^-(S)$ cresça após a operação.

Dado um vetor x e um par de elementos s e t de S, chamamos de st**-potencial com relação a** x (ou simplesmente st**-potencial**), o maior valor de α tal que $x' := x + \alpha(\chi^t - \chi^s)$ é ainda um vetor em $B(f)$. Denotamos o st-potencial com relação a x por $\tilde{\alpha}(x, s, t)$. Formalmente,

$$\tilde{\alpha}(x, s, t) := \max\left\{\alpha \colon x + \alpha(\chi^t - \chi^s) \in B(f)\right\}. \tag{4.7}$$

Segue de sua própria definição que $\tilde{\alpha}(x, s, t) \geq 0$.

Calcular st-potenciais e, portanto, saber quando é possível pivotar s e t, também não é uma tarefa fácil. Para que x' obtido em (4.6) seja ainda um vetor em $B(f)$, devemos ter

$$f(U) - x'(U) \geq 0, \text{ para todo } U \subseteq S. \tag{4.8}$$

Como $f(U) - x(U) \geq 0$ vale para toda parte U de S, uma vez que x está em $B(f)$, para garantir que a propriedade (4.8) valha para x', deve existir certo controle sobre o valor que x' assume nas partes U de S tais que $x'(U) > x(U)$. Pela forma como x' é obtido a partir de x, tais partes são exatamente aquelas que contêm t e não contêm s. Sendo assim, o valor do st-potencial com relação a x é determinado pelo subconjunto U que minimiza o valor $f(U) - x(U)$, com t em U e s em $S \setminus U$. Esse é o valor máximo com que podemos aumentar o componente $x(t)$ de modo a produzir x' que não viole a propriedade (4.8).

Segue que o st-potencial $\tilde{\alpha}(x, s, t)$ é também dado por

$$\tilde{\alpha}(x, s, t) := \min\Big\{f(U) - x(U) \colon t \in U \subseteq S \setminus \{s\}\Big\}. \tag{4.9}$$

Calcular st-potenciais resume-se, portanto, a uma nova instância do problema de minimização de funções submodulares. Tal fato apresenta-se como outro obstáculo aos algoritmos combinatórios para minimização de funções submodulares.

st-potenciais de fácil cálculo

Bixby, Cunningham e Topkis [BCT85] mostraram que se $x^{\prec}$ é um ponto extremo de $B(f)$, o st-potencial para alguns pares de elementos distintos s e t em S pode ser calculado de forma eficiente, com o auxílio do método guloso. Se $\prec$ é uma ordem linear de S que gera $x^{\prec}$, tais pares são aqueles em que s **precede imediatamente** t **na ordem** $\prec$. Isto é, os elementos s e t são tais que $s \prec t$ e não existe u tal que $s \prec u \prec t$, $u \neq s$ e $u \neq t$. Nesses casos, o st-potencial com relação a $x^{\prec}$ é exatamente o valor α dado pelo teorema 2.13. Ou seja,

$$\tilde{\alpha}(x^{\prec}, s, t) = f\big([t]_{\prec} \setminus \{s\}\big) - x^{\prec}\big([t]_{\prec} \setminus \{s\}\big). \tag{4.10}$$

Denote $\alpha := f\big([t]_{\prec} \setminus \{s\}\big) - x^{\prec}\big([t]_{\prec} \setminus \{s\}\big)$. Para observar a validade de (4.10), note que, pela definição (4.7) de st-potencial, $\tilde{\alpha}(x^{\prec}, s, t) \geq \alpha$, uma vez que, pelo teorema 2.13, $x^{\prec} + \alpha(\chi^t - \chi^s)$ é um ponto extremo de $B(f)$. Por outro lado, como t está em $[t]_{\prec} \setminus \{s\}$, segue da definição alternativa de st-potencial, dada por (4.9), que $\tilde{\alpha}(x^{\prec}, s, t) \leq \alpha$. Portanto, $\tilde{\alpha}(x^{\prec}, s, t) = \alpha$.

Quando fazemos uma pivotação desse tipo, com s precedendo imediatamente t numa ordem que gera $x^{\prec}$, e quando $\tilde{\alpha}(x^{\prec}, s, t) > 0$, estamos trocando um ponto extremo $x^{\prec}$ por um de seus pontos extremos adjacentes em $B(f)$. Além disso, temos que o vetor $x' := x^{\prec} + \tilde{\alpha}(x^{\prec}, s, t)(\chi^t - \chi^s)$, obtido a partir da pivotação entre s e t, é gerado pela ordem $\prec'$ obtida a partir de $\prec$ ao trocarmos s e t de posição.

Grafo de Cunningham

Em seu primeiro algoritmo para minimizar funções submodulares, Bixby, Cunningham e Topkis restringiram as pivotações aos casos em que o st-potencial pode ser facilmente calculado. Em cada iteração, seu algoritmo mantém uma representação de um vetor x em $B(f)$, conforme (4.5). Com o objetivo de guiar as pivotações a serem realizadas, definem o **grafo de Cunningham para f e x**, que denotamos por $D(f,x)$.

O grafo de Cunningham $D(f,x)$ é orientado e tem como conjunto de vértices o conjunto base S da função f. Para definirmos seu conjunto de arcos, começamos atribuindo, para cada par s e t de elementos distintos de S, o seguinte conjunto:

$$F(s,t) := \big\{ i \in I_x \colon s \text{ precede imediatamente } t \text{ na ordem } \prec_i \big\}. \tag{4.11}$$

Feito isso, colocamos um arco de s a t em $D(f,x)$ se, e somente se, $F(s,t) \neq \emptyset$.

Note que o grafo de Cunningham possui um arco de s a t se, e somente se, existe i em I_x tal que a pivotação entre s e t em $x^{\prec_i}$ dá origem a um ponto extremo adjacente a $x^{\prec_i}$ em $B(f)$, dado que $\tilde{\alpha}(x^{\prec_i}, s, t) > 0$. Além disso, para tais pares s e t sabemos calcular o st-potencial com relação a $x^{\prec_i}$ de forma simples, conforme (4.10).

Bixby, Cunningham e Topkis utilizam determinados caminhos sobre o grafo $D(f,x)$ para realizar pivotações sobre pontos extremos que compõem a representação mantida de x. Em cada iteração, trocam pontos extremos $x^{\prec_i}$ por outros $x^{\prec_j}$, de maneira que

$$x^{\prec_j} := x^{\prec_i} + \tilde{\alpha}(x^{\prec_i}, s, t)\big(\chi^t - \chi^s\big), \tag{4.12}$$

para algum par s e t em S. Feito isso, os novos pontos extremos $x^{\prec_j}$ são utilizados para a obtenção do novo vetor x' em $B(f)$, de maneira que $x'^{-}(S) \geq x^{-}(S)$.

No caso do problema da pertinência no politopo dos conjuntos independentes de um matróide, considerar apenas pivotações de forma a realizar trocas entre pontos extremos adjacentes não foi um empecilho para que se obtivesse um algoritmo polinomial.

Já no caso do problema de minimização de funções submodulares, o grafo auxiliar tornou-se "pequeno" e "instável" e nenhuma complexidade razoável pôde ser demonstrada. Por exemplo, não se conseguiu garantir propriedades sobre o comprimento de um caminho mínimo ou sobre um caminho de capacidade máxima nesse grafo ao longo das iterações, conforme expõe Fleischer [Fle00]. Discutiremos um pouco sobre essa questão no capítulo 5. Os algoritmos de Schrijver [Sch00] e Iwata, Fleischer e Fujishige [IFF01], apresentados nos próximos capítulos, resolvem esse problema adaptando e ampliando de alguma forma os arcos propostos para o primeiro grafo auxiliar – o grafo de Cunningham.

4.4 Descrição do método

O método de Cunningham para minimização de funções submodulares, cujos principais elementos estão descritos na seção anterior, pode ser resumido nos seguintes princi-

pais tópicos:

- **(relação min-max)** abordagem de caminhos de aumento baseada na relação min-max (4.3):
$$\max\{x^-(S)\colon x \in B(f)\} = \min\{f(U)\colon U \subseteq S\};$$
- **(representação de x)** manutenção de uma representação de um vetor x em $B(f)$ como combinação convexa de pontos extremos de $B(f)$, conforme (4.5);
- **(caminhos de aumento)** realização de sucessivos aumentos em $x^-(S)$ através de st-pivotações nos pontos extremos que compõem a representação de x, com auxílio do grafo de Cunningham $D(f,x)$.

Com base nessas idéias, apresentamos abaixo um esquema de algoritmo combinatório para minimizar funções submodulares. Mostramos também como Bixby, Cunningham e Topkis [BCT85] sugeriram realizar os aumentos em $x^-(S)$ com auxílio do grafo de Cunningham, na primeira versão do algoritmo. Nos próximos capítulos, ressaltamos como essa versão original foi modificada de modo a resultar em algoritmos fortemente polinomiais.

Método CUNNINGHAM(S,f): *Recebe uma função submodular f sobre S tal que $f(\emptyset)=0$ e devolve uma parte W de S que minimiza f.*

O método de Cunningham é iterativo. Cada iteração começa com uma representação de um vetor x em $B(f)$, conforme (4.5), ou seja,

- um conjunto de índices I_x, com $|I_x| \leq |S|+1$;
- para cada i em I_x, um número real positivo λ_i;
- para cada i em I_x, uma ordem linear $\prec_i$ de S;

tais que $x = \sum_{i\in I_x} \lambda_i x^{\prec_i}$ e $\sum_{i\in I_x}\lambda_i = 1$. Na primeira iteração, temos $I_x := \{1\}$, $\lambda_1 := 1$ e $\prec_1$ sendo uma ordem linear qualquer de S, de maneira que $x = x^{\prec_1}$.

Cada iteração consiste no seguinte. Sejam

$P := \{s \in S\colon x(s) > 0\}$; $N := \{s \in S\colon x(s) < 0\}$ e $D := D(f,x)$.

Caso 1: Não existe caminho de P a N em D.
Seja W o conjunto de vértices que acessam N em D.
Devolva W.

Caso 2: Existe um caminho de P a N em D.
$Q \leftarrow$ ESCOLHE-CAMINHO(D,P,N).
$x'' \leftarrow$ AUMENTA(S,f,x,Q).
$x' \leftarrow$ COMPACTA(S,f,x'').
Comece nova iteração com x' no papel de x.

Na descrição do algoritmo, ESCOLHE-CAMINHO, AUMENTA e COMPACTA são rotinas auxiliares.

A rotina ESCOLHE-CAMINHO é responsável pela escolha de um caminho Q de P a N em $D(f,x)$ a partir do qual será realizado o aumento em x. Sua especificação está a seguir.

Rotina ESCOLHE-CAMINHO(D,P,N): *Recebe um grafo D e dois subconjuntos de vértices P e N de D e devolve um caminho Q de P a N nesse grafo.*

Quando descreveram seu primeiro algoritmo para minimizar funções submodulares, Bixby, Cunningham e Topkis [BCT85] não fixaram nenhuma maneira de selecionar o caminho para realizar o aumento. Conforme veremos nos próximos capítulos, os demais algoritmos combinatórios selecionam o caminho de alguma forma específica nessa etapa.

A rotina AUMENTA representa o **passo aumentador** que origina um vetor x' em $B(f)$ tal que $x'^{-}(S) \geq x^{-}(S)$. Sua especificação está a seguir.

Rotina AUMENTA(S,f,x,Q): *Recebe a representação de um vetor x em $B(f)$ e um caminho Q em $D(f,x)$ e devolve a representação de um vetor x' em $B(f)$, obtida com o auxílio dos arcos de Q, tal que*
(i) $x' = x + \delta(\chi^t - \chi^s)$, para elementos s e t distintos em Q e algum $\delta \geq 0$;
(ii) $x'^{-}(S) \geq x^{-}(S)$.

O passo aumentador descrito pela rotina AUMENTA é implementado de maneiras distintas pelos algoritmos combinatórios para minimizar funções submodulares que descrevemos ao longo deste texto. A prova do teorema 4.4, na próxima seção, apresenta a implementação sugerida por Bixby, Cunningham e Topkis em seu primeiro algoritmo [BCT85].

A rotina COMPACTA é responsável por garantir que a representação mantida do vetor x em $B(f)$ seja sempre composta por um número reduzido de pontos extremos, através de técnicas de álgebra linear. O teorema 2.1 de Carathéodory garante que isso é de fato possível. A especificação dessa rotina está a seguir.

Rotina COMPACTA(S,f,x): *Recebe a representação de um vetor x em $B(f)$ e devolve uma representação desse mesmo vetor composta por no máximo $|S|+1$ dos pontos extremos que compunham originalmente a representação de x.*

Como menciona Cunningham [Cun85], tal procedimento é comumente utilizado para transformar soluções viáveis em soluções viáveis básicas em algoritmos de programação linear. Além disso, com a utilização da eliminação Gaussiana, a rotina COMPACTA pode ser realizada em tempo proporcional a $|S|^3$, sempre que a representação do vetor recebida como entrada seja composta por $O(|S|)$ pontos extremos. Todos os algoritmos combinatórios para minimizar funções submodulares que apresentamos ao longo do texto realizam esse passo ao final de cada iteração visando a garantir um consumo de tempo razoável. Estabelecemos o resultado abaixo para referências futuras.

Proposição 4.3: *A rotina COMPACTA(S,f,x) pode ser implementada de modo a*

consumir tempo $O(|S|^3)$, sempre que a representação do vetor x recebida como entrada seja composta por $O(|S|)$ pontos extremos. ∎

4.5 Correção do método

No trabalho em que apresentam a primeira versão de algoritmo combinatório para minimização de funções submodulares, Bixby, Cunningham e Topkis justificam a abordagem de caminhos de aumento através dos resultados abaixo. Conforme mencionado, a prova do próximo teorema 4.4 também apresenta a implementação que sugeriram para a rotina AUMENTA(S, f, x, Q).

Para cada arco (s,t) em $D(f,x)$, defina sua **capacidade** $u(s,t)$ como

$$u(s,t) := \sum_{i \in F(s,t)} \tilde{\alpha}(x^{\prec_i}, s, t), \tag{4.13}$$

em que $F(s,t)$ é definido como em (4.11). Defina também a **capacidade de um caminho** Q em $D(f,x)$ como $\min\{u(s,t)\colon (s,t) \in Q\}$.

Teorema 4.4: *Sejam P, N, x e $D(f,x)$ considerados no início de uma iteração qualquer do método* CUNNINGHAM(S,f). *Se existe um caminho de P a N em $D(f,x)$ com capacidade não-nula, então existe x' em $B(f)$ tal que $x'^{-}(S) > x^{-}(S)$.*

Demonstração: Suponha que exista um caminho Q de P a N em $D(f,x)$, com capacidade não-nula e seqüência de vértices $s = s_1, s_2, \ldots, s_m = t$. Vamos utilizar o caminho Q para obter $x' = x + \delta(\chi^t - \chi^s)$, para algum $\delta > 0$, de maneira que x' esteja em $B(f)$ e satisfaça $x'^{-}(S) > x^{-}(S)$.

Considere a representação de x como $\sum_{i \in I_x} \lambda_i x^{\prec_i}$, com $\lambda_i > 0$, para todo i em I_x, e $\sum_{i \in I_x} \lambda_i = 1$, mantida pelo algoritmo.

Para cada j, com $1 \leq j < m$, escolhemos um índice $i(j)$ em $F(s_j, s_{j+1})$, conforme definido em (4.11), para "representar" o arco (s_j, s_{j+1}). O índice $i(j)$ justifica a existência do arco (s_j, s_{j+1}) no grafo $D(f,x)$.

Para cada i em I_x, defina $n(i) := \big|\{1 \leq j < m\colon i = i(j)\}\big|$, como o número de vezes que o índice i foi escolhido para representar um arco no caminho Q.

Defina

$$\delta' := \min\left\{\frac{\lambda_i}{n(i)}\colon n(i) > 0,\ i \in I_x\right\}; \tag{4.14}$$

$$\alpha' := \min\left\{\tilde{\alpha}(x^{\prec_{i(j)}}, s_j, s_{j+1})\colon 1 \leq j < m\right\}; \tag{4.15}$$

$$\alpha := \min\{\alpha', x(s)\}. \tag{4.16}$$

Observe que $\alpha \geq 0$ e $\delta' > 0$.

Para cada $1 \leq j < m$, consideramos ainda o vetor

$$z^j = x^{\prec_{i(j)}} + \alpha(\chi^{s_{j+1}} - \chi^{s_j}). \tag{4.17}$$

Por fim, tomamos $\lambda'_i := \lambda_i - \delta' n(i)$, para cada i em I_x, e definimos

$$\begin{aligned} x' &:= \sum_{i \in I} \lambda'_i x^{\prec_i} + \sum_{1 \leq j < m} \delta' z^j \\ &= \sum_{i \in I} \lambda_i x^{\prec_i} + \delta' \alpha(\chi^t - \chi^s) \\ &= x + \delta' \alpha(\chi^t - \chi^s). \end{aligned} \tag{4.18}$$

Tome $\delta := \delta' \alpha$. Observe que, por definição, temos $0 \leq \alpha < x(s)$ e $0 < \delta' < 1$, de maneira que $0 \leq \delta < x(s)$. Segue que $x'^-(S) \geq x^-(S)$. Ademais, como Q é um caminho de capacidade não-nula, então, para cada $1 \leq j < m$, podemos escolher um índice $i(j)$ para representar o arco (s_j, s_{j+1}) de maneira que o (s_j, s_{j+1})-potencial com relação a $x^{\prec_{i(j)}}$ seja não-nulo. Nesse caso, segue que α (e conseqüentemente δ) será estritamente positivo, de maneira a podermos garantir que $x'^-(S) > x^-(S)$.

Temos também que x' está em $B(f)$ pois é combinação convexa de elementos de $B(f)$, conforme (4.18). Mais especificamente, mostramos abaixo que é possível obter uma representação de x' como combinação convexa de pontos extremos de $B(f)$.

Para cada $1 \leq j < m$, considere o ponto extremo

$$x^j = x^{\prec_{i(j)}} + \tilde{\alpha}(x^{\prec_{i(j)}}, s_j, s_{j+1})(\chi^{s_{j+1}} - \chi^{s_j}),$$

obtido através da pivotação entre s_j e s_{j+1} em $x^{\prec_{i(j)}}$. Se $\tilde{\alpha}(x^{\prec_{i(j)}}, s_j, s_{j+1}) = 0$, então $z^j = x^j$, com z^j definido conforme (4.17). Por outro lado, se $\tilde{\alpha}(x^{\prec_{i(j)}}, s_j, s_{j+1}) \neq 0$, então z^j é combinação convexa de x^j e $x^{\prec_{i(j)}}$, como segue:

$$z^j = \left(\frac{\alpha}{\tilde{\alpha}(x^{\prec_{i(j)}}, s_j, s_{j+1})} \right) x^j + \left(1 - \frac{\alpha}{\tilde{\alpha}(x^{\prec_{i(j)}}, s_j, s_{j+1})} \right) x^{\prec_{i(j)}}.$$

Concluímos, assim, que x', definido em (4.18), pode ser escrito como combinação convexa de $x^{\prec_i}$, i em I_x, e x^j, $1 \leq j < m$, todos pontos extremos de $B(f)$. ∎

O próximo resultado, juntamente com a relação min-max (4.3), garante que, se o método de Cunningham pára, então ele de fato devolve um minimizador de f.

Teorema 4.5: *Suponha que o caso 1 ocorra em determinada iteração do método* CUNNINGHAM(S, f). *Seja W o conjunto de vértices que acessam N em $D(f, x)$, conforme definido nessa iteração. Então $x^-(S) = f(W)$, de maneira que W minimiza f.*

Demonstração: Seja W conforme definido pelo caso 1 do método CUNNINGHAM(S, f). Considere também a representação do vetor x mantida pelo algoritmo, conforme (4.5). Se $W = \emptyset$, então $N = \emptyset$, de modo que $x^-(S) = 0 = f(\emptyset)$ e segue o resultado. Suponha, então, que $W \neq \emptyset$. Para cada elemento t de W, temos que $[t]_{\prec_i}$ está em W, para todo i em I_x. De fato, pela definição do grafo $D(f, x)$, para todo s que precede t em $\prec_i$, existe um caminho de s a t em $D(f, x)$. Como não existem arcos entrando em W, segue que todo elemento que precede t em $\prec_i$ está em W.

Agora, como pelo teorema 2.10 temos que $[t]_{\prec_i}$ é $x^{\prec_i}$-justo, concluímos que W é a união de conjuntos $x^{\prec_i}$-justos, de onde segue que W é também um conjunto $x^{\prec_i}$-justo, para cada i em I_x. Isso nos permite concluir também que W é um conjunto x-justo, como segue:

$$x(W) = \sum_{i \in I_x} \lambda_i x^{\prec_i}(W) = \sum_{i \in I_x} \lambda_i f(W) = f(W),$$

já que $\sum_{i \in I_x} \lambda_i = 1$.

Pela definição de W, temos também que $P \cap W = \emptyset$ e $N \subseteq W$, de maneira que

$$x^-(S) = x(W) = f(W),$$

de onde segue o resultado. ∎

Em seu primeiro trabalho, Bixby, Cunningham e Topkis mostraram que é possível garantir que o método CUNNINGHAM(S, f) pára, mas não obtiveram sucesso em demonstrar nenhuma complexidade razoável para o algoritmo. Um pouco mais tarde, Cunningham [Cun85] demonstrou um consumo de tempo pseudo-polinomial para a implementação sugerida por Bixby, Cunningham e Topkis para o algoritmo, sob a hipótese de que a função submodular f é inteira. Nos próximos capítulos, mostramos como outras idéias foram adicionadas ao método de Cunningham de forma a originar algoritmos fortemente polinomiais para minimizar funções submodulares.

Capítulo 5

Algoritmo de Schrijver

No capítulo 4, apresentamos algumas idéias combinatórias propostas por Cunningham para o problema de minimização de funções submodulares. Dentre tais idéias, Cunningham sugere uma abordagem de caminhos de aumento que se assemelha muito àquela proposta por Ford e Fulkerson [FF56] para o problema do fluxo máximo. No caso do problema do fluxo máximo, para que se obtivesse algoritmos polinomiais baseados em caminhos de aumento, Edmonds e Karp [EK72] sugeriram duas abordagens: **caminhos mínimos** (*shortest paths*) e **caminhos de maior aumento** (*fattest paths*).

Conforme apresentamos neste capítulo, nenhuma dessas duas abordagens aplicadas diretamente sobre o método de Cunningham resulta na obtenção de um algoritmo polinomial. Entretanto, através da introdução de idéias adicionais, Schrijver [Sch00] desenvolveu um algoritmo fortemente polinomial para minimizar funções submodulares que traz uma abordagem de caminhos mínimos em sua essência. Este capítulo é dedicado principalmente à apresentação desse algoritmo. No capítulo 6, veremos como Iwata, Fleischer e Fujishige [IFF01] conseguiram também derivar um algoritmo fortemente polinomial partindo do método de Cunningham e de idéias envolvidas numa abordagem de caminhos de maior aumento.

Baseados no algoritmo de Schrijver e em idéias apresentadas por Goldberg e Tarjan [GT88] para o problema do fluxo máximo, Fleischer e Iwata [FI03] desenvolveram um algoritmo *push-relabel* para o problema de minimização de funções submodulares. Esse algoritmo também está descrito neste capítulo.

5.1 Caminhos mínimos e caminhos de maior aumento

Nesta seção, apresentamos alguma intuição sobre porque a aplicação das abordagens de caminhos mínimos ou caminhos de maior aumento diretamente sobre o método de Cunningham pode não ser suficiente para a obtenção de algoritmos polinomiais. Schrijver [Sch00] e Iwata, Fleischer e Fujishige [IFF01] solucionaram os problemas que apresentamos aqui ao desenvolver seus algoritmos fortemente polinomiais.

Na abordagem de caminhos mínimos para o problema do fluxo máximo, seleciona-se

o caminho com o menor número de arcos para realizar um aumento. A polinomialidade do algoritmo resultante baseia-se no fato de que o comprimento de um caminho mínimo nunca diminui, mas possivelmente aumenta, após um número polinomial de iterações.

No método de Cunningham (capítulo 4), os aumentos na solução mantida são realizados com o auxílio do grafo de Cunningham, conforme detalhado nas seções 4.3 e 4.4. Dada uma função submodular f sobre S, uma representação de um vetor x no polimatróide das bases $B(f)$ e elementos s e t distintos de S, existe um arco de s a t no grafo de Cunningham $D(f,x)$ se, e somente se, s precede imediatamente t em alguma ordem $\prec_i$ que compõem a representação de x.

Note que se após a realização de um aumento mantivermos na representação de x um ponto extremo $x^{\prec_i}$ e acrescentarmos outro ponto extremo obtido a partir da pivotação entre s e t em $x^{\prec_i}$, estamos adicionando ao grafo $D(f,x)$ um "atalho" que pode diminuir o comprimento de um caminho mínimo.

Para observar esse fato, suponha que u preceda imediatamente s em $\prec_i$ e que pivotamos s e t em $x^{\prec_i}$, de forma a produzir o ponto extremo $x^{\prec_j}$. Suponha ainda que optamos por manter $x^{\prec_i}$ e $x^{\prec_j}$ na representação de x mantida pelo algoritmo. Nesse caso, note que o arco (u,t) passa a fazer parte do grafo $D(f,x)$ após a pivotação. Além disso, tal arco representa um atalho entre u e t com relação ao caminho que passa pelos arcos (u,s) e (s,t), que já pertenciam ao grafo $D(f,x)$. Atalhos desse tipo podem reduzir o comprimento de um caminho mínimo no grafo $D(f,x)$ de uma iteração para outra.

Schrijver procurou resolver essa "instabilidade" no comprimento dos caminhos mínimos no grafo de Cunningham adicionando arcos de s a t a esse grafo sempre que s precede t (não necessariamente imediatamente) em alguma ordem de S que compõe a representação de x. A existência desses arcos impede o surgimento de atalhos como o exemplificado acima.

Na abordagem de caminhos de maior aumento, seleciona-se o caminho que propicia maior acréscimo na solução mantida para realizar o aumento a cada iteração. Tal caminho corresponde ao caminho de maior capacidade no grafo auxiliar. No caso do problema do fluxo máximo, consegue-se demonstrar que, através de caminhos desse tipo, pode-se realizar aumentos polinomiais a cada iteração. Mais especificamente, demonstra-se que a diferença entre a solução ótima e a solução mantida é reduzida polinomialmente quando os aumentos são realizados através de caminhos desse tipo. Isso garante que o algoritmo termina após um número polinomial de aumentos.

Cunningham [Cun85] apresentou um exemplo para o qual um caminho de maior aumento no grafo $D(f,x)$ nem sempre permite que aumentos polinomiais sejam realizados. Além disso, ainda que existam caminhos com capacidade polinomial, cabe lembrar que, quando realizamos uma pivotação entre s e t em algum ponto extremo $x^{\prec_i}$ que compõe a representação de x mantida pelo algoritmo, o acréscimo em $x^-(S)$ que obtemos ao adicionarmos $x^{\prec_j} := x^{\prec_i} + \delta(\chi^t - \chi^s)$ à representação de x não é diretamente proporcional a δ. Ao invés disso, o aumento é proporcional a $\lambda_j\delta$, em que λ_j é o coeficiente de $x^{\prec_j}$ na combinação convexa que compõe a nova representação de x. Conforme observou

Fleischer [Fle00], isso pode ser um problema, uma vez que o coeficiente λ_j pode ser exponencialmente pequeno, por exemplo, dependendo inversamente de um polinômio em $\max_{U \subseteq S} |f(U)|$, que é um limitante para o quanto $x^-(S)$ precisa crescer até que se atinja o ótimo.

Iwata, Fleischer e Fujishige [IFF01] aumentam a capacidade de cada arco do grafo auxiliar através de uma relaxação que propõem ao problema, de maneira a garantir a realização de aumentos polinomiais a cada iteração. Essas idéias estão expostas com mais detalhes no capítulo 6.

5.2 Rotina PIVOTA: limitantes para st-potenciais

Conforme mencionamos na seção anterior, com o objetivo de buscar uma abordagem de caminhos mínimos, Schrijver aumenta o grafo auxiliar sobre o qual seu algoritmo trabalha. Além dos arcos pertencentes ao grafo de Cunningham $D(f, x)$ (definido na seção 4.3), Schrijver adiciona ao seu grafo auxiliar um arco de s a t sempre que s precede t em alguma ordem que compõe a representação mantida do vetor x em $B(f)$. Ao fazer isso, Schrijver já não se limita a trabalhar com pares s e t para os quais o st-potencial pode ser calculado facilmente, conforme definido em (4.10), na seção 4.3.

Para resolver o problema do cálculo dos potenciais para os pares que não são adjacentes no grafo de Cunningham, Schrijver propõem a rotina que detalhamos nesta seção. Tal rotina descreve como calcular limitantes inferiores para o st-potencial relativo a alguns pares s e t que são suficientes para garantir uma complexidade razoável ao seu algoritmo. O algoritmo de Schrijver está descrito na próxima seção.

Seja $\prec$ uma ordem linear de S. Dados elementos s e t distintos de S, defina

$$(s,t]_{\prec} := \{u \neq s\colon\ s \prec u \prec t\}. \tag{5.1}$$

Dados elementos distintos s e u de S, com $s \prec u$, denote por $\prec^{s,u}$ a ordem linear de S obtida quando redefinimos $v \prec u$ como $u \prec v$ para todo v tal que $s \prec v \prec u$. Equivalentemente, a ordem $\prec^{s,u}$ é obtida quando fazemos com que u preceda imediatamente s na ordem $\prec$. Note que, se u está em $(s,t]_{\prec}$, então $(s,t]_{\prec^{s,u}} = (s,t]_{\prec} \setminus \{u\}$.

A rotina proposta por Schrijver tem a seguinte especificação.

Rotina PIVOTA$(S, f, \prec, s, t)$: *Recebe uma função submodular f sobre S tal que $f(\emptyset) = 0$, uma ordem linear $\prec$ de S e dois elementos distintos s e t em S, com $s \prec t$, e devolve $\delta \geq 0$ e um vetor $\mu \geq 0$, indexado por $(s,t]_{\prec}$, tais que*

$$\begin{array}{ll} (i) & x^{\prec} + \delta(\chi^t - \chi^s) = \sum_{u \in (s,t]_{\prec}} \mu(u) x^{\prec^{s,u}} \\ (ii) & \sum_{u \in (s,t]_{\prec}} \mu(u) = 1. \end{array} \tag{5.2}$$

Observe que a rotina PIVOTA devolve uma representação do vetor $x^{\prec} + \delta(\chi^t - \chi^s)$, dada através do vetor μ e das ordens $\prec^{s,u}$, com u em $(s,t]_{\prec}$. As idéias para sua implemen-

tação estão no restante da seção.

Considere uma função submodular f sobre S, uma ordem linear $\prec$ de S e elementos distintos s e t em S, com $s \prec t$, que compõem a entrada da rotina PIVOTA. Começamos com observações importantes: para todo u em S, com $s \prec u$, temos

$$\begin{array}{ll} x^{\prec^{s,u}}(v) \leq x^{\prec}(v), & \text{se } s \prec v \prec u,\ v \neq u \\ x^{\prec^{s,u}}(v) \geq x^{\prec}(v), & \text{se } v = u, \\ x^{\prec^{s,u}}(v) = x^{\prec}(v), & \text{caso contrário,} \end{array} \tag{5.3}$$

para todo v em S. A prova das observações segue diretamente da forma como os vetores $x^{\prec}$ e $x^{\prec^{s,u}}$ são gerados respectivamente através das ordens $\prec$ e $\prec^{s,u}$ pelo método guloso, conforme (2.17), na seção 2.6.

Observamos também que as condições definidas em (5.2) valem para o escalar $\delta \geq 0$ e para o vetor $\mu \geq 0$, indexado por $(s,t]_{\prec}$, se, e somente se, vale também que

$$\delta(\chi^t - \chi^s) = \sum_{u \in (s,t]_{\prec}} \mu(u)\big(x^{\prec^{s,u}} - x^{\prec}\big), \tag{5.4}$$

com $\sum_{u \in (s,t]_{\prec}} \mu(u) = 1$.

Denote $(s,t]_{\prec}$ por $\{u_1, \ldots, u_k = t\}$, com $u_i \prec u_{i+1}$, para $1 \leq i < k$. Considere a matriz M com k linhas, em que a linha i representa $x^{\prec^{s,u_i}} - x^{\prec}$, para $1 \leq i \leq k$. Utilizando (5.3), podemos construir a seguinte representação das entradas de M. Nessa representação, um sinal "$+$" denota um valor ≥ 0 e um sinal "$-$" denota um valor ≤ 0.

								$M_{(s,t]}$							
	$u \prec s$			s	u_1			$\cdots$			$u_k = t$		$t \prec u$		
$x^{\prec^{s,u_1}} - x^{\prec}$	0	$\cdots$	0	$-$	$+$	0	$\cdots$	$\cdots$	$\cdots$	0	0	0	$\cdots$	0
	$\vdots$		$\vdots$	$-$	$-$	$+$	$\ddots$			$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$\vdots$		$\vdots$	$-$	$-$	$-$	$\ddots$	$\ddots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\ddots$	$\ddots$	0	0	$\vdots$		$\vdots$
	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$			$\ddots$	$+$	0	$\vdots$		$\vdots$
$x^{\prec^{s,u_k}} - x^{\prec}$	0	$\cdots$	0	$-$	$-$	$-$	$\cdots$	$\cdots$	$\cdots$	$-$	$+$	0	$\cdots$	0

Para obter (5.4), precisamos escrever $\delta(\chi^t - \chi^s)$ como combinação convexa das linhas de M. Chamemos de m_i a linha da matriz M que representa o vetor $x^{\prec^{s,u_i}} - x^{\prec}$, para cada $1 \leq i \leq k$. O elemento da linha m_i que está na coluna referente a cada u em S será denotado por $m_i(u)$.

