Fluxos em Redes

Juliana Barby Simão
Apoio financeiro da FAPESP
Processo 04/00580-8

Marcelo Hashimoto
Apoio financeiro da FAPESP
Processo 04/00581-4

Orientador: José Coelho de Pina

Sumário

Capítulo 1

Introdução

Pode-se dizer que redes estão constantemente presentes em nossas vidas e em inúmeros lugares e situações. As redes elétricas, por exemplo, distribuem a energia das usinas entre nossas casas, garantindo iluminação e conforto. Redes de distribuição de água e gás também fornecem recursos indispensáveis em nosso cotidiano. São as redes telefônicas, a *Internet* e mesmo as redes de distribuição dos correios que permitem às pessoas comunicarem-se à distância. Atividades simples, como o transporte de cargas, dependem de redes de estradas, ferrovias, rios ou mesmo simples corredores.

Em todos os exemplos acima mencionados, objetiva-se transportar alguma coisa de um ponto a outro, ou distribuí-la entre vários pontos específicos, através de vias bem definidas. Obviamente, sempre é interessante garantir que esse transporte seja feito da forma mais eficiente possível. A definição de eficiência pode variar dependendo do problema. Podemos estar interessados, por exemplo, em maximizar a quantidade transportada de um ponto a outro ou em minimizar o custo de uma distribuição.

Nessas situações, queremos atribuir uma certa quantidade de fluxo em cada via, de modo a obter eficiência máxima sob certas restrições. Esse conjunto de problemas, mais conhecido como *problemas sobre fluxos em redes*, é um representante clássico dos problemas de *otimização combinatória*, ou seja, problemas de otimização formulados sobre estruturas discretas.

Uma característica comum dos problemas de otimização combinatória é o fato de que o conjunto de soluções viáveis, embora seja finito, pode ser muito grande. Isso implica que estratégias de força bruta, como testar todas as atribuições possíveis, consomem tempo excessivo ao ponto de inviabilizar sua utilização até mesmo nos computadores mais poderosos. É preciso buscar estratégias que permitam resolver o problema em tempo razoável.

Este texto apresenta os principais conceitos e problemas na teoria dos fluxos em redes e descreve algoritmos eficientes para resolvê-los.

Capítulo 2

Redes

Este capítulo introduz formalmente o conceito de redes e redes simétricas e de conceitos relacionados, como caminhos, circuitos, capacidades, custos, separadores e cortes. Também apresenta uma seção a respeito de resultados sobre caminhos de custo mínimo.

2.1 Redes

Uma **rede** é um par (N, A), onde N é um conjunto finito qualquer e A é um multiconjunto de pares ordenados de elementos de N. Os elementos de N são chamados de **vértices** (*nodes*) e os de A são chamados de **arcos** (*arcs*). O conjunto de vértices de uma rede G é denotado por N_G, o conjunto de arcos por A_G, o número de vértices por $n(G)$ e o número de arcos por $m(G)$. Quando a rede está implícita pelo contexto, podemos utilizar simplesmente as notações N, A, n e m respectivamente. Pode-se representar naturalmente uma rede através de um diagrama, como o da figura 2.1, onde os vértices são pequenos círculos e os arcos são flechas ligando os círculos.

Seja a um arco (i, j) de uma rede. Os vértices i e j são chamados de **pontas** de a. Mais especificamente, i é a **ponta inicial** e j é a **ponta final**. Diz-se também que o arco a **incide** nos vértices i e j, ou ainda, que o arco a **sai** do vértice i e **entra** em j.

O conjunto de todos os arcos de uma rede G que incidem em um vértice i é denotado por $\delta_G(i)$. Mais especificamente, o conjunto de todos os arcos que saem de i é denotado por $\delta_G^-(i)$ e o conjunto de todos os arcos que entram em i é denotado por $\delta_G^+(i)$. Denota-se tais conjuntos simplesmente por $\delta(i)$, $\delta^+(i)$ e $\delta^-(i)$ quando a rede está implícita pelo contexto.

Por conveniência de notação, são utilizadas algumas convenções:

O conjunto dos números inteiros é denotado por $\mathbb{Z}$ e o conjunto dos inteiros não-negativos por $\mathbb{Z}_{\geq}$. Analogamente, o conjunto dos números racionais é denotado por $\mathbb{Q}$ e o conjunto dos racionais não-negativos por $\mathbb{Q}_{\geq}$.

Se f é uma função de A em algum dos conjuntos numéricos definidos

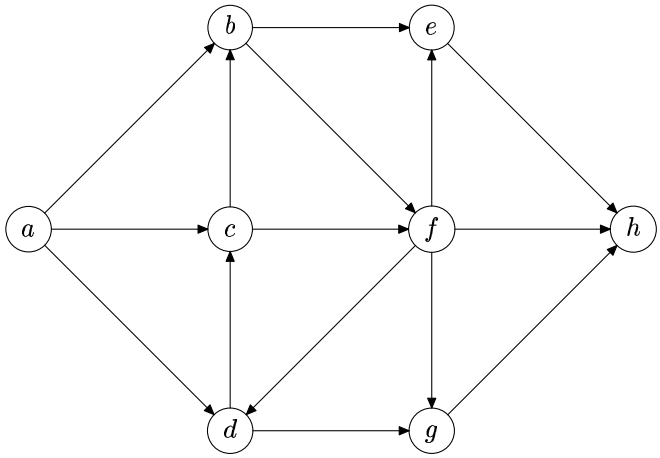

Figura 2.1: Representação da rede $(\{a,\ b,\ c,\ d,\ e,\ f,\ g,\ h\},\ \{(a,b),\ (a,c),\ (a,d),\ (b,c),\ (d,c),\ (b,e),\ (b,f),\ (c,f),\ (f,d),\ (d,g),\ (e,h),\ (f,h),\ (g,h)\})$.

acima, então o valor de f em relação a um arco a é denotado por f_a. Se B é um subconjunto de A, então $f(B)$ denota a soma $\sum_{a\in B} f_a$.

Se g é uma função de N em algum dos conjuntos numéricos definidos acima, então o valor de g em relação a um vértice i é denotado por $g(i)$. Se M é um subconjunto de N, então $g(M)$ denota a soma $\sum_{i\in M} g(i)$.

2.2 Caminhos e circuitos

Um **caminho** P em uma rede é uma seqüência $\langle i_0, a_1, i_1, a_2, i_2, \ldots, a_k, i_k\rangle$, onde $i_0, \ldots, i_k$ são vértices dois-a-dois distintos e $a_1, \ldots, a_k$ são arcos tais que $a_j = (i_{j-1}, i_j)$ para todo j entre 1 e k. O valor k é o **comprimento** do caminho, o vértice i_0 é a sua **origem** e o vértice i_k é seu **destino**. Diz-se também que P é um caminho **entre** i_0 e i_k e que i_0 e i_k são os seus **extremos**. Os conjuntos de vértices e arcos de P são denotados respectivamente por N_P e A_P. A figura 2.2 mostra um exemplo de caminho.

Seja P um caminho $\langle i_0, a_1, i_1, a_2, i_2, \ldots, a_k, i_k\rangle$ em uma rede. Dado um vértice i_j, com j entre 1 e k, o **vértice predecessor** de i_j em P é o vértice i_{j-1}, e o **arco predecessor** de i_j em P é o arco a_j.

Dados dois vértices i e j de uma rede, dizemos que j é **acessível** a partir de i se existe um caminho entre i e j na rede. Se j é acessível a partir de i, um caminho P entre i e j é chamado de **mínimo** se não existe outro caminho entre i e j com comprimento menor que o de P. A **distância** entre i e j é o comprimento de um caminho mínimo entre i e j. Se j não é acessível a partir de i, diz-se que a distância entre i e j é **infinita**.

Um **circuito** W em uma rede é, similarmente a um caminho, uma seqüên-

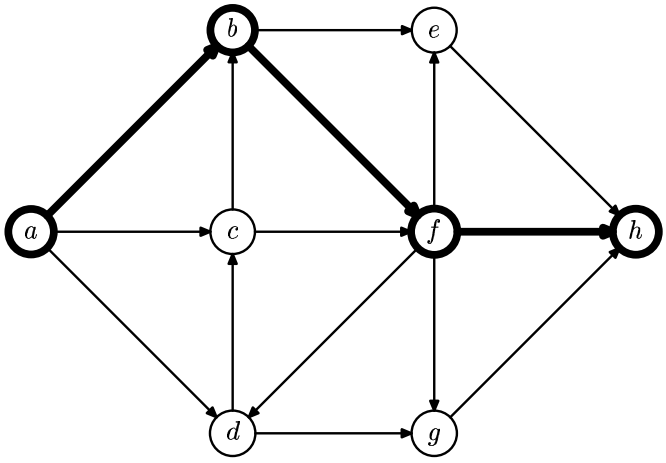

Figura 2.2: Representação do caminho $\langle a, (a,b), b, (b,f), f, (f,h), h\rangle$.

cia $\langle i_0, a_1, i_1, a_2, i_2, \ldots, a_{k-1}, i_{k-1}, a_k, i_0\rangle$, onde $k \geq 1$, $i_0, \ldots, i_{k-1}$ são vértices dois-a-dois distintos, $a_1, \ldots, a_{k-1}$ são arcos tais que $a_j = (i_{j-1}, i_j)$ para todo j entre 1 e $k-1$ e a_k é um arco tal que $a_k = (i_{k-1}, i_0)$. Os conjuntos de vértices e arcos de W são denotados respectivamente por N_W e A_W. A figura 2.3 mostra um exemplo de circuito.

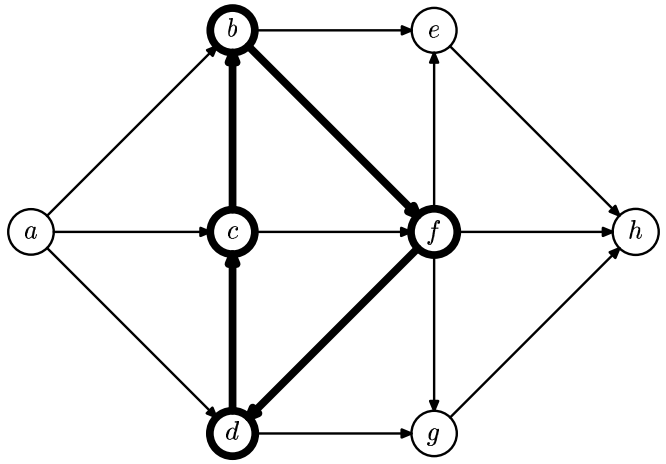

Figura 2.3: Representação do circuito $\langle b, (b,f), f, (f,d), d, (d,c), c, (c,b), b\rangle$.

Uma rede é dita **acíclica** se não possui circuitos.

Vamos apresentar agora alguns resultados sobre caminhos e distâncias. Para tanto, considere a seguinte notação: se i e j são vértices de uma rede G, então $d(i,j)$ é a distância entre i e j em G.

Lema 2.1. *Seja G uma rede, seja P um caminho entre os vértices i e w em G de comprimento r_1 e seja Q um caminho entre os vértices w e j em G de comprimento r_2. Então existe um caminho entre i e j em G, cujo comprimento não é superior a $(r_1 + r_2)$.*

Prova: Seja P o caminho $\langle i = i_0, a_1, i_1, \ldots, i_{r_1-1}, a_{r_1}, i_{r_1} = w\rangle$ e seja Q o caminho $\langle w = j_0, b_1, j_1, \ldots, j_{r_2-1}, b_{r_2}, j_{r_2} = j\rangle$.

Seja k o menor índice tal que $i_k = j_l$, para algum l, $0 \leq l \leq r_2$, e tal que $0 \leq k \leq r_1$. Tal índice está bem definido, uma vez que os caminhos possuem pelo menos o vértice w em comum. Podemos considerar, então, o caminho $P' = \langle i, a_1, \ldots, a_k, i_k = j_l, b_{l+1}, \ldots, b_{r_2}, j\rangle$. Pela escolha de k, é claro que os vértices de P' são todos distintos e, portanto, que P' é de fato um caminho entre i e j em G. Ademais, o comprimento de P' é $k + (r_2 - l) \leq r_1 + r_2$. □

Lema 2.2. *Seja G uma rede, seja s um vértice e seja $a = (i, j)$ um arco de G. Então $d(s, j) \leq d(s, i) + 1$.*

Prova: Se $d(s, i) = \infty$, então o resultado vale diretamente. Caso contrário, então, pela definição de distância, existe um caminho P entre s e i em G de comprimento $d(s, i)$. Além disso, $Q = \langle i, a, j\rangle$ é um caminho em G entre i e j de comprimento 1. Portanto, podemos aplicar o Lema 2.1 sobre os caminhos P e Q e concluir que existe um caminho entre s e j em G de comprimento não superior a $d(s, i) + 1$. Disso segue o resultado. □

Lema 2.3. *Seja G uma rede e seja $P = \langle i_0, a_1, i_1, \ldots, a_k, i_k\rangle$ um caminho mínimo entre i_0 e i_k em G. Então o caminho $P_l = \langle i_0, a_1, i_1, \ldots, a_l, i_l\rangle$, contido em P, para qualquer $0 \leq l \leq k$, é um caminho mínimo entre i_0 e i_l em G.*

Prova: Suponha por contradição que exista um caminho Q_l entre i_0 e i_l em G e de comprimento l', com $l' < l$. Notemos, então, que o caminho $\langle i_l, a_{l+1}, \ldots, a_k, i_k\rangle$, contido em P, é um caminho entre i_l e i_k de comprimento $k - l$. Assim, pelo Lema 2.1, existe um caminho entre i_0 e i_k de comprimento não superior a $l' + (k - l) < k$. Mas isso é uma contradição, pois P é um caminho de comprimento mínimo entre i_0 e i_k em G e seu comprimento é k. Logo, P_l é um caminho mínimo entre i_0 e i_l em G. □

Lema 2.4. *Seja G uma rede, seja P um caminho mínimo com origem s em G e seja $a = (i, j)$ um arco de P. Então $d(s, j) = d(s, i) + 1$.*

Prova: Seja $P = \langle s = i_0, a_1, i_1, \ldots, a_k, i, a, j, a_{k+2}, \ldots, a_l, i_l = t\rangle$. Denote por P_i o caminho $\langle s, a_1, i_1, \ldots, a_k, i\rangle$, contido em P, e por P_j o caminho $\langle s, a_1, i_1, \ldots, a_k, i, a, j\rangle$, também contido em P.

Pelo Lema 2.3, P_i é um caminho de comprimento mínimo entre s e i e P_j é um caminho de comprimento mínimo entre s e j. Ou seja, $d(s, i) = k$ e $d(s, j) = k + 1$. Disso segue o resultado. □

2.3 Separadores e cortes

Um **separador** de uma rede é um subconjunto de seus vértices.

Se S é um separador de uma rede G, o conjunto de todos os arcos dessa rede com uma ponta em S e a outra fora de S é um **corte**, denotado por $\delta_G(S)$. Mais especificamente, o conjunto de todos os arcos com ponta inicial em S e ponta final fora de S é denotado por $\delta_G^-(S)$ e o conjunto de todos os arcos com ponta inicial fora de S e ponta final em S é denotado por $\delta_G^+(S)$. Diz-se também que os arcos de $\delta_G^-(S)$ **saem** de S e que os arcos de $\delta_G^+(S)$ **entram** em S. Se a rede está implícita pelo contexto, pode-se denotar tais conjuntos por $\delta(S)$, $\delta^+(S)$ e $\delta^-(S)$.

Dados dois vértices s e t de uma rede, um separador S dessa rede que contém s e não contém t é um st-**separador** e o corte $\delta(S)$ é um st-**corte**. A figura 2.4 mostra um exemplo de separador e corte.

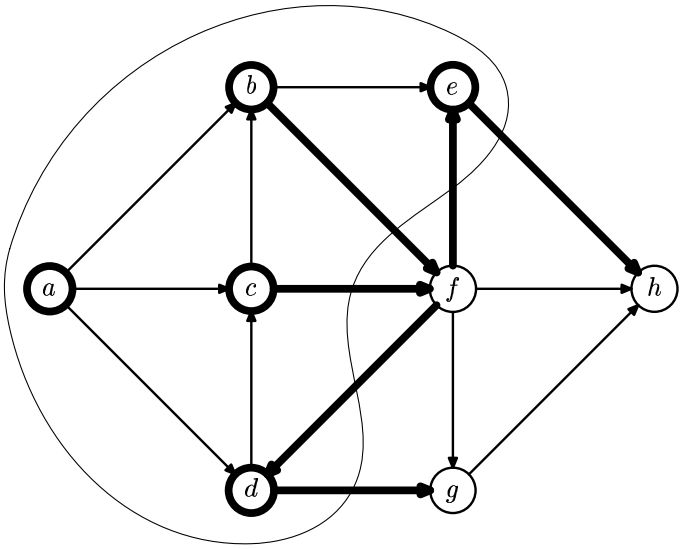

Figura 2.4: Separador $\{a, b, c, d, e\}$ e seu respectivo corte.

2.4 Capacidades, custos e demandas

A **capacidade** de uma rede (N, A) é uma função u de A em $\mathbb{Q}_{\geq}$. A maior capacidade de um arco, $\max_{a \in A} u_a$, é denotada por U. Uma rede é **capacitada** se existe uma função capacidade definida sobre seus arcos.

Se S é um separador de uma rede com capacidade u, a **capacidade** de S, denotada por $\text{cap}(S)$, é o valor $u(\delta^-(S))$.

O **custo** de uma rede (N, A) é uma função c de A em $\mathbb{Q}$. O maior valor absoluto do custo de um arco, $\max_{a \in A} |c_a|$, é denotado por C.

Se P é um caminho de uma rede com custo c, o **custo** de P, denotado por custo(P), é o valor $c(A_p)$. Analogamente, se W é um circuito de uma rede com custo c, o **custo** de W, denotado por custo(W), é o valor $c(A_W)$. Se W é um circuito de custo negativo, isto é, se custo(W) < 0, então W é dito um **circuito negativo**.

A **demanda** de uma rede (N, A) é uma função b de N em $\mathbb{Q}$. Um vértice i é um **produtor** se $b(i) < 0$ e é um **consumidor** se $b(i) > 0$.

2.5 Redes simétricas

Seja (N, A) uma rede e seja a um arco em A. O arco **irmão** de a, denotado por $\overline{a}$, é um arco definido da seguinte forma: se i é a ponta inicial e j é a ponta final de a, então j é a ponta inicial e i é a ponta final de $\overline{a}$.

O conjunto dos arcos irmãos de todos os arcos em A é denotado por $\overline{A}$ e a definição de arco irmão é estendida para $\overline{A}$ pela propriedade $\overline{\overline{a}} = a$. O conjunto dos irmãos de um conjunto de arcos B qualquer é denotado por $\overline{B}$.

A rede $(N, A \cup \overline{A})$ é chamada de rede **simétrica** de (N, A). A figura 2.5 mostra um exemplo de construção de uma rede simétrica.

2.6 Caminhos de custo mínimo

Seja G uma rede com custo c e seja P um caminho entre os vértices i e j em G. Dizemos que P é um caminho de **custo mínimo** entre i e j se não existe outro caminho entre i e j com custo menor do que custo(P).

Caminhos de custo mínimo são uma ferramenta essencial para a obtenção de resultados fundamentais na teoria de fluxos em redes. Esta seção será dedicada ao estudo de algumas de suas propriedades.

Vamos considerar a seguinte notação: dados dois vértices i e j de uma rede G, o valor $c(i, j)$ é o custo de um caminho de custo mínimo entre i e j em G. Se não existe caminho entre i e j em G, definimos que $c(i, j) = \infty$.

Além disso, se $P = \langle i, a_1, \ldots, a_k, w\rangle$ e $Q = \langle w, b_1, \ldots, b_q, j\rangle$ são caminhos em G, então PQ denota a seqüência $\langle i, a_1, \ldots, a_k, w, b_1, \ldots, b_q, j\rangle$, resultante da concatenação dos caminhos P e Q. Se o único vértice comum a P e Q é o vértice w, então PQ é um caminho entre i e j em G. Ademais, se $i = j$ e w é o único vértice comum aos caminhos, então PQ é um circuito em G. Note que, em ambos os casos, custo(PQ) = custo(P) + custo(Q).

Lema 2.5. *Seja G uma rede com custo c, seja s um vértice e seja $a = (i, j)$ um arco de G. Suponha que G não possua circuitos negativos. Então $c(s, i) + c_a \geq c(s, j)$.*

Prova: Suponha por contradiçao que $c(s, i) + c_a < c(s, j)$. Se $c(s, i) = \infty$, temos a contradição de que $\infty + c_a < c(s, j)$. Vamos, portanto, assumir,

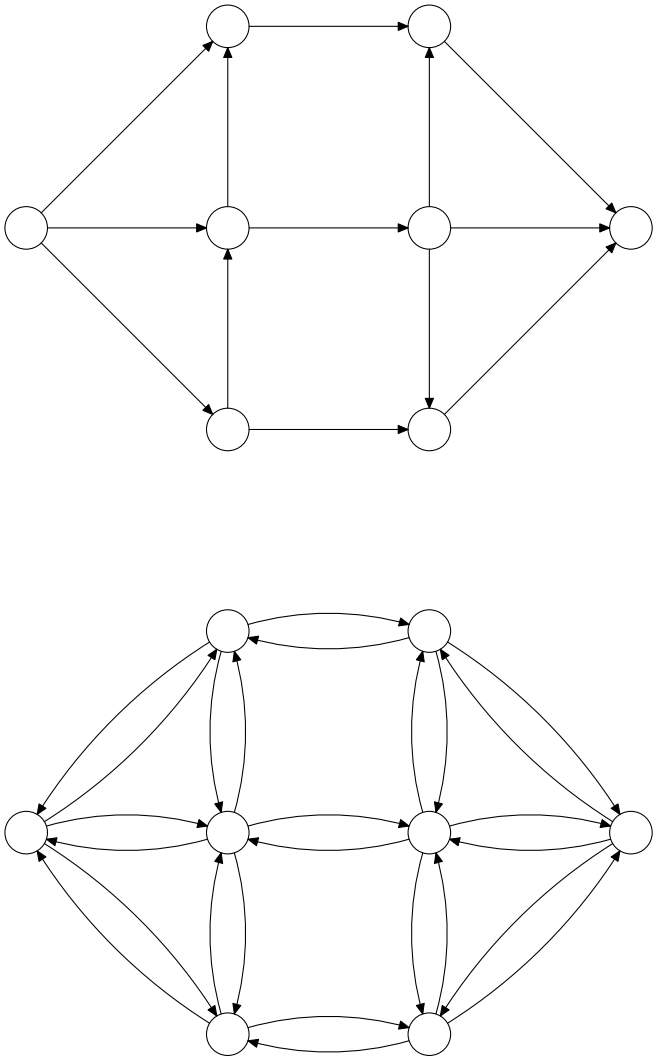

Figura 2.5: Rede original e sua respectiva rede simétrica.

sem perda de generalidade, que $c(s,i) \neq \infty$. Neste caso, existe um caminho $P = \langle s, a_1, i_1, \ldots, a_k, i\rangle$ cujo custo é $c(s,i)$.