Como $x^{\prec^{s,u}}(S) = x^{\prec}(S) = f(S)$, para todo $s \prec u \prec t$, já que todos os vetores considerados estão no polimatróide das bases $B(f)$, temos que a soma das entradas de cada linha de M é zero. Sabemos também, por (5.3), que o único elemento que pode ser positivo de cada linha m_i é $m_i(u_i) = (x^{\prec^{s,u_i}} - x^{\prec})(u_i)$.

Consideramos, então, dois casos: (i) $m_i(u_i) = 0$ para algum i e (ii) $m_i(u_i) > 0$ para todo i.

Se $m_i(u_i) = 0$ para algum i, $1 \leq i \leq k$, então toda a linha m_i é composta por zeros. Nesse caso, tomamos $\delta := 0$, de maneira que $\delta(\chi^t - \chi^s) = x^{\prec_{s,u_i}} - x^{\prec}$. A rotina PIVOTA$(S, f, \prec, s, t)$ pode, então, devolver $\delta = 0$ e o vetor μ definido por $\mu(u_i) := 1$ e $\mu(u) := 0$, se $u \neq u_i$.

Suponha agora que $m_i(u_i) > 0$ para todo i. Primeiramente, vamos escrever $\chi^t - \chi^s$ como combinação não-negativa das linhas de M. Chamando de $M_{(s,t]}$ a submatriz de M formada pelas colunas em $(s,t]_{\prec} = \{u_1, \ldots, u_k\}$, temos que o subsistema

$$yM_{(s,t]} = (0, \ldots, 0, 1) \tag{5.5}$$

tem solução única $\beta := \big(\beta(u_1), \ldots, \beta(u_k)\big) \geq \mathbf{0}$.

De fato, como $M_{(s,t]}$ é uma matriz triangular inferior, a solução do sistema acima é da forma:

$$\beta(u_k) := \tfrac{1}{m_k(u_k)} \quad \text{e} \quad \beta(u_i) := \tfrac{\sum_{i<j\leq k}\big(\beta(u_j)|m_j(u_i)|\big)}{m_i(u_i)},\ 1 \leq i < k. \tag{5.6}$$

Note que $\beta(u_k) > 0$ e $\beta(u_i) \geq 0$, para todo $1 \leq i < k$, uma vez que estamos considerando o caso em que todos os elementos da diagonal de $M_{(s,t]}$ são positivos e que os demais elementos não nulos da submatriz são não-positivos.

Mostramos agora que $\chi^t - \chi^s$ pode ser escrito como combinação das linhas de M com coeficientes β. Ou seja, denotando $\eta := \beta M$, vamos mostrar que $\eta = \chi^t - \chi^s$. Mais especificamente, queremos mostrar que $\eta(t) = 1$, $\eta(s) = -1$ e que $\eta(u) = 0$, para todo u em S tal que $u \neq s$ e $u \neq t$.

Como o vetor β é solução de (5.5), temos que $\eta(u_i) = 0$, para cada $1 \leq i < k$ e $\eta(u_k = t) = 1$. Além disso, para u tal que $u \prec s$ ou $t \prec u$, com $u \neq s$ e $u \neq t$, temos $m_i(u) = 0$, para todo $1 \leq i \leq k$, de maneira que $\eta(u) = \sum_{1\leq i\leq k} \beta(u_i) m_i(u) = 0$, como queríamos.

Resta mostrarmos que $\eta(s) = \sum_{1\leq i\leq k} \beta(u_i) m_i(s) = -1$. Como cada linha da matriz M soma zero, temos $m_i(s) = -\sum_{1\leq j\leq i} m_i(u_j)$, para todo i. Logo,

$$\begin{aligned}
\sum_{1\leq i\leq k} \beta(u_i) m_i(s) &= -\sum_{1\leq i\leq k} \beta(u_i) \Bigg(\sum_{1\leq j\leq i} m_i(u_j) \Bigg) \\
&= -\sum_{1\leq i\leq k} \Bigg(\sum_{1\leq j\leq i} \beta(u_i) m_i(u_j) \Bigg) \\
&= -\sum_{1\leq j\leq k} \Bigg(\sum_{j\leq i\leq k} \beta(u_i) m_i(u_j) \Bigg) \\
&= -\sum_{1\leq j\leq k} \eta(u_j) \\
&= -\eta(u_k) = -\eta(t) = -1,
\end{aligned} \tag{5.7}$$

também como queríamos. Note que, na terceira igualdade acima, deixamos de fazer a soma por linhas da matriz M e passamos a fazê-la por colunas.

Segue que $\chi^t - \chi^s$ pode ser escrito como combinação não-negativa das linhas de M com coeficientes β. Em particular, como $\beta(u_k) > 0$ e $\beta(u_i) \geq 0, 1 \leq i < k$, podemos definir $\delta := \left(\sum_{1 \leq i \leq k} \beta(u_i)\right)^{-1}$. Assim, temos que $\delta > 0$ e que $\delta(\chi^t - \chi^s)$ é combinação convexa de $x^{\prec_{s,u}} - x^{\prec}$, para $u \in (s,t]_{\prec}$, com coeficientes μ dados por $\mu(u) := \delta\beta(u)$. O escalar δ e o vetor μ podem, então, ser devolvidos pela rotina PIVOTA$(S, f, \prec, s, t)$.

Quando s e t recebidos pela rotina PIVOTA são consecutivos na ordem $\prec$, também recebida como entrada, estamos no caso em que sabemos calcular o st-potencial com relação a $x^{\prec}$ de maneira simples, conforme (4.10), na seção 4.3. Cabe observar que, nesse caso, o valor δ devolvido pela rotina PIVOTA é exatamente o st-potencial $\tilde{\alpha}(x^{\prec}, s, t)$. De fato, quando s precede imediatamente t na ordem $\prec$, temos $(s,t]_{\prec} = \{t\}$ de forma que, pelas descrições acima, PIVOTA$(S, f, \prec, s, t)$ devolve

$$\delta = x^{\prec_{s,t}}(t) - x^{\prec}(t) \quad \text{e} \quad \mu(t) = 1,$$

de maneira que $x^{\prec} + \delta(\chi^t - \chi^s) = x^{\prec_{s,t}}$. Disso concluímos, pelo teorema 2.13 e pela definição de st-potencial dada em (4.10), que

$$\delta = f\big([t]_{\prec} \setminus \{s\}\big) - x^{\prec}\big([t]_{\prec} \setminus \{s\}\big) = \tilde{\alpha}(x^{\prec}, s, t).$$

Esse fato é interessante pois mostra que não estamos "perdendo" quando aplicamos a rotina PIVOTA sobre pares s e t adjacentes no grafo de Cunningham.

Por fim, tecemos algumas considerações sobre o consumo de tempo da rotina PIVOTA. Um primeiro passo para sua implementação consiste na obtenção dos vetores $x^{\prec}$ e $x^{\prec_{s,u_i}}$, para $1 \leq i \leq k$. Cada um desses vetores pode ser obtido em tempo $O(|S|\gamma)$ através do método guloso (teorema 3.2), de maneira que o processo completo consome tempo $O(|S|^2\gamma)$. O segundo passo principal consiste em verificar se $m_i(u_i) > 0$ para todo i e, em caso afirmativo, obter o vetor β como em (5.6). Esse processo pode ser realizado em tempo $O(|S|^2)$. Feito isso, o escalar δ e o vetor μ podem ser obtidos em tempo proporcional a $|S|$, com a utilização do vetor β. Podemos, então, estabelecer o seguinte resultado.

Proposição 5.1: *A rotina* PIVOTA$(S, f, \prec, s, t)$ *pode ser implementada de modo a consumir tempo* $O(|S|^2\gamma)$. ∎

5.3 Descrição do algoritmo

O algoritmo de Schrijver para minimização de funções submodulares pode ser facilmente interpretado como uma extensão do método de Cunningham, descrito na seção 4.4.

Para conseguir aplicar uma abordagem de caminhos mínimos ao método e demonstrar que seu algoritmo pára após um número polinomial de aumentos, Schrijver considera mais arcos, além daqueles já presentes no grafo de Cunningham, em seu grafo auxiliar. Dados uma função submodular f sobre S e uma representação de um vetor x em $B(f)$, o **grafo de Schrijver para** f **e** x, denotado por $D_S(f, x)$, é também um grafo orientado e

definido sobre o conjunto base S da função f. Seu conjunto de arcos é dado por

$$A_S := \Big\{(s,t)\colon \exists i \in I_x,\ s \prec_i t,\ s \neq t\Big\}. \tag{5.8}$$

Ou seja, o grafo de Schrijver $D_S(f,x)$ possui um arco de s a t se, e somente se, s precede t em alguma ordem que compõe a representação mantida de x.

Retomando a definição de st-potencial dada em (4.9), temos que se t precede s em alguma ordem $\prec_i$ que compõem a representação de x, então o st-potencial com relação a $x^{\prec_i}$ não pode ser positivo, uma vez que, nesse caso, $[t]_{\prec_i}$ é um conjunto $x^{\prec_i}$-justo que contém t e não contém s. Portanto, Schrijver apenas não põe em seu grafo arcos ligando s e t quando já se sabe a priori que não se poderia obter ganhos pivotando s e t. Note também que os arcos do grafo de Cunningham estão contidos no grafo de Schrijver.

Além disso, sempre que s e t estão ligados por um arco no grafo de Schrijver, através da rotina PIVOTA, descrita na seção anterior, podemos calcular limitantes inferiores para o st-potencial com relação a algum $x^{\prec_i}$ que compõe a representação de x.

Implementação das rotinas ESCOLHE-CAMINHO e AUMENTA

Explicamos aqui, em linhas gerais, como Schrijver implementou as rotinas ESCOLHE-CAMINHO e AUMENTA, especificadas na seção 4.4, durante a descrição do método de Cunningham. Mais abaixo apresentamos pseudo-códigos com essas implementações. Nas próximas seções, detalhamos como as implementações sugeridas por Schrijver garantem uma complexidade fortemente polinomial ao algoritmo.

Cada iteração do algoritmo de Schrijver também começa definindo os conjuntos $P := \{s \in S\colon x(s) > 0\}$ e $N := \{s \in S\colon x(s) < 0\}$.

Para realizar o passo aumentador e obter x' tal que $x'^{-}(S) \geq x^{-}(S)$, Schrijver escolhe sempre caminhos mínimos de P a N no grafo auxiliar $D_S(f,x)$. Para cada vértice u do grafo $D_S(f,x)$, denote por $d(u)$ o comprimento de um caminho mínimo de P a u em $D_S(f,x)$. Se não existe um caminho de P a u, definimos $d(u) := \infty$. Dizemos que $d(u)$ é a **distância** de P a u em $D_S(f,x)$. Se t é a ponta final de um caminho Q de P a N, dizemos que Q é um **caminho mínimo** de P a N se seu comprimento é exatamente igual a $d(t)$.

Assim como fez Cunningham [Cun84] em seu algoritmo para o problema da pertinência no politopo dos matróides (cujas principais idéias estão descritas na seção 4.1), Schrijver utiliza uma regra lexicográfica para selecionar um caminho mínimo em determinada iteração. Antes mesmo do trabalho de Cunningham sobre matróides, tal técnica já aparecia em trabalhos de Lawler e Martel [LM82] e Schönsleben [Sch80] sobre intersecção de matróides.

Para escolher os caminhos de forma "consistente" em cada iteração, de modo a garantir progressos, Schrijver renomeia os elementos do conjunto base S como $\{1,\ldots,|S|\}$ logo no início do algoritmo, de maneira arbitrária. Feito isso, a cada iteração, escolhe o caminho mínimo Q de P a N que é lexicograficamente maior com relação à tripla $(d(t_Q), t_Q, s_Q)$ dentre todos os demais caminhos mínimos de P a N, em que (s_Q, t_Q) representa o último

arco do caminho Q.

Escolhido um caminho no grafo $D_S(f,x)$, Schrijver utiliza apenas seu último arco para realizar o passo aumentador. Tal passo é realizado com o auxílio da rotina PIVOTA, que encontra um limitante inferior para o st-potencial com relação a algum $x^{\prec_i}$ na representação de x, sendo (s,t) o último arco do caminho escolhido para a realização do aumento. A representação de $x^{\prec_i} + \delta(\chi^t - \chi^s)$ devolvida pela rotina PIVOTA$(S, f, \prec_i, s, t)$ é utilizada, então, para a obtenção de uma representação para $y = x + \lambda_i\delta(\chi^t - \chi^s)$.

Observe que, dependendo do valor de $\lambda_i\delta$, $y(s)$ pode decrescer muito com relação a $x(s)$, o que poderia resultar em um decréscimo em $y^-(S)$ quando comparado a $x^-(S)$. Isso não estaria de acordo com o desejado, uma vez que buscamos nos aproximar do valor máximo de $x^-(S)$ a cada iteração, lembrando que o algoritmo trabalha sobre a relação min-max (4.3).

Para evitar esse problema, Schrijver realiza ainda uma última operação após o aumento: ao invés de substituir a representação de x pela representação de y e iniciar nova iteração, substitui x pela representação de um vetor x' que é uma combinação convexa apropriada de x e y, conforme especificado nos pseudo-códigos que descrevem o algoritmo de Schrijver abaixo.

Algoritmo

Algoritmo SCHRIJVER(S, f): *Recebe uma função submodular f sobre S tal que $f(\emptyset) = 0$ e devolve uma parte W de S que minimiza f.*

Cada iteração do algoritmo começa com uma representação de um vetor x em $B(f)$, conforme (4.5), composta por

- um conjunto de índices I_x, com $|I_x| \leq |S| + 1$;
- para cada i em I_x, um número real positivo λ_i;
- para cada i em I_x, uma ordem linear $\prec_i$ de S;

tais que $x = \sum_{i\in I_x} \lambda_i x^{\prec_i}$ e $\sum_{i\in I_x} \lambda_i = 1$.

Assim como no método CUNNINGHAM, na primeira iteração, temos $I_x := \{1\}$, $\lambda_1 := 1$ e $\prec_1$ sendo uma ordem linear qualquer de S, de maneira que $x = x^{\prec_1}$. Além disso, com o objetivo de realizar a escolha consistente dos caminhos de aumento e garantir um consumo de tempo razoável, Schrijver ordena os elementos de S de maneira arbitrária, porém fixa, e os renomeia como $\{1, \ldots, |S|\}$, logo no início do algoritmo.

Segue a descrição de uma iteração qualquer do algoritmo SCHRIJVER. Sejam

$P := \{s \in S\colon x(s) > 0\}$; $N := \{s \in S\colon x(s) < 0\}$ e $D_S := D_S(f,x)$.

Caso 1: Não existe caminho de P a N em D_S.
Seja W o conjunto de vértices que acessam N em D_S.
Devolva W.

Caso 2: Existe um caminho de P a N em D_S.
$Q \leftarrow$ ESCOLHE-CAMINHO-SCHRIJVER(D_S, P, N).
$x'' \leftarrow$ AUMENTA-SCHRIJVER(S, f, x, Q).
$x' \leftarrow$ COMPACTA(S, f, x'').
Comece nova iteração com x' no papel de x.

No pseudo-código acima, chamamos de ESCOLHE-CAMINHO-SCHRIJVER e AUMENTA-SCHRIJVER respectivamente as implementações sugeridas por Schrijver para as rotinas ESCOLHE-CAMINHO e AUMENTA. As descrições e implementações dessas rotinas estão a seguir. A rotina COMPACTA é a mesma utilizada pelo método CUNNINGHAM, especificada na seção 4.4.

Rotina ESCOLHE-CAMINHO-SCHRIJVER(D_S, P, N): *Recebe um grafo D_S e dois subconjuntos de vértices P e N de D_S e devolve um caminho mínimo Q de P a N nesse grafo.*

Seja $d(u)$ a distância de P a u em D_S, para cada vértice u.
Seja $\mathcal{Q}$ a família de todos os caminhos mínimos de P a N em D_S.
Para cada caminho Q em $\mathcal{Q}$, seja (s_Q, t_Q) seu último arco.
Dentre todos os caminhos em $\mathcal{Q}$, escolha Q tal que

$$\big(d(t_Q), t_Q, s_Q\big) \geq \big(d(t_{Q'}), t_{Q'}, s_{Q'}\big)$$

para todo caminho Q' em $\mathcal{Q}$.
Devolva Q.

Rotina AUMENTA-SCHRIJVER(S, f, x, Q): *Recebe a representação de um vetor x em $B(f)$ e um caminho Q em $D_S(f, x)$ e devolve a representação de um vetor x' em $B(f)$, obtida a partir do último arco (s, t) de Q, tal que*
(i) $x' = x + \delta(\chi^t - \chi^s)$, para algum $\delta \geq 0$;
(ii) $x'^{-}(S) \geq x^{-}(S)$.

Seja (s, t) o último arco do caminho Q.
Seja $\alpha := \max\big\{|(s, t]_{\prec_i}| : i \in I_x\big\}$.
Seja j um índice em I_x tal que $|(s, t]_{\prec_j}| = \alpha$.
$(\delta, \mu) \leftarrow$ PIVOTA$(S, f, \prec_j, s, t)$.
Utilizando a representação de x, bem como δ e μ, obtenha uma representação para

$$y = x + \lambda_j \delta(\chi^t - \chi^s)$$

como combinação convexa de $x^{\prec_i}$, $i \neq j$, i em I_x, e $x^{\prec_j^{s,u}}$, $u \in (s, t]_{\prec_j}$.
Seja x' o ponto mais próximo de y no segmento de reta $\overline{xy}$ tal que $x'(t) \leq 0$.

Utilizando as representações de x e y, obtenha uma representação para x' como combinação convexa de $x^{\prec_i}$, i em I_x, e $x^{\prec_j^{s,u}}$, $u \in (s,t]_{\prec_j}$.
Devolva x'.

5.4 Correção do algoritmo

Pela descrição do algoritmo SCHRIJVER na seção anterior, quando não existem mais caminhos de P a N no grafo de Schrijver $D_S(f,x)$, o algoritmo pára devolvendo o conjunto de vértices que acessam N em $D_S(f,x)$.

Sabemos que o grafo de Schrijver $D_S(f,x)$ contém todos os arcos do grafo de Cunningham $D(f,x)$, definido na seção 4.3. Portanto, todo caminho no grafo $D(f,x)$ é também um caminho no grafo de Schrijver $D_S(f,x)$. Por outro lado, se existe um arco (s,t) no grafo de Schrijver, pela maneira como os grafos são construídos, é fácil ver que existe um caminho de s a t no grafo de Cunningham. Concluímos, assim, que existe um caminho entre dois vértices quaisquer em $D_S(f,x)$ se, e somente se, existe um caminho entre esses mesmos vértices no grafo $D(f,x)$. Logo, se o caso 1 ocorre no algoritmo SCHRIJVER, o conjunto W devolvido pelo algoritmo é exatamente o conjunto de vértices que acessam N no grafo de Cunningham $D(f,x)$.

Assim, o teorema 4.5, que garante a correção do método CUNNINGHAM, juntamente com a relação min-max (4.3), nos garantem que, se o algoritmo SCHRIJVER pára, então ele de fato devolve um minimizador de f. Podemos, portanto, estabelecer o seguinte resultado.

Teorema 5.2: *Suponha que o caso 1 ocorra em determinada iteração do algoritmo* SCHRIJVER(S,f). *Seja W o conjunto de vértices que acessam N em $D_S(f,x)$, conforme definido nessa iteração. Então $x^-(S) = f(W)$, de maneira que W minimiza f.* ∎

Na próxima seção limitamos o número de iterações do algoritmo, de forma a demonstrar que o caso 1 ocorre em algum momento.

5.5 Consumo de tempo

No trabalho em que descreve seu algoritmo, Schrijver mostra que um minimizador para a função submodular é encontrado após $O(|S|^6)$ iterações. Pouco tempo depois, Vygen [Vyg03] refinou a análise de complexidade do algoritmo e mostrou que o número de iterações poderia ser limitado como $O(|S|^5)$. A demonstração de consumo de tempo que apresentamos a seguir baseia-se principalmente no trabalho de Vygen. Como sempre, denotamos $n := |S|$.

Dada uma iteração qualquer do algoritmo, considere a representação do vetor x em $B(f)$ mantida, além dos objetos D_S, P, N, d, s, t, α, j e x' definidos durante a iteração. Os objetos D_S, P e N são definidos logo no início da iteração; d, s e t são determinados pela rotina

ESCOLHE-CAMINHO-SCHRIJVER e α, j e x' são definidos por AUMENTA-SCHRIJVER. Defina também

$$\beta := \left|\left\{i \in I_x \colon |(s,t]_{\prec_i}| = \alpha\right\}\right|,$$

como o número de ordens que podem ser escolhidas para a aplicação da subrotina PIVOTA durante a iteração, na rotina AUMENTA-SCHRIJVER. O objeto β é definido implicitamente em cada iteração.

Note que $\beta \leq n+1$ ao longo de todo o algoritmo, uma vez que o número de termos na combinação convexa que descreve x é sempre reduzido a $n+1$ ao final de cada iteração, com o auxílio da rotina COMPACTA.

Vamos chamar respectivamente de D'_S, P', N', d', α' e β' os objetos D_S, P, N, d, α e β definidos no início da iteração seguinte.

Apresentamos a seguir uma série de resultados que nos ajudarão a limitar o número de iterações executadas pelo algoritmo até que um minimizador para a função submodular f seja encontrado.

Lema 5.3: *Se um arco (v,w) não está no grafo D_S no início de determinada iteração, mas está no grafo D'_S no início da iteração seguinte, então*

$$s \prec_j w \prec_j v \prec_j t.$$

Demonstração: Como (v,w) não é um arco de D_S no início da iteração, então $w \prec_j v$. Por outro lado, como (v,w) está em D'_S, então deve existir u em $(s,t]_{\prec_j}$ tal que $\prec_j^{s,u}$ está na representação obtida do vetor x' e tal que $v \prec_j^{s,u} w$. Lembre que ordens da forma $\prec_j^{s,u}$ são as únicas que podem estar na representação de x' e não na de x. Por fim, da definição de $\prec_j^{s,u}$, concluímos que devemos ter $u = v$, além de $s \prec_j w \prec_j v \prec_j t$. ∎

Lema 5.4: *A distância de P a qualquer vértice v em S nunca diminui ao longo das iterações. Isto é,*

$$d'(v) \geq d(v), \text{ para todo } v \in S. \tag{5.9}$$

Demonstração: Pela descrição do algoritmo SCHRIJVER, temos que $P' \subseteq P$. De fato, observe que t é o único elemento em S tal que $x'(t) \geq x(t)$. Mas, como vale também que $x'(t) \leq 0$, pela própria definição de x' durante a iteração, t não faz parte de P' no início da iteração seguinte. Logo, se (5.9) não vale, ou seja, se alguma distância diminuiu, deve existir um arco (v,w) em D'_S que não está em D_S e tal que $d(w) \geq d(v)+2$. Agora, pelo lema anterior, devemos ter $s \prec_j w \prec_j v \prec_j t$. Além disso, temos também que $d(s)+1 = d(t)$, uma vez que (s,t) é escolhido como o último arco de um caminho mínimo de P a t no início da rotina AUMENTA-SCHRIJVER. Disso concluímos

$$d(w) \leq d(s)+1 = d(t) \leq d(v)+1,$$

o que é uma contradição. Segue a validade de (5.9). ∎

A partir dos lemas anteriores podemos concluir que, intuitivamente, todo arco que é adicionado ao grafo de Schrijver de uma iteração para a seguinte está "voltando" e, portanto, não pode reduzir o comprimento de um caminho mínimo nesse grafo.

Chamemos de **bloco** uma seqüência de iterações consecutivas em que o mesmo par s e t é selecionado para a aplicação da rotina PIVOTA$(S, f, \prec_j, s, t)$. Limitamos a seguir o número de iterações de um bloco, bem como o número total de blocos executados pelo algoritmo.

Lema 5.5: *O número de iterações em um bloco é $O(n^2)$.*

Demonstração: Considere duas iterações consecutivas do bloco. Vamos mostrar que $(\alpha', \beta') < (\alpha, \beta)$, no sentido lexicográfico, em que α e β são os objetos definidos pelo algoritmo na primeira iteração considerada e α' e β' são os mesmos objetos definidos na iteração seguinte. Como α e β são limitados respectivamente por n e $n+1$, disso concluiremos o resultado.

Primeiramente, observe que se uma ordem linear de S compõe a representação de x' mas não compõe a representação mantida de x, então ela é da forma $\prec_j^{s,u}$, para algum u em $(s,t]_{\prec_j}$. Como $(s,t]_{\prec_j^{s,u}}$ é sempre um subconjunto próprio de $(s,t]_{\prec_j}$ pela própria definição, segue que $\alpha' \leq \alpha$.

Vamos supor, então, que $\alpha' = \alpha$. Como o mesmo par s e t é selecionado na iteração seguinte do bloco, então $x'(t) < 0$, já que t é a ponta final de um caminho de P' a N' em D'_S. Nesse caso, na iteração em que x' foi obtido, vale que $x' = y$. Lembre-se que x' é o ponto mais próximo de y com $x'(t) \leq 0$ no segmento $\overline{xy}$. Segue que o ponto extremo $x^{\prec_j}$ já não está na representação mantida de x'. Além disso, conforme observamos acima, qualquer ordem $\prec_j^{s,u}$ que esteja na representação de x' e não esteja na representação de x é tal que $|(s,t]_{\prec_j^{s,u}}| < \alpha$. Segue que

$$\beta' = \big|\{i \in I_{x'} \colon |(s,t]_{\prec_i}| = \alpha\}\big| < \big|\{i \in I_x \colon |(s,t]_{\prec_i}| = \alpha\}\big| = \beta,$$

de onde segue o resultado. ∎

Lema 5.6: *Se $d'(v) = d(v)$, para todo v em S, então o número*

$$d^* := \max\big\{d(v) \colon v \in N\big\}$$

não aumenta de uma iteração para a seguinte. Além disso, enquanto d^* é constante, nenhum elemento é adicionado ao conjunto de elementos que atingem $d(v)$ máximo.

Demonstração: O único elemento que pode estar em $N' \setminus N$ é s. De fato, note que s é o único elemento tal que $x'(s) \leq x(s)$. Logo, o único elemento que poderia passar a fazer parte do conjunto de elementos que atingem $d(v)$ máximo é s. Agora, como $d(s) < d(t)$ pela escolha do par s e t pelo algoritmo, se as distâncias não mudam de uma iteração

para a seguinte, então o máximo d^* também não aumenta. Além disso, se d^* se manteve constante, nenhum elemento novo passa a atingir o máximo na próxima iteração. ∎

Lema 5.7: *Para todo t^* em S, existem $O(n^2)$ iterações tais que $t = t^*$ e $x'(t^*) = 0$.*

Demonstração: Se $x'(t^*) = 0$, então t^* saiu de N ao final da iteração. Pelo lema 5.4, temos que a distância $d(t^*)$ de P a t^* não cai ao longo das iterações. Além disso, quando t^* é escolhido como t em determinada iteração, temos $d(t^*) = \max\{d(v) : v \in N\}$. Logo, pelo lema anterior, antes que t^* seja escolhido novamente como t em alguma iteração, devem ocorrer duas iterações consecutivas tais que $d'(v) > d(v)$, para algum v em S. Ou seja, entre duas iterações tais que $t = t^*$ e $x'(t) = 0$, pelo menos um vértice deve ter sua distância com relação a P aumentada. Como $d(v) < n$, para todo v em S, chegamos ao limitante desejado. ∎

Dados v e t em S, chamamos v de ***t*-boring** se (v, t) não é um arco do grafo de Schrijver $D_S(f, x)$ ou se $d(t) \leq d(v)$.

Lema 5.8: *Sejam s^* e t^* elementos de S e considere o período entre uma iteração tal que $s = s^*$ e $t = t^*$ e o próximo aumento em $d(t^*)$. Então*

(i) qualquer elemento $v > s^*$ de S é t^*-boring durante todo o período;
(ii) se s^* se torna t^*-boring em algum momento do período, então permanece t^*-boring até o final do período.

Demonstração: A prova será por indução no número de iterações do período.

Na primeira iteração, todo $v > s^*$ é t^*-boring em função da escolha $s = s^*$ pelo algoritmo. De fato, como $v > s^*$ e v não foi escolhido no lugar de s^*, então ou (v, t^*) não é um arco de D_S ou $d(v) \geq d(t^*)$.

Como $d(t^*)$ se mantém constante durante todo o período e como $d(v)$ nunca diminui, para todo v, precisamos analisar apenas o caso em que o arco (v, t^*) é inserido no grafo D_S, com $v > s^*$, ou o caso em que o arco (s^*, t^*) deixa de existir em D_S e depois é inserido novamente. Ou seja, supondo que $v \geq s^*$ seja t^*-boring no início da iteração e que o arco (v, t^*) seja inserido no grafo, queremos mostrar que v continua t^*-boring ao final da iteração. Em particular, queremos provar que $d'(t^*) \leq d'(v)$ vale no início da iteração seguinte.

Suponha que o arco (v, t^*) seja inserido no grafo em uma iteração que escolheu o par s e t para realizar a pivotação. Pelo lema 5.3, temos

$$s \prec_j t^* \prec_j v \prec_j t \tag{5.10}$$

e, portanto,

$$d(t^*) \leq d(s) + 1 = d(t) \leq d(v) + 1. \tag{5.11}$$

Se $s > v$, então s é t^*-boring no início da iteração pela hipótese de indução. Isso implica

que

$$d(s) \geq d(t^*), \tag{5.12}$$

já que, por (5.10), ou $t^* = s$ ou (s, t^*) é um arco de D_S. De (5.11) e (5.12) segue que $d(t^*) \leq d(v)$. Logo, como $d'(t^*) = d(t^*)$ e $d'(v) \geq d(v)$, temos $d'(t^*) \leq d'(v)$, exatamente como queríamos.

Por outro lado, se $s < v$, então, pela escolha de s, temos que ou $t = v$ ou $d(v) \geq d(t)$, já que o arco (v, t) está em D_S (novamente por (5.10)). Segue que $d(v) \geq d(t)$ e, também pela equação (5.11), que $d(t^*) \leq d(v)$. Disso concluímos, assim como no caso anterior, que $d'(t^*) \leq d'(v)$ e segue o resultado. ∎

Lema 5.9: *O algoritmo* SCHRIJVER *executa* $O(n^3)$ *blocos.*

Demonstração: Quando estamos na última iteração de um bloco, isto é, quando o par (s, t) selecionado para realizar a pivotação é diferente do par (s', t') a ser selecionado na iteração seguinte, ocorre uma das seguintes situações:

(1) t é removido de N ao final da iteração.
Pela descrição do algoritmo, temos que $x'(t) \leq 0$. Logo, para que t seja removido de N devemos ter $x'(t) = 0$ ao final da iteração. Assim, pelo lema 5.7, existem $O(n^2)$ iterações em que determinado vértice t pode ser escolhido como t e sair de N nessa mesma iteração. Portanto, como o número de vértices candidatos a t é limitado por n, no máximo $O(n^3)$ blocos podem terminar dessa forma.

(2) O arco (s, t) foi removido do grafo $D_S(f, x)$.
Se o arco (s, t) é removido do grafo $D_S(f, x)$, então s se torna t-boring. Ademais, enquanto s é t-boring, o par s e t não será novamente escolhido para a realização de uma pivotação. Agora, pelo lema 5.8, s continuará t-boring enquanto $d(t)$ se mantiver constante. Portanto, para que o par s e t seja novamente escolhido, $d(t)$ deve aumentar. Como $d(t)$ é limitado por n, temos $O(n)$ blocos terminando com determinado arco (s, t) fixo sendo removido do grafo D_S. Variando s e t, temos $O(n^3)$ blocos terminando dessa forma.

(3) A distância $d(v)$ foi alterada para algum v em S.
Como as distâncias nunca diminuem e como $d(v)$ é limitado por n para cada v, $O(n^2)$ blocos podem terminar dessa maneira.

Segue que o algoritmo SCHRIJVER executa $O(n^3)$ blocos. ∎

Pelos lemas 5.5 e 5.9, temos que o algoritmo SCHRIJVER executa $O(n^3)$ blocos, cada um com $O(n^2)$ iterações. Podemos, portanto, estabelecer o seguinte resultado.

Teorema 5.10 (do número de iterações de SCHRIJVER):
O algoritmo SCHRIJVER(S, f) *encontra um minimizador para a função submodular* f *em* $O(n^5)$ *iterações.* ∎

Por fim, estabelecemos um limitante para o consumo de tempo total do algoritmo.

Teorema 5.11 (do consumo de tempo de SCHRIJVER):
O algoritmo SCHRIJVER(S, f) *é fortemente polinomial e pode ser implementado de modo a consumir tempo* $O(n^8 + n^7\gamma)$.

Demonstração: Em cada iteração, o grafo de Schrijver pode ser construído em tempo proporcional a n^2. Além disso, o caminho a partir do qual o algoritmo realizará o aumento também pode ser encontrado em tempo $O(n^2)$ através de uma adaptação de uma busca em largura nesse grafo. Conforme proposições 5.1 e 4.3, temos que as rotinas PIVOTA e COMPACTA podem ser implementadas respectivamente de modo a consumir tempo $O(n^2\gamma)$ e $O(n^3)$. Para a garantia do consumo de tempo da rotina COMPACTA, observe que o número de pontos extremos na combinação convexa que descreve x aumenta em no máximo n durante uma iteração, de maneira que permanece $O(n)$ ao longo de todo o algoritmo. Segue que uma iteração do algoritmo SCHRIJVER pode ser implementada de modo a consumir tempo $O(n^3 + n^2\gamma)$. O resultado segue, então, do teorema 5.10 acima. ∎

5.6 Arcabouço push-relabel

Goldberg e Tarjan [GT88] propuseram uma abordagem alternativa aos caminhos de aumento para o problema do fluxo máximo, conhecida como **push-relabel**. Através dessa abordagem, desenvolveram algoritmos mais eficientes e também mais flexíveis, no sentido de que puderam ser estendidos para outros problemas relacionados, como, por exemplo, problemas sobre intersecção de polimatróides (Fujishige e Zhang [FZ92]) e problemas de fluxo máximo paramétrico (Gallo, Grigoriadis e Tarjan [GGT89]).