Primeiramente, suponha que j seja um vértice de P. Neste caso, temos que P é da forma $\langle s, a_1, i_1, \ldots, a_l, j, a_{l+1}, \ldots, a_k, i\rangle$. Denote por P_1 o caminho $\langle s, a_1, i_1, \ldots, a_l, j\rangle$ e por P_2 o caminho $\langle j, a_{l+1}, \ldots, a_k, i\rangle$. Temos que $c(s,i) = \text{custo}(P) = \text{custo}(P_1) + \text{custo}(P_2)$. Ademais, como P_1 é um caminho entre s e j, então $c(s,j) \leq \text{custo}(P_1)$. Assim, vale que $c(s,i) + c_a = \text{custo}(P) + c_a = \text{custo}(P_1) + \text{custo}(P_2) + c_a < c(s,j) \leq \text{custo}(P_1)$, de onde concluímos que $\text{custo}(P_2) + c_a < 0$. Notemos, então, que $\text{custo}(P_2) + c_a$ é o custo do circuito W em G definido por $\langle j, a_{l+1}, \ldots, a_k, i, a, j\rangle$. Isto é, W é formado pela adição do arco a ao caminho P_2. Mas então W é um circuito negativo na rede G, o que não pode ocorrer por hipótese. Chegamos a uma contradição. Assim, vale que $c(s,i) + c_a \geq c(s,j)$.

Suponha agora que j não seja um vértice de P. Neste caso, o caminho definido por $\langle s, a_1, i_1, \ldots, a_k, i, a, j\rangle$ é um caminho entre s e j em G cujo custo é $c(s,i) + c_a$. Mas isso é uma contradição. De fato, se $c(s,j) = \infty$, então não deveriam existir caminhos entre s e j em G. Por outro lado, se $c(s,j) \neq \infty$, basta lembrarmos que, por hipótese, $c(s,i) + c_a < c(s,j)$ e não deveriam existir caminhos em G com custo inferior a $c(s,j)$. Portanto, concluímos que $c(s,i) + c_a \geq c(s,j)$. □

Lema 2.6. *Seja G uma rede, seja P um caminho entre os vértices i e j em G e seja Q um caminho entre os vértices j e i em G, com $i \neq j$. Suponha que* $\text{custo}(P) + \text{custo}(Q) < 0$. *Então existe um circuito negativo em G.*

Prova: Vamos fazer uma prova por indução no comprimento do caminho P. Suponha inicialmente que $P = \langle i, a, j\rangle$. Neste caso, temos que PQ é claramente um circuito e o resultado segue diretamente.

Suponha agora que o comprimento de P seja maior do que 1. Se os únicos vértices comuns a P e Q são os vértices i e j, então PQ é um circuito e o resultado também vale. Suponha, então, que P e Q tenham pelo menos um vértice em comum além dos vértices i e j.

Denote por $\langle i = i_0, \ldots, i_p = j\rangle$ o caminho P e por $\langle j = j_0, \ldots, j_q = i\rangle$, o caminho Q. Seja k o menor índice tal que $i_k = j_t$, para algum t, com $0 < t < q$, e tal que $0 < k < p$. O vértice i_k é o primeiro vértice diferente de i em P que também ocorre em Q. Denote ainda por P_1 o caminho $\langle i = i_0, \ldots, i_k\rangle$, por P_2 o caminho $\langle i_k, \ldots, i_p = j\rangle$, por Q_1 o caminho $\langle j = j_0, \ldots, j_t = i_k\rangle$ e por Q_2 o caminho $\langle i_k = j_t, \ldots, j_q = i\rangle$, de forma que $P = P_1P_2$ e $Q = Q_1Q_2$.

Notemos, então, que, pela escolha do índice k, P_1Q_2 é um circuito em G. Se $\text{custo}(P_1Q_2) = \text{custo}(P_1) + \text{custo}(P_2) < 0$, então P_1Q_2 é um circuito negativo e o resultado vale. Caso contrário, devemos ter $\text{custo}(P_2) + \text{custo}(Q_1) < 0$, uma vez que $\text{custo}(P) + \text{custo}(Q) < 0$.

Observemos, então, que P_2 é um caminho entre i_k e j em G e que Q_1 é um caminho entre j e i_k em G. Ademais, o comprimento de P_2 é estritamente

menor do que o comprimento de P, uma vez que $k > 0$. Assim, podemos aplicar a hipótese de indução sobre P_2 e Q_1 e concluir que a rede G possui um circuito negativo, o que conclui a indução. □

Lema 2.7. *Seja G uma rede e seja $P = \langle i_0, a_1, i_1, \ldots, a_k, i_k \rangle$ um caminho de custo mínimo entre i_0 e i_k em G. Suponha que G não possua circuitos negativos. Então o caminho $P_l = \langle i_0, a_1, i_1, \ldots, a_l, i_l \rangle$, contido em P, para qualquer $0 \leq l \leq k$, é um caminho de custo mínimo entre i_0 e i_l em G.*

Prova: Seja P o caminho P_1P_2, em que P_1 é o caminho $\langle i_0, a_1, i_1, \ldots, a_l, i_l \rangle$ e P_2 é o caminho $\langle i_l, \ldots, a_k, i_k \rangle$. Suponha por contradição que P_1 não seja um caminho de custo mínimo entre i_0 e i_l. Seja Q um caminho de custo mínimo entre i_0 e i_l em G. Então vale que $\text{custo}(Q) < \text{custo}(P_1)$.

Se Q e P_2 não possuem vértices em comum além do vértice i_l, então o caminho QP_2 é um caminho entre i_0 e i_k tal que $\text{custo}(QP_2) = \text{custo}(Q) + \text{custo}(P_2) < \text{custo}(P_1) + \text{custo}(P_2) = \text{custo}(P)$. Mas isso é uma contradição com a minimalidade do custo do caminho P. Concluímos, portanto, que Q e P_2 possuem pelo menos um vértice em comum diferente de i_l.

Denote por $\langle i_0 = j_0, b_1, j_1, \ldots, b_q, j_q = i_l \rangle$ o caminho Q. Seja r o menor índice tal que $j_r = i_t$, para algum t, com $l < t \leq k$, e tal que $0 < r < q$. O vértice j_r é o primeiro vértice em Q que coincide com algum vértice em P_2.

Denote por Q_1 o caminho $\langle i_0 = j_0, b_1, j_1, \ldots, b_r, j_r = i_t \rangle$ e por Q_2 o caminho $\langle i_t = j_r, b_{r+1}, \ldots, b_q, j_q = i_l \rangle$, de forma que $Q = Q_1Q_2$. Denote também por P_2' o caminho $\langle i_l, \ldots, a_t, i_t \rangle$ e por P_2'' o caminho $\langle i_t, \ldots, a_k, i_k \rangle$, de forma que $P_2 = P_2'P_2''$.

Notemos, então, que o caminho Q_1P_2'' é um caminho entre i_0 e i_k em G. Portanto, devemos ter $\text{custo}(Q_1P_2'') \geq \text{custo}(P) = \text{custo}(P_1P_2'P_2'')$, pois P é um caminho de custo mínimo entre i_0 e i_k em G. Ou seja, devemos ter $\text{custo}(Q_1) \geq \text{custo}(P_1P_2') = \text{custo}(P_1) + \text{custo}(P_2')$.

Por outro lado, temos por hipótese que $\text{custo}(Q) = \text{custo}(Q_1Q_2) = \text{custo}(Q_1) + \text{custo}(Q_2) < \text{custo}(P_1)$. Concluímos, assim, que $\text{custo}(P_1) + \text{custo}(P_2') + \text{custo}(Q_2) \leq \text{custo}(Q_1) + \text{custo}(P_2) < \text{custo}(P_1)$. Ou seja, $\text{custo}(P_2') + \text{custo}(Q_2)$ é negativo.

Observemos agora que P_2' é um caminho entre i_l e i_t em G e que Q_2 é um caminho entre i_t e i_l em G. Além disso, temos que $i_l \neq i_t$. Assim, podemos aplicar o Lema 2.6 e concluir que a rede G possui um circuito negativo. Mas isso é um absurdo, pois G não contém circuitos negativos por hipótese.

Logo, P_1 é um caminho de custo mínimo entre i_0 e i_l em G. □

Lema 2.8. *Seja G uma rede com custo c, seja P um caminho de custo mínimo em G cuja origem é s e seja $a = (i, j)$ um arco de P. Suponha que G não possua circuitos negativos. Então $c(s, i) + c_a = c(s, j)$.*

Prova: Seja P o caminho $\langle s = i_0, a_1, i_1, \ldots, a_k, i, a, j, a_{k+2}, \ldots, i_l \rangle$. Denote por P_i o caminho $\langle s, a_1, i_1, \ldots, a_k, i \rangle$, contido em P, e por P_j o caminho $\langle s, a_1, i_1, \ldots, a_k, i, a, j \rangle$, também contido em P.

Note que, pela definição de custo de um caminho, vale que $\text{custo}(P_j) = \text{custo}(P_i) + c_a$. Ademais, pelo Lema 2.7, temos que $c(s,i) = \text{custo}(P_i)$ e que $c(s,j) = \text{custo}(P_j)$. Disso segue o resultado. □

Capítulo 3

Fluxos

Este capítulo introduz formalmente o conceito de fluxo e conceitos relacionados. Também são apresentadas algumas propriedades gerais de fluxos e resultados referentes a decomposições e a redes residuais.

3.1 Fluxos e excessos

Um **fluxo** em uma rede G com capacidade u é uma função x de A_G em $\mathbb{Q}_{\geq}$ que **respeita** u, isto é, tal que $x_a \leq u_a$, para cada a em A_G. A figura 3.1 mostra um exemplo de fluxo sobre uma rede.

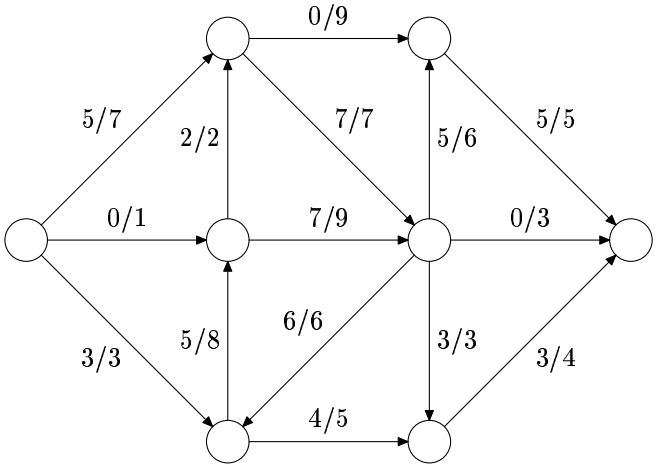

Figura 3.1: Representação de um fluxo arbitrário em uma rede capacitada. Para cada um dos arcos, o primeiro número representa o valor do fluxo e o segundo número representa o valor da capacidade.

Se a rede G também tem associada a ela uma função de custos c, o **custo** de x, denotado por custo(x), é o valor $\sum_{a \in A_G} c_a x_a$.

O valor $x(\delta^+(i))$ é o fluxo que **entra** no vértice i e, analogamente, o valor $x(\delta^-(i))$ é o fluxo que **sai** do vértice i. A diferença entre esses dois valores define o **excesso** de i em relação a x, denotado por $e_x(i)$:

$$e_x(i) = x(\delta^+(i)) - x(\delta^-(i)).$$

Pode-se denotar o excesso de i por $e(i)$ quando o fluxo está implícito.

3.2 *st*-fluxos

Dados uma rede capacitada G e dois vértices s e t dessa rede, um ***st*-fluxo** em G é um fluxo x que satisfaz a seguinte propriedade, conhecida como *lei da conservação do fluxo* ou *lei de Kirchhoff*:

$$e(i) = 0 \text{ para todo } i \text{ em } N_G \setminus \{s, t\}.$$

Os vértices s e t são chamados respectivamente de **fonte** (*source*) e **sorvedouro** (*sink*) do st-fluxo. O valor $e_x(t)$, que denotaremos por $\text{val}(x)$, é a **intensidade** de x. A figura 3.2 mostra um exemplo de st-fluxo.

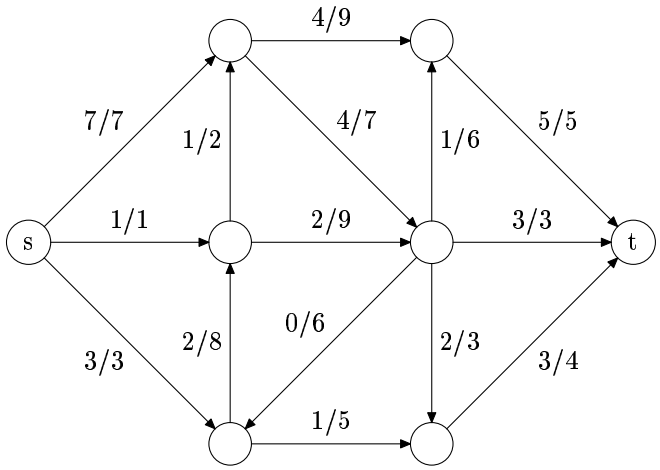

Figura 3.2: st-fluxo de intensidade 11.

3.3 Fluxos viáveis

Dada uma rede capacitada G com demanda b e um fluxo x sobre essa rede, o **desequilíbrio** de um vértice i em relação a x é a diferença $e_x(i) - b(i)$. Diz-se que x é um **fluxo viável** em G se vale a propriedade de que todos os vértices tem desequilíbrio nulo:

$$e_x(i) = b(i) \text{ para todo vértice } i \text{ em } N_G.$$

A figura 3.3 mostra um exemplo de fluxo viável.

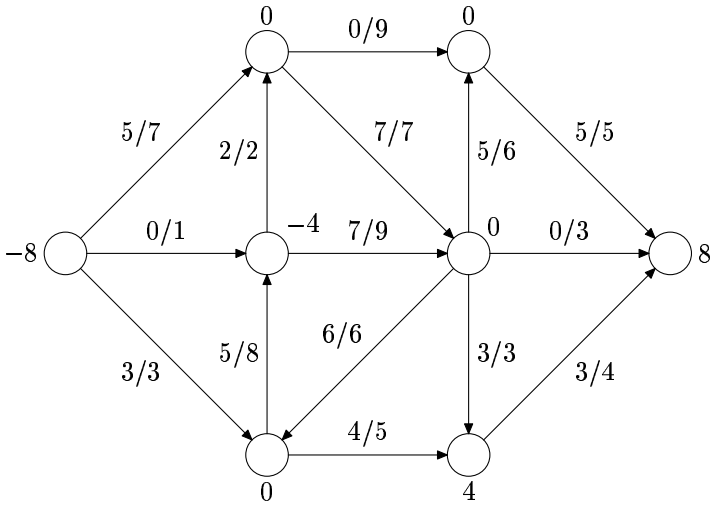

Figura 3.3: Exemplo de fluxo viável. Os valores próximos aos vértices representam suas demandas, que estão satisfeitas pelo fluxo.

3.4 Fluxos-caminho e fluxos-circuito

Um fluxo x em uma rede capacitada G é um **fluxo-caminho** se existe um caminho P em G e um racional positivo p tais que $x_a = p$ se a é um arco de P e $x_a = 0$ se a não é arco de P. O racional p é o **valor** do fluxo-caminho. Os conceitos de **extremos**, **origem** e **destino** relacionados a P são estendidos para o fluxo x nesse caso.

Analogamente, um fluxo x em uma rede capacitada G é um **fluxo-circuito** se existe um circuito W em G e um racional positivo w tais que $x_a = w$ se a é um arco de W e $x_a = 0$ se a não é arco de W. O racional w é o **valor** do fluxo-circuito.

Um st-fluxo x em uma rede G é chamado de st-fluxo **elementar** se x é um fluxo-caminho em G cuja origem é s e cujo destino é t. Note que se o st-fluxo elementar x é um fluxo-caminho de valor p, então $\text{val}(x) = p$.

3.5 Redes residuais

Seja $G = (N, A)$ uma rede capacitada e seja x um fluxo em G. A **capacidade residual** de G em relação a x é uma função r de $A \cup \overline{A}$ em $\mathbb{Q}_{\geq}$ definida da seguinte maneira:

$$r_a = \begin{cases} u_a - x_a & \text{se } a \in A, \\ x_{\overline{a}} & \text{se } a \in \overline{A}. \end{cases}$$

Seja A' o conjunto dos arcos de $A \cup \overline{A}$ com capacidade residual positiva. A **rede residual** de G em relação a x é a rede $G(x) = (N, A')$. Uma rede

residual é sempre capacitada e sua capacidade é definida pela capacidade residual correspondente. A figura 3.4 mostra um exemplo.

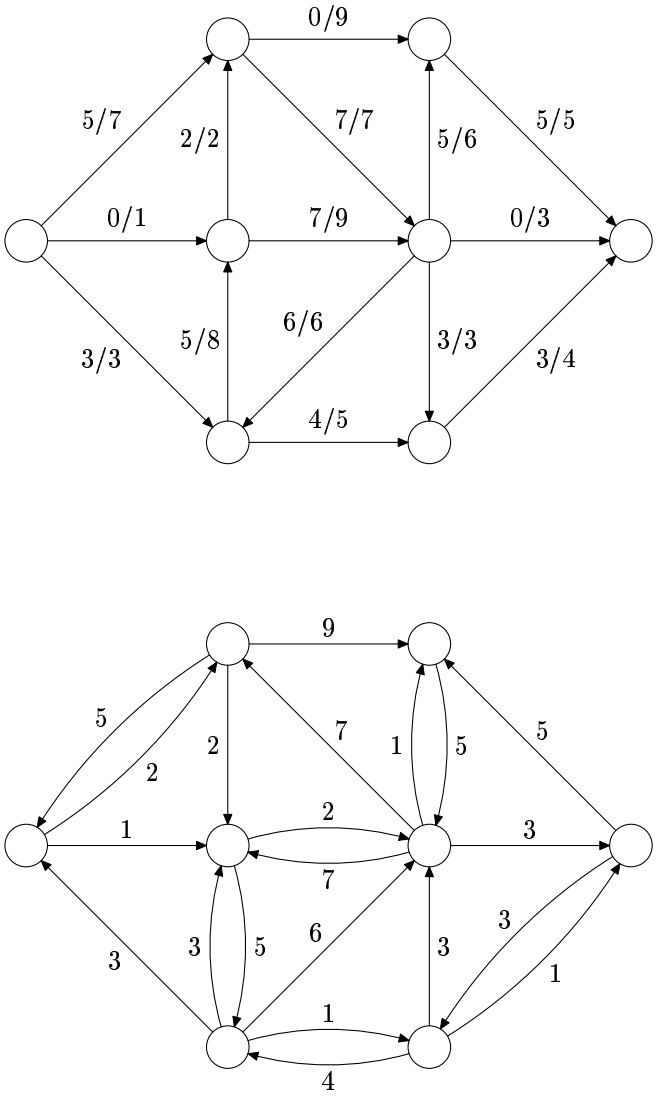

Figura 3.4: Fluxo em uma rede e a respectiva rede residual. Os números nos arcos da rede residual representam as capacidades residuais.

Se existe um custo c associado à rede G, então o custo c' da rede residual $G(x)$ é definido da seguinte maneira:

$$c'_a = \begin{cases} c_a & \text{se } a \in A, \\ -c_{\overline{a}} & \text{se } a \in \overline{A}. \end{cases}$$

Se S é um separador de G, a **capacidade residual** de S em relação a x,

denotada por $\text{res}_x(S)$, é o valor $r(\delta^-_{G(x)}(S))$. Se o fluxo está implícito pelo contexto, tal valor pode ser denotado simplesmente por $\text{res}(S)$.

3.6 Propriedades de fluxos

Aprentaremos aqui alguns resultados gerais sobre fluxos.

Lema 3.1. *Seja x um fluxo em uma rede G. Então,* $\sum_{i \in N_G} e(i) = 0$.

Prova: A validade da equação segue do fato de que, pela definição de excesso, o fluxo de cada arco (i, j) aparece exatamente duas vezes na soma: uma com sinal positivo, quando o excesso de j é contabilizado; e outra com sinal negativo, quando o excesso de i é contabilizado. □

Corolário 3.2. *Seja x um st-fluxo em uma rede G. Então,* $\text{val}(x) = -e(s)$.

Prova: Pelo Lema 3.1, temos que $\sum_{i \in N_G} e(i) = 0$. Ademais, como x é um st-fluxo, então $e(i) = 0$ para todo vértice i em $N_G \setminus \{s, t\}$. Assim, segue que $\text{val}(x) = e(t) = -\sum_{i \in N_G \setminus \{t\}} e(i) = -e(s)$. □

Lema 3.3. *Seja x um st-fluxo em uma rede (N, A). Então,* $\text{val}(x) = x(\delta^-(S)) - x(\delta^+(S))$, *para qualquer st-separador S de (N, A).*

Prova: Pelo Lema 3.2, sabemos que:

$$\text{val}(x) = -e(s) = x(\delta^-(s)) - x(\delta^+(s)).$$

Ademais, pela *lei de Kirchhoff*, temos que:

$$\sum_{i \in S \setminus \{s\}} e(i) = \sum_{i \in S \setminus \{s\}} (x(\delta^+(i)) - x(\delta^-(i))) = 0.$$

Portanto,

$$\begin{aligned}
\text{val}(x) &= x(\delta^-(s)) - x(\delta^+(s)) \\
&= x(\delta^-(s)) - x(\delta^+(s)) - \sum_{i \in S \setminus \{s\}} e(i) \\
&= x(\delta^-(s)) - x(\delta^+(s)) + \sum_{i \in S \setminus \{s\}} (x(\delta^-(i)) - x(\delta^+(i))) \\
&= \sum_{i \in S} (x(\delta^-(i)) - x(\delta^+(i))).
\end{aligned}$$

Considere um arco (i, j) na rede. Se i e j pertencem ambos a S, então o fluxo nesse arco aparece exatamente duas vezes na soma, uma com sinal positivo e outra com sinal negativo, já que o arco está em $\delta^-(i)$ e em $\delta^+(j)$. Portanto, o fluxo referente a um arco a só é contabilizado se apenas uma de

suas pontas está em S. Mais ainda, x_a aparece com sinal positivo, somente se a está em $\delta^-(S)$, e x_a aparece com sinal negativo, somente se a está em $\delta^+(S)$. Assim,

$$\begin{aligned} \text{val}(x) &= \sum_{i \in S} (x(\delta^-(i)) - x(\delta^+(i))) \\ &= x(\delta^-(S)) - x(\delta^+(S)). \end{aligned}$$

□

3.7 Decomposição de fluxos

Dado um fluxo x em uma rede capacitada G, uma **decomposição** de x é um conjunto X de fluxos-caminho e fluxos-circuito em G que satisfaz $x_a = \sum_{x' \in X} x'_a$ para todo arco a de G. A figura 3.5 mostra um exemplo.