Baseando-se nas idéias do algoritmo de Schrijver e na abordagem de Goldberg e Tarjan, Fleischer e Iwata [FI03] desenvolveram um arcabouço push-relabel para o problema de minimização de funções submodulares. O algoritmo proposto tem consumo de tempo equivalente ao do algoritmo de Schrijver (quando consideramos a análise de Vygen, exposta na seção anterior). Entretanto, Fleischer [Fle00] afirma que, em função de sua maior modularidade, a versão push-relabel do algoritmo pode ser mais facilmente estendida para a resolução de outros problemas mais gerais, como aconteceu com o algoritmo de Goldberg e Tarjan para fluxo máximo. De fato, utilizando o arcabouço que propõem, Fleischer e Iwata [FI00a] apresentam o primeiro algoritmo polinomial para o problema de fluxos submodulares máximos que não depende de uma rotina para minimizar funções submodulares. A seção 7.2 traz uma breve introdução sobre fluxos submodulares.

Na abordagem push-relabel para o problema do fluxo máximo, os algoritmos mantêm vértices com **excesso** de fluxo. Isto é, vértices que possivelmente possuem mais fluxo entrando do que saindo. Iterativamente, os excessos vão sendo "escoados" para os demais vértices, com o auxílio de operações **push** e **relabel**, de maneira que se obtenha um fluxo viável quando o algoritmo pára.

De maneira semelhante e baseado também na relação min-max (4.3):

$$\max\{x^-(S)\colon x \in B(f)\} = \min\{f(U)\colon U \subseteq S\},$$

o algoritmo push-relabel para funções submodulares procura "escoar" o excesso nos vértices s com $x(s) > 0$ para vértices t com $x(t) < 0$. Para tanto, se utiliza de operações PUSH, RELABEL e da rotina PIVOTA, descrita na seção 5.2.

Chamaremos de PUSH-RELABEL o algoritmo push-relabel de Fleischer e Iwata para minimização de funções submodulares. As idéias que permeiam o algoritmo estão descritas com mais detalhes no restante da seção.

Rótulos distância

Seja f uma função submodular tal que $f(\emptyset) = 0$. Novamente definimos $n := |S|$. Assim como o algoritmo de Schrijver, o algoritmo PUSH-RELABEL mantém uma representação de um vetor x em $B(f)$ e trabalha sobre o grafo de Schrijver $D_S(f,x)$, definido na seção 5.3. O algoritmo considera também as partes $P := \{s \in S\colon x(s) > 0\}$ e $N := \{s \in S\colon x(s) < 0\}$ de S. Além disso, de maneira semelhante aos algoritmos push-relabel para o problema do fluxo máximo, trabalha com funções-distância válidas sobre o conjunto de vértices do grafo $D_S(f,x)$.

Uma **função-distância** é uma função que atribui um número inteiro a cada vértice do grafo $D_S(f,x)$. Se d é uma função-distância e u é um elemento de S e, portanto, um vértice de $D_S(f,x)$, dizemos que $d(u)$ é o seu **rótulo**. De forma mais geral, dizemos que d representa os **rótulos** dos vértices do grafo $D_S(f,x)$.

Uma função-distância $d\colon S \to \mathbb{Z}$ é **válida** se $d(t) = 0$ para todo vértice t em N e $d(u) \leq d(v) + 1$ para todo arco (u,v) no grafo $D_S(f,x)$. Se d é uma função-distância válida, dizemos também que d representa **rótulos válidos** para os vértices de $D_S(f,x)$. Observe que a função d tem aqui um significado diferente do apresentado na seção 5.3, em que $d(u)$ representava a distância entre P e um vértice qualquer u no grafo $D_S(f,x)$.

Rótulos válidos representam um limitante inferior para o comprimento de um caminho mínimo de u a N, para qualquer vértice u em $D_S(f,x)$, conforme estabelece o próximo resultado.

Proposição 5.12: *Seja d uma função-distância sobre os vértices do grafo $D_S(f,x)$. Se d é válida, então $d(u)$ é um limitante inferior para a distância de u a N em $D_S(f,x)$.*

Demonstração: Seja Q um caminho de u a N em $D_S(f,x)$ com seqüência de vértices $u = u_1, u_2, \ldots, u_m = t$, em que t é um elemento de N. Como os rótulos d são válidos, temos $d(u_i) - d(u_{i+1}) \leq 1$, para todo $1 \leq i < m$, além de $d(t) = 0$, de maneira que

$$|Q| = m - 1 \geq \sum_{i=1}^{m-1} d(u_i) - d(u_{i+1}) = d(u) - d(t) = d(u).$$

Ou seja, $d(u)$ limita inferiormente o comprimento de qualquer caminho Q de u a N em $D_S(f,x)$. ∎

Pushes e relabels

O algoritmo PUSH-RELABEL é iterativo e mantém uma função-distância válida d, além da representação de um vetor x em $B(f)$. Suas operações básicas são definidas pelas rotinas PUSH e RELABEL. Ambas as operações são aplicadas somente sobre vértices s de $D_S(f,x)$ que estão em em P e tais que $d(s) < n$.

O algoritmo também é baseado na relação min-max (4.3). Desse modo, busca-se novamente aumentar o valor de $x^-(S)$, reduzindo os valores de componentes s de x tais que $x(s) > 0$ e aumentando os valores de componentes t de x tais que $x(t) < 0$.

Defina $R := \{s\colon s \in P,\ d(s) < n\}$.

Um PUSH(S,f,x,s,t) é aplicado sobre um par de vértices s e t tais que $d(s) = d(t) + 1$, com s em R, quando existe um arco (s,t) em $D_S(f,x)$. O objetivo de uma operação PUSH é obter um vetor x' tal que $x'(s) = 0$ ou tal que (s,t) já não seja um arco em $D_S(f,x')$.

Fazendo uma analogia com o problema do fluxo máximo, podemos imaginar que, intuitivamente, um PUSH procura "empurrar" o excesso no vértice s com $x(s) > 0$ para algum vértice t que esteja mais próximo de um vértice t' tal que $x(t') < 0$ no grafo auxiliar.

A especificação da rotina PUSH(S,f,x,s,t) está a seguir.

> **Rotina** PUSH(S,f,x,s,t): *Recebe uma função submodular f sobre S tal que $f(\emptyset) = 0$, uma representação de um vetor x em $B(f)$ e dois elementos distintos s e t em S. Devolve a representação de um vetor x' em $B(f)$ tal que $x'(s) = 0$ ou $t \prec_i s$ para todo i em I_x.*

A rotina PUSH é iterativa e mantém a representação de um vetor x em $B(f)$. A cada iteração, seleciona-se j em I_x tal que o tamanho do intervalo $(s,t]_{\prec_j}$ seja máximo. Aplica-se, então, a subrotina PIVOTA$(S,f,\prec_j,s,t)$, definida na seção 5.2, de maneira a se obter $\delta \geq 0$ e uma representação de $x^{\prec_j} + \delta(\chi^t - \chi^s)$ como combinação convexa de $x^{\prec_j^{s,u}}$, para u em $(s,t]_{\prec_j}$. Feito isso, define-se $x' := x + \epsilon(\chi^t - \chi^s)$, com $\epsilon := \min\{x(s), \lambda_j\delta\}$. A definição de ϵ se justifica pois não queremos que $x'(s)$ se torne negativo. Ao final da iteração, o número de elementos na combinação convexa que compõe a representação de x' é reduzido a no máximo $n+1$, através da rotina COMPACTA, especificada na seção 4.4.

Essa seqüência de passos é repetida até que se obtenha $x'(s) = 0$ ou até que (s,t) deixe de pertencer ao grafo $D_S(f,x')$, o que ocorre quando se tem $t \prec_i s$ para toda ordem $\prec_i$ que compõe a representação de x'.

Ao final da execução da rotina, se o arco (s,t) não está no grafo $D_S(f,x')$, dizemos que o PUSH foi **saturador**. Caso contrário, dizemos que ocorreu um PUSH **não-saturador**.

O pseudo-código abaixo descreve a rotina PUSH(S,f,x,s,t).

Caso 1: $x(s) = 0$ ou $t \prec_i s$ para todo i em I_x.
Devolva x.

Caso 2: $x(s) > 0$ e $s \prec_i t$ para algum i em I_x.
Seja $\alpha := \max\{|(s,t]_{\prec_i}| : i \in I_x\}$.
Seja j um índice em I_x tal que $|(s,t]_{\prec_j}| = \alpha$.
$(\delta, \mu) \leftarrow \text{PIVOTA}(S, f, \prec_j, s, t)$.
$\epsilon \leftarrow \min\{x(s), \lambda_j \delta\}$.
Utilizando a representação de x, bem como δ e μ, obtenha uma representação para

$$x'' = x + \epsilon(\chi^t - \chi^s),$$

como combinação convexa de $x^{\prec_i}$, i em I_x, e $x^{\prec_j^{s,u}}$, u em $(s,t]_{\prec_j}$.
$x' \leftarrow \text{COMPACTA}(S, f, x'')$.
Comece nova iteração com x' no papel de x.

No código acima, se $\epsilon = x(s)$ em determinada iteração, então o caso 1 ocorre na iteração seguinte e a rotina PUSH termina. Caso contrário, temos que, após cada chamada da rotina PIVOTA, $\max_{i \in I_x} |(s,t]_{\prec_i}|$ diminui ou o número de índices em I_x que atingem esse máximo diminui. Lembre-se que após uma aplicação de $\text{PIVOTA}(S, f, \prec_j, s, t)$ trocamos a ordem $\prec_j$ por ordens $\prec_j^{s,u}$, para u em $(s,t]_{\prec_j}$ na representação mantida de x. Além disso, sabemos que $|(s,t]_{\prec_j^{s,u}}| < |(s,t]_{\prec_j}|$, para todo u em $(s,t]_{\prec_j}$. Disso podemos concluir o seguinte resultado.

Proposição 5.13: *A rotina* PUSH *invoca* $O(n^2)$ *vezes a rotina* PIVOTA. ∎

Quando vale $d(s) \leq d(t)$ para todo arco (s,t) em $D_S(f,x)$, nenhum PUSH pode ser aplicado sobre o vértice s. Nesse caso, é aplicada a rotina $\text{RELABEL}(D_S, d, s)$, que é responsável por atualizar o rótulo de s para que novas chamadas a PUSH possam ser realizadas em iterações posteriores. Sua especificação está a seguir.

Rotina $\text{RELABEL}(D_S, d, s)$: *Recebe um grafo D_S, uma função-distância d sobre os vértices de D_S e um vértice s de D_S. Se $d(s) \leq d(t)$, para todo arco (s,t) em D_S, devolve uma função-distância d' tal que $d'(s) = d(s) + 1$ e $d'(u) = d(u)$, para todo vértice $u \neq s$. Caso contrário, devolve a própria função-distância d.*

Descrição do algoritmo

Segue a descrição do algoritmo PUSH-RELABEL por completo.

Algoritmo $\text{PUSH-RELABEL}(S, f)$: *Recebe uma função submodular f sobre S tal que $f(\emptyset) = 0$ e devolve uma parte W de S que minimiza f.*

Cada iteração do algoritmo começa com:

- uma representação de um vetor x em $B(f)$, conforme (4.5), composta por
  - um conjunto de índices I_x, com $|I_x| \leq n + 1$;

  - para cada i em I_x, um número real positivo λ_i;
  - para cada i em I_x, uma ordem linear $\prec_i$ de S;

  tais que $x = \sum_{i \in I_x} \lambda_i x^{\prec_i}$ e $\sum_{i \in I_x} \lambda_i = 1$;

- rótulos válidos $d(s)$, para s em S;
- ponteiros $\pi(s)$, para cada s em S, representando o próximo vértice candidato a t para a execução de um PUSH(S, f, x, s, t).

Assim como no algoritmo SCHRIJVER, logo no início do algoritmo PUSH-RELABEL, é fixada uma ordenação para os elementos de S, que são renomeados como $\{1, \ldots, |S|\}$. Essa ordenação fixada dos elementos de S, bem como a manutenção dos ponteiros π, são necessárias para que se garanta um consumo de tempo razoável, como veremos mais abaixo.

No início da primeira iteração, temos

- $I_x := \{1\}$, $\lambda_1 := 1$ e $\prec_1$ sendo uma ordem linear qualquer de S, de maneira que $x = x^{\prec_1}$;
- $d(s) = 0$, para todo s em S;
- $\pi(s) = 1$, para todo s em S;

Uma iteração qualquer do algoritmo PUSH-RELABEL está descrita a seguir. Sejam

$P := \{s \in S \colon x(s) > 0\}$; $N := \{s \in S \colon x(s) < 0\}$; $D_S := D_S(f, x)$ e $R := \{s \in P \colon d(s) < n\}$;

Caso 1: $R = \emptyset$ ou $N = \emptyset$.
Seja W o conjunto de vértices que acessam N em D_S.
Devolva W.

Caso 2: $R \neq \emptyset$ e $N \neq \emptyset$.
Seja s o elemento em R com $d(s)$ máximo.

Caso 2A: $\pi(s) > n$.
$d' \leftarrow$ RELABEL(D_S, d, s).
$\pi'(s) \leftarrow 1$; $\pi'(u) \leftarrow \pi(u)$, para todo $u \neq s$.
Comece nova iteração com d' no papel de d e π' no papel de π.

Caso 2B: $\pi(s) \leq n$.
$t \leftarrow \pi(s)$.
Se $d(s) = d(t) + 1$ então
    $x' \leftarrow$ PUSH(S, f, x, s, t).
Senão $x' \leftarrow x$.

Caso 2B1: PUSH foi não-saturador.
Comece nova iteração com x' no papel de x.

Caso 2B2: PUSH foi saturador ou não ocorreu um PUSH.
$\pi'(s) \leftarrow t+1$; $\pi'(u) \leftarrow \pi(u)$, para todo $u \neq s$.
Comece nova iteração com x' no papel de x e π' no papel de π.

Correção e consumo de tempo

Dada uma iteração qualquer do algoritmo, considere a representação do vetor x em $B(f)$ mantida, além da função-distância d e dos ponteiros π. Considere também os objetos D_S, P, N, R, s, t, j e x' definidos durante a iteração.

Apresentamos a seguir alguns resultados que garantem que o algoritmo PUSH-RELABEL devolve um minimizador para a função submodular f em um número fortemente polinomial de iterações.

Lema 5.14: *O algoritmo* PUSH-RELABEL *mantém uma função-distância d válida.*

Demonstração: Pela maneira como é definida, a função-distância d é válida no início do algoritmo. Por sua própria especificação, a rotina RELABEL(D_S, d, S) só altera a função-distância d se $d(s) \leq d(t)$, para todo arco (s,t) em D_S, de maneira que d continua válida quando ocorre o caso 2A.

Como o caso 2B não altera a função-distância d, resta verificarmos se d permanece válida quando um arco (u,v) é adicionado ao grafo após a execução da rotina PIVOTA$(S, f, \prec_i, s, t)$, dentro de uma operação PUSH. Se o arco (u,v) foi adicionado ao grafo, então, pelo lema 5.3, temos $s \prec_j v \prec_j u \prec_j t$, logo no início do caso 2B. Como os rótulos são válidos no início da iteração e como o arco (s,t) foi escolhido para a aplicação do PUSH, segue que $d(u) \leq d(t)+1 = d(s) \leq d(v)+1$, de maneira que os rótulos continuam válidos quando consideramos o par u e v. Assim, segue que os rótulos continuam válidos após a iteração. ∎

O próximo teorema estabelece a correção do algoritmo PUSH-RELABEL, dado que o algoritmo pára. Ele é essencialmente um corolário do teorema 5.2, que garante a correção do algoritmo SCHRIJVER.

Teorema 5.15: *Quando ocorre o caso 1 do algoritmo* PUSH-RELABEL*, o conjunto de vértices W que acessam N em $D_S(f,x)$ é um minimizador de f.*

Demonstração: Quando ocorre o caso 1, ou $N = \emptyset$ ou $R = \emptyset$. Se $N = \emptyset$, então $W = \emptyset$. Além disso, nesse caso, temos $x^-(S) = 0 = f(\emptyset) = f(W)$, de maneira que W minimiza f pela relação min-max (4.3).

Se $R = \emptyset$, então não existem elementos s em P tais que $d(s) < n$. Como o algoritmo mantém uma função-distância válida, então, pela proposição 5.12, $d(s)$ é um limitante inferior para a distância de s a N em $D_S(f,x)$. Disso concluímos que, nesse caso, não

existem vértices em P que acessam N em $D_S(f,x)$. Ou seja, quando $R = \emptyset$, não existem caminhos de P a N em $D_S(f,x)$ e estamos exatamente na condição do caso 1 do algoritmo SCHRIJVER. Assim, o teorema 5.2, que garante a correção do algoritmo SCHRIJVER, garante também que, se o caso 1 ocorre no algoritmo PUSH-RELABEL, então ele devolve um conjunto W que minimiza f. ∎

Seguimos com resultados para limitar o número de iterações do algoritmo.

Lema 5.16: *Antes da execução de um* RELABEL(D_S, d, s) *em uma iteração em que ocorre o caso 2A no algoritmo* PUSH-RELABEL*, vale que* $d(s) \leq d(t)$*, para todo arco* (s,t) *em* D_S*.*

Demonstração: Vamos mostrar que, no início de qualquer iteração do algoritmo,

$$\text{se } (u,v) \text{ é um arco de } D_S, \text{ com } v < \pi(u), \text{ então } d(u) \leq d(v). \tag{5.13}$$

Como quando ocorre o caso 2A temos $\pi(s) > n$, seguirá que $d(s) \leq d(t)$ para todo arco (s,t), antes da execução de rotina RELABEL(D_S, d, s).

Provaremos por indução no número de iterações. Observe que (5.13) vale trivialmente no início da primeira iteração. Suponha agora que a afirmação valha no início de uma iteração qualquer do algoritmo. Vamos mostrar que continua válida no início da iteração seguinte.

Se (5.13) vale no início de determinada iteração em que ocorre o caso 2A, então continua valendo ao seu final. Note que o caso 2A altera apenas o rótulo de s e redefine o ponteiro $\pi(s)$ como 1.

Consideremos agora uma iteração em que ocorre o caso 2B. Analisemos primeiro o caso em que um arco (u,v) é inserido no grafo $D_S(f,x')$, com $v < \pi(u)$, logo após a chamada da rotina PIVOTA$(S, f, \prec_j, s, t)$ durante um PUSH. Precisamos mostrar que, nesse caso, $d(u) \leq d(v)$ logo ao final da iteração. Utilizando novamente o resultado do lema 5.3, como o arco (u,v) foi inserido no grafo D_S, temos $s \prec_j v \prec_j u \prec_j t$, de onde concluímos que os arcos (s,v) e (u,t) estavam no grafo $D_S(f,x)$ no início da iteração. Assim, como pelo lema 5.14 a função-distância d mantida é válida e como a rotina PUSH foi aplicada sobre o par s e t, vale que $d(u) \leq d(t) + 1 = d(s) \leq d(v) + 1$.

Se $t < \pi(u)$, então, pela hipótese de indução, a primeira desigualdade acima pode ser redefinida como $d(u) \leq d(t)$, de onde segue $d(u) \leq d(v)$, como queríamos. Por outro lado, se $v < \pi(u) \leq t$, como o arco (s,v) já existia no grafo D_S e como t é definido como $\pi(s)$ no início do caso 2B, podemos aplicar novamente a hipótese de indução e concluir que $d(s) \leq d(v)$, de onde segue também que $d(u) \leq d(v)$.

Quando ocorre o caso 2B, $\pi(s)$ é eventualmente incrementado. Como t é definido como $\pi(s)$ no início da iteração, resta verificar se os rótulos continuam válidos para o par s e t quando o arco (s,t) permanece no grafo D_S após o incremento. Agora, se $\pi(s)$ é

incrementado ao final da iteração e o arco (s,t) permanece no grafo D_S, então $d(s) \leq d(t)$ no início da iteração, pois o PUSH não foi executado. Como os rótulos d não são alterados pelo caso 2B, segue que continuam válidos para s e t ao final da iteração. ∎

Partindo do lema acima, bem como da especificação da rotina RELABEL, chegamos ao seguinte corolário, que nos permite também limitar o número de ocorrências do caso 2A, como vemos abaixo.

Corolário 5.17: *Sempre que ocorre o caso 2A no algoritmo* PUSH-RELABEL*, a rotina* RELABEL(D_S, d, s) *incrementa o rótulo* $d(s)$. ∎

Proposição 5.18: *O caso 2A ocorre no máximo* n^2 *vezes durante toda a execução do algoritmo* PUSH-RELABEL*.*

Demonstração: Conforme concluímos, sempre que ocorre o caso 2A em determinada iteração do algoritmo, a rotina RELABEL(D_S, d, s) incrementa o rótulo $d(s)$. Além disso, o algoritmo só aplica RELABEL sobre um vértice s se s está em R e, portanto, se $d(s) < n$. Como $d(s) = 0$, para todo s em S, no início do algoritmo, segue que RELABEL(D_S, d, s) pode ser executada no máximo n vezes, para s fixo. Disso segue o resultado. ∎

O corolário 5.17 também nos permite limitar o número de iterações do caso 2B em que ocorre um PUSH saturador ou não ocorre um PUSH.

Corolário 5.19: *O algoritmo* PUSH-RELABEL *executa no máximo* n^3 *iterações do caso 2B2.*

Demonstração: Quando ocorre um PUSH saturador ou não ocorre um PUSH, durante uma iteração do caso 2B em que s é escolhido como um vértice s^* fixado, temos que $\pi(s^*)$ é incrementado em 1. Logo, ocorrem no máximo n iterações desse tipo em que $s = s^*$ antes que s^* sofra um RELABEL. Como pelo corolário 5.17 todo RELABEL(D_S, d, s^*) incrementa o rótulo $d(s^*)$ e como $0 \leq d(s^*) \leq n$ durante todo o algoritmo, segue que ocorrem no máximo n^2 iterações do caso 2B desse tipo, para cada vértice s^* fixado. Variando s^* sobre todos os possíveis vértices, chegamos ao resultado. ∎

Os próximos resultados limitam o número de iterações do caso 2B em que ocorre um PUSH não-saturador.

Lema 5.20: *Considere uma iteração em que ocorre o caso 2B1 na qual* s *é escolhido como um vértice* s^* *fixado. Antes que* s^* *seja escolhido novamente como* s *no início de alguma iteração, o algoritmo* PUSH-RELABEL *executa um* RELABEL(D_S, d, u)*, para algum* u *em* S*.*

Demonstração: Quando ocorre um PUSH não-saturador numa iteração que definiu $s = s^*$, temos $x'(s^*) = 0$, de maneira que s^* não está mais em R no início da iteração seguinte. Portanto, para que s^* volte a R e possa ser selecionado novamente como s no início de al-

guma iteração, o valor de $x(s^*)$ deve crescer através de alguma operação PUSH(S, f, x, u, s^*) para algum u em S. Ademais, para que essa operação PUSH ocorra, devemos ter $d(u) = d(s^*) + 1$. Como s é sempre escolhido como o elemento de maior rótulo e como s^* tinha o maior rótulo na iteração em que sofreu o PUSH não-saturador, deve ocorrer pelo menos uma operação RELABEL antes que PUSH(S, f, x, u, s^*) seja invocado. ∎

Corolário 5.21: *O algoritmo* PUSH-RELABEL *executa no máximo* n^3 *iterações do caso 2B1.*

Demonstração: Como o algoritmo executa no máximo n^2 operações RELABEL, então, pelo lema anterior, o número de operações PUSH não-saturadoras sobre cada vértice fixado é limitado em n^2. Logo, o número total de execuções do caso 2B1 é limitado por n^3. ∎

Utilizando a proposição 5.18, bem como os corolários 5.19 e 5.21, chegamos ao seguinte resultado.

Teorema 5.22 (do número de iterações de PUSH-RELABEL):
O algoritmo PUSH-RELABEL(S, f) *encontra um minimizador para a função submodular* f *em* $O(n^3)$ *iterações.* ∎

Por fim, podemos estabelecer um limitante para o consumo de tempo total do algoritmo PUSH-RELABEL.

Teorema 5.23 (do consumo de tempo de PUSH-RELABEL):
O algoritmo PUSH-RELABEL(S, f) *é fortemente polinomial e pode ser implementado de modo a consumir tempo* $O(n^8 + n^7\gamma)$.

Demonstração: Pela proposição 5.13, cada PUSH executa $O(n^2)$ chamadas à rotina PIVOTA e executa ainda uma chamada à rotina COMPACTA. Utilizando então as proposições 5.1 e 4.3, que estabelecem os consumos de tempo dessas rotinas, concluímos que cada PUSH pode ser implementado de forma a consumir tempo $O(n^5 + n^4\gamma)$. Observe que antes da execução da rotina COMPACTA durante um PUSH, o número de pontos extremos na representação de x cresce em no máximo n, de forma que podemos utilizar a proposição 4.3. Além disso, cada operação RELABEL pode ser implementada de forma a consumir tempo constante. Assim, como acabamos de mostrar que o algoritmo PUSH-RELABEL executa $O(n^3)$ iterações e como em cada iteração ocorre um PUSH ou um RELABEL, segue o resultado. ∎

Capítulo 6

Algoritmo IFF

Conforme discutimos no capítulo 5, uma das dificuldades encontradas na busca da polinomialidade do método de Cunningham, descrito no capítulo 4, concentra-se no fato de que nem sempre existem caminhos de aumento com capacidades suficientemente grandes. Buscando solucionar essa questão, Iwata, Fleischer e Fujishige [IFF01], inspirados também em idéias presentes em algoritmos para fluxos submodulares [Iwa97, IMS99, FIM02], propuseram relaxar o problema de minimizar uma função submodular e aplicar a técnica de *scaling*. Na relaxação proposta, as capacidades dos arcos do grafo auxiliar são aumentadas em função de um parâmetro fixado, de maneira que se consiga obter caminhos de aumento com capacidades grandes o suficiente para a garantia da polinomialidade do algoritmo.

Em seu trabalho, Iwata, Fleischer e Fujishige começam apresentando um algoritmo fracamente polinomial para minimizar funções submodulares, cujo consumo de tempo depende não apenas do tamanho do conjunto base sobre o qual a função está definida, mas também do maior valor assumido pela função. Utilizando chamadas a um conjunto de iterações desse primeiro algoritmo, propõem também uma versão fortemente polinomial. Os dois algoritmos estão descritos neste capítulo.

6.1 Fluxos δ-viáveis e poliedro dos excessos

Definimos nesta seção os fluxos δ-viáveis e o poliedro dos excessos, objetos que foram explorados por Iwata, Fleischer e Fujishige [IFF01] para a relaxação do problema e o desenvolvimento de seus algoritmos, conforme veremos nas próximas seções.

Seja H um grafo orientado completo sobre S, sem laços. Isto é, H possui um arco (u, v) ligando quaisquer dois elementos distintos u e v em S. Fixe um parâmetro $\delta > 0$.

Um **fluxo** φ em H é uma função que atribui um valor $\varphi(u, v)$ para cada par de elementos distintos u e v em S. O fluxo φ é dito **δ-viável** se satisfaz, para todo par de elementos distintos u e v em S,

$$0 \leq \varphi(u, v) \leq \delta \quad \text{e} \quad \varphi(u, v) > 0 \Longrightarrow \varphi(v, u) = 0. \tag{6.1}$$

Seja φ um fluxo em H. Dado v em S, defina o **excesso** $\partial\varphi(v)$ **do fluxo** φ **em** v como a diferença entre a quantidade de fluxo que sai e a quantidade de fluxo que entra em v. Mais especificamente,

$$\partial\varphi(v) = \varphi(\delta^{\text{out}}(v)) - \varphi(\delta^{\text{in}}(v)).$$

Note que, para cada fluxo φ, $\partial\varphi$ é um vetor indexado por S. Chamamos $\partial\varphi$ de **vetor dos excessos de** φ.

Fixado δ, podemos considerar o seguinte conjunto, que é um poliedro, como veremos:

$$\partial\Phi_\delta = \left\{\partial\varphi\colon \varphi \text{ fluxo } \delta\text{-viável}\right\}. \tag{6.2}$$

O poliedro $\partial\Phi_\delta$ é o **poliedro dos excessos** dos fluxos δ-viáveis.

Atribuindo capacidade δ a cada arco de H, temos que a função

$$\kappa_\delta(U) = \delta|U||S \setminus U|, \text{ para todo } U \subseteq S, \tag{6.3}$$

é a função submodular que representa a capacidade de cada corte em H. A função capacidade de cortes é um dos exemplos de funções submodulares apresentados na seção 1.2.

Utilizando o teorema de Gale [Gal57], sobre a existência de fluxos que satisfazem determinadas restrições de capacidades e demandas em uma rede, pode-se provar que $\partial\Phi_\delta$ é exatamente o polimatróide das bases, conforme definido na seção 2.6, associado à função κ_δ. Isto é,

$$\partial\Phi_\delta = B(\kappa_\delta) = \left\{x \in \mathbb{R}^S\colon x(S) = \kappa_\delta(S) = 0,\ x(U) \leq \kappa_\delta(U), \text{ para todo } U \subseteq S\right\}. \tag{6.4}$$

6.2 Uma relaxação do problema

Iwata, Fleischer e Fujishige [IFF01] adotaram a estratégia de caminhos de maior aumento com o objetivo de transformar o método de Cunningham em um algoritmo polinomial para minimizar funções submodulares. Entretanto, como nem sempre existem caminhos de aumento com capacidades suficientemente grandes no grafo de Cunningham (seção 4.3), propõem que seu algoritmo aplique a técnica de *scaling* sobre uma relaxação do problema.

Na verdade, Iwata, Fleischer e Fujishige apresentam dois algoritmos em seu trabalho. O primeiro algoritmo é fracamente polinomial, isto é, cujo consumo de tempo depende não só do tamanho do conjunto base sobre o qual a função submodular está definida, como também do maior valor assumido pela função. Esse algoritmo, o qual chamaremos **algoritmo IFF**, trabalha diretamente sobre a relaxação proposta do problema. O segundo algoritmo apresentado é fortemente polinomial e está descrito na seção 6.7. Essa segunda versão de algoritmo se utiliza de chamadas a um conjunto de iterações do algoritmo IFF. As idéias que apresentamos nesta e na próxima seção são utilizadas diretamente pelo algoritmo IFF.

Na abordagem sugerida por Iwata, Fleischer e Fujishige, o problema de minimizar uma função submodular é relaxado em função de um parâmetro $\delta > 0$ fixado, que vai sendo reduzido ao longo do algoritmo. Quanto menor o parâmetro δ, mais próximo o problema relaxado está do problema original. Isso é o que chamamos de **scaling** aplicado ao parâmetro δ. A técnica de scaling é bastante utilizada na tentativa de transformar algoritmos pseudo-polinomiais em polinomiais e foi introduzida por Edmonds e Karp [EK72] quando propuseram o primeiro algoritmo polinomial para o problema do fluxo de custo mínimo.

O algoritmo IFF é iterativo e executa uma seqüência de δ-fases. Uma **δ-fase** é uma seqüência de iterações em que o parâmetro δ está fixado. Cada δ-fase do algoritmo trabalha sobre uma instância do problema relaxado associada ao parâmetro δ.

Seja f uma função submodular sobre S tal que $f(\emptyset) = 0$ e considere novamente a função κ_δ, como definida em (6.3). Como sempre, definimos $n := |S|$. O algoritmo IFF encontra um minimizador para a função f, buscando, em cada δ-fase, resolver de forma aproximada o seguinte par relaxado de problemas duais:

$$\max\left\{z^-(S)\colon z \in B(f+\kappa_\delta)\right\} = \min\left\{f(U)+\kappa_\delta(U)\colon U \subseteq S\right\}. \tag{6.5}$$

Observe que κ_δ é também uma função submodular e que a soma de funções submodulares também resulta em uma função submodular. Portanto, a validade da relação de dualidade acima segue da relação min-max (4.3), na qual são baseados os demais algoritmos para minimização de funções submodulares que descrevemos nos capítulos anteriores.

Lembrando agora que $B(\kappa_\delta)$ é justamente o poliedro dos excessos dos fluxos δ-viáveis, conforme (6.4), podemos representar qualquer vetor z em $B(f+\kappa_\delta)$ como a soma de um vetor x em $B(f)$ e um vetor $\partial\varphi$ dos excessos de um fluxo φ δ-viável.

Desse modo, os problemas duais em (6.5) podem ser reescritos como

$$\begin{gathered}\max\left\{(x+\partial\varphi)^-(S)\colon x \in B(f),\ \varphi \text{ fluxo } \delta\text{-viável}\right\}\\ =\\ \min\left\{f(U)+\kappa_\delta(U)\colon U \subseteq S\right\}.\end{gathered} \tag{6.6}$$

Para trabalhar sobre o par de problemas duais acima, o algoritmo IFF mantém um vetor x em $B(f)$ e um fluxo δ-viável φ, de forma que $z := x + \partial\varphi$ seja um vetor em $B(f+\kappa_\delta)$. Assim como sugerido por Cunningham, o vetor x é mais uma vez mantido através de uma representação (seção 4.3), como combinação convexa de pontos extremos de $B(f)$. Busca-se, então, maximizar $z^-(S)$ através de caminhos de aumento em um grafo auxiliar, obtido com o auxílio do fluxo φ e dos arcos do grafo de Cunningham $D(f,x)$.

Quanto menor o parâmetro δ, mais próximo z está de um vetor em $B(f)$. Além disso, é interessante observar que um limitante inferior para $z^-(S)$ fornece também um limitante inferior para $x^-(S)$ em função da seguinte relação:

$$x^-(S) \geq z^-(S) - \delta n^2/4. \tag{6.7}$$

Note que, para alguma parte U de S, temos

$$x^-(S) = x(U) = z(U) - \partial\varphi(U) \geq z^-(S) - \partial\varphi(U).$$

Além disso, vale que

$$\partial\varphi(U) \leq \varphi(\delta^{\text{out}}(U)) = \delta|U||S \setminus U| \leq \delta n^2/4,$$

para toda parte U de S. Disso segue a relação (6.7).

6.3 Uma δ-fase de scaling

Conforme mencionamos na seção anterior, uma δ-fase do algoritmo IFF é uma seqüência de iterações em que o parâmetro δ se mantém constante. Em cada δ-fase, o algoritmo mantém um vetor x em $B(f)$ e um fluxo δ-viável φ, de forma que $z := x + \partial\varphi$ esteja em $B(f + \kappa_\delta)$, e busca aumentar o valor de $z^-(S)$ com auxílio do fluxo φ e do grafo de Cunningham $D(f, x)$.

Como nem sempre aumentos suficientemente grandes podem ser realizados através dos arcos do grafo $D(f, x)$, conforme discutimos na seção 5.1, os aumentos em $z^-(S)$ são, em princípio, realizados exclusivamente através do fluxo φ. Quando isso não é mais possível, procura-se realizar um aumento através de pivotações nos pontos extremos que compõem a representação do vetor x em $B(f)$, com o auxílio dos arcos de $D(f, x)$. Quando aumentos de "tamanho suficiente" não podem ser realizados seja através de φ ou através de $D(f, x)$, a fase de scaling termina.

Rotina DELTA-AUMENTO: aumentos através de δ-caminhos

Dado um fluxo δ-viável φ, defina $H(\varphi)$ como o grafo completo sobre S com conjunto de arcos

$$A(\varphi) := \Big\{(u, v) \colon u, v \in S,\ u \neq v,\ \varphi(v, u) = 0\Big\}. \tag{6.8}$$

Intuitivamente, o grafo $H(\varphi)$ contém um arco de u a v se podemos devolver δ unidades de fluxo de v a u sem que o fluxo φ deixe de ser δ-viável.

Defina também

$$\begin{gathered} P_\delta(x, \varphi) := \big\{v \in S \colon x(v) + \partial\varphi(v) \geq \delta\big\} \\ \text{e} \\ N_\delta(x, \varphi) := \big\{v \in S \colon x(v) + \partial\varphi(v) \leq -\delta\big\}. \end{gathered} \tag{6.9}$$

Chamamos de **δ-caminho** um caminho de $P_\delta(x, \varphi)$ a $N_\delta(x, \varphi)$ em $H(\varphi)$. Em cada iteração dentro de determinada δ-fase, o algoritmo IFF procura aumentar $z^-(S)$ em δ, "devolvendo" δ unidades de fluxo através de um δ-caminho. Mais especificamente, a devolução de fluxo é realizada de modo a produzir um aumento em $\partial\varphi^-(S)$.

Se Q é um δ-caminho, um **δ-aumento através de φ e Q** é definido como

$$\begin{aligned} \varphi(v,u) &:= \delta - \varphi(u,v), \\ \varphi(u,v) &:= 0, \end{aligned} \tag{6.10}$$

para cada arco (u,v) em Q. Os demais arcos de $H(\varphi)$ não têm seu fluxo φ alterado após um δ-aumento.

Observe que, se Q é um δ-caminho de s a t, após um δ-aumento temos que:

$$\begin{aligned} &\partial\varphi(s) \text{ diminui em } \delta; \\ &\partial\varphi(t) \text{ aumenta em } \delta; \\ &\partial\varphi(v) \text{ não se altera, para todo } v \text{ em } Q,\ v \neq s,t. \end{aligned} \tag{6.11}$$

Logo, como s é um elemento de $P_\delta(x,\varphi)$ e t é um elemento de $N_\delta(x,\varphi)$, $z^-(S)$ aumenta em δ. Observe também que o fluxo φ continua δ-viável após um δ-aumento.

O algoritmo IFF realiza uma série de δ-aumentos através de chamadas à rotina DELTA-AUMENTO, cuja especificação está a seguir.

Rotina DELTA-AUMENTO(S,δ,φ,Q): *Recebe um parâmetro $\delta > 0$, um fluxo δ-viável φ e um δ-caminho Q. Devolve um fluxo δ-viável φ' que é resultante de um δ-aumento através de φ e Q.*

Rotina PIVOTAÇÃO-DUPLA: utilizando os arcos de $D(f,x)$

Em cada iteração de uma δ-fase do algoritmo IFF, se existe um δ-caminho, um δ-aumento é realizado. Caso contrário, intuitivamente, o algoritmo procura repassar um pouco do fluxo φ para arcos do grafo de Cunningham $D(f,x)$ visando produzir um novo δ-caminho e continuar efetuando δ-aumentos, sempre de maneira a não reduzir $z^-(S)$.

Suponha que não exista um δ-caminho e denote por W o conjunto dos vértices que acessam $N_\delta(x,\varphi)$ no grafo $H(\varphi)$. Seja x o vetor em $B(f)$ mantido pelo algoritmo na iteração. Considere a representação mantida do vetor x. Valem os seguintes resultados.

Lema 6.1: *Se W é x-justo e se (s,t) é um arco de $D(f,x)$ que entra em W, então o st-potencial com relação a qualquer $x^{\prec_i}$ na representação de x é nulo.*

Demonstração: Suponha que W seja x-justo. Como $x = \sum_{i\in I_x} \lambda_i x^{\prec_i}$, com cada $x^{\prec_i}$ sendo um ponto extremo em $B(f)$, então W é $x^{\prec_i}$-justo, para cada $\prec_i$ na representação de x. Segue que se s está em $S \setminus W$ e se t está em W, então o st-potencial $\tilde{\alpha}(x^{\prec_i},s,t)$ é zero, para toda ordem $\prec_i$, pela definição de st-potencial dada em (4.9), na seção 4.3, por

$$\tilde{\alpha}(x^{\prec_i},s,t) := \min\Big\{f(U) - x^{\prec_i}(U) \colon t \in U \subseteq S \setminus \{s\}\Big\}.$$

∎

Lema 6.2: *Se W não é x-justo, então existe pelo menos um arco de $D(f,x)$ entrando*

em W.

Demonstração: Suponha que W não seja x-justo. Como $x = \sum_{i \in I_x} \lambda_i x^{\prec_i}$, com cada $x^{\prec_i}$ sendo um ponto extremo em $B(f)$, então W não é $x^{\prec_i}$-justo, para alguma $\prec_i$ na representação de x. Conseqüentemente, W não é um segmento inicial da ordem $\prec_i$, isto é, W não é da forma $[s]_{\prec_i}$, para nenhum s em S (teorema 2.10). Segue que existem elementos s e t em S, com s em $S \setminus W$, t em W e tais que s precede imediatamente t em $\prec_i$, de maneira que (s,t) é um arco de $D(f,x)$ que entra em W. ∎

Se W é x-justo, não será possível utilizar o grafo $D(f,x)$ na tentativa de produzir um δ-caminho. Conforme veremos abaixo, para que isso seja possível, é necessário que haja pelo menos um arco de $D(f,x)$ entrando em W com capacidade não nula, o que não ocorre, pelo lema 6.1. Lembre que a capacidade de cada arco (s,t) no grafo de Cunningham $D(f,x)$ é definida como a soma dos st-potenciais referentes a cada $x^{\prec_i}$ que está na representação mantida de x e tal que s precede imediatamente t na ordem $\prec_i$, conforme (4.13). Isso marca o final da δ-fase de scaling.

Analisemos então o caso em que W não é x-justo. Nesse caso, pelo último lema 6.2 acima, existe um arco (s,t) de $D(f,x)$ entrando em W. Pela definição do grafo $D(f,x)$ (seção 4.3), segue que existe um índice i na representação mantida do vetor x tal que s precede imediatamente t na ordem $\prec_i$. Um trio (i,s,t), com i sendo um índice na representação de x, s em $S \setminus W$, t em W e tal que s precede imediatamente t na ordem $\prec_i$ é chamado **ativo**.

Nesse momento, o algoritmo aplica a rotina PIVOTAÇÃO-DUPLA sobre um trio ativo (i,s,t), com um dos seguintes objetivos:

- aumentar W, buscando "criar" um δ-caminho (para que possamos efetuar mais δ-aumentos);
- ou tornar W mais próximo de ser x-justo (para que possamos finalizar a δ-fase de scaling).

A descrição de PIVOTAÇÃO-DUPLA está a seguir.

Rotina PIVOTAÇÃO-DUPLA$(S, f, \delta, x, \varphi, (i,s,t))$: *Recebe uma função submodular f sobre S tal que $f(\emptyset) = 0$, um parâmetro $\delta > 0$, a representação de um vetor x em $B(f)$, um fluxo δ-viável φ e um trio ativo (i,s,t). Devolve um vetor x' em $B(f)$, obtido com o auxílio de uma st-pivotação em $x^{\prec_i}$, e um fluxo δ-viável φ', de maneira que*

$$\begin{array}{ll} (i) & (x' + \partial\varphi')^-(S) = (x + \partial\varphi)^-(S); \\ (ii) & \varphi'(t,s) \leq \varphi(t,s). \end{array} \tag{6.12}$$

A rotina começa realizando uma pivotação entre s e t em $x^{\prec_i}$, de maneira a obter o ponto extremo $x^{\prec_j}$, isto é,

$$x^{\prec_j} := x^{\prec_i} + \tilde{\alpha}(x^{\prec_i}, s, t)(\chi^t - \chi^s),$$

em que $\tilde{\alpha}(x^{\prec_i}, s, t) = f\big([t]_{\prec_i} \setminus \{s\}\big) - x\big([t]_{\prec_i} \setminus \{s\}\big)$, conforme (4.10), na seção 4.3. A ordem $\prec_j$ é obtida simplesmente trocando-se s e t de posição em $\prec_i$, conforme teorema 2.13.

Feito isso, toma-se $\epsilon := \min\big\{\delta, \lambda_i \tilde{\alpha}(x^{\prec_i}, s, t)\big\}$ e obtém-se uma representação para

$$x' := x + \epsilon(\chi^t - \chi^s), \tag{6.13}$$

com o auxílio da representação de x e de $x^{\prec_j}$, como segue. Comece definindo a representação de x' como a representação de x. Se $\epsilon = \lambda_i \tilde{\alpha}(x^{\prec_i}, s, t)$, troque $x^{\prec_i}$ por $x^{\prec_j}$ na representação. O coeficiente λ_j de $x^{\prec_j}$ na combinação convexa que descreve x' será exatamente λ_i nesse caso. Se $\epsilon = \delta < \lambda_i \tilde{\alpha}(x^{\prec_i}, s, t)$, mantenha $x^{\prec_i}$ na representação, mas redefina λ_i como $\lambda_i - (\delta / \tilde{\alpha}(x^{\prec_i}, s, t))$. Além disso, adicione $x^{\prec_j}$ à representação de x', com coeficiente $\lambda_j := \delta / \tilde{\alpha}(x^{\prec_i}, s, t)$.

Por fim, obtém-se o fluxo φ' da seguinte maneira:

$$\begin{array}{lcl} \varphi'(s,t) & := & \max\big\{0, \epsilon - \varphi(t,s)\big\}, \\ \varphi'(t,s) & := & \max\big\{0, \varphi(t,s) - \epsilon\big\}, \\ \varphi'(u,v) & := & \varphi(u,v), \ \text{ para } u \neq s \text{ ou } v \neq t. \end{array} \tag{6.14}$$

Quando $\epsilon = \lambda_i \tilde{\alpha}(x^{\prec_i}, s, t)$, e, portanto, quando o ponto extremo $x^{\prec_i}$ deixa de pertencer à representação de x', dizemos que PIVOTAÇÃO-DUPLA foi **saturadora**. Caso contrário, PIVOTAÇÃO-DUPLA foi **não-saturadora**.

A rotina PIVOTAÇÃO-DUPLA realiza uma st-pivotação sobre $x^{\prec_i}$. Podemos imaginar a obtenção do fluxo φ' como uma pivotação entre t e s sobre $\partial\varphi$. Disso vem o nome da rotina.

Observe também que, após uma chamada à rotina PIVOTAÇÃO-DUPLA, como reduzimos o fluxo φ em um arco que sai de W, um δ-caminho pode ser criado. Lembre-se que, por (6.8), $H(\varphi)$ tem um arco entrando em W se, e somente se, existe um arco com fluxo nulo saindo de W. Além disso, com a redução do coeficiente λ_i na representação do vetor x mantido, estamos caminhando para a eliminação do arco (s,t) que entra em W no grafo $D(f,x)$. Isso faz com que W se torne mais próximo de ser x-justo. Note que o índice i é um dos que justificam a presença do arco (s,t) em $D(f,x)$. Note também que o novo índice j não justifica a presença desse arco uma vez que, pela definição de $x^{\prec_j}$, s não precede t em $\prec_j$ (teorema 2.13). Ademais, se não existem arcos de $D(f,x)$ entrando em W, então W é x-justo, pelo lema 6.2. Ou seja, após uma chamada à PIVOTAÇÃO-DUPLA também estamos caminhando para o final da δ-fase.

Cabe notar também que se o arco (s,t) possui capacidade nula em $D(f,x)$, isto é, se $\tilde{\alpha}(x^{\prec_i}, s, t) = 0$, então a rotina PIVOTAÇÃO-DUPLA não realiza progressos no sentido de produzir um δ-caminho. Logo, como pelo lema 6.1 todos os arcos que entram em um conjunto x-justo têm capacidade nula em $D(f,x)$, se W é x-justo, a δ-fase pode terminar.

Por fim, ressaltamos que o vetor x' e o fluxo φ' devolvidos pela rotina PIVOTAÇÃO-

DUPLA satisfazem:

$$x'(t) = x(t) + \epsilon \quad \text{e} \quad \partial\varphi'(t) = \partial\varphi(t) - \epsilon;$$
$$x'(s) = x(s) - \epsilon \quad \text{e} \quad \partial\varphi'(s) = \partial\varphi(s) + \epsilon.$$

Além disso, $x'(v) = x(v)$ e $\partial\varphi'(v) = \partial\varphi(v)$, para todo $v \neq s, t$, de maneira que $(x' + \partial\varphi')^-(S) = (x + \partial\varphi)^-(S)$.

Esses argumentos nos trazem alguma intuição sobre o fato de que a rotina atende os objetivos a que se propõe.

Rotina DELTA-FASE: execução de uma δ-fase

Para facilitar a descrição do algoritmo IFF, segue abaixo o pseudo-código da rotina DELTA-FASE que representa a execução de uma δ-fase, cujas idéias descrevemos ao longo da seção.

> **Rotina** DELTA-FASE$(S, f, \delta, x, \varphi)$: *Recebe uma função submodular f sobre S, um parâmetro $\delta > 0$, a representação de um vetor x em $B(f)$ e um fluxo δ-viável φ. Devolve a representação de um vetor x' em $B(f)$ e um fluxo δ-viável φ', de maneira que não exista um δ-caminho em $H(\varphi')$. Além disso, o vetor x' é tal que o conjunto W dos vértices que acessam*
>
> $$N_\delta(x', \varphi') := \{v \in S \colon x'(v) + \partial\varphi'(v) \leq -\delta\}$$
>
> *em $H(\varphi')$ é x'-justo.*

Cada iteração de uma δ-fase começa com um vetor x em $B(f)$ e um fluxo δ-viável φ. Na primeira iteração, x e φ são os parâmetros recebidos pela rotina.

Uma iteração qualquer da rotina DELTA-FASE está descrita a seguir. Sejam

$$P_\delta(x, \varphi) := \{v \in S \colon x(v) + \partial\varphi(v) \geq \delta\} \text{ e } N_\delta(x, \varphi) := \{v \in S \colon x(v) + \partial\varphi(v) \leq -\delta\}.$$

Caso 1: Existe um δ-caminho Q em $H(\varphi)$.
$\varphi' \leftarrow$ DELTA-AUMENTO(S, δ, φ, Q).
$x' \leftarrow$ COMPACTA(S, f, x).
Comece nova iteração com φ' e x' respectivamente nos papéis de φ e x.

Caso 2: Não existe δ-caminho em $H(\varphi)$.
Seja W o conjunto de vértices que acessam $N_\delta(x, \varphi)$ em $H(\varphi)$.

Caso 2A: W é x-justo.
$x' \leftarrow$ COMPACTA(S, f, x).
Devolva φ e x'.

Caso 2B: W não é x-justo.
Escolha um trio ativo (i, s, t).

$(x', \varphi') \leftarrow$ PIVOTAÇÃO-DUPLA$(S, f, \delta, x, \varphi, (i, s, t))$.
Comece nova iteração com x' e φ' respectivamente nos papéis de x e φ.

No código acima, a rotina COMPACTA é a mesma definida na seção 4.4, durante a descrição do método de Cunningham. Quando um δ-aumento não pode ser realizado, a rotina PIVOTAÇÃO-DUPLA é executada. Cada chamada de PIVOTAÇÃO-DUPLA pode acrescentar um ponto extremo à representação mantida de x. Por esse motivo, para que a representação de x continue com poucos pontos extremos, de forma que se consiga garantir um consumo de tempo menor ao algoritmo IFF, algumas chamadas à rotina COMPACTA se fazem necessárias.

A análise de consumo de tempo do algoritmo IFF, na seção 6.6, nos mostrará que as iterações de DELTA-FASE em que COMPACTA é invocada são momentos adequados, visando a manter sempre $O(|S|)$ pontos extremos na combinação convexa que descreve x.

6.4 Descrição do algoritmo

Após discussão das idéias que permeiam o funcionamento do algoritmo IFF, apresentamos sua descrição completa. A descrição apresentada é também bastante sucinta, uma vez que se baseia principalmente na rotina DELTA-FASE, introduzida na seção anterior.

Algoritmo IFF(S, f): *Recebe uma função submodular f* inteira *sobre S tal que $f(\emptyset) = 0$ e devolve uma parte W de S que minimiza f.*

Uma função submodular f é inteira se $f(U)$ é um número inteiro para cada parte U de S. A restrição de que a função f recebida seja inteira é necessária para a demonstração da correção do algoritmo IFF. Entretanto, conforme veremos na seção 6.7, a versão fortemente polinomial do algoritmo, que utiliza chamadas às δ-fases do algoritmo IFF, não exige nenhuma restrição sobre a função a ser minimizada.

Cada iteração do algoritmo IFF começa com:

- um parâmetro $\delta \geq 0$;
- uma representação de um vetor x em $B(f)$, conforme (4.5), composta por
  - um conjunto de índices I_x;
  - para cada i em I_x, um número real positivo λ_i;
  - para cada i em I_x, uma ordem linear $\prec_i$ de S;

  tais que $x = \sum_{i \in I_x} \lambda_i x^{\prec_i}$ e $\sum_{i \in I_x} \lambda_i = 1$;
- um fluxo δ-viável φ.

No início da primeira iteração, assim como nos algoritmos descritos anteriormente, temos $I_x := \{1\}$, $\lambda_1 := 1$ e $\prec_1$ sendo uma ordem linear qualquer de S, de maneira que

$x = x^{\prec_1}$. Além disso, o fluxo φ é inicializado como o fluxo identicamente nulo, isto é, $\varphi(u,v) = 0$, para todo par de elementos distintos u e v em S. O fluxo identicamente nulo é claramente δ-viável, para qualquer $\delta \geq 0$.

Dado x em $\mathbb{R}^S$, defina $x^+(s) := \max\{0, x(s)\}$. O parâmetro δ tem uma inicialização específica para a garantia do consumo de tempo fracamente polinomial do algoritmo, como segue:

$$\delta := \min\{|x^-(S)|, x^+(S)\}/n^2. \tag{6.15}$$

Os critérios de parada garantem a correção do algoritmo. O caso 0 cobre uma situação particular. Esses detalhes serão apresentados na próxima seção.

Uma iteração qualquer do algoritmo está descrita a seguir.

Caso 0: $\delta = 0$.
Se $\delta = x^-(S)$, devolva $\emptyset$.
Se $\delta = x^+(S)$, devolva S.

Caso 1: $\delta > 0$.
$(x', \varphi') \leftarrow$ DELTA-FASE$(S, f, \delta, x, \varphi)$.
Seja $N_\delta(x', \varphi') := \{v \in S : x'(v) + \partial\varphi'(v) \leq -\delta\}$.
Seja W o conjunto de vértices que acessam $N_\delta(x', \varphi')$ em $H(\varphi')$.

Caso 1A: $\delta < 1/n^2$.
Devolva W.

Caso 1B: $\delta \geq 1/n^2$.
$\delta' \leftarrow \delta/2$.
$\varphi' \leftarrow \varphi'/2$.
Comece nova iteração com δ', x' e φ' respectivamente nos papéis de δ, x e φ.

6.5 Correção do algoritmo

Apresentamos agora os resultados que garantem que se o algoritmo IFF(S, f) pára e se a função submodular f recebida como parâmetro é inteira, então ele devolve um minimizador de f. Na próxima seção limitamos o número de iterações da rotina DELTA-FASE e, conseqüentemente, estabelecemos o consumo de tempo do algoritmo IFF.

O algoritmo inicializa $\delta := \min\{|x^-(S)|, x^+(S)\}/n^2$. Quando ocorre o caso 0, se $\delta = x^-(S) = 0$, então $x^-(S) = f(\emptyset) = 0$. Por outro lado, se $\delta = x^+(S) = 0$, então $x^-(S) = x(S) = f(S)$. Segue pela relação min-max (4.3) que o algoritmo devolve um minimizador de f em ambos os casos.

Vamos mostrar a partir dos próximos resultados que o algoritmo também devolve a resposta correta quando termina em uma iteração do caso 1A.

Teorema 6.3 (da dualidade forte aproximada): *Considere uma iteração qualquer do algoritmo* IFF *em que ocorre o caso 1. Seja W o conjunto dos vértices que aces-*

sam $N_\delta(x', \varphi') := \{v \in S \colon x'(v) + \partial\varphi'(v) \leq -\delta\}$ *em* $H(\varphi')$ *logo após a execução de* DELTA-FASE$(S, f, \delta, x, \varphi)$. *Nesse momento, vale que*

$$(x' + \partial\varphi')^-(S) \geq f(W) - n\delta. \tag{6.16}$$

Além disso,

$$x'^-(S) \geq f(W) - n^2\delta. \tag{6.17}$$

Demonstração: Começamos provando (6.16). Tome $z = x' + \partial\varphi'$ e seja $P_\delta(x', \varphi') := \{v \in S \colon x'(v) + \partial\varphi'(v) \geq \delta\}$. Após a execução de DELTA-FASE$(S, f, \delta, x, \varphi)$, temos que não existe um δ-caminho em $H(\varphi')$ e, portanto, que $N_\delta(x', \varphi') \subseteq W \subseteq S \setminus P_\delta(x', \varphi')$. Como $\delta > 0$, segue que $z(w) < \delta$, para todo w em W e que $z(w) > -\delta$, para todo w em $S \setminus W$, de modo que

$$\begin{aligned} z^-(S) &= z^-(W) + z^-(S \setminus W) \\ &\geq \big(z(W) - \delta|W|\big) - \delta|S \setminus W| \\ &= z(W) - n\delta \\ &= x'(W) + \partial\varphi'(W) - n\delta \\ &\geq f(W) - n\delta. \end{aligned} \tag{6.18}$$

Note que $x'(W) = f(W)$, pois W é x'-justo após a execução de DELTA-FASE. Além disso, como por definição não existem arcos de $H(\varphi')$ entrando em W, então todo arco (u, v) que sai de W é tal que $\varphi'(u, v) > 0$, caso exista algum. Logo, como φ' é um fluxo δ-viável, segue também que todo arco que entra em W tem fluxo nulo, de maneira que $\partial\varphi'(W) = \varphi'(\delta^{\text{out}}(W)) - \varphi'(\delta^{\text{in}}(W)) \geq 0$. Disso segue a desigualdade em (6.18).

Por fim, mostramos (6.17). Para alguma parte U de S, temos que

$$x'^-(S) = x'(U) = z(U) - \partial\varphi'(U) \geq z^-(S) - n(n-1)\delta.$$

A última desigualdade acima segue do fato de que $\partial\varphi'(v) \leq (n-1)\delta$, para todo v em S. Agora, como mostramos (6.16), então

$$x'^-(S) \geq z^-(S) - n^2\delta + n\delta \geq f(W) - n^2\delta,$$

de onde segue o resultado. ∎

Como corolário do teorema 6.3, quando o algoritmo IFF chega ao final de sua última fase de scaling, na qual $\delta < 1/n^2$, temos $x^-(S) > f(W) - 1$. Além disso, temos que $x^-(S) \leq x(W) \leq f(W)$, como também que $x^-(S) = f(U)$, para alguma parte U de S, pela relação min-max (4.3). Logo, se f é inteira, então $x^-(S) = f(W)$, de maneira que W é um minimizador de f. Disso segue a correção do algoritmo.

Teorema 6.4: *Suponha que f seja uma função submodular inteira. Quando ocorre*

o caso 1A no algoritmo IFF(S, f)*, o conjunto* W *dos vértices que acessam* $N_\delta(x', \varphi')$ *em* $H(\varphi')$*, conforme definido nessa iteração, é um minimizador de* f*.* ∎

6.6 Consumo de tempo

Para estabelecer o consumo de tempo do algoritmo IFF, começamos analisando o consumo de tempo da rotina DELTA-FASE. Vamos limitar primeiramente o número de iterações da rotina DELTA-FASE em que ocorrem δ-aumentos.

Lema 6.5 (da dualidade fraca aproximada): *Fixe* $\delta > 0$ *e seja* f *uma função submodular sobre* S*. Para todo vetor* x *em* $B(f)$ *e para todo fluxo* δ*-viável* φ*, temos*

$$(x + \partial\varphi)^-(S) \leq f(U) + \frac{n^2\delta}{4}, \text{ para todo } U \subseteq S.$$

Demonstração: Para toda parte U de S, como $|U| \leq n$, vale que

$$\begin{aligned}(x + \partial\varphi)^-(S) &\leq (x + \partial\varphi)(U) = x(U) + \partial\varphi(U) \\ &\leq f(U) + \delta|U||S \setminus U| \leq f(U) + n^2\delta/4.\end{aligned}$$

∎

Lema 6.6: *O número de* δ*-aumentos por* δ*-fase do algoritmo* IFF *é* $O(n^2)$*. Isto é, durante uma chamada à rotina* DELTA-FASE$(S, f, \delta, x, \varphi)$ *pelo algoritmo* IFF*, ocorrem* $O(n^2)$ *iterações do caso 1 dessa rotina.*

Demonstração: Considere uma iteração do algoritmo IFF em que ocorre uma chamada à rotina DELTA-FASE, com exceção da primeira. Sejam δ', x' e φ' respectivamente o parâmetro δ, o vetor x e o fluxo φ mantidos pelo algoritmo IFF no final de sua iteração anterior. Considere também $z' := x' + \partial\varphi'$. Pelo teorema 6.3, temos

$$z'^-(S) = z'(U') = x'(U') + \partial\varphi'(U') \geq f(W) - n\delta', \tag{6.19}$$

para certas partes U' e W de S.

Tomemos $z := x + \partial\varphi$, em que x e φ são respectivamente o vetor em $B(f)$ e o fluxo δ-viável que serão passados como parâmetro para a rotina DELTA-FASE nessa iteração. Considere então a chamada à rotina DELTA-FASE.

Na primeira iteração dessa chamada à rotina, temos que $x = x'$, $\partial\varphi = \partial\varphi'/2$, $\delta = \delta'/2$ e $z^-(S) = z(U)$, para alguma parte U de S.

Assim,

$$\begin{aligned} z^-(S) = z(U) &= x(U) + \partial\varphi(U) \\ &= \big(x'(U) + \partial\varphi'(U)\big) - \partial\varphi'(U)/2 \\ &\geq \big(x'(U') + \partial\varphi'(U')\big) - \partial\varphi'(U)/2 \qquad (6.20) \\ &\geq f(W) - n\delta' - \partial\varphi'(U)/2 \qquad (6.21) \\ &= f(W) - 2n\delta - \partial\varphi(U) \\ &\geq f(W) - 2n\delta - n^2\delta/4. \end{aligned}$$

Para a relação (6.20), observe que $x'(U) + \partial\varphi'(U) = z'(U) \geq z'^-(S) = z'(U') = x'(U') + \partial\varphi'(U')$. Já a relação (6.21) segue diretamente de (6.19).

Por outro lado, durante toda a execução da rotina DELTA-FASE, pelo lema 6.5 acima, temos

$$z^-(S) \leq f(W) + n^2\delta/4.$$

Segue que $z^-(S)$ pode sofrer um aumento de no máximo $(2n + n^2/2)\delta$ durante toda a δ-fase. Como $z^-(S)$ aumenta em δ a cada δ-aumento, ocorrem no máximo $O(n^2)$ δ-aumentos durante a δ-fase.

Analisemos agora o número de δ-aumentos durante a primeira invocação de DELTA-FASE pelo algoritmo IFF. No início da rotina, temos $z = \hat{x}$, para algum vetor $\hat{x}$ em $B(f)$. Além disso, pela relação de dualidade (6.6), durante toda a primeira δ-fase, temos

$$z^-(S) \leq (f + \kappa_\delta)(\emptyset) = f(\emptyset) = 0 \quad \text{e} \quad z^-(S) \leq (f + \kappa_\delta)(S) = f(S) = \hat{x}(S).$$

Utilizando os limitantes acima, concluímos que o acréscimo sofrido por $z^-(S)$ durante a primeira δ-fase está limitado por

$$\xi := \min\big\{|\hat{x}^-(S)|, \hat{x}(S) - \hat{x}^-(S)\big\} = \min\big\{|\hat{x}^-(S)|, \hat{x}^+(S)\big\}.$$

Como o parâmetro δ é inicializado com ξ/n^2, o número de δ-aumentos nessa primeira δ-fase também está limitado por n^2. ∎

Os próximos resultados limitam o número de iterações em que ocorre o caso 2 na rotina DELTA-FASE.

Lema 6.7: *Entre dois δ-aumentos ou um δ-aumento e o final de uma execução da rotina* DELTA-FASE*, ocorrem no máximo $n-1$ iterações em que a rotina* PIVOTAÇÃO-DUPLA *é não-saturadora.*

Demonstração: Quando DELTA-FASE executa uma PIVOTAÇÃO-DUPLA não-saturadora sobre o trio ativo (i, s, t), temos $\epsilon = \delta$, de maneira que o arco (t, s) têm seu fluxo zerado e, portanto, o arco (s, t) passa a fazer parte do grafo $H(\varphi)$. Segue que s passa a pertencer ao

conjunto de vértices que acessam $N_\delta(x, \varphi)$ na próxima iteração. Portanto, após no máximo $n-1$ execuções de PIVOTAÇÃO-DUPLA em que a rotina é não-saturadora, temos $W = S$, de forma que um δ-caminho é gerado ou a fase de scaling termina. ∎

Corolário 6.8: *Entre dois δ-aumentos ou um δ-aumento e o final de uma execução da rotina* DELTA-FASE*, o número de pontos extremos na representação mantida do vetor x cresce em no máximo $n-1$.*

Demonstração: O número de pontos extremos na representação mantida do vetor x só é incrementado quando a execução de PIVOTAÇÃO-DUPLA é não-saturadora. Portanto, o resultado segue diretamente do lema anterior. ∎

Lema 6.9: *Entre dois δ-aumentos ou um δ-aumento e o final de uma execução da rotina* DELTA-FASE*, ocorrem $O(n^3)$ invocações da rotina* PIVOTAÇÃO-DUPLA*.*

Demonstração: Quando ocorre uma execução de PIVOTAÇÃO-DUPLA sobre um trio ativo (i, s, t), o elemento t troca de posição com s na ordem $\prec_i$, sendo movido "para frente" e originando uma ordem $\prec_j$ que irá compor a nova representação do vetor x mantido. Se fixarmos uma ordem $\prec_i$ logo após o último δ-aumento, podem ocorrer no máximo $O(n^2)$ trocas desse tipo: entre pares de elementos em ordens que se originaram a partir de $\prec_i$. Ou seja, cada ordem fixada dá origem a no máximo $O(n^2)$ trios ativos.

Como pelo corolário 6.8 anterior o número de pontos extremos, e conseqüentemente ordens, na representação de x cresce em no máximo $n-1$ entre dois δ-aumentos e como aplicamos a rotina COMPACTA após cada δ-aumento, garantimos que o número de ordens na representação de x nunca é maior do que $2n$.

Segue que podem existir no máximo $O(n^3)$ trios ativos diferentes, de onde segue o resultado. ∎

Podemos agora limitar o número total de iterações de uma execução da rotina DELTA-FASE, bem como seu consumo de tempo.

Teorema 6.10 (do número de iterações de DELTA-FASE):
Uma chamada à rotina DELTA-FASE *pelo algoritmo* IFF *executa $O(n^5)$ iterações.*

Demonstração: Pelos lemas anteriores, sabemos que uma chamada à rotina DELTA-FASE executa $O(n^2)$ iterações em que ocorre o caso 1, que são as iterações nas quais ocorre um δ-aumento. Além disso, entre duas iterações quaisquer do caso 1, ocorrem $O(n^3)$ iterações em que ocorre o caso 2, que são as iterações em que a rotina PIVOTAÇÃO-DUPLA é executada. Disso segue o resultado. ∎

Teorema 6.11 (do consumo de tempo de DELTA-FASE):
A rotina DELTA-FASE *pode ser implementada de modo a consumir tempo $O(n^5\gamma)$ quando invocada pelo algoritmo* IFF*.*

Demonstração: No início de uma δ-fase, podemos obter os conjuntos $P_\delta(x, \varphi)$ e $N_\delta(x, \varphi)$ em tempo $O(n)$. Além disso, através de uma busca no grafo $H(\varphi)$, podemos obter o conjunto W dos vértices que acessam $N_\delta(x, \varphi)$ em tempo $O(n^2)$.