Vamos apresentar alguns resultados para demonstrar que um fluxo x qualquer admite uma decomposição que satisfaz certas propriedades.

Para as demonstrações a seguir, considere a seguinte notação. Seja P um caminho entre dois vértices i e j tal que $e(i) < 0$, $e(j) > 0$ e $x_a > 0$ para todo arco a de P. Denote por x^P o fluxo-caminho assim definido:

$$x^P_a = \begin{cases} \min\{-e(i), e(j), \min\{x_a : a \text{ é arco de } P\}\} & \text{se } a \text{ é um arco de } P; \\ 0 & \text{se } a \text{ não é arco de } P. \end{cases}$$

As propriedades de P garantem que x^P é de fato um fluxo-caminho. Denote por $x - x^P$ o fluxo abaixo:

$$(x - x^P)_a = \begin{cases} x_a - x^P_a & \text{se } a \text{ é um arco de } P; \\ x_a & \text{se } a \text{ não é arco de } P. \end{cases}$$

A definição de x^P garante que $x - x^P$ é de fato um fluxo. Analogamente, seja W um circuito na rede tal que $x_a > 0$ para todo arco a de W. Denote por x^W o fluxo-circuito assim definido:

$$x^W_a = \begin{cases} \min\{x_a : a \text{ é arco de } P\} & \text{se } a \text{ é um arco de } W; \\ 0 & \text{se } a \text{ não é arco de } W. \end{cases}$$

As propriedades de W garantem que x^W é de fato um fluxo-circuito. Denote por $x - x^W$ o fluxo abaixo:

$$(x - x^W)_a = \begin{cases} x_a - x^W_a & \text{se } a \text{ é arco de } W; \\ x_a & \text{se } a \text{ não é arco de } W. \end{cases}$$

A definição de x^W garante que $x - x^W$ é de fato um fluxo.

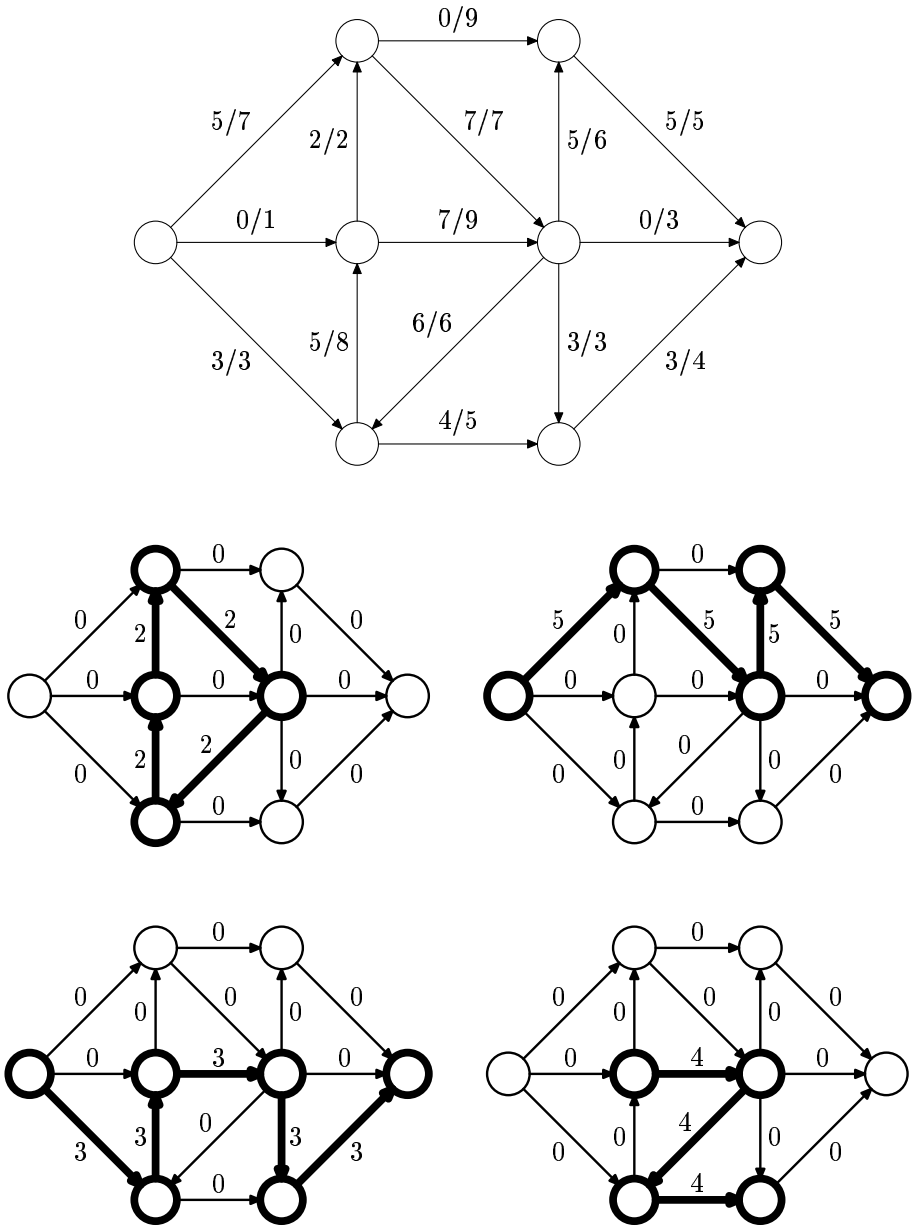

Figura 3.5: Exemplo de decomposição de um fluxo.

Lema 3.4. *Seja x um fluxo em uma rede e seja W um circuito sobre essa mesma rede tal que $x_a > 0$ para todo arco a de W. Então, $e_x(i) = e_{x-x^W}(i)$ para qualquer vértice i da rede.*

Prova: Se i não é um vértice de W, então i não é ponta de nenhum arco de W. Logo, pela definição de $x - x^W$, temos que $(x - x^W)_a = x_a$ para todo arco em $\delta(i)$ e, portanto, temos que $e_x(i) = e_{x-x^W}(i)$.

Se i é um vértice de W, então, pela definição de circuito, existe um e apenas um arco a de W tal que a está em $\delta^+(i)$ e existe um e apenas um arco b de W tal que b está em $\delta^-(i)$. Como $(x - x^W)_a = x_a - x_a^W$ e $(x - x^W)_b = x_b - x_b^W$, podemos concluir que $x(\delta^+(i)) - (x - x^W)(\delta^+(i)) = x(\delta^-(i)) - (x - x^W)(\delta^-(i))$. Portanto, temos que $e_x(i) = e_{x-x^W}(i)$. □

Lema 3.5. *Seja x um fluxo em uma rede G. Todo vértice i de G com $e(i) < 0$ é origem de um caminho P com destino j, tal que $x_a > 0$ para todo arco a de P e $e(j) > 0$. Todo vértice i de G com $e(i) > 0$ é destino de um caminho P com origem j, tal que $x_a > 0$ para todo arco a de P e $e(i) < 0$.*

Prova: Vamos provar algoritmicamente a primeira afirmação do lema. O algoritmo descrito abaixo recebe uma rede, um fluxo x sobre essa rede e um vértice i com $e(i) < 0$ e devolve um caminho P entre i e j, tal que $x_a > 0$ para todo arco a de P, e $e(j) > 0$. Cada iteração mantém um fluxo y e um caminho P. Na primeira iteração, temos que $y = x$ e $P = \langle i \rangle$.

Caso 1: Existe um circuito W tal que $y_a > 0$ para todo arco a de W.
Comece nova iteração com $y - y^W$ no papel de y.

Caso 2: Não existe circuito W tal que $y_a > 0$ para todo arco a de W.
Seja $P = \langle i, a_1, \ldots, a_k, j \rangle$.

Caso 2A: Existe um arco $a = (j, l)$ tal que $y_a > 0$ e l não está em P.
Seja P' o caminho $\langle i, a_1, \ldots, a_k, j, a, l \rangle$.
Comece nova iteração com P' no papel de P.

Caso 2B: Não existe arco $a = (j, l)$ tal que $y_a > 0$ e l não está em P.
Devolva P e pare.

No Caso 1, a definição de $y - y^W$ garante que o circuito W tem pelo menos um arco a tal que $(y - y^W)_a = 0$. Como o número de arcos é finito, após um número finito de iterações o algoritmo passa a considerar somente o Caso 2. Ademais, pelo Lema 3.4, temos também que se a afirmação é válida para o fluxo $y - y^W$, então ela também é válida para o fluxo y.

No Caso 2A, temos que P' tem mais vértices do que P. Como o conjunto de vértices é finito, isso implica que o algoritmo pára, pois após um número finito de iterações temos que no máximo todos os vértices da rede estão em P. Resta apenas provar que, na última iteração, $e(j) > 0$.

Suponha por contradição que $e(j) \leq 0$. Pela definição de excesso, isso implica que existe um arco $a = (j, l)$ tal que $y_a > 0$. Se l não está em P, as condições do Caso 2A estão satisfeitas. Se l está em P, então temos um circuito W tal que $y_a > 0$ para todo arco a de W e as condições do Caso 1 estão satisfeitas. Mas isso é um absurdo, pois na última iteração somente as condições do Caso 2B devem estar satisfeitas. Portanto, $e(j) > 0$.

A prova da segunda afirmação é análoga, bastando descrever um algoritmo similar que encontra o caminho desejado a partir de seu destino. □

Corolário 3.6. *Seja x um fluxo sobre uma rede G. Existe um vértice i de G com $e(i) < 0$ se, e somente se, existe um vértice j de G com $e(j) > 0$.*

Prova: Segue da existência dos caminhos descritos no Lema 3.5. □

Lema 3.7. *Seja x um fluxo em uma rede G. Se existe um arco a de G tal que $x_a > 0$ e se $e(i) = 0$ para todo vértice i de G, então existe um circuito W tal que $x_a > 0$ para todo arco a de W.*

Prova: Vamos provar o lema algoritmicamente. O algoritmo abaixo recebe uma rede e um fluxo x sobre essa rede tal que $e(i) = 0$ para todo vértice i e $x_a > 0$ para algum arco a, e devolve um circuito W tal que $x_a > 0$ para todo arco a em W. Cada iteração mantém um caminho P. Na primeira iteração, $P = \langle i \rangle$, onde i é a ponta inicial de um arco a tal que $x_a > 0$.

Seja j o destino do caminho P.

Caso 1: Existe um arco $a = (j, l)$ tal que $x_a > 0$ e l está em P.
Seja $P = \langle i, a_1, \ldots, a_q, l, a_{q+1}, \ldots, a_k, j \rangle$.
Seja W o circuito $\langle l, a_{q+1}, \ldots, a_k, j, a, l \rangle$.
Devolva W e pare.

Caso 2: Existe um arco $a = (j, l)$ tal que $x_a > 0$ e l não está em P.
Seja $P = \langle i, a_1, \ldots, a_k, j \rangle$.
Seja P' o caminho $\langle i, a_1, \ldots, a_k, j, a, l \rangle$.
Comece nova iteração com P' no papel de P.

Pela definição de x, o vértice i está bem definido antes do início da primeira iteração. Portanto, nesse momento, quando $j = i$, existe um arco $a = (j, l)$ tal que $x_a > 0$. No início das demais iterações, sabemos que existe um arco b em $\delta^+(j)$, o arco predecessor de j em P, tal que $x_b > 0$. Como $e(j) = 0$ pela definição de x, então $x(\delta^-(j)) > 0$ e temos que a existência de um arco $a = (j, l)$ tal que $x_a > 0$ também é garantida.

Sabemos que o conjunto de vértices é finito e que P' tem mais vértices que P. Logo, o algoritmo pára pois, após um número finito de iterações, no máximo todos os vértices estão em P e considera-se no Caso 1. □

Teorema 3.8 (decomposição do fluxo). *Seja x um fluxo em uma rede G. Então é possível obter uma decomposição X do fluxo x, não necessariamente única, que satisfaz as seguintes propriedades abaixo:*
(i) Para todo fluxo-caminho x' que está em X, a origem i de x' é tal que $e_x(i) < 0$ e o destino j de x' é tal que $e_x(j) > 0$.
(ii) Há no máximo $n + m$ fluxos e no máximo m fluxos-circuito em X.

Prova: O algoritmo abaixo recebe uma rede e um fluxo x e devolve uma decomposição de x com as propriedades desejadas. Cada iteração mantém um fluxo y e um conjunto X de fluxos-caminho e fluxos-circuito sobre a mesma rede. Na primeira iteração, temos que $y = x$ e $X = \emptyset$.

Caso 1: $y_a = 0$ para todo arco a da rede.
Devolva X e pare.

Caso 2: Existe um arco a na rede tal que $y_a > 0$.

Caso 2A: $e(i) = 0$ para todo vértice i da rede.
Seja W um circuito tal que $y_a > 0$ para todo arco a de W.
Comece nova iteração com o fluxo $y - y^W$ no papel de y e com o conjunto $X \cup \{y^W\}$ no papel de X.

Caso 2B: $e(i) < 0$ para algum vértice i da rede.
Seja P um caminho com origem i e destino j tal que $e(i) < 0$, $e(j) > 0$ e $y_a > 0$ para todo arco a de P.
Comece nova iteração com o fluxo $y - y^P$ no papel de y e com o conjunto $X \cup \{y^P\}$ no papel de X.

No Caso 2A, a existência de W é garantida pelo Lema 3.7. No Caso 2B, a existência de P é garantida pelo Lema 3.5. O Corolário 3.6 garante que se não existe um vértice i tal que $e(i) < 0$, então não existe um vértice j tal que $e(j) > 0$, ou seja, todos os vértices tem excesso zero. Portanto, podemos verificar que o algoritmo de fato considera todos os casos possíveis.

Como na última iteração temos que $y_a = 0$ para todo arco a da rede, temos, pela definição de $y - y^W$ e $y - y^P$, que na última iteração vale que X satisfaz $x_a = \sum_{x' \in X} x'_a$ para todo arco a da rede. Portanto, podemos concluir que X é de fato uma decomposição de x.

Pela escolha dos caminhos no Caso 2B, a propriedade (i) do teorema é garantida. Ademais, em cada execução dos Casos 2A e 2B, ou o fluxo em um arco torna-se zero ou o excesso em um vértice torna-se zero. Portanto, o algoritmo pára após no máximo $n + m$ iterações, uma vez que o critério de parada é o fluxo ser zero em todos os arcos. Cada iteração adiciona no máximo um fluxo a X, o que implica que X contém no máximo $n + m$ fluxos. Como cada iteração do Caso 2A zera o fluxo em um arco, X contém no máximo m fluxos-circuito e a propriedade (ii) é garantida. □

Corolário 3.9. *Seja x um st-fluxo com intensidade positiva em uma rede. Então, é possível obter uma decomposição X de x, não necessariamente única, tal que existem no máximo m fluxos-caminho em X e tal que todos os fluxos-caminho em X são st-fluxos elementares. Ademais, a intensidade de x é a soma das intensidades desses st-fluxos elementares.*

Prova: Aplicando o mesmo algoritmo utilizado na prova do Teorema 3.8, sabemos que a decomposição X devolvida é tal que, dado um fluxo-caminho x' de X, a origem i de x' é tal que $e_x(i) < 0$ e o destino j de x' é tal que $e_x(j) > 0$. Como x tem intensidade positiva, temos que $e_x(t) > 0$. Pela definição de st-fluxo, s e t são os únicos vértices cujo excesso em relação a x não é necessariamente nulo. Portanto, pelo Corolário 3.6, s é o único vértice com excesso negativo e t é o único vértice com excesso positivo. Logo, todos os fluxos-caminho de X têm origem s e destino t. Ademais, como em cada execução do Caso 2B, ou um fluxo em um arco torna-se nulo ou o excesso em um vértice torna-se nulo, o Caso 2B é executado no máximo m vezes.

Pelo Lema 3.4, sabemos que a subtração de cada fluxo-circuito de X do fluxo original x não altera o excesso de nenhum vértice da rede e, portanto, não altera a intensidade desse fluxo. Assim, podemos observar que somente as ocorrências do Caso 2A reduzem a intensidade de x. Ademais, essa redução corresponde exatamente ao valor do fluxo-caminho obtido, o que implica no resultado. □

Corolário 3.10. *Seja x um fluxo em uma rede tal que $e_x(i) = 0$ para todo vértice i. Então, é possível obter uma decomposição X de x, não necessariamente única, tal que X não contém nenhum fluxo-caminho e contém no máximo m fluxos-circuito.*

Prova: Aplicando o mesmo algoritmo utilizado na prova do Teorema 3.8, podemos observar que o Lema 3.4 garante que o Caso 2B nunca ocorre ao longo de toda a execução do algoritmo. Portanto, temos que a decomposição X devolvida não contém nenhum fluxo-caminho e a propriedade (ii) do Teorema 3.8 garante que X contém no máximo m fluxos-circuito. □

Lema 3.11. *Seja X uma decomposição de um fluxo x em uma rede G com custo c. Então,* $\text{custo}(x) = \sum_{x' \in X} \text{custo}(x')$.

Prova: Sabemos que $\text{custo}(x) = \sum_{a \in A_G} c_a x_a$ e que $x_a = \sum_{x' \in X} x'_a$ para todo arco a em A_G. Logo, temos que:

$$\begin{aligned} \text{custo}(x) &= \sum_{a \in A_G} \left(c_a \sum_{x' \in X} x'_a \right) = \sum_{a \in A_G} \sum_{x' \in X} c_a x'_a \\ &= \sum_{x' \in X} \sum_{a \in A_G} c_a x'_a = \sum_{x' \in X} \text{custo}(x'). \end{aligned}$$

□

3.8 Fluxos em redes residuais

Sejam x e y dois fluxos em uma rede G. O **fluxo-diferença** entre y e x, denotado por $y - x$, é o seguinte fluxo definido sobre $G(x)$:

$$(y-x)_a = \begin{cases} y_a - x_a & \text{se } a \text{ está em } A_G \text{ e } y_a > x_a; \\ x_{\overline{a}} - y_{\overline{a}} & \text{se } a \text{ está em } \overline{A_G} \text{ e } y_{\overline{a}} < x_{\overline{a}}; \\ 0 & \text{caso contrário;} \end{cases}$$

As propriedades de $y - x$ garantem que ele é de fato um fluxo.

Seja x um fluxo sobre uma rede G e seja y um fluxo sobre a rede residual $G(x)$. O **fluxo obtido a partir de x e y**, denotado por $x + y$, é o fluxo sobre a rede G definido da seguinte forma:

$$(x+y)_a = x_a + y_a - y_{\overline{a}} \text{ para cada arco } a \text{ de } G$$

Para que $x+y$ seja uma função bem-definida, assumimos que $y_a = 0$ se a não está em $G(x)$ e $y_{\overline{a}} = 0$ se $\overline{a}$ não está em $G(x)$. Pelas propriedades de uma rede residual, é garantido que $x + y$ é de fato um fluxo.

Vamos agora apresentar algumas propriedades sobre fluxos em redes residuais que têm aplicação fundamental em capítulos posteriores.

Lema 3.12. *Sejam x e y dois fluxos em uma rede G. Então, $y = x+(y-x)$.*

Prova: Seja a um arco de G. Sabemos que $(x + (y - x))_a = x_a + (y - x)_a - (y - x)_{\overline{a}}$. Se $y_a > x_a$, temos que $x_a + (y - x)_a - (y - x)_{\overline{a}} = x_a + (y_a - x_a) + 0 = x_a + y_a - x_a = y_a$. Se $y_a < y_a$, temos que $x_a + (y - x)_a - (y - x)_{\overline{a}} = x_a + 0 - (x_a - y_a) = x_a - x_a + y_a = y_a$. Se $y_a = x_a$, temos que $x_a + (y - x)_a - (y - x)_{\overline{a}} = x_a + 0 - 0 = x_a = y_a$. Logo, podemos concluir que $(x + (y - x))_a = x_a + y_a - x_a = y_a$ e temos $(x + (y - x)) = y$. □

Lema 3.13. *Sejam x e y dois fluxos em uma rede G. Então o fluxo diferença $y - x$ é tal que $e_{y-x}(i) = e_y(i) - e_x(i)$ para todo vértice i.*

Prova: Sejam B_1 e B_2 subconjuntos de A_G tais que um arco a de A_G está em B_1 se, e somente se, $y_a > x_a$ e está em B_2 se, e somente se, $y_a < x_a$.

Dado um vértice i de $G(x)$, temos pela definição de fluxo-diferença que:

$$\begin{aligned} e_{y-x}(i) &= (y-x)(\delta^+_{G(x)}(i)) - (y-x)(\delta^-_{G(x)}(i)) \\ &= (y-x)((\delta^+_{G(x)}(i) \cap B_1) \cup (\delta^+_{G(x)}(i) \cap \overline{B_2})) - \\ &\quad (y-x)((\delta^-_{G(x)}(i) \cap B_1) \cup (\delta^-_{G(x)}(i) \cap \overline{B_2}) \\ &= (y-x)(\delta^+_{G(x)}(i) \cap B_1) + (y-x)(\delta^+_{G(x)}(i) \cap \overline{B_2}) - \\ &\quad (y-x)(\delta^-_{G(x)}(i) \cap B_1) - (y-x)(\delta^-_{G(x)}(i) \cap \overline{B_2}). \end{aligned}$$

Se a é um arco de $A_{G(x)} \cap B_1$, então o fluxo-diferença $(y-x)_a$ é dado por $y_a - x_a$. Portanto, temos que:

$$(y-x)(A_{G(x)} \cap B_1) = y(A_{G(x)} \cap B_1) - x(A_{G(x)} \cap B_1)$$

Por outro lado, se a é um arco de $A_{G(x)} \cap \overline{B_2}$, então o fluxo-diferença $(y-x)_a$ é dado por $x_{\overline{a}} - y_{\overline{a}}$. Logo, temos que:

$$(y-x)(A_{G(x)} \cap \overline{B_2}) = x(\overline{(A_{G(x)} \cap \overline{B_2})}) - y(\overline{(A_{G(x)} \cap \overline{B_2})})$$

Assim, podemos concluir que:

$$\begin{aligned}
e_{y-x}(i) &= y(\delta^+_{G(x)}(i) \cap B_1) - x(\delta^+_{G(x)}(i) \cap B_1) + \\
&\quad x(\overline{(\delta^+_{G(x)}(i) \cap \overline{B_2})}) - y(\overline{(\delta^+_{G(x)}(i) \cap \overline{B_2})}) - \\
&\quad y(\delta^-_{G(x)}(i) \cap B_1) + x(\delta^-_{G(x)}(i) \cap B_1) - \\
&\quad x(\overline{(\delta^-_{G(x)}(i) \cap \overline{B_2})}) + y(\overline{(\delta^-_{G(x)}(i) \cap \overline{B_2})}) \\
&= y(\delta^+_{G(x)}(i) \cap B_1) + y(\overline{(\delta^-_{G(x)}(i) \cap \overline{B_2})}) - \\
&\quad y(\delta^-_{G(x)}(i) \cap B_1) - y(\overline{(\delta^+_{G(x)}(i) \cap \overline{B_2})}) - \\
&\quad x(\delta^+_{G(x)}(i) \cap B_1) - x(\overline{(\delta^-_{G(x)}(i) \cap \overline{B_2})}) + \\
&\quad x(\delta^-_{G(x)}(i) \cap B_1) + x(\overline{(\delta^+_{G(x)}(i) \cap \overline{B_2})}).
\end{aligned}$$

Se a é um arco de B_1, então $x_a < y_a \leq u_a$ e portanto a está em $G(x)$. Se a é um arco de B_2, então $x_a > y_a \geq 0$ e portanto $\overline{a}$ está em $G(x)$. Logo,

$$\begin{aligned}
e_{y-x}(i) &= y(\delta^+_G(i) \cap B_1) + y(\delta^+_G(i) \cap B_2) - \\
&\quad y(\delta^-_G(i) \cap B_1) - y(\delta^-_G(i) \cap B_2) - \\
&\quad x(\delta^+_G(i) \cap B_1) - x(\delta^+_G(i) \cap B_2) + \\
&\quad x(\delta^-_G(i) \cap B_1) + x(\delta^-_G(i) \cap B_2) \\
&= y(\delta^+_G(i) \cap (B_1 \cup B_2)) - y(\delta^-_G(i) \cap (B_1 \cup B_2)) - \\
&\quad x(\delta^+_G(i) \cap (B_1 \cup B_2)) + x(\delta^-_G(i) \cap (B_1 \cup B_2)).
\end{aligned}$$

Agora, como sabemos que $y_a = x_a$, ou seja, que $y_a - x_a = 0$, para todo arco a que não está em $B_1 \cup B_2$, podemos concluir que:

$$\begin{aligned}
e_{y-x}(i) &= y(\delta^+_G(i)) - y(\delta^-_G(i)) - x(\delta^+_G(i)) + x(\delta^-_G(i)) \\
&= e_y(i) - e_x(i).
\end{aligned}$$

□

Corolário 3.14. *Sejam x e y dois st-fluxos em uma rede G. Então o fluxo diferença $y - x$ é um st-fluxo em $G(x)$ tal que* $\mathrm{val}(y-x) = \mathrm{val}(y) - \mathrm{val}(x)$.

Prova: Pelo Lema 3.13, temos que $e_{y-x}(i) = e_y(i) - e_x(i) = 0 - 0 = 0$ para todo vértice i diferente de s e t, garantindo a conservação do fluxo. Ademais, $\text{val}(y - x) = e_{y-x}(t) = e_y(t) - e_x(t) = \text{val}(y) - \text{val}(x)$. □

Corolário 3.15. *Sejam x e y dois fluxos viáveis em uma rede G com demanda b. Então o fluxo $y - x$ é tal que $e_{y-x}(i) = 0$ para todo vértice i.*

Prova: Pelo Lema 3.13 e como x e y são ambos fluxos viáveis, temos que $e_{y-x}(i) = e_y(i) - e_x(i) = b(i) - b(i) = 0$, para todo vértice i. □

Lema 3.16. *Sejam x e y dois fluxos em uma rede G com custo c. Então o fluxo diferença $y - x$ é tal que* $\text{custo}(y - x) = \text{custo}(y) - \text{custo}(x)$.

Prova: Sejam B_1 e B_2 subconjuntos de A_G tais que um arco a de A_G está em B_1 se, e somente se, $y_a > x_a$ e está em B_2 se, e somente se, $y_a < x_a$.

Seja c' o custo da rede residual $G(x)$. Por definição temos que:

$$\text{custo}(y - x) = \sum_{a \in B_1} c'_a(y_a - x_a) + \sum_{a \in B_2} c'_{\overline{a}}(x_a - y_a).$$

E, pela definição de custo na rede residual,

$$\begin{aligned}
\text{custo}(y - x) &= \sum_{a \in B_1} c_a(y_a - x_a) - \sum_{a \in B_2} c_a(x_a - y_a) \\
&= \sum_{a \in B_1} c_a y_a - \sum_{a \in B_1} c_a x_a - \sum_{a \in B_2} c_a x_a + \sum_{a \in B_2} c_a y_a \\
&= \sum_{a \in B_1} c_a y_a + \sum_{a \in B_2} c_a y_a - \sum_{a \in B_1} c_a x_a - \sum_{a \in B_2} c_a y_a \\
&= \sum_{a \in B_1 \cup B_2} c_a y_a - \sum_{a \in B_1 \cup B_2} c_a x_a.
\end{aligned}$$

Como sabemos que $y_a = x_a$, ou seja, que $y_a - x_a = 0$, para todo arco a que não está em $B_1 \cup B_2$, podemos concluir que

$$\begin{aligned}
\text{custo}(y - x) &= \sum_{a \in A_G} c_a y_a - \sum_{a \in A_G} c_a x_a \\
&= \text{custo}(y) - \text{custo}(x).
\end{aligned}$$

□

Lema 3.17. *Seja x um fluxo sobre uma rede G e seja y um fluxo sobre $G(x)$. Então o fluxo $x + y$ é tal que $e_{x+y}(i) = e_x(i) + e_y(i)$ para todo vértice i.*

Prova: Seja i um vértice de G. Pela defingão de $x+y$ temos que:

$$\begin{aligned}
e_{x+y}(i) &= (x+y)(\delta_G^+(i)) - (x+y)(\delta_G^-(i)) \\
&= x(\delta_G^+(i)) + y(\delta_G^+(i)) - y(\overline{\delta_G^+(i)})) - \\
&\quad x(\delta_G^-(i)) - y(\delta_G^-(i)) + y(\overline{\delta_G^-(i)}) \\
&= x(\delta_G^+(i)) - x(\delta_G^-(i)) + \\
&\quad y(\delta_G^+(i)) + y(\overline{\delta_G^-(i)}) - y(\delta_G^-(i)) - y(\overline{\delta_G^+(i)}) \\
&= x(\delta_G^+(i)) - x(\delta_G^-(i)) + y(\delta_G^+(i) \cup \overline{\delta_G^-(i)}) - y(\delta_G^-(i) \cup \overline{\delta_G^+(i)}) \\
&= x(\delta_G^+(i)) - x(\delta_G^-(i)) + y(\delta_{G(x)}^+(i)) - y(\delta_{G(x)}^-(i)) \\
&= e_x(i) + e_y(i).
\end{aligned}$$

□

Corolário 3.18. *Seja x um st-fluxo sobre uma rede G e seja y um st-fluxo sobre a rede residual $G(x)$. Então o fluxo obtido $x+y$ é um st-fluxo sobre a rede G tal que* $\text{val}(x+y) = \text{val}(x) + \text{val}(y)$.

Prova: Pelo Lema 3.17, temos que $e_{x+y}(i) = e_x(i) + e_y(i) = 0 + 0 = 0$ para todo vértice i diferente de s e t, garantindo a conservação do fluxo. Ademais, temos que $\text{val}(x+y) = e_{x+y}(t) = e_x(t) + e_y(t) = \text{val}(x) + \text{val}(y)$. □

Corolário 3.19. *Seja x um fluxo viável sobre uma rede G com demanda b e seja y um fluxo em $G(x)$ tal que $e_y(i) = 0$ para todo vértice i. Então o fluxo obtido $x+y$ é um fluxo viável em G.*

Prova: Pelo Lema 3.17 e como x foi definido como fluxo viável, temos que $e_{x+y}(i) = e_x(i) + e_y(i) = b(i) + 0 = b(i)$ para todo vértice i. □

Lema 3.20. *Seja x um fluxo sobre uma rede G com custo c e seja y um fluxo sobre $G(x)$. Então o fluxo $x+y$ é tal que* $\text{custo}(x+y) = \text{custo}(x) + \text{custo}(y)$.

Prova: Temos, pela definição de $x+y$, que:

$$\begin{aligned}
\text{custo}(x+y) &= \sum_{a \in A_G} c_a(x_a + y_a - y_{\overline{a}}) \\
&= \sum_{a \in A_G} c_a x_a + \sum_{a \in A_G} c_a y_a - \sum_{a \in A_G} c_a y_{\overline{a}}.
\end{aligned}$$

Seja c' o custo da rede residual $G(x)$. Temos que:

$$\begin{aligned}
\text{custo}(x+y) &= \sum_{a\in A_G} c_a x_a + \sum_{a\in A_G} c'_a y_a + \sum_{a\in A_G} c'_{\overline{a}} y_{\overline{a}} \\
&= \sum_{a\in A_G} c_a x_a + \sum_{a\in A_G} c'_a y_a + \sum_{a\in \overline{A_G}} c'_a y_a \\
&= \sum_{a\in A_G} c_a x_a + \sum_{a\in A_G\cup\overline{A_G}} c'_a y_a \\
&= \text{custo}(x) + \text{custo}(y).
\end{aligned}$$

□

Capítulo 4

Fluxos máximos

Imagine que uma empresa deseja transportar a maior quantidade possível de produtos de uma cidade para outra, através da rede rodoviária. A restrição do transporte é que cada estrada da rede suporta um número finito de caminhões. Considerando as cidades como vértices, as estradas como arcos e o limite de caminhões como a capacidade, este problema resume-se a encontrar um st-fluxo de intensidade máxima na rede rodoviária, onde s é a cidade que envia os produtos e t é a cidade que os recebe.

Este é um exemplo de instância do *problema do fluxo máximo*, que tem aplicações em inúmeras áreas. Outro problema, por exemplo, que pode ser modelado da mesma maneira é maximizar a quantidade de dados enviada de um roteador a outro na *Internet*, sob a restrição de que cada cabo suporta uma quantidade finita de transmissão.

Este capítulo apresenta a definição formal desse problema e de conceitos relacionados, bem como a demonstração de propriedades envolvidas.

4.1 Problema

Um st-fluxo x em uma rede capacitada é **máximo** se não existe nenhum st-fluxo na mesma rede com intensidade maior que a de x.

O **problema do fluxo máximo** pode ser definido da seguinte maneira:

Problema MaxFlow(G, u, s, t): *Dados uma rede G com capacidade u e dois vértices s e t, encontrar um st-fluxo x máximo nessa rede.*

O problema MaxFlow(G, u, s, t) pode ser expresso como o seguinte programa linear: encontrar um vetor x indexado por A_G tal que:

$$\begin{array}{lrcll} \text{maximize} & e(t) & & & \\ \text{sujeito a} & e(i) & = & 0 & \text{para cada } i \text{ em } N_G \setminus \{s,t\}; \\ & x_a & \leq & u_a & \text{para cada } a \text{ em } A_G. \\ & x_a & \geq & 0 & \text{para cada } a \text{ em } A_G; \end{array}$$

Vários dos resultados que apresentamos posteriormente são baseados em resultados conhecidos de programação linear.

4.2 Fluxos máximos e redes residuais

Os resultados aqui apresentados mostram que o problema de encontrar um st-fluxo máximo em uma rede G a partir de um dado st-fluxo x em G equivale ao problema de encontrar um fluxo máximo na rede residual $G(x)$.

Lema 4.1. *Seja x^* um st-fluxo máximo em uma rede G, seja x um st-fluxo qualquer também em G e seja x' um st-fluxo máximo sobre a rede residual $G(x)$. Então* $\text{val}(x') \geq \text{val}(x^*) - \text{val}(x)$.

Prova: Pelo Corolário 3.14, sabemos que o fluxo-diferença entre x^* e x é um st-fluxo em $G(x)$ com intensidade $\text{val}(x^*) - \text{val}(x)$. Logo, um st-fluxo máximo em $G(x)$ tem intensidade maior ou igual que $\text{val}(x^*) - \text{val}(x)$. □

Lema 4.2. *Seja x^* um st-fluxo máximo em uma rede G, seja x um st-fluxo qualquer também em G e seja x' um st-fluxo máximo sobre a rede residual $G(x)$. Então* $\text{val}(x') \leq \text{val}(x^*) - \text{val}(x)$.

Prova: Pelo Corolário 3.18, temos que o fluxo obtido $x + x'$ é um st-fluxo em G tal que $\text{val}(x + x') = \text{val}(x) + \text{val}(x')$. Portanto, podemos concluir que $\text{val}(x) + \text{val}(x') \leq \text{val}(x^*)$, pois x^* é um st-fluxo máximo em G. O resultado segue diretamente dessa última desigualdade. □

Teorema 4.3. *Seja x^* um st-fluxo máximo em uma rede G, seja x um st-fluxo qualquer também em G e seja x' um st-fluxo máximo em $G(x)$. Então* $\text{val}(x') = \text{val}(x^*) - \text{val}(x)$.

Prova: O resultado segue diretamente dos Lemas 4.1 e 4.2. □

Teorema 4.4 (equivalência dos fluxos). *Seja x um st-fluxo qualquer em uma rede G e seja x' um st-fluxo máximo sobre a rede residual $G(x)$. Então o fluxo obtido a partir de x e x', $x + x'$, é um st-fluxo máximo em G.*

Prova: Suponha que $x + x'$ não seja um st-fluxo máximo em G. Pelo Lema 3.18, sabemos que $x + x'$ é um st-fluxo em G cuja intensidade é $\text{val}(x + x') = \text{val}(x) + \text{val}(x')$.

Considere, então, um st-fluxo máximo x^* em G. Pelo Lema 3.12, sabemos que $x^* = x + (x^* - x)$. Isto é, que x^* é o fluxo obtido a partir de x e $(x^* - x)$, em que $(x^* - x)$ é um fluxo sobre a rede residual $G(x)$. Como x^* e x são st-fluxos, então o Lema 3.14 garante que $(x^* - x)$ também o é. Ademais, pelo Lema 3.18, vale que $\text{val}(x^*) = \text{val}(x) + \text{val}(x^* - x)$.

Agora, como x^* é um fluxo máximo e como $x + x'$ não é um fluxo máximo em G, devemos ter $\text{val}(x^*) > \text{val}(x + x')$. Mas então, $\text{val}(x^* - x) > \text{val}(x')$, o que é uma contradição, pois $(x^* - x)$ e x' são ambos st-fluxos em $G(x)$, mas x' é máximo. Assim, $x + x'$ é um st-fluxo máximo em G. □

4.3 Fluxos máximos e cortes mínimos

O resultado apresentado a seguir fornece um limitante superior que pode ser utilizado para certificar a maximalidade de um fluxo.

Teorema 4.5 (dualidade fraca). *Seja x um st-fluxo em uma rede G. Então,* $\text{val}(x) \leq \text{cap}(S)$ *para qualquer st-separador S de G.*

Prova: Já mostramos, através do Lema 3.3, que:

$$\text{val}(x) = x(\delta^-(S)) - x(\delta^+(S)).$$

Como x respeita u, temos que $x(\delta^-(S)) \leq u(\delta^-(S))$. Ademais, como $x_a \geq 0$, para todo arco a da rede, concluímos que $x(\delta^+(S)) \geq 0$. Portanto,

$$\begin{aligned} \text{val}(x) &= x(\delta^-(S)) - x(\delta^+(S)) \\ &\leq u(\delta^-(S)) - x(\delta^+(S)) \\ &\leq u(\delta^-(S)). \end{aligned}$$

Como $\text{cap}(S) = u(\delta^-(S))$ por definição, segue o resultado. □

Corolário 4.6. *Seja x um st-fluxo e seja S um st-separador em uma rede. Se* $\text{val}(x) = \text{cap}(S)$*, então x é um fluxo máximo e S tem capacidade mínima.*

Prova: O resultado segue diretamente do Teorema 4.5. □

Capítulo 5

Método dos caminhos de aumento

Em 1956, Ford e Fulkerson [12] propuseram um algoritmo muito simples para obter um fluxo máximo em uma rede, conhecido como **método dos caminhos de aumento**. Esse método não só resolve o problema do fluxo máximo, como também é uma prova algoritmica para muitos resultados fundamentais na teoria de fluxos em redes, como o **teorema do fluxo máximo e corte mínimo** e o **teorema da integralidade**.

Neste capítulo são apresentados conceitos e resultados preliminares, seguidos da descrição do método e do enunciado dos teoremas citados.

5.1 Caminhos alternantes e de aumento

Seja G uma rede e seja x um st-fluxo sobre G. Um **caminho alternante** é um caminho na rede residual $G(x)$ e um **caminho de aumento** é um caminho alternante cuja origem é o vértice s e cujo destino é o vértice t.

A **capacidade residual** de um caminho alternante qualquer P, denotada por $\text{res}(P)$, é a menor capacidade residual de um arco em P, isto é, $\min_{a \in A_P} r_a$. Podemos observar que, pela definiçao da capacidade residual de um arco, a capacidade residual de um caminho alternante é positiva.

Considere a seguinte notação: se P é um caminho de aumento em $G(x)$, então o fluxo x^P é um st-fluxo elementar tal que $x^P_a = \text{res}(P)$, se a é um arco de P e $x^P_a = 0$, se a não é um arco de P. A definição de capacidade residual garante que x^P é de fato um st-fluxo elementar em $G(x)$.

O fluxo $x + x^P$ é o **fluxo obtido a partir de x e P**.

Lema 5.1. *Seja x um st-fluxo sobre uma rede G e seja P um caminho de aumento em $G(x)$. Então o fluxo obtido a partir de x e P é um st-fluxo de intensidade* $\text{val}(x) + \text{res}(P)$.

Prova: Considere o fluxo $x + x^P$. Como x e x^P são st-fluxos, então, pelo Corolário 3.18, temos que $x + x^P$ é um st-fluxo em G tal que $\text{val}(x + x^P) = \text{val}(x) + \text{val}(x^P)$. Além disso, temos por definição que x^P é um fluxo-caminho de valor $\text{res}(P)$. Portanto, $\text{val}(x^P) = \text{res}(P)$, de onde segue o resultado. □

Lema 5.2. *Seja x um st-fluxo sobre uma rede G, seja x' um st-fluxo elementar em $G(x)$, seja P o caminho em $G(x)$ no qual o valor de x' é positivo e seja p o valor de x'. Então P é um caminho de aumento e* $\text{res}(P) \geq p$.

Prova: O fato de P ser um caminho de aumento segue diretamente do fato de x' ser um st-fluxo elementar em $G(x)$. Como $p = x'_a \leq r_a$, para todo arco a de P, podemos concluir também que $\text{res}(P) \geq p$. □

5.2 Descrição

O Lema 5.1 que provamos anteriormente mostra que se existe um caminho de aumento para um determinado fluxo x, então podemos obter um fluxo de maior intensidade a partir desse caminho. O método dos caminhos de aumento se baseia nessa idéia e aumenta sucessivamente a intensidade de um fluxo inicial até que não existam mais caminhos de aumento.

O algoritmo iterativo abaixo recebe uma rede capacitada G e dois vértices s e t dessa rede e devolve um st-fluxo de intensidade máxima e um st-separador de capacidade mínima em G. Cada iteração do algoritmo mantém um st-fluxo x. Na primeira iteração, $x_a = 0$ para todo arco a.

Caso 1: Não existe caminho de aumento em $G(x)$.
Seja S o conjunto dos vértices acessíveis a partir de s em $G(x)$.
Devolva x e S e pare.

Caso 2: Existe um caminho de aumento em $G(x)$.
Seja P um caminho de aumento em $G(x)$.
Seja x' o fluxo obtido a partir de x e P.
Comece nova iteração com x' no papel de x.

Intuitivamente, os caminhos de aumento são caminhos entre a fonte e o sorvedouro que apresentam espaço para a passagem de mais fluxo. No caso dos arcos que tem o valor do fluxo aumentado, esse espaço é representado naturalmente pela diferença entre a capacidade e o valor anterior do fluxo. No caso dos arcos que tem o valor do fluxo reduzido, a redução representa o envio de fluxo na direção oposta, ou seja, o espaço é justamente o valor anterior do fluxo, representando o quanto ele pode ser reduzido. A figura 5.1 mostra uma simulação completa do método dos caminhos de aumento.

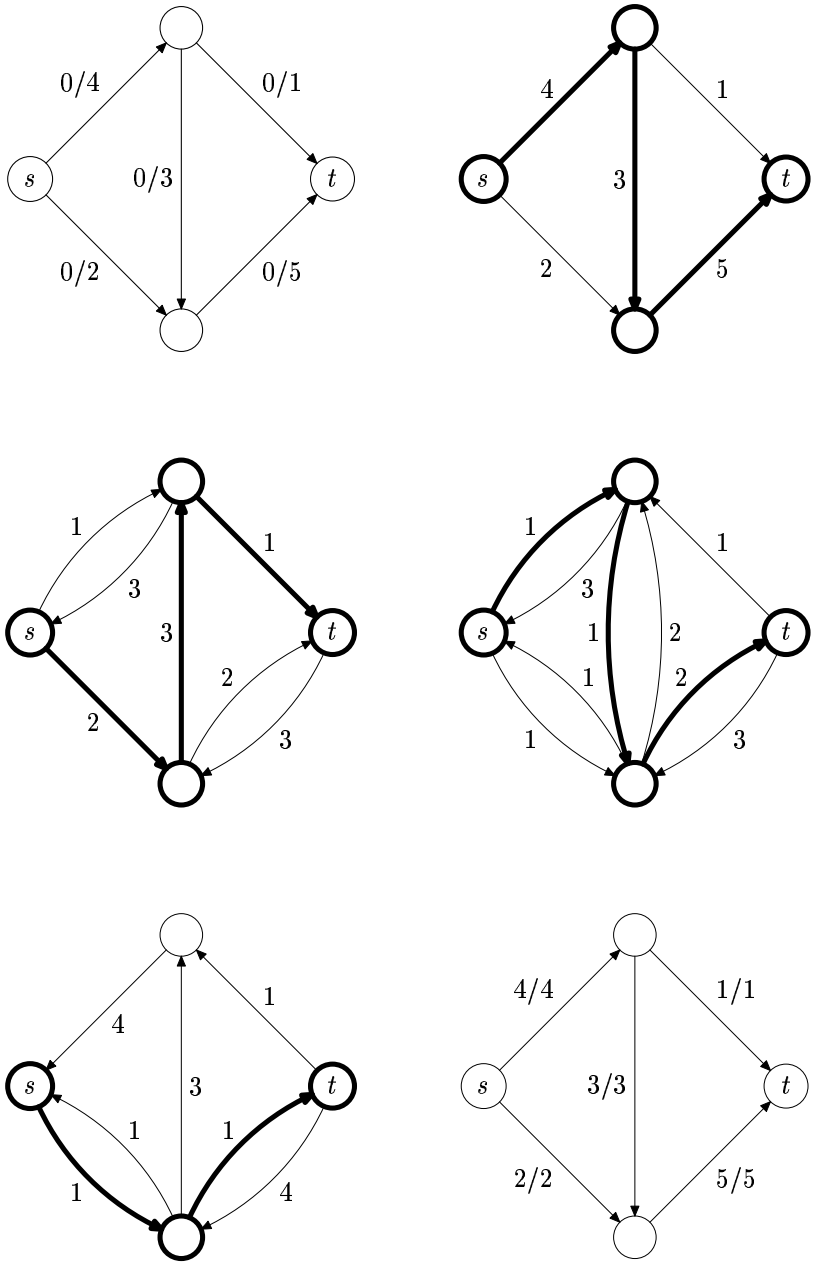

Figura 5.1: Simulação do método dos caminhos de aumento. O primeiro diagrama e o último representam o fluxo inicial e final, respectivamente, e os intermediários indicam os caminhos de aumento nas redes residuais.