Nas iterações de uma δ-fase em que ocorre um δ-aumento, cada δ-caminho pode ser obtido em tempo $O(n^2)$. Ademais, a rotina COMPACTA consome tempo $O(n^3)$, conforme teorema 4.3, uma vez que já mostramos que o número de pontos extremos na representação mantida de x é sempre limitado por $O(n)$. Após um δ-aumento, os conjuntos $P_\delta(x, \varphi)$ e $N_\delta(x, \varphi)$ também podem ser atualizados em tempo $O(1)$. Apenas as pontas inicial e final do δ-caminho podem entrar ou sair de $P_\delta(x, \varphi)$ ou $N_\delta(x, \varphi)$. Segue que uma iteração em que ocorre um δ-aumento pode ser implementada de modo a consumir tempo $O(n^3)$. Como ocorrem $O(n^2)$ δ-aumentos por δ-fase, o tempo total consumido pelas iterações em que ocorre um δ-aumento pode ser limitado por $O(n^5)$ por δ-fase.

Entre dois δ-aumentos quaisquer, pode-se realizar todas as atualizações no conjunto W em tempo $O(n^3)$. De fato, a cada uma das $O(n)$ iterações em que após PIVOTAÇÃO-DUPLA um novo elemento s passa a pertencer a W, através de uma busca simples no grafo $H(\varphi)$, pode-se adicionar a W todos os elementos que acessam s em $H(\varphi)$ em tempo $O(n^2)$.

Para encontrar um trio ativo de maneira eficiente, o algoritmo pode manter, para cada ordem na representação de x, uma lista dos elementos em $S \setminus W$ que podem compor um trio ativo: isto é, os elementos de $S \setminus W$ que precedem algum elemento de W. O algoritmo pode guardar também, para cada ordem, o último elemento s_i^* de $S \setminus W$ que já ocupa sua posição "correta" na ordem, isto é, que não precede nenhum outro elemento de W. No início da δ-fase, essa estrutura pode ser construída em tempo $O(n^2)$, uma vez que o número de ordens é limitado sempre por $O(n)$. Quando a execução de PIVOTAÇÃO-DUPLA não altera o conjunto W, a estrutura pode ser atualizada em tempo $O(1)$, bastando para isso verificar se o elemento s em $S \setminus W$ que participou do trio ativo (i, s, t) deve sair da lista mantida para a ordem i ou não. (Se s preceder s_i^* na ordem após PIVOTAÇÃO-DUPLA, pode sair da lista mantida e ocupar o novo lugar de s_i^*.) Quando o conjunto W é atualizado, toda essa estrutura precisa ser refeita, o que consome tempo $O(n^2)$. Como o conjunto W só é atualizado em $O(n)$ iterações entre dois δ-aumentos, segue que o consumo total de tempo para obtenção dos trios ativos entre dois δ-aumentos é $O(n^3)$.

Cada chamada à PIVOTAÇÃO-DUPLA executa ainda $O(1)$ chamadas ao oráculo que define a função submodular f, de maneira que se consome tempo $O(n^3\gamma)$ com chamadas ao oráculo entre dois δ-aumentos.

Disso tudo concluímos que cada δ-fase pode ser implementada de maneira a consumir tempo $O(n^3\gamma)$ com as iterações em que ocorre uma PIVOTAÇÃO-DUPLA entre dois δ-aumentos. Como ocorrem $O(n^2)$ δ-aumentos, cada δ-fase consome tempo $O(n^5\gamma)$ com as iterações em que ocorre PIVOTAÇÃO-DUPLA. Segue também que cada δ-fase pode ser implementada de modo a consumir tempo $O(n^5\gamma)$. ∎

Tendo limitado o número de iterações bem como o consumo de tempo da rotina DELTA-FASE, podemos agora estabelecer o número de iterações, além do consumo de tempo, do algoritmo IFF por completo. Consideraremos as iterações das δ-fases como iterações do próprio algoritmo IFF.

Teorema 6.12 (do número de iterações de IFF):
O algoritmo IFF(S, f) *encontra um minimizador para a função submodular inteira* f *em* $O(n^5 \log M)$ *iterações, em que* $M := \max\{|f(U)| : U \subseteq S\}$.

Demonstração: No início do algoritmo, o parâmetro δ é inicializado com ξ/n^2, em que $\xi := \min\{|\hat{x}^-(S)|, \hat{x}^+(S)\}$, sendo $\hat{x}$ o vetor em $B(f)$ com que se inicializa x. Para $U := \{v \in S : \hat{x}(v) > 0\}$, temos $\xi \leq \hat{x}^+(S) = \hat{x}(U) \leq f(U) \leq M$, de modo que o parâmetro δ satisfaz $\delta < M/n^2$ no início do algoritmo.

Como o parâmetro δ é dividido por dois após a execução de cada DELTA-FASE e como o algoritmo termina quando $\delta < 1/n^2$, segue que o algoritmo IFF executa $O(\log M)$ δ-fases. Ademais, como acabamos de mostrar, no teorema 6.10, que cada δ-fase executa $O(n^5)$ iterações, segue o resultado. ∎

Teorema 6.13 (do consumo de tempo de IFF):
O algoritmo IFF(S, f) *pode ser implementado de modo a consumir tempo fracamente polinomial* $O(n^5 \gamma \log M)$, *em que* $M := \max\{|f(U)| : U \subseteq S\}$.

Demonstração: O resultado segue diretamente do fato de que o algoritmo IFF executa $O(\log M)$ δ-fases, como mostramos na prova do teorema acima, bem como do teorema 6.11, que estabelece o consumo de tempo de cada δ-fase. ∎

6.7 Algoritmo fortemente polinomial

Partindo das idéias presentes no algoritmo IFF, Iwata, Fleischer e Fujishige [IFF01] propuseram, no mesmo trabalho, um algoritmo fortemente polinomial para minimizar funções submodulares, o qual chamaremos IFF-FOP. Esta seção é dedicada à sua descrição.

O algoritmo IFF-FOP é iterativo e recebe também uma função submodular f sobre S tal que $f(\emptyset) = 0$, mas não exige que a função recebida seja inteira. Seu desenvolvimento baseia-se principalmente no fato de que, com alguma engenhosidade, é possível encontrar, após $O(\log n)$ δ-fases de scaling do algoritmo IFF, um dentre os seguintes três objetos:

1. um novo elemento presente em todo minimizador de f;
2. um novo elemento que não esteja em nenhum minimizador de f;
3. um novo par (v, w) de elementos tais que w está em todo minimizador de f que contém v.

Como o número de objetos distintos de um dos três tipos acima é da ordem de n^2, o algoritmo IFF-FOP encontra um minimizador para a função f após $O(n^2 \log n)$ δ-fases do algoritmo IFF. O próximo resultado motiva essas idéias.

Lema 6.14: *Seja x o vetor em $B(f)$ mantido pelo algoritmo* IFF(S, f). *No final de uma δ-fase do algoritmo, vale que:*

(a) Se $x(v) < -n^2\delta$, então v pertence a qualquer minimizador de f;

(b) Se $x(v) > n^2\delta$, então v não pertence a nenhum minimizador de f.

Demonstração: Pelo teorema 6.3, no final de uma δ-fase do algoritmo IFF, existe uma parte X de S tal que $x^-(S) \geq f(X) - n^2\delta$. Além disso, para qualquer minimizador W de f, temos $f(X) \geq f(W) \geq x(W) \geq x^-(W)$. Conseqüentemente, vale que

$$x^-(S) \geq f(X) - n^2\delta \geq x^-(W) - n^2\delta. \tag{6.22}$$

Temos também que

$$x^-(W) \geq x^-(S) \geq f(X) - n^2\delta \geq x(W) - n^2\delta. \tag{6.23}$$

Seja W um minimizador de f. Suponha, então, que, no final de determinada δ-fase, valha que $x(v) < -n^2\delta$, para algum v em S. Suponha ainda por contradição que v não esteja em W. Então v está em $S \setminus W$, de maneira que

$$x^-(S) - x^-(W) = x^-(S \setminus W) \leq x(v) < -n^2\delta,$$

o que é uma contradição com (6.22).

Suponha agora que $x(v) > n^2\delta$, para algum v em S, no final de determinada δ-fase de scaling. Suponha ainda que v esteja em W. Como $x(v) > 0$, então

$$x^-(W) \leq x(W) - x(v) < x(W) - n^2\delta,$$

o que é uma contradição com (6.23). Dessas duas contradições segue o resultado. ∎

Rotina ROTULA: identificação de objetos

A cada iteração do algoritmo IFF-FOP, um dentre os três objetos descritos no início da seção é encontrado com o auxílio da rotina ROTULA, cuja especificação está a seguir.

Rotina ROTULA(S, f, y, η): *Recebe uma função submodular f sobre S, um ponto extremo y de $B(f)$ e um parâmetro $\eta \geq 0$ tais que $f(S) \geq \eta/3$ ou $f(U) \leq -\eta/3$, para alguma parte U de S. Devolve um elemento v de S tal que v está em qualquer minimizador de f ou v não está em minimizador algum de f.*

Para encontrar o elemento v como descrito acima, a rotina ROTULA faz uma seqüência de invocações à rotina DELTA-FASE, descrita na seção 6.3.

A rotina ROTULA é iterativa e mantém um parâmetro $\delta \geq 0$, uma representação de um vetor x em $B(f)$ e um fluxo δ-viável φ. No início da primeira iteração, o parâmetro δ é inicializado com η, o vetor x com o ponto extremo y e φ é o fluxo identicamente nulo. As δ-fases são executadas até que se tenha $\delta < \eta/(3n^3)$, onde $n := |S|$.

Uma iteração da rotina ROTULA está descrita a seguir.

Caso 1: $\delta < \eta/(3n^3)$.
Se $f(S) \geq \eta/3$, seja v tal que $x(v) > n^2\delta$. [v em nenhum minimizador de f]
Senão, seja v tal que $x(v) < -n^2\delta$. [v em qualquer minimizador de f]
Devolva v.

Caso 2: $\delta \geq \eta/(3n^3)$.
$(x', \varphi') \leftarrow$ DELTA-FASE$(S, f, \delta, x, \varphi)$.
$\delta' \leftarrow \delta/2$.
$\varphi' \leftarrow \varphi'/2$.
Comece nova iteração com δ', x' e φ' respectivamente nos papéis de δ, x e φ.

Note que, como δ é inicializado com η, então a rotina ROTULA executa $O(\log n)$ δ-fases. Podemos, portanto, estabelecer o seguinte resultado para referências futuras.

Lema 6.15: *A rotina* ROTULA(S, f, y, η) *executa* $O(\log n)$ δ*-fases.* ∎

Utilizando o resultado do lema 6.14, mostramos que, após a execução da última δ-fase de scaling pela rotina ROTULA, um elemento v nas condições especificadas pode ser encontrado. O lema abaixo garante a correção da rotina.

Lema 6.16: *Seja x o vetor em $B(f)$ mantido por* ROTULA(S, f, y, η) *na iteração em que a rotina termina, isto é, quando $\delta < \eta/(3n^3)$. Nesse momento, vale que:*

(a) Se $f(S) \geq \eta/3$, então existe um elemento v em S que satisfaz $x(v) > n^2\delta$. Conseqüentemente, v não pertence a nenhum minimizador de f.

(b) Se $f(U) \leq -\eta/3$, para alguma parte U de S, então existe um elemento v em S que satisfaz $x(v) < -n^2\delta$. Conseqüentemente, v é um elemento de qualquer minimizador de f.

Demonstração: Sabemos que $x^-(S) \leq x(U) \leq f(U)$, para qualquer parte U de S. Além disso, como x é um vetor em $B(f)$, então $x(S) = f(S)$, de modo que, se $f(S) \geq \eta/3$ quando a rotina ROTULA termina, então

$$x(S) = f(S) \geq \eta/3 > n^3\delta,$$

já que $\delta < \eta/(3n^3)$ nesse momento. Segue que existe pelo menos um v em S que satisfaz $x(v) > n^2\delta$. Além disso, pelo lema 6.14, v não está em nenhum minimizador de f.

Por outro lado, quando a rotina ROTULA termina, se $f(U) \leq -\eta/3$, para algum U, então

$$x^-(S) \leq f(U) \leq -\eta/3 < -n^3\delta,$$

já que $\delta < \eta/(3n^3)$ nesse momento. Analogamente, segue que existe pelo menos um v em S que satisfaz $x(v) < -n^2\delta$, de maneira que, pelo lema 6.14, v está em qualquer minimizador de f. ∎

Uma iteração do algoritmo

Descrevemos aqui as idéias envolvidas em uma iteração do algoritmo IFF-FOP. Mais abaixo apresentamos um pseudo-código completo.

Cada iteração do algoritmo IFF-FOP mantém:

- Uma parte W de S contida em qualquer minimizador de f;
- Uma parte V de $S \setminus W$ e uma coleção de conjuntos disjuntos $\Gamma := \{\Gamma(v) : v \in V\}$, com v em $\Gamma(v)$ e $\Gamma(v)$ contido em $S \setminus W$, para cada v em V. Para cada parte V' de V, denotamos $\Gamma(V') := \cup_{v \in V'} \Gamma(v)$;
- Uma função submodular $\hat{f}$ sobre V tal que se V' minimiza $\hat{f}$, então $W \cup \Gamma(V')$ minimiza f.
- Um grafo orientado D sobre V tal que existe um arco (v, w) em D se, e somente se, w está em todo minimizador de $\hat{f}$ que contém v. Se V' é uma parte de V, denotamos por $D(V')$ o subgrafo de D induzido por V'.

Durante a demonstração da correção do algoritmo, mostraremos que os objetos mantidos satisfazem de fato as propriedades descritas acima e que, portanto, essas relações são invariantes do algoritmo.

Seguimos com mais algumas definições. Para cada v em V, defina $R(v)$ como o conjunto de vértices acessíveis a partir de v em D. Defina também $\hat{f}_v$ como a **contração de $\hat{f}$ por $R(v)$** da seguinte maneira:

$$\hat{f}_v(U) := \hat{f}(U \cup R(v)) - \hat{f}(R(v)), \text{ para todo } U \subseteq V \setminus R(v). \tag{6.24}$$

Observe que a função $\hat{f}_v$ é também uma função submodular. Além disso, um minimizador de $\hat{f}_v$ é uma parte de U de $V \setminus R(v)$ tal que $\hat{f}(U \cup R(v))$ é mínimo.

Uma ordem linear $\prec$ de V é **consistente com o grafo** D se $w \prec v$ sempre que o arco (v, w) está em D. Um ponto extremo de $B(\hat{f})$ gerado por uma ordem linear consistente de V é chamado **consistente**. O próximo lema é importante para a correção do algoritmo.

Lema 6.17: *Seja y um ponto extremo consistente de $B(\hat{f})$. Então, para cada v em V, vale que*

$$y(v) \leq \hat{f}\big(R(v)\big) - \hat{f}\big(R(v) \setminus \{v\}\big). \tag{6.25}$$

Demonstração: Seja $\prec$ a ordem consistente de V que gera y. Pela definição de ordem consistente, temos que $R(v) \subseteq [v]_\prec$, para cada v em V. Ademais, pela forma como o vetor y é definido pelo método guloso em (2.17), na seção 2.6, vale que $y(v) = \hat{f}\big([v]_\prec\big) - \hat{f}\big([v]_\prec \setminus \{v\}\big)$. Logo, pela submodularidade de $\hat{f}$, temos

$$y(v) = \hat{f}\big([v]_\prec\big) - \hat{f}\big([v]_\prec \setminus \{v\}\big) \leq \hat{f}\big(R(v)\big) - \hat{f}\big(R(v) \setminus \{v\}\big).$$

∎

Cada iteração do algoritmo IFF-FOP começa definindo

$$\eta := \max\Big\{\hat{f}\big(R(v)\big) - \hat{f}\big(R(v) \setminus \{v\}\big) : v \in V\Big\}, \tag{6.26}$$

que será utilizado para a chamada da rotina ROTULA. Essa escolha de η é determinada de forma que o número de δ-aumentos na primeira δ-fase da rotina não seja tão grande, conforme veremos na análise de consumo de tempo do algoritmo.

Se $V = \emptyset$ ou $\eta \leq 0$, o algoritmo pára devolvendo $W \cup \Gamma(V)$. Conforme veremos mais abaixo durante a prova da correção do algoritmo, as propriedades invariantes aos objetos mantidos garantem que esse conjunto é de fato um minimizador da função f nesses casos.

Se $\eta > 0$, considere um elemento v em V que atinja o máximo em (6.26). Como

$$\begin{aligned}\eta &= \hat{f}\big(R(v)\big) - \hat{f}\big(R(v) \setminus \{v\}\big) \\ &= \Big(\hat{f}(V)\Big) + \Big(-\hat{f}\big(R(v) \setminus \{v\}\big)\Big) + \Big(\hat{f}\big(R(v)\big) - \hat{f}(V)\Big),\end{aligned}$$

então devemos ter

$$\max\Big\{\hat{f}(V), -\hat{f}\big(R(v) \setminus \{v\}\big), \hat{f}\big(R(v)\big) - \hat{f}(V)\Big\} \geq \eta/3. \tag{6.27}$$

O algoritmo segue, então, com o primeiro caso que for aplicável.

Se $\hat{f}(V) \geq \eta/3 > 0$, o algoritmo aplica ROTULA$(V, \hat{f}, y, \eta)$, em que y é um ponto extremo de $B(\hat{f})$ consistente com D, e encontra um elemento w de V que não está em nenhum minimizador de $\hat{f}$ (lema 6.16). Lembre que $\hat{f}$ é uma função submodular definida sobre V. Nesse caso, se w está em $R(u)$, algum u, então u também não está em nenhum minimizador de $\hat{f}$, pela definição de $R(u)$. É suficiente, portanto, minimizar $\hat{f}$ sobre $V \setminus \{u : w \in R(u)\}$. Excluímos, então, $\{u : w \in R(u)\}$ de V, obtendo V', e restringimos os demais objetos definidos sobre V a V', antes de partir para a próxima iteração. Restringir Γ a V' significa manter na coleção apenas os conjuntos $\Gamma(v)$, com v em V'. No caso de $\hat{f}$, a função passa a estar definida apenas sobre as partes de V'. Por fim, tomamos o grafo $D(V')$, induzido por V', no lugar de D.

Se $\hat{f}\big(R(v) \setminus \{v\}\big) \leq -\eta/3$, o algoritmo aplica ROTULA$(V, \hat{f}, y, \eta)$, novamente com y sendo um ponto extremo de $B(\hat{f})$ consistente com D, e encontra um novo elemento w que está em todo minimizador de $\hat{f}$. Nesse caso, todo minimizador de $\hat{f}$ contém $R(w)$

e, pela maneira como os objetos são definidos, todo minimizador da função original f contém $\Gamma(R(w))$. Assim, incluímos $\Gamma(R(w))$ em W e redefinimos $\hat{f}$ como $\hat{f}_w$ antes de partir para a próxima iteração. Excluímos também $R(w)$ de V e restringimos Γ e D ao novo V.

Por fim, se $\hat{f}(V) - \hat{f}(R(v)) = \hat{f}_v(V \setminus R(v)) \leq -\eta/3$, o algoritmo aplica a rotina ROTULA$(V \setminus R(v), \hat{f}_v, y_v, \eta)$, em que y_v é um ponto extremo de $B(\hat{f}_v)$ consistente com o grafo $D(V \setminus R(v))$, que é o subgrafo de D induzido pelo conjunto sobre o qual a função $\hat{f}_v$ está definida. Nesse caso, a rotina ROTULA encontra um elemento w que está em todo minimizador de $\hat{f}_v$, também pelo lema 6.16. Conseqüentemente, w está em todo minimizador de $\hat{f}$ que contém v. O algoritmo, portanto, adiciona o arco (v, w) ao grafo D. Se com a adição desse arco criamos um circuito em D, então ou todos os elementos que estão em caminhos de w a v em D estão contidos em um minimizador de $\hat{f}$ ou nenhum deles está. Assim, antes de seguir para a próxima iteração, o algoritmo transforma todo esse conjunto de elementos em um único vértice do grafo D e modifica V e $\hat{f}$ de maneira a refletir essa nova situação.

Descrição do algoritmo

Segue descrição do algoritmo IFF-FOP.

Algoritmo IFF-FOP(S, f): *Recebe uma função submodular f sobre S tal que $f(\emptyset) = 0$ e devolve uma parte W de S que minimiza f.*

No início da primeira iteração, temos:

- $W := \emptyset$;
- $V := S$, $\Gamma(v) := \{v\}$, para cada v em V;
- $\hat{f} := f$;
- D como o grafo sobre V sem nenhum arco.

Uma iteração qualquer do algoritmo está a seguir. Cabe lembrar que mais de um dentre os casos 2A, 2B e 2C descritos abaixo podem ocorrer simultaneamente. O algoritmo deve seguir com o primeiro caso que for aplicável. Seja

$$\eta := \max\left\{\hat{f}(R(v)) - \hat{f}(R(v) \setminus \{v\}) \colon v \in V\right\}$$

Caso 1: $V = \emptyset$ ou $\eta \leq 0$.
Devolva $W \cup \Gamma(V)$.

Caso 2: $\eta > 0$.
Seja v um elemento tal que $\hat{f}\big(R(v)\big) - \hat{f}\big(R(v) \setminus \{v\}\big) = \eta$.

Caso 2A: $\hat{f}(V) \geq \eta/3$.

Seja y um ponto extremo de $B(\hat{f})$ consistente com D.
$w \leftarrow \textsc{Rotula}(V, \hat{f}, y, \eta)$. [$w$ em nenhum minimizador de $\hat{f}$]
$V' \leftarrow V \setminus \{u : w \in R(u)\}$.
$\Gamma', \hat{f}', D' \leftarrow$ restrição de $\Gamma, \hat{f}, D$ a V'.
Comece nova iteração com V', Γ', $\hat{f}'$ e D' respectivamente nos papéis de V, Γ, $\hat{f}$ e D.

Caso 2B: $\hat{f}(R(v) \setminus \{v\}) \leq -\eta/3$.

Seja y um ponto extremo de $B(\hat{f})$ consistente com D.
$w \leftarrow \textsc{Rotula}(V, \hat{f}, y, \eta)$. [$w$ em todo minimizador de $\hat{f}$]
$W' \leftarrow W \cup \Gamma(R(w))$.
$V' \leftarrow V \setminus R(w)$.
$\hat{f}' \leftarrow \hat{f}_w$.
$\Gamma', D' \leftarrow$ restrição de Γ, D a V'
Comece nova iteração com W', V', Γ', $\hat{f}'$ e D' respectivamente nos papéis de W, V, Γ, $\hat{f}$ e D.

Caso 2C: $\hat{f}_v(V \setminus R(v)) = \hat{f}(V) - \hat{f}(R(v)) \leq -\eta/3$.

Seja D_v o subgrafo de D induzido por $V \setminus R(v)$.
Seja y_v um ponto extremo de $B(\hat{f}_v)$ consistente com D_v.
$w \leftarrow \textsc{Rotula}(V \setminus R(v), \hat{f}_v, y_v, \eta)$. [$w$ em todo minimizador de $\hat{f}$ que contém v]

Caso 2C1: v está em $R(w)$. [v em todo minimizador de $\hat{f}$ que contém w]

Seja $K := \{u : u \in R(w),\ v \in R(u)\}$.
Seja D' o grafo obtido quando se contrai K a um único vértice v em D.
$V' \leftarrow (V \setminus K) \cup \{v\}$.
Seja $\hat{f}'$ a função submodular sobre V' tal que, para cada U em V',

$$\begin{aligned} \hat{f}'(U) &= \hat{f}(U), && \text{se } v \text{ não está em } U; \\ \hat{f}'(U) &= \hat{f}(U \cup K), && \text{se } v \text{ está em } U. \end{aligned}$$

$\Gamma(v) \leftarrow \Gamma(v) \cup K$.
$\Gamma' \leftarrow$ restrição de Γ a V'.
Comece nova iteração com V', Γ', $\hat{f}'$ e D' respectivamente nos papéis de V, Γ, $\hat{f}$ e D.

Caso 2C2: v não está em $R(w)$.

Seja D' o grafo obtido ao adicionarmos o arco (v, w) a D.
Comece nova iteração com D' no papel de D.

Correção e consumo de tempo

Começamos mostrando mais formalmente que de fato são invariantes as propriedades enunciadas sobre os objetos mantidos pelo algoritmo.

Lema 6.18: *Sejam W, V, Γ, $\hat{f}$ e D os objetos mantidos pelo algoritmo* IFF-FOP(S, f). *No início de cada iteração desse algoritmo, vale que:*

> (i) *W está contido em qualquer minimizador de f.*
> (ii) *Se $V' \subseteq V$ minimiza a função $\hat{f}$, então $W \cup \Gamma(V')$ minimiza f.*
> (iii) *Existe um arco (v, w) no grafo D se, e somente se, w está em todo minimizador de $\hat{f}$ que contém v.*

Demonstração: Pela maneira como os objetos W, V, Γ, $\hat{f}$ e D são definidos no começo do algoritmo, é fácil perceber que as propriedades (i), (ii) e (iii) são válidas no início da primeira iteração. Vamos supor, então, que elas valham no início de uma iteração qualquer do algoritmo e mostrar que continuam válidas ao seu final.

Quando ocorre o caso 2A, pelo lema 6.16, temos que o elemento w devolvido pela rotina ROTULA$(V, \hat{f}, y, \eta)$ não está em nenhum minimizador de $\hat{f}$. Além disso, pela validade de (iii) no início da iteração, se w está em $R(u)$, então w está em todo minimizador de $\hat{f}$ que contém u. Conseqüentemente, se w está em $R(u)$, então u também não está em nenhum minimizador de $\hat{f}$. Segue que a atualização em V realizada durante a iteração mantém a validade das três propriedades.

Quando ocorre o caso 2B, também pelo lema 6.16, o elemento w devolvido por ROTULA$(V, \hat{f}, y, \eta)$ está em todo minimizador de $\hat{f}$. Assim, novamente pela validade da propriedade (iii) no início da iteração, temos que $R(w)$ também está em todo minimizador de $\hat{f}$. Disso segue que, se U é um minimizador de $\hat{f}_w$, então $U \cup R(w)$ é um minimizador de $\hat{f}$. Conseqüentemente, pela validade de (ii) no início da iteração, temos que $W \cup \Gamma(R(w))$ está em todo minimizador de f, bem como que se U minimiza $\hat{f}_w$, então $W \cup \Gamma(R(w) \cup U) = W \cup \Gamma(R(w)) \cup \Gamma(U)$ minimiza f. Segue que as atualizações realizadas nos objetos durante a iteração mantêm a validade das propriedades no início da iteração seguinte.

Quando ocorre o caso 2C, o lema 6.16 nos garante que o elemento w devolvido por ROTULA$(V \setminus R(v), \hat{f}_v, y_v, \eta)$ está em todo minimizador de $\hat{f}_v$. Conseqüentemente, w está em todo minimizador de $\hat{f}$ que contém $R(v)$ e, portanto, em todo minimizador de $\hat{f}$ que contém v. Logo, a adição do arco (v, w) ao grafo D realizada no caso 2C2 mantém a validade das três propriedades. Já quando ocorre o caso 2C1, temos que K representa um conjunto de elementos tais que se um deles está em algum minimizador de $\hat{f}$, então todos os outros também estão. Assim, quando fazemos com que todo o conjunto K seja tratado como um único elemento, seja no grafo D ou no conjunto V, estamos mantendo a validade das propriedades na próxima iteração. ∎

Utilizando especialmente a relação invariante (ii) no lema acima, mostramos agora que, se o algoritmo IFF-FOP pára, então ele devolve um minimizador da função f.

Teorema 6.19: *Suponha que o caso 1 ocorra em determinada iteração do algoritmo* IFF-FOP(S, f). *O conjunto $W \cup \Gamma(V)$, devolvido pelo algoritmo nessa iteração, é*

um minimizador da função f.

Demonstração: Se $V = \emptyset$, então o resultado segue diretamente da propriedade (ii) no lema 6.18 acima. Note que, nesse caso, $\hat{f}$ está definida sobre $\emptyset$ e que $\Gamma(V) = \emptyset$.

Por outro lado, se $\eta \leq 0$, então, pelo lema 6.17, qualquer ponto extremo y em $B(\hat{f})$ consistente com D satisfaz $y(v) \leq 0$, para todo v em V. Conseqüentemente, $y^-(V) = y(V) = \hat{f}(V)$, já que y está em $B(\hat{f})$ e que a função $\hat{f}$ é definida sobre V. Disso concluímos, pela relação min-max (4.3), que V é um minimizador de $\hat{f}$, de maneira que, pela propriedade (ii) no lema 6.18, $W \cup \Gamma(V)$ é um minimizador da função f original. ∎

Por fim, limitamos o número de iterações, bem como o consumo de tempo do algoritmo IFF-FOP.

Teorema 6.20 (do número de iterações de IFF-FOP):
O algoritmo IFF-FOP(S, f) *encontra um minimizador para a função submodular f em $O(n^2)$ iterações.*

Demonstração: Em cada iteração do algoritmo IFF-FOP em que ocorrem os casos 2A, 2B ou 2C1 temos que o elemento w é excluído do conjunto V mantido. Além disso, quando ocorre o caso 2C2, um novo arco (v, w) é adicionado ao grafo D. Note que, nesse caso, como w é um elemento de $V \setminus R(v)$, o arco (v, w) de fato ainda não existia em D. Segue que o algoritmo executa no máximo $O(n^2)$ iterações. ∎

Teorema 6.21 (do consumo de tempo de IFF-FOP):
O algoritmo IFF-FOP(S, f) *é fortemente polinomial e pode ser implementado de modo a consumir tempo $O(n^7 \gamma \log n)$.*

Demonstração: Pelo lema 6.15, em cada iteração do algoritmo IFF-FOP, cada chamada à rotina ROTULA executa $O(\log n)$ δ-fases, representadas pelas chamadas à rotina DELTA-FASE, de maneira que o algoritmo como um todo executa $O(n^2 \log n)$ δ-fases.

Se mostrarmos que a primeira DELTA-FASE de cada chamada à rotina ROTULA pelo algoritmo IFF-FOP também executa $O(n^2)$ δ-aumentos, de forma semelhante ao que acontecia no algoritmo IFF, concluiremos que o lema 6.6, que estabelece o número de δ-aumentos por δ-fase, bem como todos os demais resultados sobre a rotina DELTA-FASE no algoritmo IFF, seguem válidos para as δ-fases do algoritmo IFF-FOP. Em particular, poderemos utilizar o teorema 6.11 e concluir que cada δ-fase do algoritmo IFF-FOP pode ser implementada de modo a consumir tempo $O(n^5\gamma)$, de maneira que chegaremos ao limitante $O(n^7 \gamma \log n)$ para o consumo de tempo do algoritmo IFF-FOP.

A única diferença entre uma seqüência de chamadas a DELTA-FASE em uma execução de ROTULA ou no algoritmo IFF está na forma como o parâmetro δ é inicializado. Em uma execução de DELTA-FASE pela rotina ROTULA, o parâmetro δ é inicializado com $\eta := \max\{\hat{f}(R(v)) - \hat{f}(R(v) \setminus \{v\}) : v \in V\}$. Nesse caso, pelo lema 6.17, temos $y(u) \leq \eta$, para todo u em V e qualquer ponto extremo y em $B(\hat{f})$ consistente com D. Assim, quando o

algoritmo IFF-FOP executa ROTULA$(V, \hat{f}, y, \eta)$, temos $y^+(V) \leq n\eta$. Retomando agora a prova do lema 6.6, temos que o número de δ-aumentos na primeira δ-fase de uma chamada a ROTULA$(V, \hat{f}, y, \eta)$ é limitado por $y^+(V)/\delta = y^+(V)/\eta \leq n$. Lembre-se que o vetor x é inicializado com y e o parâmetro δ com η pela rotina ROTULA$(V, \hat{f}, y, \eta)$.

Analogamente, utilizando a submodularidade de $\hat{f}$, concluímos que $y_v(u) \leq \hat{f}_v(R(u)) - \hat{f}_v(R(u) \setminus \{u\}) \leq \hat{f}(R(u)) - \hat{f}(R(u) \setminus \{u\}) \leq \eta$, para cada u em $V \setminus R(v)$ e qualquer ponto extremo y_v em $B(\hat{f}_v)$ consistente com $D(V \setminus R(v))$. Conseqüentemente, o número de δ-aumentos na primeira δ-fase de uma chamada a ROTULA$(V \setminus R(v), \hat{f}_v, y_v, \eta)$ também está limitado por $y^+(V \setminus R(v))/\eta \leq n$.

Concluímos, assim, que todas as δ-fases no algoritmo IFF-FOP também executam $O(n^2)$ δ-aumentos, de onde segue o resultado. ∎

6.8 Outros avanços

Quando apresentou seu algoritmo, Schrijver [Sch00] acreditava que seria possível o desenvolvimento de um algoritmo para minimizar funções submodulares que fosse **totalmente combinatório**, no sentido de utilizar somente adições, subtrações, comparações e chamadas ao oráculo que define a função submodular. Note que os algoritmos que apresentamos até aqui realizam também multiplicações e divisões, apesar dessas operações não estarem envolvidas na definição do problema que resolvem. O algoritmo de Cunningham para o problema da pertinência no politopo dos conjuntos independentes de um matróide, discutido na seção 4.1, é totalmente combinatório nesse sentido. Conforme expõe Iwata [Iwa02], uma algoritmo totalmente combinatório para minimizar funções submodulares apresenta ainda como vantagem o fato de poder ser mais facilmente estendido para resolver o problema sobre qualquer grupo totalmente ordenado.

Um algoritmo totalmente combinatório para minimizar uma função submodular certamente não pode se utilizar da rotina COMPACTA. Iwata, Fleischer e Fujishige [IFF01] mostraram que a utilização dessa rotina não é estritamente necessária para que se garanta a polinomialidade dos algoritmos IFF e IFF-FOP, como discutimos a seguir. Esse fato já não é verdade quando consideramos o algoritmo de Schrijver, descrito no capítulo 5. Se a rotina COMPACTA não for utilizada, o número de pontos extremos na representação que o algoritmo SCHRIJVER mantém do vetor x pode se tornar exponencial em n, bem como o seu número de iterações e consumo de tempo.