5.3 Correção e desempenho

Se o método dos caminhos de aumento termina, então ele devolve um st-fluxo. De fato, basta observarmos que x é um st-fluxo no início da primeira iteração e que, em qualquer iteração, se x é um st-fluxo, então o fluxo obtido a partir de x e P, para algum caminho de aumento P, também o é.

Vamos agora demonstrar resultados que provam que se x é o fluxo devolvido pelo método, então tal fluxo é máximo. Para tanto, utilizaremos o separador S, também devolvido pelo algoritmo, como um certificado.

Lema 5.3. *Seja G uma rede capacitada, seja x o st-fluxo devolvido pelo método dos caminhos de aumento e seja S o separador também devolvido pelo método. Então S é um st-separador de G e* $\mathrm{val}(x) = \mathrm{cap}(S)$.

Prova: Como o fluxo x é tal que não existem caminhos entre s e t em $G(x)$, a definição de S garante que ele é de fato um st-separador.

Pelo Lema 3.3, sabemos que:

$$\mathrm{val}(x) = x(\delta_G^-(S)) - x(\delta_G^+(S)).$$

Seja a um arco em $\delta_G^-(S)$. Pela definição de S, a não é um arco da rede residual $G(x)$. Logo, $r_a = 0$. Além disso, como a é um arco de G, então $r_a = u_a - x_a$. Segue que $x_a = u_a$, para todo arco a em $\delta_G^-(S)$. Ou seja,

$$x(\delta_G^-(S)) = u(\delta_G^-(S))$$

Seja a um arco em $\delta_G^+(S)$. Vamos mostrar que $x_a = 0$. Suponha que não. Neste caso, o arco $\overline{a}$, irmão de a, está em $G(x)$. Mais ainda, como a é um arco de $\delta_G^+(S)$, então $\overline{a}$ é um arco de $\delta_{G(x)}^-(S)$. Mas isso é uma contradição, pois, por definição, temos que $\delta_{G(x)}^-(S) = \emptyset$. Concluímos, assim, que $x_a = 0$, para todo arco a em $\delta_G^+(S)$. Isto é, $x(\delta_G^+(S)) = 0$. Portanto,

$$\mathrm{val}(x) = x(\delta_G^-(S)) - x(\delta_G^+(S)) = u(\delta_G^-(S)) = \mathrm{cap}(S).$$

□

Corolário 5.4. *Seja G uma rede capacitada, seja x o st-fluxo devolvido pelo método dos caminhos de aumento e seja S o st-separador também devolvido pelo método. Então x tem intensidade máxima e S tem capacidade mínima.*

Prova: O resultado segue diretamente do Lema 5.3 e do Corolário 4.6. □

Os resultados obtidos provam que se o método pára, então o fluxo devolvido com certeza é máximo. No entanto, para garantir que o método funciona é preciso demonstrar que, para qualquer rede, ele sempre pára após um número finito de iterações. O resultado a seguir prova esse fato fornecendo um limitante superior finito para o numero de iterações.

Proposição 5.5. *O método dos caminhos de aumento encontra um fluxo máximo em $O(mU)$ iterações.*

Prova: Como as capacidades dos arcos são números racionais, então existe um inteiro positivo K tal que Ku_a é um inteiro, para cada arco a da rede. Podemos escolher K, por exemplo, como o mínimo múltiplo comum entre os denominadores dos valores de todas as capacidades. Assim, temos que, em cada iteração, a capacidade residual do caminho de aumento escolhido é pelo menos $1/K$ e que, portanto, a intensidade do fluxo mantido pelo método aumenta em pelo menos $1/K$.

Pelo Teorema 4.5, sabemos que a intensidade do fluxo máximo em uma rede é limitada pela capacidade de qualquer st-separador nessa mesma rede. Podemos considerar, então, o st-separador $\{s\}$ e observar que sua capacidade está limitada superiormente por mU. Assim, se x^* é um fluxo máximo em G, então $\text{val}(x^*) \leq mU$. Logo, podemos concluir que o método dos caminhos de aumento devolve um fluxo máximo em no máximo KmU iterações. □

Cabe observar que a definição das capacidades como sendo números racionais é essencial para a correção do método dos caminhos de aumento. Ford e Fulkerson [13] mostraram que se as capacidades forem irracionais, o método pode convergir para uma resposta incorreta ou pode não parar.

Como demonstramos que o método dos caminhos de aumento sempre pára e devolve um fluxo x e um separador S tais que $\text{val}(x) = \text{cap}(S)$, ele representa uma prova algoritmica para o resultado abaixo.

Teorema 5.6 (fluxo máximo e corte mínimo). *Seja G uma rede capacitada e sejam s e t dois vértices de G. Então a intensidade de um st-fluxo máximo é igual à capacidade de um st-separador de capacidade mínima.*

Um st-fluxo x em uma rede é dito **inteiro** se o valor x_a é inteiro para todo arco a dessa rede. O seguinte teorema é também uma conseqüência direta da correção do método dos caminhos de aumento.

Teorema 5.7 (integralidade). *Seja G uma rede cujas capacidades dos arcos são todas inteiras. Então existe um fluxo máximo inteiro em G.*

Prova: Se as capacidades são valores inteiros e o st-fluxo x é inteiro no início de determinada iteração do método dos caminhos de aumento, então a capacidade residual de um caminho de aumento nessa iteração é inteira.

Ou seja, se existe um caminho de aumento em $G(x)$ nessa iteração, então o valor do fluxo em cada arco é alterado de um valor inteiro e x permanece um st-fluxo inteiro no início da próxima iteração. Notemos, então, que o st-fluxo x definido no início da primeira iteração é inteiro. Logo, o resultado segue por indução no número de iterações. □

Como última observação, cabe mencionar que demonstramos que o método dos caminhos de aumento é um algoritmo de complexidade pseudo-polinomial, pois sua complexidade depende de U. A figura 5.2 mostra o quanto esse fato pode prejudicar o desempenho do algoritmo.

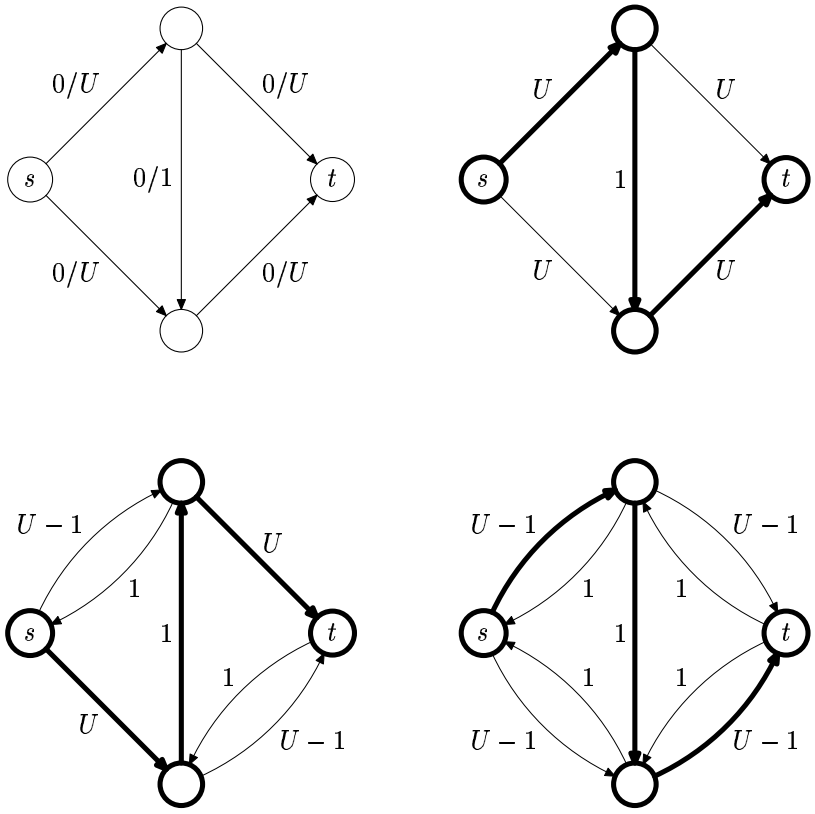

Figura 5.2: Exemplo ruim para o método dos caminhos de aumento. Como cada iteração aumenta o fluxo em apenas 1, são necessárias $2U$ iterações no total e U pode ser muito grande, apesar de a rede ser bem pequena.

Nos próximos capítulos, apresentamos implementações específicas do método que viabilizam uma melhoria substancial de desempenho.

Capítulo 6

Caminhos de aumento de comprimento mínimo

Uma das implementações mais simples para o método dos caminhos de aumento consiste em escolher sempre o caminho de aumento com o menor número de arcos possível. Isso pode ser feito facilmente através de uma busca em largura e é provavelmente a implementação mais natural do método.

Edmonds e Karp [10] demonstraram em 1972 que essa simples restrição na escolha dos caminhos de aumento garante que o método executa em tempo polinomial. Baseados nessa observação, eles propuseram o **algoritmo dos caminhos de aumento de comprimento mínimo**.

Este capítulo contém a descrição completa do algoritmo e a demonstração de que sua complexidade é polinomial.

6.1 Descrição

O algoritmo dos caminhos de aumento de comprimento mínimo é uma implementação do método dos caminhos de aumento que, a cada iteração, seleciona um caminho de aumento mínimo para realizar o aumento do fluxo.

O algoritmo recebe uma rede capacitada G e dois vértices s e t nessa rede, e devolve um st-fluxo x de intensidade máxima e um st-separador S de capacidade mínima em G. Cada iteração começa com um st-fluxo x. Na primeira iteração, x é um fluxo tal que $x_a = 0$, para todo arco a de G.

A descrição do algoritmo está abaixo.

Caso 1: Não existe caminho de aumento em $G(x)$.
Seja S o conjunto dos vértices acessíveis a partir de s em $G(x)$.
Devolva x e S e pare.

Caso 2: Existe um caminho de aumento em $G(x)$.
Seja P um caminho de aumento de comprimento mínimo.

Seja x' o fluxo obtido a partir de x e P.
Comece nova iteração com x' no papel de x.

Como o algoritmo só difere do método dos caminhos de aumento na maneira pela qual o caminho de aumento é selecionado a cada iteração, então sua corretude está garantida pela corretude do método.

6.2 Desempenho

Seja G a rede recebida como parâmetro pelo algoritmo dos caminhos de aumento de comprimento mínimo e sejam s e t respectivamente os vértices fonte e sorvedouro também recebidos como parâmetros.

Vamos denotar por x_k o st-fluxo com o qual o algoritmo inicia sua iteração k, para $k \geq 1$, e, por $d_k(i)$, a distância entre a fonte s e um vértice i qualquer na rede residual $G(x_k)$. É evidente que $d_k(s) = 0$, para todo k.

Seja P o caminho de aumento selecionado em determinada iteração do algoritmo. Dizemos que um arco a de P é **saturado** durante essa iteração se $r_a = \text{res}(P)$. Isto é, se sua capacidade residual é reduzida a zero durante o aumento do fluxo.

Pela definição de capacidade residual de um caminho de aumento, é claro que cada iteração satura pelo menos um arco da rede simétrica de G. Note também que se a é um arco saturado durante a iteração k, então a não está em $G(x_{k+1})$, enquanto seu arco irmão está. Além disso, se um arco está em $G(x_{k+1})$ e não está em $G(x_k)$, então seu arco irmão foi saturado durante a iteração k.

Lema 6.1. *Seja (i,j) um arco da rede residual $G(x_k)$, para algum $k \geq 1$. Então, $d_k(j) \leq d_k(i) + 1$. Ademais, se (i,j) é um arco de um caminho mínimo em $G(x_k)$ com origem s, então $d_k(j) = d_k(i) + 1$.*

Prova: O resultado é uma particularização dos Lemas 2.2 e 2.4. □

Corolário 6.2. *Seja $a = (i,j)$ um arco saturado durante a iteração k. Então $d_k(j) = d_k(i) + 1$.*

Prova: O resultado segue do Lema 6.1, pois todo arco saturado em determinada iteração do algoritmo pertence a um caminho de aumento de comprimento mínimo na rede residual do início dessa iteração. □

Lema 6.3. *Durante a execução do algoritmo dos caminhos de aumento de comprimento mínimo, a distância na rede residual entre a fonte s e um vértice i qualquer não decresce de uma iteração para a seguinte.*

Prova: Vamos provar que $d_{k+1}(i) \geq d_k(i)$, para todo vértice i da rede.

Suponha por contradição que exista um conjunto de vértices M tal que $d_{k+1}(i) < d_k(i)$, para todo vértice i em M. Seja j o vértice de M tal que $d_{k+1}(j)$ seja mínimo.

Pela definição de d_k e como $d_{k+1}(j) < d_k(j)$, então $d_{k+1}(j) \neq \infty$. Portanto, existe um caminho P de comprimento $d_{k+1}(j)$ em $G(x_{k+1})$, com origem s e destino j.

Seja i o vértice predecessor de j no caminho P e seja $a = (i, j)$ o arco predecessor de j em P. Como P é um caminho mínimo em $G(x_{k+1})$ com origem s, então, pelo Lema 6.1, $d_{k+1}(j) = d_{k+1}(i) + 1$.

Existem duas possibilidades: (i) a é um arco de $G(x_k)$ ou (ii) a não é um arco de $G(x_k)$.

Se (i) ocorre, então, pelo Lema 6.1, vale que $d_k(j) \leq d_k(i) + 1$. Por outro lado, sabemos que $d_{k+1}(j) = d_{k+1}(i) + 1$ e que $d_{k+1}(j) < d_k(j)$. Concluímos, assim, que $d_{k+1}(i) < d_k(i)$. Mas, como $d_{k+1}(i) < d_{k+1}(j)$, então i não está em M e, portanto, $d_{k+1}(i) \geq d_k(i)$, o que é uma contradição.

Logo, (i) não ocorre.

Se (ii) ocorre, então $\overline{a}$, o arco irmão de a, com ponta inicial j e ponta final i, é um arco de $G(x_k)$. Mais ainda, o arco $\overline{a}$ foi saturado na iteração k, pois a rede residual $G(x_{k+1})$ contém o arco a. Neste caso, pelo Corolário 6.2, temos que $d_k(i) = d_k(j) + 1$. Ademais, sabemos que $d_{k+1}(j) = d_{k+1}(i) + 1$. Logo, $d_{k+1}(i) < d_{k+1}(j)$, de onde concluímos que i não é um arco de M, uma vez que j é o vértice cuja distância em relação a s na iteração $k+1$ é mínima nesse conjunto. Portanto, $d_{k+1}(i) \geq d_k(i)$. Mas, então, $d_{k+1}(j) = d_{k+1}(i) + 1 \geq d_k(i) + 1 = d_k(j) + 2$. Ou seja, $d_{k+1}(j) \geq d_k(j)$, uma contradição.

Portanto, (ii) também não ocorre e nossa hipótese sobre a existência de M é inválida. Assim, $d_{k+1}(i) \geq d_k(i)$, para todo vértice i. □

Lema 6.4. *Cada arco da rede simétrica de G é saturado no máximo $n/2 - 1$ vezes durante toda a execução do algoritmo dos caminhos de aumento de comprimento mínimo.*

Prova: Seja $a = (i, j)$ um arco da rede simétrica de G e seja k_α uma iteração em que a foi saturado. Nesta iteração, pelo Corolário 6.2, vale que $d_{k_\alpha}(j) = d_{k_\alpha}(i) + 1$. Suponha que exista uma outra iteração k_γ, com $k_\gamma > k_\alpha$, em que a seja novamente saturado.

Sabemos que o arco a não existia na rede residual $G(x_{k_\alpha+1})$. Mas, como a pertence à rede residual $G(x_{k_\gamma})$, houve uma iteração k_β, $k_\alpha < k_\beta < k_\gamma$, na qual o arco $\overline{a} = (j, i)$ foi saturado. Assim, na iteração k_β, pelo Corolário 6.2, valia que $d_{k_\beta}(i) = d_{k_\beta}(j) + 1$.

Agora, pelo Lema 6.3, $d_{k_\beta}(j) \geq d_{k_\alpha}(j)$. Logo, $d_{k_\beta}(i) = d_{k_\beta}(j) + 1 \geq d_{k_\alpha}(j) + 1 = d_{k_\alpha}(i) + 2$. E, novamente pelo Lema 6.3, $d_{k_\gamma}(i) \geq d_{k_\beta}(i)$, de onde concluímos $d_{k_\gamma}(i) \geq d_{k_\alpha}(i) + 2$. Portanto, na iteração k_γ, quando o arco a volta a ser saturado, a distância entre s e i na rede residual foi acrescida em pelo menos 2 desde a iteração k_α.

Se um arco $a = (i, j)$ é saturado, então existe um caminho de aumento na rede residual que contém o vértice i. Como i é a ponta inicial de a, então i não é o sorvedouro t. Sendo assim, é fato que só existe um caminho de aumento que contém i enquanto sua distância com relação a s na rede residual é menor do que $n - 2$, uma vez que a distância entre s e t é, no máximo, $n - 1$. Além disso, sabemos que entre duas saturações de a, a distância entre s e i aumenta em pelo menos 2 unidades.

Dessa forma, concluímos que o arco a pode ser saturado no máximo $(n - 2)/2 = n/2 - 1$ vezes. □

Podemos, agora, limitar o número de iterações do algoritmo.

Proposição 6.5. *O algoritmo dos caminhos de aumento de comprimento mínimo encontra um fluxo máximo em $O(nm)$ iterações.*

Prova. A rede simétrica de G contém $2m$ arcos e sabemos que cada iteração do algoritmo satura pelo menos um de seus arcos. Pelo Lema 6.4, cada um desses $2m$ arcos é saturado no máximo $n/2 - 1$ vezes. Logo, o algoritmo executa no máximo $2m \cdot (n/2 - 1) = O(nm)$ iterações. □

Por fim, observamos que é possível realizar uma implementação do algoritmo na qual o consumo de tempo de cada iteração seja $O(n+m)$. De fato, um caminho de aumento de comprimento mínimo pode ser encontrado em tempo $O(n + m)$ através de uma busca em largura na rede residual. Além disso, para realizar o aumento da intensidade do fluxo, basta visitar no máximo uma vez cada um dos arcos do caminho de aumento selecionado pela iteração e seus arcos irmãos. Tal procedimento pode ser facilmente realizado em tempo $O(n)$. Portanto, fica estabelecido o seguinte resultado.

Proposição 6.6. *O algoritmo dos caminhos de aumento de comprimento mínimo pode ser implementado de modo a executar em tempo $O(nm(n+m))$.*

Capítulo 7

Fluxos de aumento bloqueadores

Claramente, o método dos caminhos de aumento tende a consumir maior tempo se existem muitos caminhos de aumento a serem considerados. Por outro lado, observamos no capítulo anterior que escolher caminhos de aumento com o menor comprimento garante complexidade polinomial.

Uma observação fundamental que possibilita uma melhoria de desempenho é o fato de que é possível enviar fluxo através de vários caminhos de aumento de uma só vez. Baseado nessa idéia, Dinits [9] sugeriu uma variação do algoritmo dos caminhos de aumento de comprimento mínimo conhecida como **algoritmo dos fluxos bloqueadores de aumento**.

Este capítulo apresenta os conceitos básicos envolvidos na descrição do algoritmo e a demonstração de sua complexidade.

7.1 Fluxos bloqueadores

Seja x um st-fluxo em uma rede G. Um arco a de G é dito **saturado**, se sua capacidade residual é nula com relação ao fluxo x.

Um st-fluxo em uma rede G é um **fluxo bloqueador** se cada caminho de s a t em G possui um arco saturado. Podemos dizer também que um fluxo bloqueador em uma rede G é um fluxo maximal nesta rede.

7.2 Redes residuais de caminhos mínimos

Seja G uma rede capacitada e seja x um st-fluxo em G.

A **rede residual de caminhos mínimos** de G em relação a x, denotada por $G^*(x)$, é a rede (N, A^*), em que N é o conjunto de vértices de G e A^* é o conjunto dos arcos da rede residual $G(x)$ que estão em algum caminho mínimo entre s e t nessa rede.

Lema 7.1. *Seja G uma rede e seja x um st-fluxo em G. Então a rede residual de caminhos mínimos $G^*(x)$ é uma rede acíclica.*

Prova: Denotemos por $d(i)$ a distância entre a fonte s e um vértice i qualquer na rede residual $G(x)$. Suponha por contradição que $G^*(x)$ não seja acíclica.

Seja $W = \langle i_0, a_1, i_1, \ldots, i_{k-1}, a_k, i_k = i_0 \rangle$ um circuito em $G^*(x)$. Pela definição de rede residual de caminhos mínimos, cada um dos arcos de W está em um caminho mínimo entre s e t em $G(x)$. Assim, pelo Lema 2.4, devemos ter $d(i_{l+1}) = d(i_l) + 1$, para todo $0 \leq l \leq k-1$. Mas, então, concluímos que $d(i_k) > d(i_0)$, o que é uma contradição, pois $i_0 = i_k$.

Portanto, não existem circuitos em $G^*(x)$. □

7.3 Descrição

O algoritmo dos fluxos bloqueadores de aumento mantém um st-fluxo e procura aumentar sua intensidade iterativamente através de fluxos bloqueadores na rede residual de caminhos mínimos. Tal procedimento objetiva aumentar o fluxo através de um conjunto de caminhos de aumento de comprimento mínimo de maneira simultânea na mesma iteração.

A descrição do algoritmo está abaixo. Como uma implementação do método dos caminhos de aumento, ele recebe uma rede capacitada G, dois vértices s e t, e devolve um st-fluxo x de intensidade máxima e um st-separador S de capacidade mínima em G. Cada iteração começa com um st-fluxo x. Na primeira iteração, x é tal que $x_a = 0$, para todo arco a de G.

Caso 1: Não existe um caminho de aumento em $G(x)$.
Seja S o conjunto dos vértices acessíveis a partir de s em $G(x)$.
Devolva x e S e pare.

Caso 2: Existe um caminho de aumento em $G(x)$.
Seja x^* um fluxo bloqueador em $G^*(x)$.
Seja x' o fluxo obtido a partir de x e x^*.
Comece nova iteração com x' no papel de x.

Pela corretude do método dos caminhos de aumento, temos que se o algoritmo dos fluxos bloqueadores de aumento pára, então ele devolve a resposta correta. Na próxima seção estabelecemos um limitante polinomial para o seu número de iterações e também mencionamos a complexidade de uma de suas possíveis implementações. Conforme poderemos verificar, a idéia de aumentar o fluxo simultaneamente através de um conjunto de caminhos de aumento de fato traz uma melhoria significante de desempenho.

7.4 Desempenho

Seja G a rede recebida como parâmetro pelo algoritmo dos fluxos bloqueadores de aumento e sejam s e t respectivamente os vértices fonte e sorvedouro também recebidos como parâmetros. Vamos mostrar que o número de iterações efetuadas pelo algoritmo é limitado polinomialmente.

Lema 7.2. *Durante a execução do algoritmo dos fluxos bloqueadores de aumento, a distância na rede residual entre a fonte s e um vértice i qualquer não decresce de uma iteração para a seguinte.*

Prova: Seja x o st-fluxo com o qual o algoritmo inicia determinada iteração, seja x^* o fluxo bloqueador obtido sobre $G^*(x)$ nessa mesma iteração e seja $x + x^*$ o fluxo obtido a partir de x e x^*.

Pelo Teorema da decomposição do fluxo (3.8) e pelo Corolário 3.9, é possível obter uma decomposição X do fluxo x^* em não mais do que $m(G^*(x))$ st-fluxos elementares. Tal decomposição não contém fluxos-circuito, uma vez que x^* é um fluxo sobre a rede $G^*(x)$, a qual, pelo Lema 7.1, é uma rede acíclica. Ademais, pelo Lema 5.2, existe um caminho de aumento P_i em $G(x)$ associado a cada st-fluxo elementar x^i em X, cuja capacidade residual é pelo menos o valor de x^i. Como x^* é um fluxo bloqueador na rede residual de caminhos mínimos $G^*(x)$, então cada um desses caminhos de aumento tem comprimento mínimo.

Logo, a obtenção do fluxo $x + x^*$ pode ser interpretada como o aumento consecutivo de x através dos caminhos de aumento de comprimento mínimo associados a cada um dos st-fluxos elementares que constituem a decomposição X de x^*. Ou seja, uma iteração do algoritmo dos fluxos bloqueadores de aumento equivale a um conjunto de iterações consecutivas do algoritmo dos caminhos de aumento de comprimento mínimo.

Portanto, o resultado segue do Lema 6.3. □

Lema 7.3. *A distância na rede residual entre a fonte s e o sorvedouro t aumenta em pelo menos uma unidade entre duas iterações consecutivas do algoritmo dos fluxos bloqueadores de aumento.*

Prova: Denote por x_k o st-fluxo com o qual o algoritmo inicia sua iteração k, para $k \geq 1$, e por $d_k(i)$ a distância entre a fonte s e um vértice i qualquer na rede residual $G(x_k)$.

Pelo Lema 7.2, temos que $d_k(t) \leq d_{k+1}(t)$. Vamos mostrar que $d_k(t) < d_{k+1}(t)$. Para tanto, suponha por contradição que $d_{k+1}(t) = d_k(t)$.

Primeiramente, notemos que $d_{k+1}(t) \neq \infty$. De fato, do contrário, teríamos $d_k(t) = \infty$ e, portanto, não existiriam caminhos de aumento em $G(x_k)$. Logo, o Caso 1 ocorreria na iteração k e esta seria a útima iteração.

Como $d_{k+1}(t) \neq \infty$, então o sorvedouro t é acessível a partir da fonte s na rede residual $G(x_{k+1})$. Assim, podemos considerar um caminho P entre s

e t em $G(x_{k+1})$, de comprimento $d_{k+1}(t)$. Como P é um caminho mínimo entre s e t, então, pela definição de rede residual de caminhos mínimos, P é um caminho em $G^*(x_{k+1})$.

Notemos, então, que existe pelo menos um arco de P que não é arco de $G(x_k)$. De fato, do contrário, P seria um caminho em $G^*(x_k)$, pois $d_{k+1}(t) = d_k(t)$. Mas, neste caso, pelo menos um arco de P teria sido saturado pelo fluxo bloqueador da iteração k e P não seria um caminho em $G^*(x_{k+1})$, o que é uma contradição. Assim, podemos considerar o arco $a = (i, j)$, o último arco de P que não está em $G(x_k)$.

Como a está em $G(x_{k+1})$ e não está em $G(x_k)$, então $\overline{a} = (j, i)$ está em $G(x_k)$ e foi saturado pelo fluxo bloqueador da iteração k. Portanto, existe um caminho mínimo em $G(x_k)$ entre s e t que contém $\overline{a}$. Assim, pelo Lema 6.1, $d_k(i) = d_k(j) + 1$.

Pelo Lema 7.2, $d_{k+1}(i) \geq d_k(i)$ e $d_{k+1}(j) \geq d_k(j)$. E, como a é um arco de um caminho mínimo em $G(x_{k+1})$, então, também pelo Lema 6.1, $d_{k+1}(j) = d_{k+1}(i) + 1$. Logo, podemos concluir que $d_{k+1}(j) = d_{k+1}(i) + 1 \geq d_k(i) + 1 = d_k(j) + 2$.

Agora, como $a = (i, j)$ é o último arco de P que não está em $G(x_k)$, então o caminho P' entre j e t e contido em P é um caminho em $G(x_{k+1})$ que também está em $G(x_k)$. Além disso, como P é um caminho mínimo, podemos utilizar o Lema 2.3 e concluir que P' é um caminho de comprimento $d_{k+1}(t) - d_{k+1}(j)$. Ademais, como $d_k(j)$ é a distância entre s e j em $G(x_k)$, então existe um caminho Q em $G(x_k)$, entre s e j e de comprimento $d_k(j)$. Mas, então, pelo Lema 2.1, existe um caminho entre s e t em $G(x_k)$ de comprimento não superior a $d_k(j) + (d_{k+1}(t) - d_{k+1}(j)) \leq d_k(j) + d_k(t) - d_k(j) - 2 = d_k(t) - 2$. Chegamos a uma contradição, uma vez que $d_k(t)$ é a distância entre s e t na rede $G(x_k)$.

Portanto, $d_{k+1}(t) > d_k(t)$. □

Proposição 7.4. *O algoritmo dos fluxos bloqueadores de aumento encontra um fluxo máximo em $O(n)$ iterações.*

Prova: É fato que a distância entre a fonte s e o sorvedouro t nunca atinge valor superior a n, enquanto existem caminhos de aumento na rede residual. Além disso, o Lema 7.3 garante que a distância entre s e t aumenta em pelo menos uma unidade a cada ocorrência do Caso 2. Sendo assim, concluímos que o algoritmo não pode executar mais do que $n - 1$ iterações. □

Por fim, vamos fazer algumas considerações sobre o consumo de tempo de uma possível implementação do algoritmo.

Lema 7.5. *Seja (N, A) uma rede capacitada e acíclica. É possível encontrar um st-fluxo bloqueador sobre (N, A) em tempo $O(nm)$.*

Prova: Considere o seguinte procedimento iterativo, que recebe uma rede acíclica (N, A) com capacidade u e dois vértices s e t, e devolve um st-fluxo

bloqueador x sobre a mesma. Cada iteração começa com um st-fluxo x, um subconjunto B dos arcos de A e uma função capacidade r. Na primeira iteração, $x_a = 0$, para todo arco a em A, $B = A$ e $r = u$.

Caso 1: Não existe caminho entre s e t em (N, B).
Devolva x e pare.

Caso 2: Existe um caminho entre s e t em (N, B).
Seja P um caminho entre s e t em (N, B).
Defina $\alpha := \min\{r_a : a \text{ é arco de } P\}$.
Defina $r'_a := r_a - \alpha$, se a é um arco de P, e $r'_a := r_a$, caso contrário.
Defina $x'_a := x_a + \alpha$, se a é um arco de P, e $x'_a := x_a$, caso contrário.
Seja B' o conjunto obtido após a remoção de todos os arcos a tais que $r'_a = 0$ do conjunto B.
Comece nova iteração com B' no papel de B, r' no papel de r e x' no papel de x.

Se o algoritmo pára, então é fácil verificar que x é um fluxo na rede (N, A). Temos também que cada iteração aumenta a intensidade do fluxo x através de caminhos de s a t e de forma a reduzir a capacidade residual de pelo menos um arco de cada um desses caminhos a zero. Ademais, na última iteração, todos os caminhos de s a t na rede já foram considerados. Portanto, o fluxo x devolvido é de fato um fluxo bloqueador.

Notemos também que, em cada iteração, a cardinalidade do conjunto B diminui em pelo menos 1, uma vez que pelo menos um arco do caminho P é removido dele. Como no início da primeira iteração temos $B = A$, então não mais do que m iterações podem ocorrer. Ademais, como (N, A) é uma rede acíclica, então um caminho entre s e t pode ser determinado em tempo $O(n)$ através de uma busca em profundidade nessa rede a partir de s. Ou seja, cada iteração do algoritmo pode ser realizada em tempo $O(n)$. Portanto, o algoritmo descrito encontra um fluxo bloqueador em tempo $O(nm)$. □

Em cada iteração do algoritmo dos fluxos bloqueadores de aumento, é possível obter a rede residual de caminhos mínimos em tempo $O(n + m)$, através de uma busca em largura na rede residual a partir do vértice fonte s. Ademais, pelo Lema 7.1, sabemos que a rede residual de caminhos mínimos é uma rede acíclica. Portanto, um fluxo bloqueador sobre esta rede pode ser obtido em tempo $O(nm)$, conforme estabelecido pelo Lema 7.5. Assim, cada iteração do algoritmo dos fluxos bloqueadores de aumento pode ser implementada com consumo de tempo $O(nm)$.

Feitas essas observações, o resultado abaixo segue da Proposição 7.4.

Proposição 7.6. *O algoritmo dos fluxos bloqueadores de aumento pode ser implementado de modo a executar em tempo* $O(n^2 m)$.

A Proposição 7.6 mostra que o algoritmo dos fluxos bloqueadores de aumento de fato apresenta um melhor desempenho teórico quando comparado ao algoritmo dos caminhos de aumento de comprimento mínimo. Uma vez que sua complexidade depende quadraticamente apenas do número de vértices da rede, essa melhoria de desempenho deve ocorrer especialmente quando a rede recebida como parâmetro é **densa**, isto é, quando possui seu número de arcos muito maior do que seu número de vértices.

Capítulo 8

Caminhos de maior aumento

Intuitivamente, ao se utilizar o método dos caminhos de aumento para resolver o problema do fluxo máximo, é natural esperar que a escolha de um caminho de aumento com a maior capacidade residual possível seja uma restrição capaz de reduzir significativamente o número total de iterações.

De fato, Edmonds e Karp [10] provaram que essa implementação para o método dos caminhos de aumento, conhecida como **algoritmo dos caminhos de maior aumento**, garante consumo de tempo polinomial.

Este capítulo descreve o algoritmo e demonstra sua complexidade.

8.1 Descrição

A única diferença entre o algoritmo que descrevemos aqui e a versão genérica do método dos caminhos de aumento é o fato de que o caminho de aumento escolhido tem sempre a maior capacidade residual possível. Esse tipo de caminho é chamado de **caminho de maior aumento**.

O algoritmo iterativo abaixo recebe uma rede capacitada G e dois vértices s e t dessa rede e devolve um st-fluxo de intensidade máxima e um st-separador de capacidade mínima em G. Cada iteração do algoritmo mantém um st-fluxo x. Na primeira iteração, $x_a = 0$ para todo arco a.

Caso 1: Não existe caminho de aumento em $G(x)$.
Seja S o conjunto dos vértices acessíveis a partir de s em $G(x)$.
Devolva x e S e pare.

Caso 2: Existe um caminho de aumento em $G(x)$.
Seja P um caminho de maior aumento em $G(x)$.
Seja x' o fluxo obtido a partir de x e P.
Comece nova iteração com x' no papel de x.

8.2 Desempenho

Os resultados demonstrados a seguir garantem que o número de iterações do método dos caminhos de maior aumento é limitado polinomialmente.

Lema 8.1. *Seja x um st-fluxo em uma rede capacitada G e seja x^* um st-fluxo máximo sobre a mesma rede. Então, o caminho de maior aumento P em relação a x é tal que* $\text{res}(P) \geq (\text{val}(x^*) - \text{val}(x))/2m$.

Prova: Considere o fluxo-diferença $x^* - x$. Sabemos pelo Corolário 3.14 que $\text{val}(x^* - x) = \text{val}(x^*) - \text{val}(x)$. Pelo Corolário 3.9, temos que existe uma decomposição X de $x^* - x$ em não mais do que $m(G(x))$ fluxos-caminho e fluxos-circuito tal que todos os fluxos-caminho de X são st-fluxos elementares e tal que a intensidade de $x^* - x$ é a soma das intensidades destes.

Isso implica que existe pelo menos um fluxo-caminho x' em X tal que o valor de x' é pelo menos $\text{val}(x^* - x)/m(G(x)) = (\text{val}(x^*) - \text{val}(x))/m(G(x))$. Logo, pelo Lema 5.2, existe um caminho de aumento P em $G(x)$ tal que $res(P) \geq (\text{val}(x^*) - \text{val}(x))/m(G(x))$, de onde segue o resultado. □

Proposição 8.2. *O algoritmo dos caminhos de maior aumento encontra um fluxo máximo em $O(m \log mU)$ iterações.*

Prova: Seja x um fluxo em uma rede G e seja x^* um fluxo máximo sobre G. Pelo Lema 8.1, sabemos que o caminho de maior aumento em relação a x tem capacidade residual maior ou igual a $(\text{val}(x^*) - \text{val}(x))/2m$.

Considere uma seqüência de $4m$ iterações do algoritmo dos caminhos de maior aumento sobre G a partir de um fluxo inicial x_1. Se em cada uma delas encontra-se um caminho de aumento com capacidade residual maior ou igual a $(\text{val}(x^*) - \text{val}(x_1))/4m$, um fluxo máximo é obtido.

Por outro lado, suponha que em uma dessas $4m$ iterações do algoritmo encontra-se um caminho de aumento com capacidade residual menor do que $(\text{val}(x^*) - \text{val}(x_1))/4m$. Seja x_2 o fluxo em G nessa iteração. Como sabemos que o caminho de maior aumento em relação a x_2 tem capacidade residual maior ou igual a $(\text{val}(x^*) - \text{val}(x_2))/2m$, então podemos concluir que

$$(\text{val}(x^*) - \text{val}(x_2))/2m \leq (\text{val}(x^*) - \text{val}(x_1))/4m$$

Ou seja, temos que

$$\text{val}(x^*) - \text{val}(x_2) \leq (\text{val}(x^*) - \text{val}(x_1))/2$$

O que significa que, após essas $4m$ iterações, a diferença entre a intensidade do fluxo mantido pelo algoritmo e a intensidade do fluxo máximo foi reduzida pela metade. Como existem no máximo m arcos em $\delta^+(s)$ e o fluxo em um arco vale no máximo U, a intensidade do fluxo máximo em G é limitada superiormente por mU e concluímos que o algoritmo encontra um fluxo máximo em não mais que $4m \log mU = O(m \log mU)$ iterações. □

Um caminho de maior aumento pode ser encontrado através de um algoritmo análogo ao algoritmo de Dijkstra [8] para caminhos de custo mínimo. Ademais, Cormen, Leiserson, Rivest e Stein [6] mostraram que o algoritmo de Dijkstra pode ser implementado de modo a executar em tempo $O(n + m \log n)$ usando a estrutura de dados *heap*. Disso segue o resultado abaixo.

Proposição 8.3. *O algoritmo dos caminhos de maior aumento pode ser implementado de modo a executar em tempo* $O(m \log mU(n + m \log n))$.

Capítulo 9

Capacity scaling

Como provamos anteriormente, utilizar o algoritmo dos caminhos de maior aumento para implementar o método dos caminhos de aumento permite encontrar um fluxo máximo em tempo polinomial. Entretanto, uma das maiores desvantagens desse algoritmo é justamente encontrar o caminho com maior capacidade residual: trata-se de um processo que exige estruturas de dados relativamente sofisticadas que prejudicam o desempenho.

Uma simplificação desse algoritmo que elimina tais problemas de implementação é, ao invés de encontrar o caminho com maior capacidade residual, encontrar um caminho com capacidade residual suficientemente grande. Nessa idéia simples, mas surpreendentemente poderosa, baseia-se o **algoritmo capacity scaling**, concebido por Ahuja e Orlin [3].

Este capítulo introduz conceitos necessários para desenvolver o algoritmo, descreve seu funcionamento e demonstra sua complexidade.

9.1 Redes residuais restritas

Seja x um fluxo em uma rede capacitada G e seja Δ um valor qualquer. Uma rede residual em relação a x e G **restrita por** Δ é uma rede $G(x, \Delta)$ tal que $N_{G(x,\Delta)} = N_G$ e $A_{G(x,\Delta)}$ é um conjunto de arcos tal que a está em $A_{G(x,\Delta)}$ se, e somente se, a é um arco de $G(x)$ e $r_a \geq \Delta$.

Uma característica importante, que é utilizada pelo algoritmo, de uma rede restrita por Δ é o fato de que um caminho em $G(x, \Delta)$ é um caminho de aumento em $G(x)$ cuja capacidade residual é maior ou igual a Δ.

9.2 Descrição

A idéia do algoritmo capacity scaling é modificar a versão genérica do método dos caminhos de aumento de tal forma que, em cada iteração, exista um limitante inferior para a capacidade residual de um caminho de aumento.

Esse limitante é gradualmente reduzido durante a execução do algoritmo, de forma que a correção é garantida pelo caso em que ele é mínimo.

O algoritmo iterativo abaixo recebe uma rede capacitada G e dois vértices s e t dessa rede e devolve um st-fluxo de intensidade máxima e um st-separador de capacidade mínima em G. Cada iteração do algoritmo mantém um st-fluxo x e um limitante inferior Δ. Na primeira iteração, temos que $x_a = 0$ para todo arco a e tambem que $\Delta = 2^{\lfloor \log U \rfloor}$.

Caso 1: $\Delta < 1$.
Seja S o conjunto dos vértices acessíveis a partir de s em $G(x, \Delta)$.
Devolva x e S e pare.

Caso 2: $\Delta \geq 1$.

Caso 2A: Não existe caminho de aumento em $G(x, \Delta)$.
Comece nova iteração com $\Delta/2$ no papel de Δ.

Caso 2B: Existe um caminho de aumento em $G(x, \Delta)$.
Seja P um caminho de aumento em $G(x, \Delta)$.
Seja x' o fluxo obtido a partir de x e P.
Comece nova iteração com x' no papel de x.

Um seqüência maximal de iterações executadas sem que o valor de Δ se altere é uma **fase de scaling**. São executadas $O(\log U)$ dessas fases.

Podemos observar que a correção do algoritmo só é garantida se não existirem caminhos de aumento com capacidade residual positiva e estritamente inferior a 1. Porém, pelo mesmo argumento da demonstração do Teorema 5.7, sabemos que essa restrição pode ser obtida se as capacidades dos arcos da rede forem todas inteiras. Portanto, como as capacidades são originalmente racionais, podemos garantir a propriedade através de um pré-processamento que multiplica todas as capacidades por uma constante K, que sabemos existir, tal que todas elas tornam-se números inteiros e um pós-processamento que divide o fluxo obtido por K, adaptando-o para a rede original.

9.3 Desempenho

Apesar do algoritmo capacity scaling ser mais simples, e aparentemente menos poderoso, do que o algoritmo dos caminhos de maior aumento, os resultados abaixo demonstram que existe um bom limitante superior para seu número de iterações, comprovando desempenho polinomial.

Proposição 9.1. *O algoritmo capacity scaling encontra um fluxo máximo em $O(m \log U)$ iterações.*

Prova: Primeiramente, podemos observar que, durante uma fase de scaling, temos que a capacidade residual de qualquer caminho de aumento é limitada superiormente por 2Δ. De fato, isso é verdade na primeira fase, pois o valor inicial de Δ é $2^{\lfloor \log U \rfloor}$, e as condições necessárias para a ocorrência do Caso 2A garantem que a propriedade continua satisfeita para as fases seguintes.

Seja x^* um fluxo máximo sobre G, seja x o fluxo mantido pelo algoritmo no início de uma fase de scaling e considere o fluxo-diferença $x^* - x$. Sabemos pelo Corolário 3.14 que $\text{val}(x^* - x) = \text{val}(x^*) - \text{val}(x)$. Pelo Corolário 3.9, temos que existe uma decomposição X de $x^* - x$ em não mais do que $m(G(x))$ fluxos-caminho e fluxos-circuito tal que os fluxos-caminho de X são st-fluxos elementares e a intensidade de $x^* - x$ é a soma das intensidades destes.

Pelo Lema 5.2, sabemos que o caminho correspondente a cada st-fluxo elementar x' de X é um caminho de aumento cuja capacidade residual é limitada inferiormente pelo valor de x'. Por outro lado, sabemos que essa capacidade residual também é limitada superiormente por 2Δ. Portanto, o valor de qualquer st-fluxo elementar de X é menor ou igual a 2Δ, o que implica que $\text{val}(x^*) - \text{val}(x) = \text{val}(x^* - x) \leq 2\Delta m(G(x)) \leq 4\Delta m(G)$.

Para concluir, podemos observar que, como qualquer caminho de aumento em $G(x, \Delta)$ tem capacidade residual maior ou igual que Δ, o Lema 5.1 garante que cada iteração aumenta a intensidade do fluxo mantido em pelo menos Δ. Como vimos que $\text{val}(x^*) - \text{val}(x) \leq 4\Delta m(G)$, o algoritmo não executa mais do que $4m(G)$ iterações em uma mesma fase de scaling, pois esse número de iterações é suficiente para encontrar um fluxo máximo.