Seja I_x o conjunto de índices que compõe a representação do vetor x mantido pela rotina DELTA-FASE, quando esta é invocada pelo algoritmo IFF. O corolário 6.8 nos garante que $|I_x|$ cresce em no máximo $n-1$ entre dois δ-aumentos. Além disso, conforme podemos observar através da prova do corolário 6.6, o número de δ-aumentos por δ-fase não depende de $|I_x|$. Ou seja, o número de δ-aumentos por chamada à DELTA-FASE continua limitado por $O(n^2)$ independentemente da aplicação da rotina COMPACTA. Uma vez que o algoritmo IFF invoca $O(\log M)$ vezes a rotina DELTA-FASE, segue que ocor-

rem $O(n^2 \log M)$ δ-aumentos durante uma execução desse algoritmo, como também que o número de pontos extremos na representação de x pode ser limitado por $O(n^3 \log M)$ durante todo o algoritmo IFF, quando a rotina COMPACTA não é utilizada.

Apesar de não alterar o número de δ-aumentos por δ-fase, o tamanho do conjunto I_x afeta o número de chamadas à rotina PIVOTAÇÃO-DUPLA dentro de cada δ-fase. Na prova do lema 6.9, explicamos que, entre dois δ-aumentos, o número de invocações da rotina PIVOTAÇÃO-DUPLA é limitado por $O(n^2)$ vezes o número de pontos extremos que compõem a representação de x. Como concluímos que $|I_x|$ é limitado por $O(n^3 \log M)$, segue que cada chamada à rotina DELTA-FASE pelo algoritmo IFF invoca $O(n^5 \log M)$ vezes a rotina PIVOTAÇÃO-DUPLA, entre dois δ-aumentos, se COMPACTA não for utilizada. Como ocorrem $O(n^2)$ δ-aumentos por δ-fase, concluímos que a rotina DELTA-FASE pode ser implementada de modo a consumir tempo $O(n^7 \gamma \log M)$ e, conseqüentemente, que essa versão "mais combinatória" do algoritmo IFF, sem a utilização de COMPACTA, pode ser implementada de modo a consumir tempo $O(n^7 \gamma \log^2 M)$.

De forma semelhante, o algoritmo IFF-FOP não depende da aplicação da rotina COMPACTA para a garantia de seu consumo de tempo fortemente polinomial. Pelo lema 6.15 e pela prova do teorema 6.21, cada chamada à rotina ROTULA executa $O(\log n)$ δ-fases e, portanto, $O(n^2 \log n)$ δ-aumentos. Como cada chamada à rotina ROTULA começa com um único ponto extremo na representação de x e como entre dois δ-aumentos $|I_x|$ cresce em no máximo $n-1$, segue que $|I_x|$ está limitado por $O(n^3 \log n)$ durante toda a rotina ROTULA.

Disso concluímos que o número de invocações à rotina PIVOTAÇÃO-DUPLA dentro de cada δ-fase de ROTULA e entre dois δ-aumentos é limitado por $O(n^5 \log n)$, de onde segue que a rotina DELTA-FASE pode ser implementada de modo a consumir tempo $O(n^7 \gamma \log n)$ quando invocada por IFF-FOP, se COMPACTA não é utilizada. Como também pela prova do teorema 6.21 sabemos que o algoritmo IFF-FOP executa $O(n^2 \log n)$ δ-fases, segue que seu consumo de tempo pode ser limitado por $O(n^9 \gamma \log^2 n)$, sem a utilização de COMPACTA.

Utilizando essas versões dos algoritmos IFF e IFF-FOP que não se utilizam de COMPACTA, Iwata [Iwa02] resolveu a questão proposta por Schrijver e apresentou uma implementação fortemente polinomial e totalmente combinatória para o algoritmo IFF-FOP.

Além de não utilizar COMPACTA, uma implementação de IFF-FOP totalmente combinatória precisa tomar algum cuidado com os coeficientes que compõem a combinação convexa que descreve o vetor x em $B(f)$. Conforme expõe Iwata em seu trabalho [Iwa02], em um algoritmo totalmente combinatório e fortemente polinomial, além de realizar adições, subtrações e comparações, podemos também simular multiplicações se os fatores são inteiros e limitados por um polinômio na dimensão n do problema. Podemos ainda arredondar uma fração de dois números, desde que a resposta seja também limitada por um polinômio de dimensão n. No algoritmo que propõe, Iwata faz uma adaptação na rotina PIVOTAÇÃO-DUPLA, de maneira a garantir que os coeficientes na representação de x sejam sempre números racionais com um denominador comum limitado por um polinômio em n.

Assim como Fleischer e Iwata fizeram com o algoritmo de Schrijver, conforme descrevemos na seção 5.6, Iwata [Iwa03] refinou ainda o algoritmo IFF, embutindo-o num arcabouço push-relabel. Iwata também utilizou idéias presentes no algoritmo de Schrijver, como a realização de uma série de pivotações simultâneas. A partir disso, obteve um novo algoritmo combinatório que minimiza uma função submodular inteira f em tempo fracamente polinomial $O((n^4\gamma + n^5)\log M)$. Seu algoritmo também é baseado em δ-fases. Substituindo as δ-fases no algoritmo IFF-FOP pelas δ-fases desse último algoritmo desenvolvido, Iwata obteve, no mesmo trabalho, um algoritmo fortemente polinomial que minimiza uma função submodular qualquer em tempo $O((n^6\gamma + n^7)\log n)$.

Utilizando as mesmas idéias que transformaram o algoritmo IFF-FOP em totalmente combinatório, Iwata mostra, ainda no mesmo trabalho, que a nova versão fortemente polinomial mais rápida do algoritmo pode também ser implementada de forma totalmente combinatória, com consumo de tempo limitado por $O(n^8\gamma\log^2 n)$.

Capítulo 7

Submodularidade em combinatória

Em vários teoremas e problemas combinatórios submodularidade está envolvida de uma forma ou outra e freqüentemente desempenha um papel essencial em uma demonstração ou na eficiência de um algoritmo. Este capítulo exemplifica esse fenômeno com mais detalhes e procura motivar o estudo dos aspectos teóricos e algorítmicos associados a funções submodulares apresentados nos capítulos anteriores.

Começamos com uma breve visão geral sobre modelos poliédricos envolvendo funções submodulares. Tais modelos se traduzem em arcabouços bastante poderosos que generalizam e unificam diversos resultados importantes em otimização combinatória. Na verdade, seria possível dedicar um trabalho inteiro a esse assunto, de maneira que nos limitamos a apresentar idéias gerais.

Feito isso, apresentamos demonstrações de alguns resultados em otimização combinatória e em teoria dos grafos que se utilizam de submodularidade. A submodularidade de certas funções tem se mostrado importante ferramenta para o desenvolvimento de demonstrações combinatórias curtas e elegantes para diversos problemas.

Uma resenha de Frank [Fra93a] aponta que técnicas em submodularidade como as utilizadas nas demonstrações que apresentamos neste capítulo têm sido freqüentes e cruciais na resolução de problemas relacionados a orientações, empacotamentos e coberturas de árvores e arborescências, aumento de conexidade e caminhos disjuntos em grafos.

7.1 Cortes, famílias laminares e famílias livres de cruzamentos

Apresentamos aqui algumas definições e conceitos que serão utilizados em várias demonstrações do capítulo. Conceitos utilizados apenas em seções específicas serão definidos nas próprias seções.

Seja $D = (V, A)$ um grafo orientado. Para cada parte U de V, denotamos por $\delta^{\text{in}}_D(U)$ o conjunto de arcos que entram em U e por $\delta^{\text{out}}_D(U)$ o conjunto de arcos que saem de U. Ademais, denotamos $d^{\text{in}}_D(U) := |\delta^{\text{in}}_D(U)|$ e $d^{\text{out}}_D(U) := |\delta^{\text{out}}_D(U)|$. Quando o grafo D está implícito pelo contexto, denotamos apenas $\delta^{\text{in}}(U)$, $\delta^{\text{out}}(U)$, $d^{\text{in}}(U)$ e $d^{\text{out}}(U)$.

Conforme já mencionamos no capítulo 1, as funções d_D^{in} e d_D^{out} são submodulares. Na verdade, para quaisquer subconjuntos T e U de vértices de D, vale que

$$d^{\text{in}}(T) + d^{\text{in}}(U) = d^{\text{in}}(T \cap U) + d^{\text{in}}(T \cup U) + d(T, U), \tag{7.1}$$

em que $d(T, U)$ indica o número de arcos entre $T \setminus U$ e $U \setminus T$. A relação (7.1) pode ser verificada através de um simples argumento de contagem.

Seja $G = (V, E)$ um grafo. Para cada parte U de V, denotamos por $\delta_G(U)$ o conjunto de arestas entre U e $V \setminus U$ em G. Ademais, denotamos $d_G(U) := |\delta_G(U)|$. Quando o grafo G está implícito pelo contexto, denotamos simplesmente $\delta(U)$ e $d(U)$. A função d_G também é submodular. A submodularidade das funções d_D^{in}, d_D^{out} e d_G é um ingrediente chave em várias das demostrações apresentadas neste capítulo.

Seja V um conjunto finito. Duas partes T e U de V são **intersectantes** se nenhum dos conjuntos $T \cap U$, $T \setminus U$ e $U \setminus T$ é vazio. Se além disso $V \setminus (T \cup U)$ é não-vazio, então dizemos que T e U **se cruzam**. Seja $\mathcal{F}$ uma família de subconjuntos de V. A família $\mathcal{F}$ é **livre de cruzamentos** se não possui dois membros que se cruzam. Ademais, $\mathcal{F}$ é **laminar** se não possui dois membros intersectantes.

Uma **subpartição** $\mathcal{F}$ de V é uma coleção de subconjuntos de V dois-a-dois disjuntos. Uma **partição** $\mathcal{P}$ de V é uma subpartição de V tal que $\bigcup_{P \in \mathcal{P}} P = V$.

Se $G = (V, E)$ é um grafo e se e, f são arestas de G, então $G - e + f$ é o grafo $G' := \big(V, (E \setminus \{e\}) \cup \{f\}\big)$. Analogamente, se $D = (V, A)$ é um grafo orientado e se a, b são arcos de D, então $D - a + b$ é o grafo orientado $D' := \big(V, (A \setminus \{a\}) \cup \{b\}\big)$.

7.2 Restrições submodulares

Seguindo a linha dos polimatróides de Edmonds e de suas intersecções, muitos modelos em combinatória poliédrica com restrições super ou submodulares foram propostos. Tipicamente, esses modelos são centrados em sistemas totalmente duais integrais e implicam em várias relações min-max combinatórias como casos particulares. Dentre tais relações, citamos o teorema de Kőnig sobre emparelhamentos [Kön31], o do fluxo máximo e corte mínimo [FF56], o da arborescência ótima de Fulkerson [Ful74] e o de Lucchesi-Younger [LY78].

Seja S um conjunto finito e seja $\mathcal{C}$ uma família de subconjuntos de S. Podemos estender o conceito de função submodular para funções f definidas sobre $\mathcal{C}$ que satisfaçam

$$f(T) + f(U) \geq f(T \cap U) + f(T \cup U),$$

sempre que T, U, $T \cap U$ e $T \cup U$ estão em $\mathcal{C}$.

Seja (V, A) um grafo orientado, seja $\mathcal{C}$ uma família de subconjuntos de V e seja f uma função submodular definida sobre $\mathcal{C}$. Os modelos de que estamos tratando generalizam

ou são baseados em restrições da forma:

$$x(\delta^{\text{in}}(U)) - x(\delta^{\text{out}}(U)) \leq f(U), \text{ para todo } U \in \mathcal{C},$$

e

$$x(\delta^{\text{in}}(U)) \leq f(U), \text{ para todo } U \in \mathcal{C}.$$

São exemplos desses modelos os fluxos submodulares de Edmonds e Giles [EG77], os *kernel systems* de Frank [Fra79b] e os *generalized polymatroids*, também de Frank [Fra84a].

A técnica de *uncrossing*, que Lovász atribui a Neil Robertson [EG77], consiste em transformar famílias de conjuntos em famílias livres de cruzamentos. A demonstração da total dual integralidade dos sistemas acima pode ser feita através de um conhecido método que combina submodularidade e *uncrossing*. Tal método reúne idéias de Edmonds, Giles, Hoffman, Johnson, Lovász e Robertson [Edm70, EG77, Hof74, Lov76].

Em linhas gerais, dado um programa linear com um conjunto de restrições como acima, o método consiste em mostrar que é possível escolher uma solução ótima dual tal que a família de conjuntos associada às suas variáveis não-nulas é "boa" (por exemplo, livre de cruzamentos). A submodularidade é um ingrediente essencial para tanto. Feito isso, mostra-se que famílias "boas" estão associadas a submatrizes totalmente unimodulares da matriz de restrições. A total dual integralidade é obtida como conseqüência desses fatos.

Também é possível demonstrar a existência de alguns objetos combinatórios mostrando-se que certos poliedros inteiros, baseados em sistemas totalmente duais integrais que envolvem restrições submodulares, são não-vazios. Uma demonstração desse tipo para o teorema da orientação de Nash-Williams foi dada por Frank [Fra84b].

Uma resenha sobre a inter-relação dentre vários modelos baseados em restrições super ou submodulares foi escrita por Schrijver [Sch84]. Nesse mesmo trabalho, Schrijver forneceu um arcabouço que inclui os demais modelos apresentados.

Algoritmos polinomiais estão disponíveis para otimização sobre modelos poliédricos associados a funções submodulares. Uma resenha sobre algoritmos para fluxos submodulares foi escrita por Fujishige e Iwata [FI00b]. Especialmente após o desenvolvimento dos algoritmos combinatórios e fortemente polinomiais para minimização de funções submodulares, muitos avanços em algoritmos para fluxos submodulares foram alcançados [FI00a, FIM02, IMS05]. Antes disso, tais algoritmos eram baseados em subrotinas capazes de minimizar funções submodulares em tempo polinomial.

7.3 Emparelhamentos e caminhos disjuntos

Os teoremas de Hall e Menger são exemplos de resultados clássicos e fundamentais em teoria dos grafos que podem ser demonstrados com o auxílio de submodularidade.

O primeiro caracteriza a existência de emparelhamentos que saturam uma das partes de um grafo bipartido. O segundo estabelece uma relação min-max que caracteriza o

número máximo de caminhos disjuntos nos arcos entre dois vértices distintos de um grafo orientado.

Teorema de Hall

Uma **bipartição** de um conjunto U é um par $\{V, W\}$ de partes de U tais que $V \cup W = U$ e $V \cap W = \emptyset$. Uma **bipartição de um grafo** G é uma bipartição $\{V, W\}$ do conjunto de vértices de G tal que toda aresta de G tem uma ponta em V e a outra em W. Um grafo G é **bipartido** se possui uma bipartição $\{V, W\}$.

Denotamos por $G = (V, W; E)$ um grafo bipartido G com bipartição $\{V, W\}$ e conjunto de arestas E. Nesse caso, os conjuntos V e W são ditos **partes** de G.

Seja G um grafo bipartido, com bipartição $\{V, W\}$. Para cada $U \subseteq V$, denotamos por $N_G(U)$ o conjunto dos vizinhos de U em G e e definimos $b_G(U) := |N_G(U)|$. Quando o grafo G está implícito pelo contexto, denotamos simplesmente $N := N_G$ e $b := b_G$. Conforme apresentamos no capítulo 1, a função b é submodular.

Um **emparelhamento** em um grafo é um conjunto de arestas que não possuem pontas em comum. Um emparelhamento M **satura** um vértice v se alguma aresta de M incide em v. Analogamente, um emparelhamento M **satura** um subconjunto U de vértices, se M satura cada vértice de U. Dizemos que dois vértices u e v estão **emparelhados** em um emparelhamento M se a aresta uv está em M. Note que se as arestas de um grafo são um emparelhamento, então $b(\{v\}) = 1$, para todo vértice v desse grafo.

O teorema de Hall [Hal35] estabelece uma condição necessária e suficiente para que um grafo bipartido possua um emparelhamento que satura uma de suas partes. A prova a seguir, que utiliza a submodularidade da função b, pode ser encontrada no artigo [Fra93b], de Frank.

Teorema 7.1 (Teorema de Hall): *Seja $G = (V, W; E)$ um grafo bipartido. Então G possui um emparelhamento que satura V se, e somente se,*

$$b(U) \geq |U|, \tag{7.2}$$

para toda parte U de V.

Demonstração: Se um emparelhamento M em G satura V, então cada vértice de V deve estar emparelhado em M a um vértice distinto de W, de maneira que a necessidade da condição (7.2) é imediata.

Verificamos agora sua suficiência. Suponha que a condição (7.2) seja satisfeita. Uma parte U de V é dita um **conjunto justo** se $b(U) = |U|$. Começamos com o seguinte resultado auxiliar.

Lema 7.2: *Se T e U são justos, então $T \cup U$ e $T \cap U$ também são justos.*

Demonstração do lema: Sejam T e U conjuntos justos. Utilizando a submodularidade da

função b e aplicando (7.2) a $T \cap U$ e $T \cup U$, temos

$$\begin{aligned} |T| + |U| = b(T) + b(U) &\geq b(T \cap U) + b(T \cup U) \\ &\geq |T \cap U| + |T \cup U| = |T| + |U|, \end{aligned}$$

de maneira que todas as relações acima valem com igualdade. Em particular, temos que $T \cup U$ e $T \cap U$ são justos. ∎

Denote por $G' = (V, W; E')$ um subgrafo gerador minimal de G que satisfaça (7.2). Isto é, G' deve ser tal que, para cada aresta vw em E', $G' - vw$ não satisfaz (7.2) para alguma parte U de V. Note que, para cada aresta vw de G', com v em V,

existe um conjunto justo U que contém v e tal que v é o único vizinho de w em U.

Para cada aresta vw de G', denote por U_{vw} uma parte justa de V com a propriedade acima. Vamos mostrar que as arestas de G' são um emparelhamento em G que cobre V.

Suponha que não. Nesse caso, como vale (7.2), existe um vértice s em V tal que $b_{G'}(\{s\}) \geq 2$. Sejam u e v dois vizinhos de s em W. Seja $U := U_{su} \cap U_{sv}$. Pelo lema 7.2, U é um conjunto justo. Ademais, por definição, temos que s é o único vizinho de u em U_{su} e, portanto, em U. Analogamente, temos também que s é o único vizinho de v em U. Considere agora a parte $U \setminus \{s\}$ de V. Como G' satisfaz (7.2), devemos ter

$$b_{G'}(U \setminus \{s\}) \geq |U \setminus \{s\}|. \tag{7.3}$$

Por outro lado, temos que $N_{G'}(U \setminus \{s\}) \subseteq N_{G'}(U) \setminus \{u, v\}$, uma vez que s é o único vizinho de u e de v em U. Assim, utilizando também o fato de que U é um conjunto justo em G', temos

$$b_{G'}(U \setminus \{s\}) \leq b_{G'}(U) - 2 = |U| - 2 < |U \setminus \{s\}|. \tag{7.4}$$

Mas então (7.3) e (7.4) se contradizem, de onde concluímos que as arestas de G' são um emparelhamento em G que cobre V. ∎

Seja $G = (V, W; E)$ um grafo bipartido que satisfaça (7.2). A prova acima nos permite descrever o seguinte algoritmo para determinar um emparelhamento em G que cobre V.

- Para cada aresta vw de G, verifique se

$$\min_{U \subseteq V} \Big(\big| N_{G-vw}(U) \big| - |U| \Big) \geq 0. \tag{7.5}$$

  Em caso afirmativo, remova a aresta vw de G.

- As arestas do grafo resultante constituem um emparelhamento em G que cobre V.

Encontrar um emparelhamento que cubra uma das partes de um grafo bipartido ou, equivalentemente, encontrar um emparelhamento máximo em um grafo bipartido, é um

problema muito bem resolvido. Isto é, existem algoritmos polinomiais e satisfatórios na prática que resolvem tal tarefa (cf. [Sch03]). Entretanto, é interessante notar que, assim como em muitos outros problemas de teoria dos grafos e otimização combinatória, a minimização de uma função submodular também é um ingrediente envolvido na resolução desse problema.

Teorema de Menger

Seja D um grafo orientado e sejam s, t dois vértices distintos de D. Um st-**caminho** em D é um caminho em D com origem s e destino t. Um st-**separador** em D é um subconjunto de vértices que contém s e não contém t.

Definimos agora uma operação que aparece em muitos problemas relativos a conexidade em grafos e que também é utilizada na prova que apresentamos para o teorema de Menger. Seja D um grafo orientado. A operação **splitting off** sobre um par de arcos uv e vw de D é a construção de um grafo orientado D' a partir da remoção dos arcos uv e vw e da adição de um novo arco uw a D. Isto é $D' := (D - uv - vw) + uw$.

Passemos agora ao teorema de Menger [Men27]. Sua versão mais conhecida, no que se refere a grafos orientados e caminhos disjuntos nos arcos, é a seguinte.

Teorema 7.3 (Teorema de Menger): *Seja $D = (V, A)$ um grafo orientado e sejam s e t dois vértices distintos de D. Em D, o número máximo de st-caminhos disjuntos nos arcos é igual ao número mínimo de arcos que precisam ser removidos para que todos os st-caminhos sejam destruídos.*

Entretanto, a prova que apresentamos, devida a Frank [Fra93b], está associada à seguinte versão equivalente.

Teorema 7.4: *Seja $D = (V, A)$ um grafo orientado e sejam s e t dois vértices distintos de D. Existem k st-caminhos disjuntos nos arcos em D se, e somente se,*

$$d^{\text{out}}(U) \geq k, \tag{7.6}$$

para todo st-separador U de D.

Demonstração: Sendo a necessidade da condição (7.6) imediata, vamos verificar sua suficiência. Para tanto, faremos indução no número de arcos do grafo D.

Um st-separador U de D é **justo** se $d^{\text{out}}(U) = k$.

Lema 7.5: *Se T e U são st-separadores justos, então $T \cup U$ e $T \cap U$ também são st-separadores justos.*

Demonstração do lema: Sejam T e U st-separadores justos de D. É claro que $T \cap U$ e $T \cup U$ também são st-separadores. Podemos, então, utilizar a submodularidade da função

d^{out}, assim como a validade de (7.6), e observar que

$$k + k = d^{\text{out}}(T) + d^{\text{out}}(U) \geq d^{\text{out}}(T \cap U) + d^{\text{out}}(T \cup U) \geq k + k.$$

Concluímos, assim, que todas as relações acima valem com igualdade e que, em particular, $T \cap U$ e $T \cup U$ são st-separadores justos. ∎

Suponha que D satisfaça (7.6). Podemos supor que todo arco a de D sai de um st-separador justo. Do contrário, o arco a pode ser descartado sem que a condição (7.6) seja violada.

Se todos os arcos de D com ponta inicial em s têm a ponta final em t, então (7.6) implica que $d^{\text{out}}(s) \geq k$ e, portanto, que existem k st-caminhos disjuntos nos arcos em D. Essa é a base de nossa indução.

Suponha agora que exista pelo menos um arco $a' = su$, com $u \neq t$. Seja T a união de todos os st-separadores justos dos quais o arco a' sai. Isto é, T é a união de todos os st-separadores justos que não contêm u. Pelo lema 7.5, T é um st-separador justo.

Notemos, então, que deve existir um arco $b' = uv$, com $v \notin T$, $v \neq u$. De fato, considere o st-separador $T' := T \cup \{u\}$. Se não existe nenhum arco com ponta inicial em u e ponta final em $V \setminus T'$, então todos os arcos que saem de T' também saem de T. Além disso, sabemos que o arco $a' = su$ sai de T e não sai de T'. Portanto, como T é justo, temos

$$d^{\text{out}}(T') \leq d^{\text{out}}(T) - 1 = k - 1.$$

Mas isso não pode ocorrer pois vale (7.6).

Considere, então, o grafo D' resultante do splitting off dos arcos a' e b'. Isto é $D' := (D - su - uv) + sv$. Afirmamos que a condição (7.6) é válida para D'. De fato, notemos que se (7.6) não é satisfeita por D', então D deveria possuir um st-separador justo contendo v e não contendo u. Entretanto, T é a união de todos os st-separadores justos de D que não contêm u e T não contém v. Portanto, D' satisfaz (7.6).

Assim, podemos aplicar a hipótese de indução sobre D' e concluir que tal grafo possui k st-caminhos disjuntos nos arcos. Note que, se algum desses st-caminhos contém o arco sv, então tal arco pode ser substituído pelo caminho composto pelos arcos su e uv em D. Conseqüentemente, D também possui k st-caminhos disjuntos nos arcos, o que conclui a prova. ∎

7.4 Ramificações e arborescências

Seja $D = (V, A)$ um grafo orientado. Uma **ramificação** é um subconjunto B de A tal que B não contém circuitos (orientados ou não) e tal que, para todo vértice v em V, existe no máximo um arco em B entrando em v. Uma **raiz** de B é um vértice que não é ponta

final de nenhum arco de B. Segue da definição que cada componente de (V, B) contém uma única raiz. Dizemos que B é uma R**-ramificação** se B é uma ramificação com conjunto de raízes $R \subseteq V$.

Uma ramificação B é uma **arborescência** se o grafo orientado (V, B) é fracamente conexo ou, equivalentemente, se (V, B) é uma árvore enraizada. Logo, cada arborescência B tem uma única raiz r. Dizemos, nesse caso, que B é uma r**-arborescência**. Uma r-arborescência pode ser caracterizada também como uma árvore geradora orientada tal que todo vértice v em V é acessível a partir de r em B.

Se D é um grafo orientado e r é um vértice de D, um r**-corte** em D é um conjunto de arcos da forma $\delta^{\text{in}}(U)$, em que U é um subconjunto não-vazio de $V \setminus \{r\}$.

Em 1973, Edmonds [Edm73] apresentou um teorema muito poderoso que caracteriza a existência de ramificações disjuntas em grafos orientados. Pouco tempo depois, Lovász [Lov76] propôs uma prova alternativa para o resultado, baseada em técnicas de submodularidade. Essa é a prova que apresentamos a seguir.

Teorema 7.6 (Empacotamento de ramificações): *Seja $D = (V, A)$ um grafo orientado e sejam $R_1, \ldots, R_k$ subconjuntos de V. Existem ramificações disjuntas $B_1, \ldots, B_k$ em D tais que R_i é o conjunto das raízes de B_i, para todo $i = 1, \ldots, k$, se, e somente se,*

$$d^{\text{in}}(U) \geq \big|\{i\colon R_i \cap U = \emptyset\}\big|, \tag{7.7}$$

para toda parte não-vazia U de V.

Demonstração: Suponha que as ramificações $B_1, \ldots, B_k$ nas condições desejadas existam. Se U é uma parte de V disjunta de R_i, então a ramificação B_i deve ter pelo menos um arco entrando em U. Como os B_i são todos disjuntos, segue a necessidade da condição (7.7).

Passemos à prova de sua suficiência. Defina

$$\gamma(U) := \{i\colon R_i \cap U = \emptyset\} \quad \text{e} \quad g(U) := \big|\gamma(U)\big|.$$

para todo $U \subseteq V$, com $U \neq \emptyset$.

Começamos provando os seguintes resultados auxiliares.

Lema 7.7: *A função g é supermodular. Isto é, dados $U, W \subseteq V$, vale que*

$$g(U) + g(W) \leq g(U \cap W) + g(U \cup W). \tag{7.8}$$

Além disso, vale a igualdade em (7.8) se, e somente se,

$$\gamma(U) \cup \gamma(W) = \gamma(U \cap W).$$

Demonstração do lema: Sejam $U, W \subseteq V$. Pelas definições de γ e g, bem como por pro-

priedades básicas da soma de cardinalidades de conjuntos, temos

$$\begin{aligned} g(U) + g(W) &= \big|\gamma(U)\big| + \big|\gamma(W)\big| \\ &= \big|\gamma(U) \cap \gamma(W)\big| + \big|\gamma(U) \cup \gamma(W)\big| \\ &= \big|\gamma(U \cup W)\big| + \big|\gamma(U) \cup \gamma(W)\big| \\ &= g(U \cup W) + \big|\gamma(U) \cup \gamma(W)\big|. \end{aligned} \tag{7.9}$$

Também é claro que $\gamma(U) \subseteq \gamma(U \cap W)$, assim como que $\gamma(W) \subseteq \gamma(U \cap W)$, de maneira que $\gamma(U) \cup \gamma(W) \subseteq \gamma(U \cap W)$. Mas, então, temos

$$\big|\gamma(U) \cup \gamma(W)\big| \leq \big|\gamma(U \cap W)\big| = g(U \cap W) \tag{7.10}$$

e segue de (7.9) que g é supermodular.

Como concluímos que $g(U) + g(W) = g(U \cup W) + \big|\gamma(U) \cup \gamma(W)\big|$ e que $\gamma(U) \cup \gamma(W) \subseteq \gamma(U \cap W)$, segue direto que vale a igualdade em (7.8) se, e somente se, $\gamma(U) \cup \gamma(W) = \gamma(U \cap W)$. ∎

Um conjunto $U \subseteq V$ é dito **perigoso** se $d^{\text{in}}(U) = g(U)$. Note que V é sempre um conjunto perigoso.

Lema 7.8: *Se $U, W \subseteq V$ são conjuntos perigosos intersectantes, então $U \cap W$ também é perigoso. Nesse caso, se $i \notin \gamma(U)$ e $i \notin \gamma(W)$, para algum $1 \leq i \leq k$, então $i \notin \gamma(U \cap W)$.*

Demonstração do lema: Primeiramente, podemos utilizar a submodularidade da função d^{in}_D para obtermos a seguinte relação:

$$d^{\text{in}}_D(U \cap W) \leq d^{\text{in}}_D(U) + d^{\text{in}}_D(W) - d^{\text{in}}_D(U \cup W).$$

Como U e W são conjuntos perigosos, e como vale (7.7) para $U \cap W$, então, utilizando a relação acima, temos

$$d^{\text{in}}_D(U \cap W) \leq g(U) + g(W) - g(U \cup W).$$

Por fim, pela supermodularidade de g, segue que

$$d^{\text{in}}_D(U \cap W) \leq g(U \cap W).$$

Entretanto, como $U \cap W \neq \emptyset$, então $d^{\text{in}}_D(U \cap W) \geq g(U \cap W)$ por hipótese, de onde concluímos

$$d^{\text{in}}_D(U \cap W) = g(U \cap W).$$

Ou seja, $U \cap W$ é perigoso.

Mais ainda, temos também que todas as relações acima valem com igualdade. Em particular,

$$g(U \cap W) + g(U \cap W) = g(U) + g(W)$$

e segue do lema 7.7 que

$$\gamma(U) \cup \gamma(W) = \gamma(U \cap W). \tag{7.11}$$

Sendo assim, não pode existir um um índice i tal que $i \notin \gamma(U)$ e $i \notin \gamma(W)$ e $i \in \gamma(U \cap W)$. De fato, se isso ocorresse, o índice i estaria em $\gamma(U \cap W) \setminus \big(\gamma(U) \cap \gamma(W)\big)$ e não valeria (7.11). Logo, se $i \notin \gamma(U)$ e $i \notin \gamma(W)$, então $i \notin \gamma(U \cap W)$. ∎

Voltemos à prova da suficiência de (7.7). Faremos indução em $\sum_{i=1}^{k} |V \setminus R_i|$.

Se $R_i = V$, para todo i, então $B_1 = \ldots = B_k = \emptyset$ são ramificações nas condições desejadas e o resultado vale.

Vamos supor, então, sem perda de generalidade, que $R_1 \neq V$. Seja $U' \subseteq V$ um conjunto perigoso minimal tal que $U' \cap R_1 \neq \emptyset$ e $U' \setminus R_1 \neq \emptyset$. Isto é, U' deve ser tal que não exista $U \subset U'$ perigoso e nas condições acima. Como V é perigoso, um tal conjunto U' sempre existe.

Por (7.7) e observando que $1 \in \gamma(U' \setminus R_1)$, mas que $1 \notin \gamma(U')$, temos

$$d^{\text{in}}(U' \setminus R_1) \geq g(U' \setminus R_1) \geq g(U') + 1 > g(U') = d^{\text{in}}(U'), \tag{7.12}$$

de modo que $\delta^{\text{in}}(U' \setminus R_1) \setminus \delta^{\text{in}}(U')$ é não-vazio. Concluímos, assim, que existe um arco $a = uv$ com $u \in R_1 \cap U'$ e $v \in U' \setminus R_1$. Considere um arco $a = uv$ desse tipo.

Tome $A' := A \setminus \{a\}$, $R'_1 = R_1 \cup \{v\}$ e $R'_i = R_i$, para todo $i \neq 1$. Afirmamos que a condição (7.7) continua válida para o grafo $D' := (V, A')$ e os subconjuntos $R'_1, \ldots, R'_k$ de V.

Suponha o contrário. Defina, para cada $U \subseteq V$,

$$\gamma'(U) := \big\{i \colon R'_i \cap U = \emptyset\big\} \quad \text{e} \quad g'(U) := \big|\gamma'(U)\big|. \tag{7.13}$$

Como estamos supondo que (7.7) não vale para D' e os conjuntos de raízes $R'_1, \ldots, R'_k$, então existe $W \subseteq V$ tal que $g'(W) > d^{\text{in}}_{D'}(W)$. Como $R'_i = R_i$, para todo $i \neq 1$, e $R_1 \subset R'_1$, então $g(W) \geq g'(W) \geq g(W) - 1$. Como D' foi obtido a partir da remoção de um único arco de D, então $d^{\text{in}}_D(W) \geq d^{\text{in}}_{D'}(W) \geq d^{\text{in}}_D(W) - 1$. Temos também que $g(W) \leq d^{\text{in}}_D(W)$. Concluímos, assim, que $g'(W) = g(W) = d^{\text{in}}_D(W)$ e que $d^{\text{in}}_{D'}(W) = d^{\text{in}}_D(W) - 1$. Ou seja, o conjunto W é um conjunto perigoso e $a = uv \in \delta^{\text{in}}_D(W)$.