Portanto, podemos concluir que $O(m(G))$ iterações são feitas para cada uma das $O(\log U)$ fases de scaling do algoritmo e temos o resultado. □

Redes residuais restritas podem ser mantidas implicitamente pelo algoritmo, bastando que a busca por caminhos de aumento desconsidere arcos com capacidade residual inferior a Δ. Logo, o processo de encontrar um caminho de aumento em $G(x, \Delta)$ pode ser feito através de uma busca em largura simples, que por sua vez pode ser implementada de forma a consumir tempo $O(n + m)$. Desse fato podemos inferir o resultado abaixo.

Proposição 9.2. *O algoritmo capacity scaling pode ser implementado de modo a executar em tempo* $O(m \log U(n + m))$.

Capítulo 10

Método do pré-fluxo

Goldberg e Tarjan [14] sugeriram em 1986 uma nova abordagem para o problema do fluxo máximo que ficou conhecida como **método do pré-fluxo**. Quando implementada de maneira adequada, essa estratégia resulta em algoritmos extremamente eficientes, tanto na teoria como na prática.

Este capítulo apresenta os principais conceitos envolvidos nesse método, além de sua descrição e da demonstração de sua complexidade.

10.1 Pré-fluxos e vértices ativos

Seja G uma rede e sejam s e t dois vértices de G. Um **pré-st-fluxo** é um fluxo x em G que satisfaz a seguinte propriedade:

$$e(i) \geq 0 \text{ para todo } i \text{ em } N_G \setminus \{s,t\}.$$

Os vértices s e t são chamados respectivamente de **fonte** e **sorvedouro** de x. Quando s e t estão implícitos, o fluxo x pode ser dito simplesmente um **pré-fluxo**. Note também que todo st-fluxo é também um pré-fluxo.

Um vértice i em $N_G \setminus \{s,t\}$ é **ativo** em relação ao pré-fluxo x se $e(i) > 0$.

Lema 10.1. *Seja x um pré-st-fluxo em uma rede G. Se existe pelo menos um vértice ativo com relação a x e se $e(t) \geq 0$, então $e(s) < 0$.*

Prova: Como x é um fluxo, pelo Lema 3.1, temos que $\sum_{i \in N_G} e(i) = 0$. Logo, $e(s) = -\sum_{i \in N_G \setminus \{s\}} e(i)$. Ademais, como $e(t) \geq 0$ e existe pelo menos um vértice ativo na rede, então $\sum_{i \in N_G \setminus \{s\}} e(i) > 0$. Disso segue o resultado. □

10.2 Distâncias e arcos admissíveis

Uma **função-distância** em uma rede G é uma função de N_G em $\mathbb{Z}_{\geq}$. Diz-se que uma função distância d é **válida** com relação a um pré-st-fluxo x se essa função satisfaz as seguintes propriedades:

$d(t) = 0$;
$d(i) \leq d(j) + 1$, se existe um arco (i, j) na rede residual $G(x)$.

O valor $d(i)$ é o **rótulo-distância** do vértice i ou apenas seu **rótulo**.

Seja x um pré-fluxo em uma rede G e seja d uma função-distância válida. Dizemos que um arco (i, j) de $G(x)$ é **admissível** se $d(i) = d(j) + 1$.

Lema 10.2. *Seja d uma função-distância e seja x um pré-st-fluxo em uma rede G. Se os rótulos-distância são válidos, então $d(i) - d(j)$ é um limitante inferior para a distância entre quaisquer dois vértices i e j em $G(x)$.*

Prova: Sejam i e j vértices de G e seja $P = \langle i = i_0, a_1, i_1, \ldots, a_k, i_k = j \rangle$ um caminho qualquer entre i e j em $G(x)$.

Como os rótulos são válidos, então temos que $d(i_{l-1}) \leq d(i_l) + 1$, para todo arco (i_{l-1}, i_l) em P, e, portanto, $d(i_{l-1}) - d(i_l) \leq 1$. Assim,

$$k \leq \sum_{l=1}^{k} (d(i_{l-1}) - d(i_l)) = d(i_0) - d(i_k) = d(i) - d(j).$$

Logo, se P é um caminho entre i e j de comprimento k, então temos que $d(i) - d(j) \leq k$. Ou seja, $d(i) - d(j)$ é um limitante inferior para o comprimento de qualquer caminho entre i e j em $G(x)$ e, em particular, $d(i) - d(j)$ é um limitante inferior para a distância entre i e j em $G(x)$. □

O método do pré-fluxo utiliza os conceitos de distâncias e arcos admissíveis para fazer duas operações básicas durante sua execução: pushes e relabels. Nas próximas seções definimos essas duas operações.

10.3 Pushes

Seja x um fluxo em uma rede G, seja i um vértice ativo e seja $a = (i, j)$ um arco admissível em $G(x)$ com relação a uma função-distância.

Um **push** em a é o processo de obtenção de um fluxo x' em G, a partir de x e a, da seguinte maneira: para cada arco b em A_G,

$$x'_b = \begin{cases} x_b + \min\{r_a, e_x(i)\} & \text{se } b = a, \\ x_b - \min\{r_a, e_x(i)\} & \text{se } b = \overline{a}, \\ x_b & \text{caso contrário.} \end{cases}$$

Podemos observar que a definição do push garante que x' é de fato um fluxo. O fluxo x' é dito o **fluxo obtido após um push em** a.

Um push é **saturador** se $r_a = \min\{r_a, e_x(i)\}$ e é dito **não-saturador** caso contrário. Um arco a é dito **saturado** se ocorre um push saturador em a. Podemos observar que se a é saturado, então sua capacidade residual com relação ao fluxo x' é nula e, portanto, a não está na rede residual $G(x')$.

10.4 Relabels

Seja x um pré-st-fluxo em uma rede G, seja d uma função-distância na rede $G(x)$ e seja i um vértice ativo em relação a x.

Um **relabel** de i é o processo de obtenção de uma função-distância d', a partir de i e d, da seguinte maneira: para cada vértice k em N_G,

$$d'(k) = \begin{cases} d(k) & \text{se } k \neq i, \\ \min\{d(j)+1 : (i,j) \text{ é arco de } \delta^-_{G(x)}(i)\} & \text{caso contrário.} \end{cases}$$

A função d' é a **função-distância obtida após um relabel de** i.

Lema 10.3. *Seja x um pré-st-fluxo em uma rede G, seja d uma função-distância na rede $G(x)$ e seja i um vértice ativo em relação a x. Se d é válida, então a função-distância d' obtida após um relabel de i também é válida. Ademais, $d'(k) \geq d(k)$, para todo vértice k da rede.*

Prova: Primeiramente, podemos observar que como i é um vértice ativo, então $x(\delta^+(i)) > 0$. Portanto, $\delta^-_{G(x)}(i)$ é não-vazio, de maneira que não estamos minimizando sobre um conjunto vazio durante um relabel.

Como d é válida, então $d(i) \leq d(j)+1$, para todo arco (i,j) em $G(x)$. Conseqüentemente, $d'(i) \geq d(i)$ e, portanto, $d'(k) = d(k) \leq d(i)+1 \leq d'(i)+1$, para todo arco (k,i) em $G(x)$. Além disso, temos que $d'(i) \leq d(j)+1 = d'(j)+1$, para todo arco (i,j) em $G(x)$.

Por fim, é claro que $d'(k) \leq d'(l)+1$, para todo arco (k,l), com $k \neq i \neq l$ e que $d'(t) = d(t) = 0$, pois $i \neq t$. Concluímos, assim, que d' é válida.

As observações acima também nos permitem concluir que $d'(k) \geq d(k)$ para todo vértice k da rede G. □

10.5 Descrição

Como vimos anteriormente, o método dos caminhos de aumento mantém um st-fluxo x e busca, em cada iteração, aumentar sua intensidade. Quando o algoritmo pára, existem propriedades que garantem que a intensidade do st-fluxo mantido não pode mais ser aumentada, ou seja, x é máximo.

O método do pré-fluxo, em contrapartida, mantém um pré-st-fluxo x e busca, em cada iteração, transformar x em um st-fluxo através de pushes e relabels. O algoritmo se baseia na existência de certas propriedades que garantem que, se o pré-st-fluxo x mantido pelo algoritmo é um st-fluxo, então x com certeza é um st-fluxo de intensidade máxima.

Sob o ponto de vista de programação linear, pode-se dizer que o método dos caminhos de aumento mantém viabilidade primal e busca viabilidade dual, enquanto o método do pré-fluxo faz exatamente o contrário.

O algoritmo iterativo abaixo recebe uma rede capacitada G e dois vértices s e t dessa rede e devolve um st-fluxo de intensidade máxima e um

st-separador de capacidade mínima em G. Cada iteração começa com um fluxo x e uma função-distância d. No início da primeira iteração, temos que $x_a = u_a$ para todo arco a em $\delta^-(s)$, $x_a = 0$ para todo arco a em $N_G \setminus \delta^-(s)$, $d(s) = n$ e $d(i) = 0$ para todo vértice i em $N_G \setminus \{s\}$.

Caso 1: Não existe um vértice ativo em N_G.
Seja k um inteiro entre 1 e $n-1$ tal que $d(i) \neq k$, para todo vértice i.
Seja S o conjunto de todos os vértices em G tais que $d(i) > k$.
Devolva x e S e pare.

Caso 2: Existe um vértice ativo em N_G.
Seja i um vértice ativo em N_G.

Caso 2A: Existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja a um arco admissível em $\delta^-_{G(x)}(i)$.
Seja x' o fluxo obtido após um push em a.
Comece nova iteração com x' no papel de x.

Caso 2B: Não existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja d' a função-distância obtida após um relabel de i.
Comece nova iteração com d' no papel de d.

Intuitivamente, o método do pré-fluxo pode ser visualizado como um escoamento de água, no qual os arcos representam canos e os rótulos representam a altura dos vértices. Arcos admissíveis permitem o escoamento do excesso de água em um vértice para outros vértices que tenham altura menor. Quando não existem tais arcos, a altura do vértice aumenta para forçar o escoamento. A figura 10.1 mostra uma simulação do método.

10.6 Correção e desempenho

Lema 10.4. *No início de cada iteração do método do pré-fluxo, vale que:*
(i) x é um pré-fluxo com fonte s e sorvedouro t;
(ii) d define rótulos-distância válidos;
(iii) $d(s) = n$ e $d(t) = 0$;
(iv) $e_x(t) \geq 0$.

Prova: É fácil ver que essas propriedades são válidas no início da primeira iteração. Vamos supor, então, que elas sejam válidas no início de uma iteração qualquer e mostrar que elas continuam válidas ao seu final.

Como uma iteração na qual ocorre o Caso 1 é a última, vamos assumir sem perda de generalidade que o Caso 2 ocorre, ou seja, que executa-se um push ou um relabel. Suponha que um push em $a = (i, j)$ é executado, obtendo o fluxo x'.

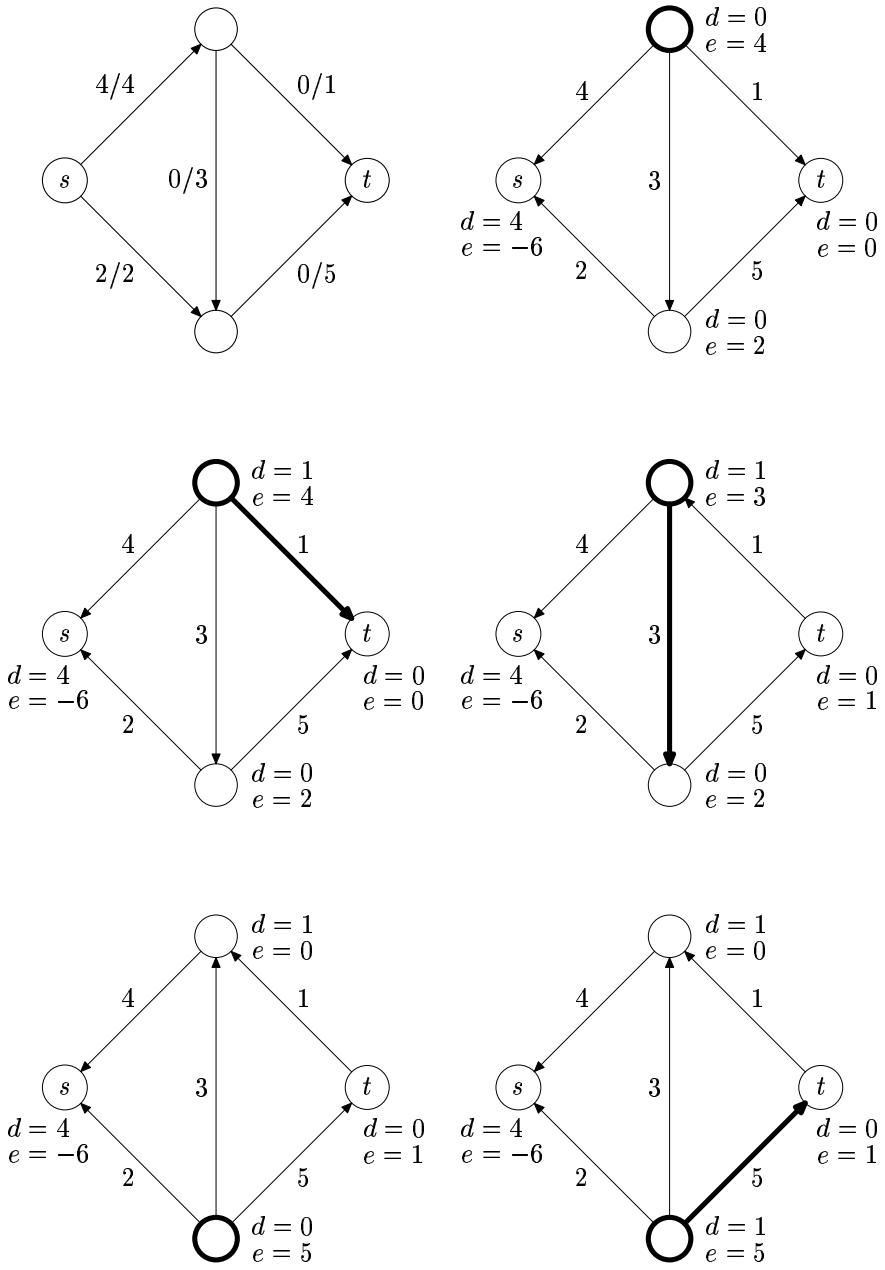

Figura 10.1: Simulação do método do pré-fluxo. O primeiro diagrama representa o pré-fluxo inicial e os seguintes indicam os pushes e relabels.

Sabemos que um push sempre produz um fluxo. Logo, temos que x' é um fluxo no final da iteração. Ademais, temos que x' satisfaz:

$$e_{x'}(i) = e_x(i) - \min\{r_a, e_x(i)\} \geq e_x(i) - e_x(i) = 0;$$

$$e_{x'}(j) = e_x(j) + \min\{r_a, e_x(i)\} \geq 0;$$

$$e_{x'}(k) = e_x(k), \text{ para todo } k \text{ em } N_G \setminus \{i, j\}.$$

Portanto, como x é um pré-fluxo em G com fonte s e sorvedouro t, então x' também o é. Suponha agora que um relabel seja executado, obtendo a função-distância d'. Sabemos, pelo Lema 10.3, que um relabel a partir de uma função-distância válida sempre produz uma função-distância válida. Portanto, temos que d' é uma função-distância válida ao final da iteração.

Desse modo, por indução no número de iterações, podemos concluir que as propriedades (i) e (ii) são válidas no início de cada iteração do método.

Como s e t são respectivamente a fonte e o sorvedouro do pré-fluxo x, então eles não são vértices ativos por definição. Logo, tais vértices nunca são selecionados pelo algoritmo no início do Caso 2 e, portanto, nunca sofrem relabel durante toda a execução do método. Assim, (iii) vale.

Para verificarmos (iv), notemos que, durante toda a execução do algoritmo, $x_a = 0$, para todo a em $\delta_G^-(t)$. De fato, se a está em $\delta_G^-(t)$, então $x_a = 0$ no início do método e, como t nunca é um vértice ativo, então a nunca sofre pushes e, conseqüentemente, nunca tem seu fluxo aumentado. Assim, em toda iteração temos que $e_x(t) = x(\delta_G^+(t)) - x(\delta_G^-(t)) = x(\delta_G^+(t))$. Por fim, pela validade da propriedade (i), temos que $x_a \geq 0$, para todo arco a da rede, o que nos permite concluir que $e_x(t) \geq 0$. □

As propriedades enunciadas no lema acima são invariantes do método do pré-fluxo. Vamos utilizá-los para mostrar que se x é o pré-st-fluxo devolvido pelo método, então x é um st-fluxo máximo. Para tanto, utilizaremos o separador S, também devolvido pelo algoritmo, como um certificado.

Lema 10.5. *Seja G uma rede capacitada, seja x o fluxo devolvido pelo método do pré-fluxo e seja S o separador também devolvido pelo método. Então x é um st-fluxo em G, S é um st-separador de G e* $\text{val}(x) = \text{cap}(S)$.

Prova: Pelo invariante (i), x é um pré-st-fluxo. Ademais, as condições do Caso 1 implicam que não existem vértices ativos em relação a x. Ou seja, $e_x(i) = 0$ para todo vértice i em $N_G \setminus \{s, t\}$. Logo, x é um st-fluxo.

Além disso, como o invariante (iii) garante que $d(s) = n$ e que $d(t) = 0$ e como o inteiro k é escolhido em um intervalo de tamanho $n - 1$ e existem apenas $n - 2$ vértices em $N_G \setminus \{s, t\}$, então a existência de k está garantida. Ademais, também pelo invariante (iii), temos que S é um st-separador.

Pelo Lema 3.3, sabemos que $\text{val}(x) = x(\delta_G^-(S)) - x(\delta_G^+(S))$. Logo, para concluir o resultado, basta mostrarmos que $x(\delta_G^-(S)) - x(\delta_G^+(S)) = \text{cap}(S)$.

Seja $a = (i, j)$ um arco em $\delta_G^-(S)$. Pela construção de S, sabemos que $d(i) > k$ e $d(j) < k$. Como pelo invariante (ii) a função-distância d é válida, então a não pode ser um arco de $G(x)$. Portanto, temos que $r_a = 0$, o que implica que $x_a = u_a$. Concluímos, assim, que $x(\delta_G^-(S)) = u(\delta_G^-(S))$.

Seja $a = (i, j)$ um arco em $\delta_G^+(S)$. Suponha que $x_a > 0$. Nesse caso, o arco $\overline{a} = (j, i)$ está em $\delta_{G(x)}^-(S)$. Entretanto, temos que $d(j) > k$ e $d(i) < k$ pela construção do separador S. Mas então, como a função-distância d é válida, $\overline{a}$ não pode ser um arco de $G(x)$, o que é uma contradição. Logo, $x_a = 0$ e vale que $x(\delta_G^+(S)) = 0$. Portanto, concluímos que

$$\text{val}(x) = x(\delta_G^-(S)) - x(\delta_G^+(S)) = u(\delta_G^-(S)) = \text{cap}(S).$$

□

Corolário 10.6. *Seja G uma rede capacitada, seja x o fluxo devolvido pelo método dos caminhos de aumento e seja S o separador também devolvido pelo método. Então x tem intensidade máxima e S tem capacidade mínima.*

Prova: O resultado segue diretamente do Lema 10.5 e do Corolário 4.6. □

Os resultados obtidos provam que se o método do pré-fluxo pára, então ele devolve um fluxo máximo. Para concluir a correção, vamos mostrar agora que o método sempre pára após um número finito de iterações.

Lema 10.7. *Durante toda a execução do método do pré-fluxo, existe um caminho entre cada vértice ativo e a fonte s na rede residual $G(x)$.*

Prova: Sabemos que $e(t) \geq 0$ pelo invariante (iv). Se a rede G possui um vértice ativo, então, pelo Lema 10.1, temos que $e(s) < 0$. Ademais, s é o único vértice com excesso negativo em relação a x. Sendo assim, pelo Lema 3.5, deve existir um caminho P entre s e i em G tal que $x_a > 0$ para todo arco a em P. Logo, todos os arcos irmãos dos arcos de P estão na rede residual $G(x)$ e, portanto, existe um caminho entre i e s em $G(x)$. □

Lema 10.8. *Durante toda a execução do método do pré-fluxo, vale que $d(i) < 2n$ para todo vértice ativo i.*

Prova: Sabemos que a função-distância d é válida no início de cada iteração. Em função disso, pelo Lema 10.2, temos que $d(i) - d(s)$ é um limitante inferior para a distância entre i e s na rede residual $G(x)$. Sendo assim, devemos ter $d(i) - d(s) \leq n - 1$, uma vez que, pelo Lema 10.7, existe um caminho entre i e s em $G(x)$ e, por definição, qualquer caminho tem no máximo $n-1$ arcos. Ademais, como sabemos que $d(s) = n$ pelo invariante (iii) do método, podemos concluir que $d(i) < 2n$. □

Proposição 10.9. *São executados no máximo $2n^2$ relabels durante todo o método do pré-fluxo.*

Prova: Quando um relabel ocorre, o rótulo do vértice i é acrescido em pelo menos uma unidade. De fato, nesse caso temos que não existem arcos admissíveis em $\delta^-_{G(x)}(i)$ e que, portanto, $d(i) < d(j) + 1$, para todo arco (i, j) em $\delta^-_{G(x)}(i)$. Cabe ressaltar que nesse momento $\delta^-_{G(x)}(i)$ é não vazio, uma vez que um relabel sempre ocorre sobre um vértice ativo. Assim, devemos ter $d'(i) = d(k)+1$, para algum arco (i, k) em $\delta^-_{G(x)}(i)$, de forma que $d'(i) > d(i)$. Conseqüentemente, e como os rótulos são todos inteiros, $d'(i) \geq d(i) + 1$.

Além disso, pelo Lema 10.8, temos que $d(i) < 2n$, para todo vértice ativo i, e apenas os rótulos de vértices ativos são alterados durante o método. Deste modo, e como $d(i) \geq 0$ para todo vértice i no início da primeira iteração, podem ocorrer no máximo $2n$ relabels de cada vértice. Disso concluímos que o número total de relabels é limitado por $2n^2$. □

Lema 10.10. *São executados no máximo n pushes saturadores sobre um determinado arco durante todo o método do pré-fluxo.*

Prova: Seja $a = (i, j)$ o arco admissível que sofre um push saturador durante determinada iteração k_α do método. Suponha que exista uma outra iteração k_γ, com $k_\gamma > k_\alpha$, em que a seja novamente saturado.