Além disso, como $g'(W) = g(W)$, então $R_1 \cap W \neq \emptyset$. De fato, note que W e R'_1 não são disjuntos, pois $v \in R'_1 \cap W$. Logo, se $R_1 \cap W = \emptyset$, então deveríamos ter $g'(W) = g(W) - 1$,

o que não ocorre.

Agora, como W e U' são perigosos e $v \in W \cap U'$, então, pelo lema 7.8, $W \cap U'$ é perigoso. Ademais, temos que $R_1 \cap (W \cap U') \neq \emptyset$. Note que como $R_1 \cap W \neq \emptyset$ e $R_1 \cap U' \neq \emptyset$, então $1 \notin \gamma(W)$ e $1 \notin \gamma(U')$, de modo que, também pelo lema 7.8, temos $1 \notin \gamma(W \cap U')$. Também é claro que $(W \cap U') \setminus R_1$ é não-vazio, pois $v \in W \cap U'$ e $v \notin R_1$, e que $(W \cap U') \subset U'$, uma vez que $u \notin W \cap U'$ e $u \in U'$.

Mas então a existência do conjunto perigoso $W \cap U'$ contradiz a minimalidade do conjunto U' inicialmente escolhido. Segue que a condição (7.7) é válida para D' e $R'_1, \ldots, R'_k$.

Sendo assim, pela hipótese de indução, existem ramificações disjuntas $B'_1, \ldots, B'_k$ em D' tais que R'_i é o conjunto das raízes de B'_i, para todo i.

Pela construção de D', é claro que $B_1 := B'_1 \cup \{a\}$ é um ramificação cujo conjunto de raízes é R_1. Assim, tomando $B_1 := B'_1 \cup \{a\}$ e $B_i := B'_i$, para todo $i \neq 1$, temos que $\{B_1, \ldots, B_k\}$ é uma coleção de ramificações disjuntas tais que R_i é o conjunto das raízes de B_i, para todo i. Isso conclui a prova. ∎

Como corolário do teorema anterior, segue a seguinte caracterização da existência de arborescências disjuntas com raízes fixadas:

Teorema 7.9: *Seja $D = (V, A)$ um grafo orientado e sejam $r_1, \ldots, r_k$ vértices em V. Existem k arborescências disjuntas $B_1, \ldots, B_k$ em D tais que B_i tem raiz r_i, para todo $i = 1, \ldots, k$, se, e somente se,*

$$d^{\text{in}}(U) \geq \big|\{i : r_i \notin U\}\big|.$$

para toda parte não-vazia U de V.

Demonstração: Basta tomar $R_i := \{r_i\}$, para cada $1 \leq i \leq k$, e aplicar o teorema 7.6. ∎

Um outro corolário importante do teorema 7.6 é a seguinte relação min-max, apresentada também por Edmonds [Edm70], que estabelece o tamanho máximo de um empacotamento de r-arborescências em um grafo orientado.

Teorema 7.10 (Empacotamento de r-arborescências): *Seja $D = (V, A)$ um grafo orientado e seja r um vértice em V. O número máximo de r-arborescências disjuntas em D é igual ao tamanho mínimo de um r-corte em D*

Demonstração: Tome $k := \min\{d^{\text{in}}(U) : \emptyset \subset U \subseteq V \setminus \{r\}\}$, $R_i := \{r\}$, para todo $1 \leq i \leq k$, e aplique o teorema 7.6. ∎

A relação min-max de Edmonds sobre r-arborescências foi posteriormente estendida por Frank [Fra81], que obteve o seguinte resultado, que caracteriza o tamanho máximo de um empacotamento de arborescências sem raízes fixadas.

Teorema 7.11 (Empacotamento de arborescências): *Seja $D = (V, A)$ um grafo*

orientado. Existem k arborescências disjuntas em D se, e somente se,

$$\sum_{i=1}^{t} d^{\text{in}}(U_i) \geq k(t-1), \tag{7.14}$$

para toda subpartição $\mathcal{F} = \{U_1, \ldots, U_t\}$ de V.

Demonstração: Vamos começar verificando a necessidade da condição (7.14). Suponha que $B_1, \ldots, B_k$ sejam k arborescências disjuntas em D. Para cada i, temos que existem arcos de B_i entrando em pelo menos $t-1$ membros de $\mathcal{F}$. Note que no máximo um membro de $\mathcal{F}$ contém a raiz de B_i. Como as k arborescências são disjuntas, segue (7.14).

Suponha agora que a condição (7.14) valha. Vamos mostrar que D possui k arborescências disjuntas.

Adicione um novo vértice r a D, além de k arcos paralelos de r a v, para todo vértice v em V. Tais arcos serão ditos **artificiais**. Claramente,

$$\text{existem } k \text{ caminhos disjuntos nos arcos de } r \text{ a cada vértice } v \in V. \tag{7.15}$$

Feito isso, remova o maior número possível de arcos artificiais, de forma que a propriedade (7.15) continue valendo. Denote por D' o grafo resultante dessas operações.

Pelo teorema 7.4 de Menger, a validade de (7.15) para D' é equivalente a

$$d^{\text{out}}_{D'}(U \cup \{r\}) \geq k, \text{ para todo } \emptyset \subseteq U \subset V. \tag{7.16}$$

Mas (7.16) é justamente o mesmo que

$$d^{\text{in}}_{D'}(U) \geq k, \text{ para todo } \emptyset \subset U \subseteq V. \tag{7.17}$$

Diremos que um subconjunto U de V é **crítico** se satisfaz (7.17) com igualdade. Considere os seguintes resultados auxiliares.

Lema 7.12: *Se T e U são conjuntos críticos intersectantes, então $T \cup U$ e $T \cap U$ também são conjuntos críticos.*

Demonstração do lema: Sejam T e U conjuntos críticos. Utilizando a submodularidade de $d^{\text{in}}_{D'}$, temos

$$k + k = d^{\text{in}}_{D'}(T) + d^{\text{in}}_{D'}(U) \geq d^{\text{in}}_{D'}(T \cap U) + d^{\text{in}}_{D'}(T \cup U) \geq k + k.$$

Para a última desigualdade, observe que, como T e U são intersectantes, então $T \cap U$ e $T \cup U$ são não-vazios e, portanto, devem satisfazer a equação (7.17).

Mas então todas as relações acima valem com igualdade e, conseqüentemente, $T \cap U$ e $T \cup U$ são críticos. ∎

Lema 7.13: *$d^{\text{in}}_{D'}(V) = k$, isto é, V é crítico.*

Demonstração do lema: Suponha por contradição que pelo menos $k+1$ arcos artificiais $a_1, \ldots, a_{k+1}$, entrem em V. Como D' é minimal, no sentido de que a remoção de qualquer um de seus arcos destrói a propriedade (7.17), temos que cada arco a_i entra em um conjunto crítico em V.

Seja $\mathcal{F} = \{U_1, \ldots, U_t\}$ a família de todos os conjuntos críticos maximais de V, isto é, que não estejam propriamente contidos em nenhum outro conjunto crítico. Como sabemos, pelo lema 7.12, que a união de conjuntos críticos é também um conjunto crítico, então todos os elementos de $\mathcal{F}$ são disjuntos. Ou seja, $\mathcal{F}$ é uma subpartição de V composta por conjuntos críticos. Assim,

$$\sum_{i=1}^{t} d^{\text{in}}_{D'}(U_i) = kt.$$

Notemos, então, que cada arco artificial a_i entra em um elemento de $\mathcal{F}$. Logo,

$$kt = \sum_{i=1}^{t} d^{\text{in}}_{D'}(U_i) \geq (k+1) + \sum_{i=1}^{t} d^{\text{in}}_{D}(U_i).$$

Mas, então,

$$\sum_{i=1}^{t} d^{\text{in}}_{D}(U_i) \leq k(t-1) - 1,$$

o que é uma contradição com a condição (7.14). Segue que $d^{\text{in}}_{D'}(V) = k$. ∎

Pela validade de (7.17), temos que todo r-corte em D' possui pelo menos k arcos. Logo, podemos utilizar o teorema 7.10 de Edmonds e concluir que D' possui k r-arborescências disjuntas. Ademais, como $d^{\text{in}}_{D'}(V) = k$, então cada uma dessas arborescências contém exatamente um arco artificial. Ou seja, a restrição das k r-arborescências disjuntas de D' aos arcos de D fornece k arborescências disjuntas em D. Mas isso é justamente o que queríamos. ∎

Se A é o conjunto dos arcos de um grafo orientado D, vamos denotar por $A(U)$ o conjunto dos arcos de D com as duas pontas em U.

A partir do teorema 7.10 de Edmonds sobre o número máximo de r-arborescências disjuntas, Frank [Fra79a] obteve o seguinte resultado.

Teorema 7.14: *O conjunto de arcos de um grafo orientado $D = (V, A)$ pode ser*

coberto por k ramificações se, e somente se,

$$d^{\text{in}}(v) \leq k, \text{ para todo } v \in V \text{ e} \tag{7.18a}$$
$$|A(U)| \leq k(|U| - 1), \text{ para todo } \emptyset \subset U \subseteq V. \tag{7.18b}$$

Demonstração: A necessidade da condição (7.18) segue facilmente da definição de ramificação. Para a suficiência, considere o grafo orientado D', construído a partir de D da seguinte maneira: adicione um vértice novo r a D, além de $k - d^{\text{in}}(v)$ arcos paralelos de r a v, para cada vértice $v \in V$. Por construção, temos que $d^{\text{in}}_{D'}(v) = k$, para todo vértice $v \in V$ e que D' possui exatamente $k|V|$ arcos.

Observe agora que entram pelo menos k arcos em cada subconjunto não-vazio U de V em D'. De fato, temos que exatamente $k|U|$ arcos têm sua ponta final em U e que, por (7.18b), no máximo $k(|U| - 1)$ arcos têm as duas pontas em U. Ou seja, entram pelo menos k arcos em U. Disso concluímos que todo r-corte em D' tem cardinalidade pelo menos k. Assim, pelo teorema 7.10 de Edmonds, D' possui k r-arborescências disjuntas.

Como cada arborescência tem exatamente $|V(D')| - 1 = |V|$ arcos e D' possui exatamente $k|V|$ arcos, então as k r-arborescências fornecidas pelo teorema de Edmonds são uma partição do conjunto de arcos de D'. Além disso, temos que a restrição das k r-arborescências disjuntas aos arcos de D é uma coleção de k ramificações disjuntas em D. Concluímos, assim, que existe uma partição dos arcos de D em k ramificações, como queríamos. ∎

7.5 Árvores geradoras

Tratemos agora do problema do empacotamento de árvores geradoras em grafos não-orientados. O primeiro resultado da área é devido a Nash-Williams [NW61] e Tutte [Tut61], que estabeleceram condições necessárias e suficientes para a existência de k árvores geradoras disjuntas em grafos não-orientados.

A submodularidade também possibilita o desenvolvimento de uma prova alternativa para esse famoso resultado. A prova que apresentamos a seguir é devida a Frank [Fra93b].

Teorema 7.15 (Empacotamento de árvores geradoras): *Seja $G = (V, E)$ um grafo. Existem k árvores geradoras disjuntas em G se, e somente se,*

$$e_{\mathcal{F}} := \sum_{i=1}^{t} \frac{d_G(V_i)}{2} \geq k(t-1), \tag{7.19}$$

para toda partição $\mathcal{F} = \{V_1, \ldots, V_t\}$ de V.

Demonstração: Seja $\mathcal{F} = \{V_1, \ldots, V_t\}$ uma partição de V. É claro que cada árvore geradora de G deve possuir pelo menos $t - 1$ arestas ligando as t partes de V em $\mathcal{F}$. Logo, se G possui k árvores geradoras disjuntas, então o número de arestas com pontas em partes

diferentes de $\mathcal{F}$ deve ser pelo menos $k(t-1)$. Disso concluímos a necessidade da condição (7.19).

Para verificar sua suficiência, começamos mostrando o teorema abaixo.

Teorema 7.16 (Frank [Fra81]): *Seja $G = (V, E)$ um grafo e seja s um vértice de G. Existe uma orientação D de G para a qual*

$$d_D^{\text{in}}(X) \geq k, \tag{7.20}$$

para todo $\emptyset \subset X \subseteq V \setminus \{s\}$, se, e somente se, a condição (7.19) é verificada por G.

Demonstração do teorema 7.16: Seja $\mathcal{F} = \{V_1, \ldots, V_t\}$ uma partição de V. Se D é uma orientação de G para a qual $d_D^{\text{in}}(X) \geq k$, para todo $\emptyset \subset X \subseteq V \setminus \{s\}$, então, em particular, $d_D^{\text{in}}(V_i) \geq k$, para toda parte V_i que não contém s. Portanto,

$$\sum_{i=1}^{t} d_D^{\text{in}}(V_i) = \sum_{i=1}^{t} \frac{d_G(V_i)}{2} \geq k(t-1).$$

Suponha agora que a condição (7.19) seja verificada por G. Suponha também, por contradição, que G não possua a orientação desejada.

Adicione o menor número possível de arestas da forma sv a G, com $v \in V$, de forma que a orientação D desejada possa ser obtida. Como usual, as arestas adicionadas serão chamadas **artificiais**. Como a reorientação dos arcos com ponta final em s não afeta a condição (7.20), então podemos supor $d_D^{\text{in}}(s) = 0$.

Diremos que um conjunto não-vazio $X \subseteq V \setminus \{s\}$ é **crítico** se $d_D^{\text{in}}(X) = k$.

Seja $a = st$ um arco de D que é a orientação de uma aresta artificial. Seja T o conjunto dos vértices acessíveis a partir de t em D.

Lema 7.17: *Se Z é um conjunto crítico e $Z \cap T \neq \emptyset$, então $Z \subseteq T$.*

Demonstração do lema: Suponha por contradição que $Z \not\subseteq T$. Tome $Y := V \setminus T$. Pela definição de T, temos $d_D^{\text{in}}(Y) = 0$. Ademais, como Z é crítico, então $d_D^{\text{in}}(Z) = k$. Logo, utilizando a igualdade (7.1), que versa sobre a função d_D^{in}, temos

$$k = d_D^{\text{in}}(Z) + d_D^{\text{in}}(Y) = d_D^{\text{in}}(Z \cup Y) + d_D^{\text{in}}(Z \cap Y) + d(Y, Z) \geq k. \tag{7.21}$$

Para a última desigualdade, observe que, como a orientação D satisfaz (7.20) e $s \notin Z \cap Y$, então $d^{\text{in}}(Z \cap Y) \geq k$. Ademais, temos $d^{\text{in}}(Z \cup Y) \geq 0$ e $d(Y, Z) \geq 0$. Assim, de (7.21), concluímos que $d^{\text{in}}(Z \cup Y) = d(Y, Z) = 0$.

Agora, se $d^{\text{in}}(Z \cup Y) = 0$, então não entram arcos em Z a partir de T. Mas, como $Z \cap T \neq \emptyset$, então devemos ter $t \in Z$, pela definição de T. Por outro lado, se $d(Y, Z) = 0$,

então não existem arcos entre $Y \setminus Z$ e $Z \setminus Y = Z \cap T$. Entretanto, sabemos que $s \in Y \setminus Z$. Note que $s \notin Z$ por definição e que $s \in Y$, uma vez que $d^{\text{in}}(s) = 0$. Além disso, sabemos que existe o arco $a = st$. Mas então a é um arco de $Y \setminus Z$ a $Z \cap T$, o que é uma contradição. Tal contradição nos permite concluir que $Z \subseteq T$, como queríamos. ∎

Dada a afirmação do lema acima, existem dois casos a serem considerados.

Caso 1: Existe um vértice u em T que não está em nenhum conjunto crítico. Considere, então, um caminho P de t a u em D. Reoriente os arcos de P no sentido de u a t e denote por D' a nova orientação de G. Afirmamos que, feito isso, o arco st pode ser descartado, sem que a condição (7.20) deixe de ser válida para D'. Mas isso é uma contradição com a minimalidade do número de arestas artificiais adicionadas a G para que a orientação D pudesse existir.

Caso 2: Todos os vértices em T estão em conjuntos críticos. Pelo lema 7.17, todos os conjuntos críticos que contêm elementos de T estão totalmente contidos em T. Assim, se considerarmos a família $\mathcal{F}' = \{V_1, \ldots, V_{t-1}\}$ de todos os conjuntos críticos maximais contidos em T, teremos que $\mathcal{F}'$ é uma partição de T. Tal observação está baseada também no lema 7.12, que garante que a união de conjuntos críticos intersectantes também é um conjunto crítico. Podemos tomar, então, $V_t = V \setminus T$ e definir a partição $\mathcal{F} := \mathcal{F}' \cup \{V_t\}$ de V.

Temos assim que $d^{\text{in}}_D(V_i) = k$, para $1 \leq i \leq t-1$, pois tais conjuntos são críticos. Ademais, temos que $d^{\text{in}}_D(V_t) = d^{\text{in}}(V \setminus T) = 0$, pela definição de T. Logo,

$$\sum_{i=1}^{t} d^{\text{in}}_D(V_i) = k(t-1) \geq \sum_{i=1}^{t} \frac{d_G(V_i)}{2} + 1,$$

uma vez que existe pelo menos a aresta artificial que deu origem ao arco $a = st$. Ou seja,

$$\sum_{i=1}^{t} \frac{d_G(V_i)}{2} < k(t-1),$$

o que também não pode ocorrer.

Portanto, nenhuma aresta artificial precisa ser adicionada para que G possua a orientação desejada. ∎

Podemos agora concluir a prova do teorema 7.15 de Tutte e Nash-Williams da seguinte maneira. Primeiro tomamos a orientação D do grafo G fornecida pelo teorema 7.16. Feito isso, o teorema 7.10 de Edmonds nos garante a existência de k arborescências geradoras disjuntas com raiz s em D. Como existe uma bijeção entre o conjunto de arestas de G e o conjunto de arcos de D, segue a existência de k árvores geradoras disjuntas em G. ∎

7.6 Aumento de conexidade

Um grafo orientado $D = (V, A)$ é **k-arco-conexo** se $d^{in}(U) \geq k$, para todo subconjunto próprio U de V. Pelo teorema 7.4 de Menger, isso é equivalente a dizer que existem k caminhos disjuntos nos arcos entre quaisquer dois vértices de D.

O aumento de conexidade em grafos é um problema para o qual técnicas envolvendo submodularidade têm se mostrado bastante eficazes. Em uma de suas versões, deseja-se tornar um grafo orientado k-arco-conexo através da adição do menor número possível de arcos. No final desta seção, apresentamos um resultado, devido a Frank [Fra92], que estabelece esse número mínimo de arcos para a realização de tal tarefa. Apresentamos também uma caracterização de grafos k-arco-conexos, de Mader [Mad74, Mad82]

Começamos com alguns resultados auxiliares relacionados à conexidade.

Teorema 7.18 (Mader [Mad82]): *Seja $D = (V' + s, A)$ um grafo orientado. Suponha que $d^{in}(s) = d^{out}(s)$ e que*

$$\text{existem } k \text{ caminhos disjuntos nos arcos entre quaisquer vértices } x \text{ e } y, \text{ com } x, y \in V' \text{ e } x \neq y. \tag{7.22}$$

Então, para todo arco st, existe um arco vs tal que a operação splitting off (seção 7.3) pode ser realizada sobre vs e st sem que a propriedade (7.22) seja destruída.

Demonstração: Defina $V := V' + s$. Vamos denotar por $d(X, Y)$ o número de arcos entre $X \setminus Y$ e $Y \setminus X$ em D e por $\bar{d}(X, Y)$ o número de arcos entre $X \cap Y$ e $V \setminus (X \cup Y)$, para quaisquer $X, Y \subseteq V$. Nos dois casos acima, consideramos os arcos nos dois sentidos na contagem.

Começamos observando que, para qualquer grafo orientado (V, A), se X e Y são subconjuntos de vértices tais que $d^{in}(X \cap Y) = d^{out}(X \cap Y)$, então

$$d^{out}(X) + d^{out}(Y) = d^{out}(X \setminus Y) + d^{out}(Y \setminus X) + \bar{d}(X, Y). \tag{7.23}$$

Um argumento simples de contagem é suficiente para verificar esse fato.

Notamos também que, pelo teorema 7.4 de Menger, a propriedade (7.22) é equivalente a

$$d^{in}(X) \geq k \quad \text{e} \quad d^{out}(X) \geq k, \tag{7.24}$$

para todo subconjunto próprio $\emptyset \neq X \subset V$.

Para provar o teorema 7.18, vamos enunciar alguns lemas sobre D que utilizam as relações (7.23) e (7.24).

Lema 7.19: *Se X e Y são subconjuntos intersectantes de V tais que $X \cap Y = \{s\}$ e $d^{out}(X) = d^{out}(Y) = k$, então $d^{out}(X \setminus Y) = d^{out}(Y \setminus X) = k$ e $\bar{d}(X, Y) = 0$.*

Demonstração: [Prova do lema.] Como $X \cap Y = \{s\}$ e $d^{\text{in}}(s) = d^{\text{out}}(s)$, podemos utilizar (7.23) e concluir que

$$\begin{aligned} k + k &= d^{\text{out}}(X) + d^{\text{out}}(Y) \\ &= d^{\text{out}}(X \setminus Y) + d^{\text{out}}(Y \setminus X) + \bar{d}(X, Y) \\ &\geq k + k + \bar{d}(X, Y). \end{aligned} \tag{7.25}$$

Para a última passagem acima, observamos que $X \setminus Y \neq \emptyset \neq Y \setminus X$, uma vez que X e Y são intersectantes, e que $s \notin X \setminus Y,\ Y \setminus X$, de maneira que (7.24) se aplica a esses conjuntos. Mas, então, $d^{\text{out}}(X \setminus Y) = d^{\text{out}}(Y \setminus X) = k$ e $\bar{d}(X, Y) = 0$, como queríamos. ∎

Lema 7.20: *Sejam $A, B \subseteq V$ tais que*

$$d^{\text{in}}(A) = d^{\text{in}}(B) = k \leq \min\{d^{\text{in}}(A \cap B), d^{\text{in}}(A \cup B)\}.$$

Então $d^{\text{in}}(A \cap B) = d^{\text{in}}(A \cup B) = k$ e $d(A, B) = 0$.

Demonstração: [Prova do lema.] Utilizando a relação (7.1), temos

$$\begin{aligned} k + k &= d^{\text{in}}(A) + d^{\text{in}}(B) \\ &= d^{\text{in}}(A \cup B) + d^{\text{in}}(A \cap B) + d(A, B) \\ &\geq k + k + d(A, B), \end{aligned} \tag{7.26}$$

de onde segue o resultado. ∎

Um subconjunto $\emptyset \subset X \subset V'$ é dito **e-crítico** se $d^{\text{in}}(X) = k$. Analogamente, X é dito **s-crítico** se $d^{\text{out}}(X) = k$. Ademais, X é **crítico** se for e-crítico ou s-crítico.

Lema 7.21: *Sejam A e B conjuntos críticos intersectantes. Então (i) $A \cup B$ é crítico ou (ii) $B \setminus A$ é crítico e $\bar{d}(A, B) = 0$.*

Demonstração: [Prova do lema.] Se A e B são e-críticos e $A \cup B \neq V'$, então vale (i) pelo lema 7.20. Note que, nesse caso, como A e B são intersectantes, então (7.24) vale para $A \cap B$ e para $A \cup B$, de modo que as hipóteses do lema 7.20 são claramente satisfeitas.

Por outro lado, se $A \cup B = V'$, defina $X := (V' + s) \setminus A = (B \setminus A) + s$ e $Y := (V' + s) \setminus B = (A \setminus B) + s$. Como $d^{\text{in}}(A) = k$, então $d^{\text{out}}(X) = d^{\text{out}}((V' + s) \setminus A) = k$. Como $d^{\text{in}}(B) = k$, então $d^{\text{out}}(Y) = d^{\text{out}}((V' + s) \setminus B) = k$. Além disso, é claro que $X \cap Y = \{s\}$ e que $X \setminus Y \neq \emptyset \neq Y \setminus X$, pois A e B são intersectantes. Logo, o lema 7.19 se aplica e, portanto, vale (ii).

O caso em que A e B são ambos s-críticos é completamente análogo.

Suponha agora que um dos conjuntos seja e-crítico e o outro s-crítico. Podemos supor,

sem perda de generalidade, que A é e-crítico e B s-crítico. Tome $X := A$ e $Y := (V' + s) \setminus B$. Temos $d^{in}(X) = d^{in}(A) = k$ e $d^{in}(Y) = d^{out}(B) = k$. Além disso, temos que $\emptyset \neq A \setminus B$, $B \setminus A \neq V'$, pois A e B são intersectantes, e que $s \notin A \setminus B$, pois $A, B \subseteq V'$. Logo, a relação (7.24) é válida para $A \setminus B$ e $B \setminus A$, de maneira que $d^{in}(X \cap Y) = d^{in}(A \setminus B) \geq k$ e $d^{in}(X \cup Y) = d^{out}(B \setminus A) \geq k$. Assim, estamos novamente nas condições do lema 7.20 e podemos utilizá-lo para concluir a validade de (ii). ∎

Retomemos, agora, a prova do teorema 7.18.

Como $d^{in}(s) = d^{out}(s)$, se existe um arco st em D', então também existe um arco vs. Temos também que o par $\{vs, st\}$ só não pode sofrer splitting off sem violar a propriedade (7.22) se existir um conjunto crítico contendo v e t.

Considere, então, um arco da forma st. Se A e B são conjuntos críticos intersectantes que contêm t, então, pelo lema 7.21, $A \cup B$ é crítico. Note que s não pertence a nenhum conjunto crítico por definição e que existe o arco st. Logo, $\bar{d}(A, B) > 0$ e o item (ii) do lema 7.21 não se aplica. Conseqüentemente, a união M de todos os conjuntos críticos que contêm t é um conjunto crítico.

Afirmamos que existe um arco vs, com $v \in V' \setminus M$. Suponha que não. Se M é e-crítico e se não existem arcos de $V' \setminus M$ para $\{s\}$, então todos os arcos que saem de $V' \setminus M$ também saem de $(V' \setminus M) + s$. Além disso, existe o arco st que sai de $(V' \setminus M) + s$ e não sai de $V' \setminus M$. Logo,

$$d^{out}(V' \setminus M) < d^{out}((V' \setminus M) + s) = d^{in}(M) = k, \tag{7.27}$$

uma vez que M é e-crítico. Mas isso é uma contradição com a validade de (7.24).

Suponha então que M seja s-crítico. É claro que $\{\{s\}, M, V' \setminus M\}$ é uma partição de V. Como $d^{in}(s) = d^{out}(s)$ e não existem arcos de $V' \setminus M$ para $\{s\}$, então o número de arcos que saem de M para $\{s\}$ deve ser igual ao número de arcos que saem de $\{s\}$ para V'. Ou seja, se k_1 arcos saem de $\{s\}$ para M e k_2 arcos saem de $\{s\}$ para $V' \setminus M$, então $k_1 + k_2$ arcos saem de M para $\{s\}$, de modo que

$$d^{in}(V' \setminus M) = (d^{out}(M) - (k_1 + k_2)) + k_2 = d^{out}(M) - k_1. \tag{7.28}$$

Agora, como existe o arco st, então $k_1 \geq 1$, de modo que

$$d^{in}(V' \setminus M) = d^{out}(M) - k_1 \leq d^{out}(M) - 1 = k - 1. \tag{7.29}$$

Mas isso é uma contradição com (7.24).

Pela escolha de M, concluímos que nenhum conjunto crítico contém v e t. Logo, vs e st podem sofre splitting off sem que a propriedade (7.22) seja violada. Isso conclui a prova. ∎

Um grafo orientado D é k-arco-conexo **minimal** se é k-arco conexo, mas $D - a$ não o é, para qualquer arco a de D.

Teorema 7.22 (Mader [Mad74]): *Todo grafo orientado $D = (V, A)$ k-arco-conexo minimal com $|V| \geq 2$ possui pelo menos dois vértices com grau de entrada e grau de saída k.*

Demonstração: Como D é k-arco-conexo, então

$$d^{\text{in}}(X) \geq k, \tag{7.30}$$

para todo $\emptyset \subset X \subset V$.

Um subconjunto próprio X de V é dito **crítico** se $d^{\text{in}}(X) = k$.

Lema 7.23: *Se X e Y são conjuntos críticos que se cruzam, então $X \cap Y$ e $X \cup Y$ também são críticos. Ademais, $d(X, Y) = 0$.*

Demonstração: [Prova do lema.] Por (7.1), temos

$$k + k = d^{\text{in}}(X) + d^{\text{in}}(Y) = d^{\text{in}}(X \cap Y) + d^{\text{in}}(X \cup Y) + d(X, Y) \geq k + k, \tag{7.31}$$

onde a última desigualdade segue do fato de que, como X e Y se cruzam, então $\emptyset \neq X \cap Y$, $X \cup Y \neq V$, de modo que (7.30) vale para $X \cap Y$ e para $X \cup Y$. Disso segue o lema. ∎

Seja $\mathcal{R}$ uma família de conjuntos críticos de D que cobre A. Dizemos que um arco é **coberto** pela família $\mathcal{R}$ se ele entra em pelo menos um elemento de $\mathcal{R}$ e que a família $\mathcal{R}$ cobre A se todos os arcos em A são cobertos por $\mathcal{R}$. Escolha a família $\mathcal{R}$ de forma que ela seja minimal no sentido de que nenhum de seus elementos possa ser descartado sem que algum arco em A deixe de ser coberto. Como D é k-aresta-conexo minimal, então tal família sempre existe.

Podemos supor também que $\mathcal{R}$ é livre de cruzamentos. (Isto é, que não existem dois membros que se cruzam em $\mathcal{R}$.) De fato, se $\mathcal{R}$ possui elementos X e Y que se cruzam, troque-os por $X \cap Y$ e $X \cup Y$. Pelo lema 7.23, $X \cap Y$ e $X \cup Y$ são críticos e $d(X, Y) = 0$. Logo, $\mathcal{R}$ continua contendo apenas conjuntos críticos e cobrindo todo o conjunto de arcos A, uma vez que não existem arcos entre $X \setminus Y$ e $Y \setminus X$. Tal processo pode ser repetido até que uma família livre de cruzamentos seja obtida. Além disso, o processo é finito pois $|X|^2 + |Y|^2 < |X \cap Y|^2 + |X \cup Y|^2$.

Vamos mostrar que, para todo $s \in V$, existe um vértice $t \neq s$ tal que $d^{\text{in}}(t) = d^{\text{out}}(t) = k$. Como $|V| \geq 2$, disso concluiremos o resultado.

Fixe $s \in V$. Sejam $\mathcal{F} := \{X \in \mathcal{R} : s \notin X\}$ e $\mathcal{H} := \{V \setminus X : s \in X \in \mathcal{R}\}$, de modo que $d^{\text{in}}(X) = k$, para todo $X \in \mathcal{F}$ e $d^{\text{out}}(X) = k$, para todo $X \in \mathcal{H}$. Defina também $\mathcal{L} := \mathcal{H} \cup \mathcal{F}$. Suponha ainda que $\sum\{|X| : X \in \mathcal{L}\}$ seja mínimo.

Como $\mathcal{R}$ é livre de cruzamentos, então $\mathcal{L}$ é laminar. Além disso, a família $\mathcal{L}$ é tal que

todo arco de D entra em membro de $\mathcal{F}$ ou sai de membro de $\mathcal{H}$. (7.32)

Suponha primeiro que todo membro de $\mathcal{L}$ é um conjunto unitário.

Tome $X := \{x \in V - s : \{x\} \in \mathcal{F}\}$ e $Y := \{y \in V - s : \{y\} \in \mathcal{H}\}$. Se $X \cap Y \neq \emptyset$, então qualquer elemento $t \in X \cap Y$ satisfaz $d^{\text{in}}(t) = d^{\text{out}}(t) = k$ e estamos feitos. De fato, como $t \in X$, então $\{t\} \in \mathcal{R}$ é crítico. Ou seja, $d^{\text{in}}(t) = k$. Como $t \in Y$, então $V - y \in \mathcal{R}$ é crítico. Ou seja, $d^{\text{out}}(t) = k$. Vamos mostrar então que, nesse caso, temos de fato $X \cap Y \neq \emptyset$. Suponha o contrário.

Pela propriedade (7.32) e como estamos supondo que todo membro de $\mathcal{L}$ é um conjunto unitário, então todo arco de D tem ponta final em X ou ponta inicial em Y. Assim, já que $X \cap Y = \emptyset$, se um arco tem ponta inicial em X, então deve ter ponta final também em X. Ou seja, temos que $d^{\text{out}}(X) = 0$. Mas, então, como vale (7.30), devemos ter $X = \emptyset$ ou $X = V$. O caso em que $X = \emptyset$ não pode ocorrer, uma vez que, como $d^{\text{out}}(s) \geq k$, então existem arestas da forma su e, pela construção das famílias de conjuntos, devemos ter $u \in X$. Por outro lado, $X = V$ também não pode ocorrer, uma vez que $s \notin X$. Tal contradição nos garante que $X \cap Y \neq \emptyset$, como queríamos.

Suponha agora que existe $X \in \mathcal{L}$ tal que $|X| \geq 2$. Tome X minimal, no sentido de que não existe $Y \subseteq X$, com $Y \in \mathcal{L}$ e $|Y| \geq 2$.

Por simetria, vamos supor $X \in \mathcal{F}$.

Lema 7.24: *O subgrafo induzido $D[X]$ é fortemente conexo.*

Demonstração: [Prova do lema.] Suponha que exista um subconjunto próprio Y de X tal que não exista arco de $X \setminus Y$ a Y. Por (7.30), temos que $d^{\text{in}}(Y) \geq k$. Ademais, pela construção da família $\mathcal{F}$, temos $d^{\text{in}}(X) = k$. Como não entram arcos em Y a partir de $X \setminus Y$, então todo arco que entra em X também entra em Y. Logo, $d^{\text{in}}(Y) = k$. Mas então Y pode substituir X em $\mathcal{F}$, uma contradição com a minimalidade da soma das cardinalidades dos conjuntos na família $\mathcal{L}$. ∎

Tome $A := \{x \in X : \{x\} \in \mathcal{F}\}$ e $B := \{y \in X : \{y\} \in \mathcal{H}\}$. Se $A \cap B \neq \emptyset$, conforme já argumentado acima, estamos feitos. Senão, afirmamos que (i) $A = \emptyset$ e que (ii) $B = X$.

Para verificar (i), começamos notando que $A \neq X$. De fato, do contrário, o conjunto X poderia ser removido da família inicial $\mathcal{R}$, o que seria uma contradição com sua minimalidade. Por outro lado, se $A \neq \emptyset$, então, pelo lema 7.24, existe um arco uv com $u \in A$ e $v \in X - A$. Pela propriedade (7.32) da família $\mathcal{L}$ e como $\mathcal{L}$ é laminar, u deve estar em algum subconjunto de X que também é membro de $\mathcal{H}$ ou v deve estar em algum subconjunto de X que também é membro de $\mathcal{F}$. Pela forma como foi escolhido, X não contém como subconjunto nenhum membro de $\mathcal{L}$ que não seja unitário. Logo, todos os membros

de $\mathcal{H}$ que são subconjuntos de X são os subconjuntos unitários de B e todos os membros de $\mathcal{F}$ que são subconjuntos de X são os subconjuntos unitários de A. Agora, como estamos supondo $A \cap B = \emptyset$ e $u \in A$, então uv não sai de nenhum elemento de $\mathcal{H}$. Além disso, como $v \in X - A$, então uv também não entra em nenhum elemento de $\mathcal{F}$. Sendo assim, a existência do arco uv contradiz a propriedade (7.32) da família $\mathcal{L}$. Segue que o arco uv não pode existir e, portanto, que $A = \emptyset$.

A prova de (ii) é semelhante. Temos que todo arco induzido por X deve possuir sua ponta final em B e que, portanto, B é não-vazio. Por outro lado, se $B \neq X$, pelo lema 7.24, deve existir um arco uv, com $u \in X - B$ e $v \in B$. Entretanto, por argumento análogo ao apresentado acima, a existência de tal arco também contradiz a construção da família $\mathcal{L}$, de onde concluímos que $B = X$.

Concluímos, assim, que $d^{\text{in}}(X) = k$ e que $d^{\text{out}}(x) = k$, para todo $x \in X$. Note que $X \in \mathcal{F}$ e que, como $B = X$, então todo $x \in X$ é tal que $\{x\} \in \mathcal{H}$. Mas, então,

$$\begin{aligned} k|X| &= \sum_{x \in X} d^{\text{out}}(x) = d^{\text{out}}(X) + |A(X)| \geq k + |A(X)| \\ &= k + \left(\sum_{x \in X} d^{\text{in}}(x) \right) - d^{\text{in}}(X) = \sum_{x \in X} d^{\text{in}}(x) \geq k|X|. \end{aligned} \tag{7.33}$$

Ou seja, todas as relações acima valem com igualdade, de onde concluímos que $d^{\text{in}}(x) = k$, para todo $x \in X$. Temos, então, que $s \notin X$ e que $d^{\text{in}}(x) = d^{\text{out}}(x) = k$, para todo $x \in X$. Mas isso era justamente o que queríamos. ∎

Podemos, agora, combinar os teoremas 7.18 e 7.22 e obter a seguinte caracterização dos grafos k-arco-conexos, devida a Mader [Mad74, Mad82].

Teorema 7.25: *Um grafo orientado D é k-arco-conexo se, e somente se, pode ser obtido a partir de um único vértice através da aplicação em qualquer ordem de uma das seguintes operações:*

Operação A: Adição de um novo arco ligando vértices já existentes.

Operação B: Escolha de k arcos quaisquer, subdivisão de cada um deles com um novo vértice e contração dos k novos vértices em um único.

Por fim, apresentamos um resultado de Frank [Fra92] que estabelece o número mínimo de arcos que precisa ser adicionado a um grafo orientado de modo a torná-lo k-arco-conexo. Tal resultado é também uma aplicação de técnicas envolvendo funções submodulares.

Teorema 7.26: *Pode-se transformar um grafo orientado $D = (V, A)$ em k-arco-conexo através da adição de no máximo γ arcos se, e somente se,*

$$\sum_{i=1}^{t} \left(k - d^{\text{in}}(X_i) \right) \leq \gamma \tag{7.34a}$$

e

$$\sum_{i=1}^{t} \left(k - d^{\text{out}}(X_i)\right) \leq \gamma, \tag{7.34b}$$

para toda subpartição $\{X_1, X_2, \ldots, X_t\}$ *de* V.

Demonstração: Começamos pela necessidade da condição (7.34). Suponha que γ novos arcos tenham sido adicionados a D de forma a torná-lo k-arco-conexo. Seja F o conjunto dos novos arcos e seja $D' = (V, E \cup F)$ o grafo k-arco-conexo resultante da adição dos arcos em F a D. Para todo subconjunto próprio X de V, temos $d^{\text{in}}_{D'}(X) \geq k$ e, portanto, $|F \cap \delta^{\text{in}}_{D'}(X)| \geq k - d^{\text{in}}_{D}(X)$. Assim, se $\{X_1, \ldots, X_t\}$ é um subpartição de V, então $\gamma = |F| \geq \sum_{i=1}^{t} |F \cap \delta^{\text{in}}_{D'}(X_i)| \geq \sum_{i=1}^{t} \left(k - d^{\text{in}}(X_i)\right)$. Disso segue a necessidade de (7.34a). A prova da necessidade de (7.34b) é completamente análoga.

Para verificar a suficiência de (7.34), começamos provando o seguinte lema.

Lema 7.27: *Um grafo orientado* $D = (V, E)$ *pode ser estendido a um grafo orientado* $D' = (V + s, E')$ *através da adição de um novo vértice* s*, de* γ *novos arcos entrando em* s *e de* γ *novos arcos saindo de* s*, de maneira que*

$$d^{\text{in}}_{D'}(X) \geq k \tag{7.35a}$$

e

$$d^{\text{out}}_{D'}(X) \geq k, \tag{7.35b}$$

para todo subconjunto próprio X *de* V.

Demonstração: [Prova do lema.] Vamos provar que é possível estender D a D' através da adição do vértice s e de γ arcos saindo de s de maneira que (7.35a) seja satisfeita. A prova de que com a adição de γ arcos entrando em s podemos obter D' tal que (7.35b) também seja satisfeita é completamente análoga.

Comece adicionando o vértice s a D e uma quantidade suficiente de arcos saindo de s, de maneira que (7.35a) seja satisfeita. Observe que a adição de k arcos de s a v, para cada vértice $v \in V$, é sempre suficiente para tanto. Feito isso, remova cada arco da forma $a = sv$, com $v \in V$, se $D - a$ continua satisfazendo (7.35a). Denote por $D' = (V + s, E')$ o grafo obtido ao final desse processo. Vamos provar que $d^{\text{out}}_{D'}(s) \leq \gamma$. Disso concluiremos que a adição de no máximo γ arcos saindo de s é suficiente para satisfazer (7.35a).

Chamemos um subconjunto $\emptyset \subset X \subset V$ de **crítico** se $d^{\text{in}}_{D'}(X) = k$.

Seja $S := \{v \in V : sv \in E'\}$. Um arco $a = sv$ só não pode ser removido de D' sem que a condição (7.35a) deixe de ser satisfeita se a entra em algum conjunto crítico. Logo, pela construção de D', temos que existe uma família $\mathcal{F} = \{X_1, X_2, \ldots, X_t\}$ de conjuntos críticos tal que $S \subseteq \bigcup_{i=1}^{t} X_i$. Podemos, então, escolher uma família $\mathcal{F}$ como acima de forma que t seja mínimo. Temos, agora, dois casos a considerar.

Caso 1: Os conjuntos críticos que compõem a família $\mathcal{F}$ são mutuamente disjuntos. Neste caso, temos

$$kt = \sum_{i=1}^{t} d^{\text{in}}_{D'}(X_i) = d^{\text{out}}_{D'}(s) + \sum_{i=1}^{t} d^{\text{in}}_{D}(X_i). \tag{7.36}$$

Utilizando agora a relação acima, como também (7.34a), temos

$$d^{\text{out}}_{D'}(s) = kt - \sum_{i=1}^{t} d^{\text{in}}_{D}(X_i) = \sum_{i=1}^{t} \left(k - d^{\text{in}}(X_i)\right) \leq \gamma, \tag{7.37}$$

justamente como queríamos.

Caso 2: Suponha agora que existam dois conjuntos críticos intersectantes A e B em $\mathcal{F}$. Se $A \cup B \neq V$, então, pelo lema 7.20, temos que $A \cup B$ é também crítico. Nesse caso, poderíamos substituir A e B por $A \cup B$ em $\mathcal{F}$ e obter uma família $\mathcal{F}'$, de cardinalidade menor do que $\mathcal{F}$, o que seria uma contradição à escolha de $\mathcal{F}$. Concluímos, assim, que $A \cup B = V$.

Defina, então, $Y_1 := V \setminus A$ e $Y_2 := V \setminus B$, de maneira que $d^{\text{out}}_{D}(Y_1) = d^{\text{in}}_{D}(A)$ e $d^{\text{out}}_{D}(Y_2) = d^{\text{in}}_{D}(B)$. Utilizando, então, (7.34b), concluímos

$$\gamma \geq k - d^{\text{out}}_{D}(Y_1) + k - d^{\text{out}}_{D}(Y_2) = k - d^{\text{in}}_{D}(A) + k - d^{\text{in}}_{D}(B) \tag{7.38}$$

Pela minimalidade da cardinalidade da família $\mathcal{F}$, temos também que $\mathcal{F} = \{A, B\}$. Disso concluímos que todos os arcos que saem de s em D' entram em A ou em B. Lembrando também que $A \cap B \neq \emptyset$, temos

$$d^{\text{in}}_{D'}(A) + d^{\text{in}}_{D'}(B) \geq d^{\text{in}}_{D}(A) + d^{\text{in}}_{D}(B) + d^{\text{out}}_{D'}(s) \tag{7.39}$$

Portanto,

$$\gamma \geq k - d^{\text{in}}_{D}(A) + k - d^{\text{in}}_{D}(B) \geq k - d^{\text{in}}_{D'}(A) + k - d^{\text{in}}_{D'}(B) + d^{\text{out}}_{D'}(s). \tag{7.40}$$

Como A e B são críticos segue que também neste caso temos $d^{\text{out}}_{D'}(s) \leq \gamma$. ∎

Para concluir a prova do teorema 7.26, primeiro estendemos o grafo D ao grafo D' através da adição de 2γ arcos, como no lema 7.27, de forma que as condições em (7.35) sejam satisfeitas. Feito isso, aplicamos γ vezes o teorema 7.18 sobre os arcos incidentes no vértice s em D' e obtemos o grafo D'', tal que $d^{\text{in}}_{D''}(s) = d^{\text{out}}_{D''}(s) = 0$. A existência do grafo D'' garante a suficiência da condição (7.34). ∎

Considerações finais

Como idéia inicial, pensamos em desenvolver uma dissertação de mestrado que fosse uma grande resenha de diversos aspectos de submodularidade em otimização combinatória. Mais especificamente, seria interessante cobrir os seguintes tópicos:

- aspectos teóricos de funções submodulares [Fuj05, Lov83];
- modelos em combinatória poliédrica que envolvem submodularidade, tais como polimatróides, fluxos submodulares e *generalized polymatroids*, bem como suas aplicações e conseqüências em otimização combinatória [Edm70, EG77, Fra84b, FT88, Fuj97, Sch84];
- aplicações de técnicas envolvendo funções submodulares a problemas combinatórios [Fra90, Fra93b, Fra94, Lov70];
- aspectos algorítmicos de submodularidade; em particular, os algoritmos combinatórios fortemente polinomiais para minimização de funções submodulares [Cun84, BCT85, Cun85, IFF01, Sch00, Vyg03, FI03, Iwa02, Iwa03];

Entretanto, nossas investigações do assunto nos mostraram que um projeto desse tipo seria por demais ambicioso, dada a vastidão de cada tópico. Também não seria interessante cobrir os assuntos todos de forma muito superficial.

Em função disso, optamos por dedicar a dissertação à apresentação dos algoritmos combinatórios e fortemente polinomiais desenvolvidos recentemente para a minimização de funções submodulares. Conforme mencionamos ao longo do texto, esse foi um problema bastante desafiador e que permaneceu em aberto na área por cerca de 20 anos. Além disso, os algoritmos propostos combinam idéias clássicas e bastante interessantes.

A dissertação foi desenvolvida como um histórico sobre minimização de funções submodulares, com foco nas idéias que levaram ao desenvolvimento dos primeiros algoritmos combinatórios e fortemente polinomiais. Nosso objetivo foi ressaltar de onde vieram as idéias presentes nesses algoritmos e, principalmente, apresentá-los como extensões dos métodos de Cunningham, desenvolvidos no contexto do problema para matróides e apresentados no capítulo 4. A leitura direta dos trabalhos em que os algoritmos são descritos pode se apresentar um pouco árdua, não trazendo tanta intuição sobre como se chegou ao seu desenvolvimento. A dissertação traz também como contribuição a apresentação de um conjunto de algoritmos em uma mesma linguagem.

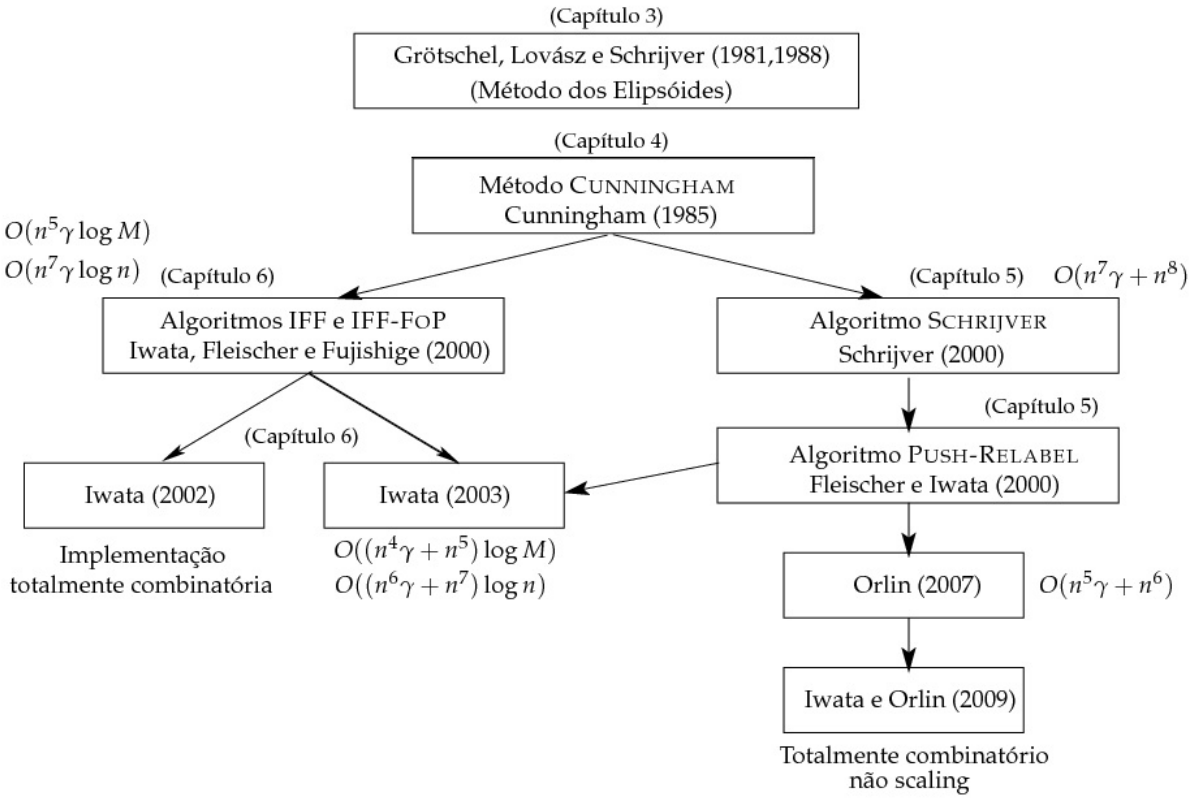
Acreditamos ter apresentado um texto auto-contido, que pode servir como um guia para o estudo de aspectos teóricos e algorítmicos associados a funções submodulares. Além da descrição dos algoritmos para minimização, mencionamos muitas referências que podem ser úteis para a investigação de outros aspectos que não foram abordados com tanta profundidade. Procuramos também motivar a importância de submodularidade em otimização combinatória, principalmente com algumas aplicações no capítulo 7.

Iwata [Iwa08] apresentou em 2008 um trabalho na linha do que estávamos desenvolvendo, mas um pouco mais resumido. O artigo também é um bom guia para o estudo do assunto, assim como uma outra resenha de McCormick [McC06]. Também mais recentemente, entre 2007 e 2008, alguns outros avanços em algoritmos para minimização de funções submodulares foram alcançados. Orlin [Orl07, Orl09] apresentou um algoritmo fortemente polinomial cujo consumo de tempo supera em mais de um fator de n o melhor consumo de tempo dentre os algoritmos para minimização desenvolvidos até então. Orlin trouxe uma abordagem um pouco diferente ao problema. Seu algoritmo trabalha diretamente sobre os vetores no polimatróide das bases $B(f)$, sem a necessidade de definir um grafo auxiliar ou utilizar técnicas de caminhos de aumento ou fluxos. Apesar disso, ainda se baseia na relação min-max

$$\max\{x^-(S)\colon x \in B(f)\} = \min\{f(U)\colon U \subseteq S\},$$

assim como mantém os vetores em $B(f)$ através de representações como combinação convexa de pontos extremos de $B(f)$, conforme sugerido por Cunningham. Um outro algoritmo baseado no trabalho de Orlin foi desenvolvido por Iwata e Orlin [IO09]. Apesar deste último algoritmo não trazer um ganho de complexidade sobre o anterior, os autores ressaltam que este é o primeiro algoritmo que pode ser implementado de forma totalmente combinatória que não seja baseado na técnica de scaling.

O esquema abaixo traz um resumo dos algoritmos para minimização de funções submodulares. A dissertação apenas não detalha os algoritmos de Iwata [Iwa03], Orlin [Orl07, Orl09] e Iwata e Orlin [IO09]. Os algoritmos de Orlin [Orl07, Orl09] e Iwata e Orlin [IO09] são os que comentamos acima. O algoritmo de Iwata [Iwa03], mencionado na seção 6.8, foi desenvolvido tendo como base idéias presentes nos algoritmos de Schrijver [Sch00] e de Iwata, Fleischer e Fujishige [IFF01], trazendo um ganho de complexidade sobre os mesmos.


(Capítulo 3)
Grötschel, Lovász e Schrijver (1981,1988)
(Método dos Elipsóides)
(Capítulo 4)
Método CUNNINGHAM
Cunningham (1985)
$O(n^5\gamma \log M)$
$O(n^7\gamma \log n)$
(Capítulo 6)
Algoritmos IFF e IFF-FOP
Iwata, Fleischer e Fujishige (2000)
(Capítulo 5)
$O(n^7\gamma + n^8)$
Algoritmo SCHRIJVER
Schrijver (2000)
(Capítulo 5)
Algoritmo PUSH-RELABEL
Fleischer e Iwata (2000)
(Capítulo 6)
Iwata (2002)
Implementação
totalmente combinatória
Iwata (2003)
$O((n^4\gamma + n^5) \log M)$
$O((n^6\gamma + n^7) \log n)$
Orlin (2007)
$O(n^5\gamma + n^6)$
Iwata e Orlin (2009)
Totalmente combinatório
não scaling

Referências Bibliográficas

[BCT85] R. E. Bixby, W. H. Cunningham e D. M. Topkis, *The partial order of a polymatroid extreme point*, Mathematics of Operations Research **10** (1985), no. 3, 367–378. 13, 23, 25, 33, 36, 39, 41, 42, 123

[Car11] C. Carathéodory, *Über den variabilitätsbereich der fourierschen konstanten von positiven harmonischen funktionen*, Rendiconto del Circolo Matematico di Palermo **32** (1911), 193–217. 14

[Cun84] W. H. Cunningham, *Testing membership in matroid polyhedra*, Journal of Combinatorial Theory, Series B **36** (1984), 161–188. 8, 33, 34, 53, 123

[Cun85] ———, *On submodular function minimization*, Combinatorica **5** (1985), 185–192. 42, 45, 48, 123

[Dil44] R. P. Dilworth, *Dependence relations in a semi-modular lattice*, Duke Mathematical Journal **11** (1944), 575–587. 10

[Din70] E.A. Dinic, *Algorithm for solution of a problem of maximum flow in a network with power estimation*, Sov. Math. Dokl. **11** (1970), no. 5, 1277–1280. 8, 33, 34

[Edm65] J. Edmonds, *Minimum partition of a matroid into independent subsets*, Journal of Research National Bureau of Standards Section B **69** (1965), 67–72. 33, 34

[Edm70] ———, *Submodular functions, matroids, and certain polyhedra*, Combinatorial Structures and Their Applications (New York) (R. Guy, H. Hanani, N. Sauer e J. Schönheim, eds.), Gordon and Breach, 1970, Proceedings Calgary International Conference on Combinatorial Structures and Their Applications, pp. 69–87. 1, 8, 11, 13, 14, 16, 20, 21, 34, 101, 109, 123

[Edm73] ———, *Edge-disjoint branchings*, Combinatorial Algorithms, Algorithmics Press, New York, 1973, pp. 91–96. 106

[EG77] J. Edmonds e R. Giles, *A min-max relation of submodular functions on graphs*, Studies in Integer Programming (P. L. Hammer et al., ed.), Annals of Discrete Mathematics, vol. 1, North-Holland, 1977, pp. 185–204. 1, 19, 21, 101, 123

[EK72] J. Edmonds e R. Karp, *Theoretical improvements in algorithmic efficiency for network flow problems*, Journal of the Association for Computing Machinery **19** (1972), 248–264. 2, 8, 33, 34, 47, 73

[FF56] L.R. Ford e D.R. Fulkerson, *Maximal flow through a network*, Canadian Journal of Mathematics **8** (1956), 399–404. 8, 33, 34, 47, 100

[FI00a] L. Fleischer e S. Iwata, *Improved algorithms for submodular function minimization and submodular flow*, STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing (New York, NY, USA), ACM, 2000, pp. 107–116. 21, 27, 61, 101

[FI00b] S. Fujishige e S. Iwata, *Algorithms for submodular flows*, IEICE Trans. Inform. Syst **83** (2000), 322–329. 1, 27, 101

[FI03] L. Fleischer e S. Iwata, *A push-relabel framework for submodular function minimization and applications to parametric optimization*, Discrete and Applied Mathematics **131** (2003), no. 2, 311–322. 2, 47, 61, 123

[FIM02] L. Fleischer, S. Iwata e S.T. McCormick, *A faster capacity scaling algorithm for minimum cost submodular flow*, Mathematical Programming, Series A **92** (2002), no. 1, 119–139. 2, 21, 27, 71, 101

[Fle00] L. Fleischer, *Recent progress in submodular function minimization*, Optima - Mathematical Programming Society Newsletter (2000), no. 64, 1–11. 36, 40, 49, 61

[Fra79a] A. Frank, *Covering branchings*, Acta Scientiarum Mathematicarum [Szeged] **41** (1979), 77–81. 111

[Fra79b] ———, *Kernel systems of directed graphs*, Acta Scientiarum Mathematicarum [Szeged] **41** (1979), 63–76. 1, 101

[Fra81] ———, *On disjoint trees and arborescences*, Algebraic Methods in Graph Theory, Vol. I (Amsterdam) (L. Lovàsz e V. T. Sos, eds.), Colloquia Mathematica Societatis János Bolyai, 25, North-Holland, 1981, pp. 159–170. 109, 113

[Fra82] ———, *An algorithm for submodular functions on graphs*, Bonn Workshop on Combinatorial Optimization (B. Korte A. Bachem, M. Grötschel, ed.), Annals of Discrete Mathematics, vol. 16, North-Holland, Amsterdam, 1982, pp. 97–120. 30

[Fra84a] ———, *Generalized polymatroids*, Finite and Infinite Sets, Vol. I (Amsterdam) (L. Lovàsz A. Hajnal e V. T. Sos, eds.), Colloquia Mathematica Societatis János Bolyai, 37, North-Holland, 1984, Proceedings Sixth Hungarian Combinatorial Colloquium, Eger, 1981, pp. 285–294. 1, 101

[Fra84b] ———, *Submodular flows*, Progress in Combinatorial Optimization (W.R. Pulleyblank, ed.), Academic Press, Toronto, Ontario, 1984, Proceedings Conference, Waterloo, Ontario, 1982, pp. 147–165. 1, 2, 21, 27, 32, 101, 123

[Fra90] ———, *Packing paths, circuits, and cuts — a survey*, Paths, Flows, and VLSI-Layout (B. Korte, L. Lovász e A. Schrijver, eds.), Springer, 1990, pp. 47–100. 1, 9, 123

[Fra92] ———, *Augmenting graphs to meet edge-connectivity requirements*, SIAM Journal on Discrete Mathematics **5** (1992), no. 1, 25–53. MR 92m:05122 115, 120

[Fra93a] ———, *Applications of submodular functions*, Surveys in combinatorics, 1993, Cambridge University Press, New York, NY, USA, 1993, pp. 85–136. 99

[Fra93b] ———, *Submodular functions in graph theory*, Discrete Mathematics **111** (1993), 231–243. 1, 102, 104, 112, 123

[Fra94] ———, *Connectivity augmentation problems in network design*, Mathematical Programming — State of the Art, The University of Michigan, Ann Arbor. Michigan, 1994, pp. 34–63. 1, 9, 123

[FT88] A. Frank e É. Tardos, *Generalized polymatroids and submodular flows*, Mathematical Programming **42** (1988), 489–563. 123

[Fuj97] S. Fujishige, *A min-max theorem for bisubmodular polyhedra*, SIAM Journal of Discrete Mathematics (1997), 294–308. 123

[Fuj03] ———, *Submodular function minimization and related topics*, The Second Japanese-Sino Optimization Meeting, Part I (Kyoto), 2003, pp. 167–180. 36

[Fuj05] ———, *Submodular functions and optimization*, Annals of Discrete Mathematics, vol. 58, Elsevier, Amsterdam, 2005. 36, 123

[Ful74] D. R. Fulkerson, *Packing rooted directed cuts in a weighted directed graph*, Mathematical Programming **6** (1974), 1–13. 100

[FZ92] S. Fujishige e X. Zhang, *New algorithms for the intersection problem of submodular systems*, Japan J. lndust. Appl. Math. **9** (1992), 369–382. 61

[Gal57] D. Gale, *A theorem on flows in networks*, Pacific Journal of Mathematics **7** (1957), 1073–1082. 72

[GGT89] G. Gallo, M.D. Grigoriadis e R.E. Tarjan, *A fast parametric maximum flow algorithm and applications*, SIAM J. Comput. **18** (1989), no. 1, 30–55. 61

[GLS81] M. Grötschel, L. Lovász e A. Schrijver, *The ellipsoid method and its consequence in combinatorial optimization*, Combinatorica **1** (1981), 169–197. 2, 27, 31, 32

[GLS88] M. Grötschel, L. Lovász e A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Algorithms and Combinatorics, vol. 2, Springer, 1988 (English). 2, 27, 31, 32

[GT88] A.V. Goldberg e R.E. Tarjan, *A new approach to the maximum flow problem*, Journal of ACM (1988), no. 35, 921–940. 8, 47, 61

[Hal35] P. Hall, *On representatives of subsets*, The Journal of the London Mathematical Society **10** (1935), no. 37, 26–30. 102

[Hof74] A. J. Hoffman, *A generalization of max flow-min cut*, Mathematical Programming **6** (1974), 352–359. 101

[IFF01] S. Iwata, L. Fleischer e S. Fujishige, *A combinatorial strongly polynomial algorithm for minimizing submodular functions*, Journal of the Association for Computing Machinery **48** (2001), 761–777. 2, 36, 40, 47, 49, 71, 72, 86, 95, 123, 124

[IMS99] S. Iwata, S. T. McCormick e M. Shigeno, *A strongly polynomial cut canceling algorithm for the submodular flow problem*, Proceedings of the 7th International IPCO Conference on Integer Programming and Combinatorial Optimization (London, UK), Springer-Verlag, 1999, pp. 259–272. 71

[IMS00] S. Iwata, S. T. McCormick e M. Shigero, *A fast cost scaling algorithm for submodular flow*, Information Processing Letters **74** (2000), no. 3-4, 123–128. 2

[IMS05] S. Iwata, S. T. McCormick e M. Shigeno, *A strongly polynomial cut canceling algorithm for minimum cost submodular flow*, SIAM J. Discret. Math. **19** (2005), no. 2, 304–320. 2, 21, 27, 101

[IO09] S. Iwata e J. B. Orlin, *A simple combinatorial algorithm for submodular function minimization*, SODA, 2009, pp. 1230–1237. 2, 124

[Iwa97] S. Iwata, *A capacity scaling algorithm for convex cost submodular flows*, Math. Program. **76** (1997), 299–308. 71

[Iwa02] ———, *A fully combinatorial algorithm for submodular function minimization*, Journal of Combinatorial Theory, Series B (2002), no. 2, 203–212. 2, 36, 95, 96, 123

[Iwa03] ———, *A faster scaling algorithm for minimizing submodular functions*, SIAM Journal on Computing **32** (2003), no. 4, 833–840. 2, 97, 123, 124

[Iwa08] ———, *Submodular function minimization*, Mathematical Programming, Ser. B **112** (2008), 45–64. 124

[Kar72] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (R. E. Miller e J. W. Thatcher, eds.), Plenum Press, 1972, pp. 85–103. 29

[Kön31] D. König, *Graphok és matrixok [hungarian; graphs and matrices]*, Matematikai és Fizikai Lapok **38** (1931), 116–119. 9, 100

[LM82] E. L. Lawler e C. U. Martel, *Computing maximal "polymatroidal" network flows*, Mathematics of Operations Research **7** (1982), 334–347. 33, 35, 53

[Lov70] L. Lovász, *A generalization of Kőnig's theorem*, Acta Mathematica Academiae Scientiarum Hungaricae **21** (1970), 443–446. 1, 123

[Lov76] ———, *On two minimax theorems in graph theory*, Journal of Combinatorial Theory, Series B **21** (1976), 96–103. 101, 106

[Lov83] ———, *Submodular functions and convexity*, Mathematical Programming — The State of the Art (Bachem A, M. Grötschel e B. Korte, eds.), Springer, 1983, pp. 234–257. 1, 2, 7, 9, 30, 32, 123

[LY78] C. L. Lucchesi e D. H. Younger, *A minimax theorem for directed graphs*, The Journal of the London Mathematical Society **17** (1978), no. 2, 369–374. 100

[Mad74] W. Mader, *Ecken vom innen- und aussengrad n in minimal n-fach kantenzusammenhängenden digraphen*, Arch. Math. **25** (1974), 107–112. 115, 118, 120

[Mad82] ———, *Konstruktion aller n–fach kantenzusammenhängenden digraphen*, European Journal of Combinatorics **3** (1982), 63–67. 115, 120

[McC06] S.T. McCormick, *Submodular function minimization*, Handbook on Discrete Optimization (K. Aardal, G. Nemhauser e R. Weismantel, eds.), vol. 12, Elsevier, 2006, pp. 321–391. 124

[Men27] K. Menger, *Zur allgemeinen kurventheorie*, Fundamenta Mathemticae **10** (1927), 96–115. 104

[NW61] C. St. J. A. Nash-Williams, *Edge-disjoint spanning trees of finite graphs*, The Journal of the London Mathematical Society **36** (1961), 445–450. 33, 112

[NW64] ———, *Decompositions of finite graphs into forests*, The Journal of the London Mathematical Society **39** (1964), 12. 33

[Orl07] J. B. Orlin, *A faster strongly polynomial time algorithm for submodular function minimization*, IPCO '07: Proceedings of the 12th international conference on Integer Programming and Combinatorial Optimization (Berlin, Heidelberg), Springer-Verlag, 2007, pp. 240–251. 2, 7, 124

[Orl09] ———, *A faster strongly polynomial time algorithm for submodular function minimization*, Math. Program. **118** (2009), no. 2, 237–251. 2, 124

[Sch80] P. Schönsleben, *Ganzzahlige polymatroid-intersektions-algorithmen*, Ph.D. thesis, Eigenossische Technische Hochschule Zurich, 1980. 33, 35, 53

[Sch84] A. Schrijver, *Total dual integrality from directed graphs, crossing families, e sub- e supremodular functions*, Progress in Combinatorial Optimization (W.R. Pulleyblank, ed.), Academic Press, Toronto, Ontario, 1984, Proceedings Conference, Waterloo, Ontario, 1982, pp. 315–361. 101, 123

[Sch00] ———, *A combinatorial algorithm minimizing submodular functions in strongly polynomial time*, Journal of Combinatorial Theory, Series B **80** (2000), no. 2, 346–355. 2, 35, 36, 40, 47, 95, 123, 124

[Sch03] ———, *Combinatorial optimization: Polyhedra and efficiency*, Algorithms and Combinatorics, vol. 24, Springer, 2003. 3, 8, 9, 21, 29, 30, 104

[Tar85] É. Tardos, *A strongly polynomial minimum cost circulations algorithm*, Combinatorica **5** (1985), 247–255. 2

[Top82] D. M. Topkis, *Adjacency on polymatroids*, Mathematical Programming **30** (1982), no. 2, 229–237. 13, 26

[Tut61] W. T. Tutte, *On a problem of decomposing a graph into n connected factors*, The Journal of the London Mathematical Society **36** (1961), 221–230. 33, 112

[Vyg03] J. Vygen, *A note on Schrijver's submodular function minimization algorithm*, Jounal of Combinatorial Theory, Series B (2003), 399–402. 56, 123

[Whi35] H. Whitney, *On the abstract properties of linear dependence*, American Journal of Mathematics **57** (1935), 509–533. 1, 7

Índice Remissivo