Sabemos que o arco a não existe na rede residual da iteração $k_\alpha+1$. Mas, como a é um arco da rede residual na iteração k_γ, existe uma iteração k_β, com $k_\alpha < k_\beta < k_\gamma$, na qual o arco $\overline{a} = (j, i)$ sofreu um push. Podemos observar que isso implica que $\overline{a}$ é um arco admissível na iteração k_β.

Denotemos por $d_k(i)$ o rótulo-distância do vértice i na iteração k. Pelo Lema 10.3, se d' é a função-distância obtida após um relabel de um vértice i, a partir de uma função-distância válida d, então $d'(i) \geq d(i)$, para todo vértice i. Logo, temos que $d_{k_\alpha}(i) = d_{k_\alpha}(j)+1$, $d_{k_\beta}(j) = d_{k_\beta}(i)+1$ e $d_{k_\beta}(j) \geq d_{k_\alpha}(j)$. Conseqüentemente, $d_{k_\alpha}(i) \leq d_{k_\beta}(j) + 1 = d_{k_\beta}(i) + 2$. Ou seja, na iteração k_γ, quando o arco a volta a ser saturado, o rótulo do vértice i foi acrescido em pelo menos 2 desde a iteração k_α.

Notemos, então, que, pelo Lema 10.8, o rótulo de qualquer vértice ativo i é sempre estritamente menor do que $2n$ e que $d(i) \geq 0$ para todo vértice i no início da primeira iteração. Disso concluímos que o arco a pode ser saturado no máximo $2n/2 = n$ vezes durante todo o método do pré-fluxo. □

Proposição 10.11. *São executados no máximo $2nm$ pushes saturadores durante todo o método do pré-fluxo.*

Prova: Como pelo Lema 10.10 cada arco sofre no máximo n pushes saturadores e como a rede residual tem no máximo o dobro de arcos que a rede original, segue que o total de pushes saturadores não passa de $2nm$. □

Proposição 10.12. *São executados $O(n^2m)$ pushes não-saturadores durante todo o método do pré-fluxo.*

Prova: Vamos recorrer a uma técnica de análise amortizada e fazer uma prova baseada em uma função potencial. Seja I o conjunto de vértices tal que $I = \{i \in N : i \text{ é vértice ativo}\}$ e seja $\Phi = \sum_{i \in I} d(i)$. No início da primeira iteração, temos que $d(i) = 0$, para todo vértice ativo i, e, portanto, $\Phi = 0$. Quando o algoritmo termina, temos $I = \emptyset$ e portanto devemos ter também $\Phi = 0$. Vamos verificar como cada tipo de operação altera Φ.

Pelo Lema 10.8, o rótulo de um vértice é sempre inferior a $2n$. Além disso, os rótulos dos vértices são acrescidos apenas durante as operações relabel. Portanto, relabels fazem o valor de Φ aumentar em no máximo $2n^2$.

Um push saturador sobre um arco admissível $a = (i, j)$ pode tornar o vértice j ativo e, pelo Lema 10.8, sabemos que $d(j) < 2n$. Sendo assim, um push saturador aumenta o valor de Φ em não mais do que $2n$. Logo, pela Proposição 10.11, podemos concluir que os pushes saturadores fazem o valor de Φ aumentar em no máximo $4n^2m$.

Quando ocorre um push não-saturador sobre um arco admissível a, o valor de Φ decresce em pelo menos 1. De fato, nesse caso o vértice i, selecionado no início da iteração, deixa de ser ativo e o vértice j, ponta final de a, pode tornar-se ativo. Mas, como a é um arco admissível, temos que $d(i) = d(j) + 1$ e, portanto, o valor de Φ decresce em pelo menos 1.

Temos, então, que $\Phi = 0$ no início da primeira iteração e que tal valor aumenta em no máximo $2n^2 + 4n^2m$ durante toda a execução do método. Ademais, sabemos que quando o método pára temos $\Phi = 0$ e que cada push não-saturador reduz o valor da função Φ em pelo menos 1.

Disso concluímos que podem ocorrer no máximo $2n^2 + 4n^2m = O(n^2m)$ pushes não-saturadores durante toda a execução do método. □

Proposição 10.13. *O método do pré-fluxo encontra um fluxo máximo em $O(n^2m)$ iterações.*

Prova: Em cada iteração do método do pré-fluxo, exceto a última, ocorre um relabel, um push saturador ou um push não-saturador. Assim, como conseqüência das Proposições 10.9, 10.11 e 10.12, temos que o método pára após no máximo $2n^2 + 2nm + 2n^2 + 4n^2m = O(n^2m)$ iterações. Logo, pelo Lema 10.5, o método encontra um fluxo máximo em $O(n^2m)$ iterações. □

Uma observação interessante que podemos fazer a partir dos resultados acima é que demonstramos que o método do pré-fluxo tem desempenho polinomial mesmo quando descrito de maneira genérica. Esse fato representa uma vantagem em relação ao método dos caminhos de aumento.

Vamos demonstrar agora alguns resultados que são necessários para limitar o tempo total de execução do método do pré-fluxo.

Proposição 10.14. *Durante a execução do método do pré-fluxo, o tempo total gasto por relabels é $O(nm)$.*

Prova: Considere um relabel feito sobre um vértice i. Para obter o valor $\min\{d(j)+1 : (i,j)$ é arco de $\delta^-_{G(x)}(i)\}$, é preciso varrer $\delta^-_{G(x)}(i)$.

Pela Proposição 10.9, sabemos que um vértice sofre no máximo $2n$ relabels durante a execução do algoritmo. Isso implica que o tempo total gasto por relabels é $O(\sum_{i\in N_G} 2n|\delta^-_{G(x)}(i)|) = O(2n\sum_{i\in N_G}|\delta^-_{G(x)}(i)|)$.

Como um arco de $G(x)$ não pode ter mais do que uma ponta inicial, temos que se j e k são vértices distintos, então $\delta^-_{G(x)}(j)$ e $\delta^-_{G(x)}(k)$ são disjuntos. Ademais, para todo arco a de $G(x)$ existe um vértice j tal que a está em $\delta^-_{G(x)}(j)$. Logo, $O(2n\sum_{i\in N_G}|\delta^-_{G(x)}(i)|) = O(2n|A_{G(x)}|) = O(nm)$. □

Lema 10.15. *Se um arco $a=(i,j)$ não é admissível no final de uma iteração k_α do método do pré-fluxo, então ele é admissível no início de uma iteração k_γ, com $k_\alpha < k_\gamma$, somente se existe uma iteração k_β, com $k_\alpha < k_\beta < k_\gamma$, na qual i sofre um relabel.*

Prova: Denotemos por $d_k(i)$ o rótulo-distância do vértice i na iteração k.

Se um arco a não é admissível no final da iteração k_α, então a não está em $G(x)$ nessa iteração ou $d_{k_\alpha}(i) < d_{k_\alpha}(j)+1$. Se o primeiro caso ocorre, então para que a esteja em $G(x)$ na iteração k_γ, é preciso ocorrer um push sobre $\overline{a}$ em alguma iteração $k_{a'}$, com $k_\alpha < k_{a'} < k_\gamma$. Mas isso implica que $d_{k_{a'}}(j) = d_{k_{a'}}(i)+1$, ou seja, $d_{k_{a'}}(i) < d_{k_{a'}}(j)+1$. Logo, podemos assumir sem perda de generalidade que só precisamos considerar o segundo caso.

Como o Lema 10.3 implica que $d_{k_\gamma}(i) \geq d_{k_\alpha}(i)$ e $d_{k_\gamma}(j) \geq d_{k_\alpha}(j)$, deve existir uma iteração k_β, com $k_\alpha < k_\beta < k_\gamma$, na qual i sofre um relabel. □

Proposição 10.16. *Durante a execução do método do pré-fluxo, o tempo total gasto para saber qual operação deve ser feita em cada iteração é $O(nm)$.*

Prova: Um vértice i deve sofrer relabel em uma iteração se nenhum arco de $\delta^-_{G(x)}(i)$ é admissível nessa iteração. Portanto, em uma análise de pior caso temos que o conjunto $\delta^-_{G(x)}(i)$ deve ser varrido em todas as iterações. Porém, o Lema 10.15 implica que se um arco foi verificado como não admissível, ele não precisa ser verificado novamente até o vértice i sofrer um relabel.

Logo, como a Proposição 10.9 implica que um vértice sofre no máximo $2n$ relabels, temos que o tempo total gasto para descobrir se os vértices ativos devem sofrer push ou relabel é $O(2n\sum_{i\in N_G}|\delta^-_{G(x)}(i)|)$ e verificamos anteriormente que $O(\sum_{i\in N_G} 2n|\delta^-_{G(x)}(i)|) = O(2n\sum_{i\in N_G}|\delta^-_{G(x)}(i)|) = O(nm)$. □

Para concluir a análise, resta verificar o tempo consumido em cada iteração pelo processo de buscar um vértice ativo. Nos capítulos seguintes, são apresentadas implementações específicas do método que utilizam estruturas de dados capazes de realizar esse processo em tempo constante. Tais implementações também permitem obter melhores limitantes para o número total de iterações.

Capítulo 11

Fila de vértices ativos

Em capítulos anteriores, vimos que o algoritmo dos caminhos de aumento de comprimento mínimo, uma das implementações mais simples do método dos caminhos de aumento, é um algoritmo eficiente. Para o método do pré-fluxo, também é possível demonstrar que uma de suas implementações mais simples pode resultar em uma melhoria significativa de desempenho.

A idéia, adaptada por Goldberg de um algoritmo de Schiloach e Vishkin [20], consiste simplesmente em ordenar os vértices ativos em uma fila comum do tipo FIFO (first-in-first-out). Por esse motivo, o algoritmo resultante da idéia é conhecido como **algoritmo da fila de vértices ativos**.

Este capítulo descreve o algoritmo e demonstra sua complexidade.

11.1 Descrição

O algoritmo da fila de vértices ativos é a implementação do método do pré-fluxo que utiliza uma fila FIFO para ordenar os vértices ativos e escolher aquele que deve ser processado durante determinada iteração.

A idéia é executar pushes saturadores sobre arcos admissíveis incidentes ao primeiro vértice da fila até que o mesmo deixe de ser ativo ou sofra um relabel. Além disso, os vértices que se tornam ativos durante a execução do algoritmo são sempre inseridos no final da fila.

Como o método geral, o algoritmo da fila de vértices ativos recebe uma rede capacitada G e dois vértices s e t, e devolve um st-fluxo de intensidade máxima e um st-separador de capacidade mínima nessa rede. Cada iteração começa com um fluxo x e uma função-distância d.

No início da primeira iteração, x é o fluxo tal que $x_a = u_a$, para todo arco a em $\delta_G^-(s)$ e $x_a = 0$, para todo arco a em $N_G \setminus \delta_G^-(s)$. Além disso, d é a função-distância tal que $d(s) = n$ e $d(i) = 0$, para todo vértice i em $N_G \setminus \{s\}$. Todos os vértices que são ponta final de um arco em $\delta_G^-(s)$ são inseridos na fila em qualquer ordem antes do início da primeira iteração.

A descrição do algoritmo está a seguir.

Caso 1: Não existe um vértice ativo em N_G.
Seja k um inteiro entre 1 e $n-1$ tal que $d(i) \neq k$, para todo vértice i.
Seja S o conjunto de todos os vértices em G tais que $d(i) > k$.
Devolva x e S e pare.

Caso 2: Existe um vértice ativo em N_G.
Seja i o vértice ativo em N_G que está no início da fila.

Caso 2A: Existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja $a = (i, j)$ um arco admissível em $\delta^-_{G(x)}(i)$.
Seja x' o fluxo obtido após um push em a.
Se o vértice j tornou-se ativo, insira-o na fila.
Se o vértice i deixou de ser ativo, remova-o da fila.
Comece nova iteração com x' no papel de x.

Caso 2B: Não existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja d' a função-distância obitida após um relabel de i.
Remova o vértice i da fila e o insira ao seu final.
Comece nova iteração com d' no papel de d.

11.2 Desempenho

Durante a execução do algoritmo, um mesmo vértice ativo é selecionado no início de uma série de iterações consecutivas até que deixe de ser ativo ou sofra um relabel. Definimos um **processamento** de um vértice como uma dessas seqüências de iterações consecutivas em que esse mesmo vértice está no início da fila de vértices ativos, terminada sempre por um relabel ou um push que torna o vértice inativo. Note que durante um processamento ocorre no máximo um relabel e no máximo um push não-saturador.

Definimos também um conjunto de **fases** do algoritmo, como segue:

- A fase 1 corresponde ao processamento dos vértices que foram inseridos na fila antes do início da primeira iteração.
- A fase k, para todo $k > 1$, corresponde ao processamento dos vértices que foram inseridos na fila durante a execução da fase $k-1$.

Cada vértice deve passar por no máximo um processamento durante uma fase. Sendo assim, se um vértice sofre um relabel durante o seu processamento em determinada fase e é movido para o final da fila, então o seu próximo processamento não deve ser incluído nessa mesma fase.

Por definição, durante cada processamento de um vértice ocorre no máximo um push não-saturador. Logo, o algoritmo efetua no máximo n pushes não-saturadores durante uma fase.

Dessa forma, se estabelecermos um limitante para o número de fases efetuadas pelo algoritmo, teremos estabelecido também um limitante para o número de pushes não-saturadores efetuados.

Lema 11.1. *O algoritmo da fila de vértices ativos executa $O(n^2)$ fases.*

Prova: Vamos fazer uma análise amortizada com o auxílio da função potencial $\Phi := \max\{d(i) : i \text{ vértice ativo}\}$. Para tanto, vamos verificar como varia o valor de Φ no decorrer de cada fase do algoritmo.

Suponha, inicialmente, que pelo menos um relabel ocorra durante determinada fase. Pelo Lema 10.8, temos que o rótulo de qualquer vértice é no máximo $2n$ e, portanto, Φ pode sofrer um acréscimo de no máximo $2n$ durante esta fase. Sendo assim, o valor de Φ recebe um acréscimo total de no máximo $2n^2$ após todas as fases em que ocorrem relabels.

Suponha agora que não ocorram relabels durante uma fase. Temos, então, que o processamento de cada vértice termina com um push que o torna inativo. Neste caso, o excesso de cada vértice processado é transferido completamente para vértices com rótulos menores, uma vez que os pushes ocorrem sempre em arcos admissíveis. Desse modo, Φ decresce em pelo menos uma unidade durante uma fase em que não ocorrem relabels.

Temos, portanto, que o valor inicial de Φ é zero e que Φ recebe um acréscimo máximo de $2n^2$ ao longo de todo o algoritmo. Ademais, quando o algoritmo termina devemos ter $\Phi = 0$, pois não existem vértices ativos.

Sabemos também que todas as fases sem relabels, e apenas elas, diminuem o valor de Φ. Mais ainda, sabemos que essas fases diminuem o valor de Φ em pelo menos uma unidade. Disso concluímos que ocorrem no máximo $2n^2$ fases sem relabels. Além disso, o número de fases em que ocorrem relabels é limitado por $2n^2$, uma vez que, pela Proposição 10.9, esse é o limitante para o número de relabels durante o método do pré-fluxo.

Assim, o algoritmo executa no máximo $4n^2 = O(n^2)$ fases. □

Proposição 11.2. *O algoritmo da fila de vértices ativos executa $O(n^3)$ pushes não-saturadores.*

Prova: Segue da definição de fase que o algoritmo executa no máximo n pushes não-saturadores durante uma fase. Logo, como conseqüência do Lema 11.1, o algoritmo executa $O(n^3)$ pushes não-saturadores. □

Corolário 11.3. *O algoritmo da fila de vértices ativos encontra um fluxo máximo em $O(n^3 + nm)$ iterações.*

Prova: Como o algoritmo é uma implementação do método do pré-fluxo, então exatamente uma dentre as operações push saturador, push não-saturador e relabel ocorre a cada iteração, com exceção da última. Assim, o resultado segue diretamente das Proposições 10.9, 10.11 e 11.2 e do Lema 10.5. □

Por fim, apresentamos um limitante para o consumo de tempo de uma possível implementação do algoritmo.

Proposição 11.4. *O algoritmo da fila de vértices ativos pode ser implementado de modo a executar em tempo $O(n^3 + nm)$.*

Prova: Pelas Proposições 10.16 e 10.15, o tempo total consumido para a escolha de cada operação a ser realizada e para a execução dos relabels é $O(nm)$.

Além disso, como uma operação push pode ser realizada em tempo constante, então, pelas Proposições 10.11 e 11.2, o tempo total consumido para a execução de todos os pushes é $O(n^3 + nm)$.

Para concluir o resultado, basta observarmos que uma estrutura de dados FIFO pode ser facilmente implementada (cf. Cormen, Leiserson, Rivest e Stein [6]) de forma que a escolha de um vértice ativo no início de cada iteração possa sempre ser realizada em tempo constante. □

Capítulo 12

Vértices ativos de maior rótulo

Uma idéia razoavelmente intuitiva para escolher os vértices ativos na implementação do método do pré-fluxo é escolher o vértice ativo com o maior rótulo possível, na tentativa de reduzir a probabilidade de que o fluxo enviado na iteração atual retorne ao vértice em iterações posteriores.

Na verdade, este critério de escolha foi sugerido pelos próprios Goldberg e Tarjan [14] quando eles idealizaram o método do pré-fluxo e resultou no desenvolvimento do **algoritmo dos vértices ativos de maior rótulo**, um algoritmo de excelente desempenho, tanto teórico como prático.

Este capítulo descreve o algoritmo e demonstra sua complexidade.

12.1 Descrição

A única diferença entre o algoritmo dos vértices ativos de maior rótulo e a versão genérica do método do pré-fluxo é o fato de que o algoritmo dos vértices de maior rótulo sempre escolhe o vértice ativo com o maior rótulo-distância possível dentre todos os vértices ativos existentes.

O algoritmo iterativo abaixo recebe uma rede capacitada G e dois vértices s e t dessa rede e devolve um st-fluxo de intensidade máxima e um st-separador de capacidade mínima em G. Cada iteração começa com um fluxo x e uma função-distância d. No início da primeira iteração, temos que $x_a = u_a$ para todo arco a em $\delta^-(s)$, $x_a = 0$ para todo arco a em $N_G \setminus \delta^-(s)$, $d(s) = n$ e $d(i) = 0$ para todo vértice i em $N_G \setminus \{s\}$.

Caso 1: Não existe um vértice ativo em N_G.
Seja k um inteiro entre 1 e $n-1$ tal que $d(i) \neq k$, para todo vértice i.
Seja S o conjunto de todos os vértices em G tais que $d(i) > k$.
Devolva x e S e pare.

Caso 2: Existe um vértice ativo em N_G.
Seja i um vértice ativo em N_G tal que $d(i)$ é máximo.

Caso 2A: Existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja a um arco admissível em $\delta^-_{G(x)}(i)$.
Seja x' o fluxo obtido após um push em a.
Comece nova iteração com x' no papel de x.

Caso 2B: Não existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja d' a função-distância obtida após um relabel de i.
Comece nova iteração com d' no papel de d.

12.2 Desempenho

Apresentaremos agora alguns resultados que permitem obter um limitante superior muito bom para o número de iterações do algoritmo dos vértices ativos de maior rótulo. A análise, entretanto, é razoavelmente sofisticada e exige a definição prévia de algumas notações auxiliares.

Dada uma iteração qualquer do algoritmo dos vértices ativos de maior rótulo, um **arco corrente** de um vértice i é um arco admissível em $\delta^-(i)$ que seria escolhido pelo algoritmo para sofrer um push se i fosse o vértice ativo escolhido pelo algoritmo nessa iteração. Como o critério para escolher qual arco admissível sofre um push não é relevante para a correção do algoritmo e não será relevante nesta análise, podemos assumir sem perda de generalidade que tal critério não depende do excesso no vértice escolhido. Logo, um vértice que possui um arco corrente não é necessariamente ativo.

A rede formada por todos os vértices da rede original e por seus respectivos arcos correntes em uma determinada iteração do algoritmo dos vértices ativos de maior rótulo é chamada de **rede corrente** dessa iteração.

Seja F a rede corrente de uma iteração do algoritmo. Se existe um caminho entre um vértice i e um vértice j em F, com $i \neq j$, então i é **descendente** de j e j é **ancestral** de i. Denota-se por $D(i)$ o conjunto de todos os descendentes de i em F. Finalmente, um vértice ativo i é um **vértice ativo maximal** se $D(i)$ não contém nenhum vértice ativo.

Lema 12.1. *Seja i um vértice ativo de maior rótulo em uma dada iteração do algoritmo. Então, i é um vértice ativo maximal nessa iteração.*

Prova: Como uma rede corrente é formada somente por arcos admissíveis e sabemos que um arco admissível $a = (j, k)$ é tal que $d(j) = d(k) + 1$, temos que o rótulo-distância de qualquer vértice em $D(i)$ é maior que o rótulo de i. Portanto, nenhum vértice ativo pode estar em $D(i)$, pois i é um vértice ativo de maior rótulo. Isso implica que i é um vértice ativo maximal. □

Lema 12.2. *Sejam i e j dois vértices ativos maximais distintos em uma dada iteração do algoritmo. Então, os conjuntos $D(i)$ e $D(j)$ são disjuntos.*

Prova: Suponha por contradição que existe um vértice k tal que k é descendente tanto de i como de j na rede corrente. Por definição, existe um caminho P_1 entre k e i e um caminho P_2 entre k e j na rede corrente.

Pela definição de arco corrente, temos que um vértice é ponta inicial de no máximo um arco da rede corrente. Como a origem de P_1 e P_2 é a mesma, isso implica que P_1 está contido em P_2 ou P_2 está contido em P_1.

Portanto, temos que i é descendente de j ou j é descendente de i. Mas isso é uma contradição, pois ambos os vértices são ativos maximais. Logo, podemos concluir que os conjuntos $D(i)$ e $D(j)$ são disjuntos. □

Vamos fazer novamente uma análise amortizada baseada em função potencial. Por conveniência de notação, vamos definir que F denota a rede corrente ao longo de toda a execução do algoritmo. Denote por H o conjunto de todos os vértices ativos maximais de F. Considere a função $\Phi = \sum_{i \in H} \Phi(i)$, onde $\Phi(i) = \max\{0, K - |D(i)|\}$ e K é um valor positivo qualquer.

Lema 12.3. *Seja i o vértice ativo escolhido pelo algoritmo em uma iteração do algoritmo dos vértices ativos de maior rótulo. Então, é válido que:*
(i) Um relabel aumenta Φ em no máximo K.
(ii) Um push saturador aumenta Φ em no máximo K.
(iii) Um push não-saturador não aumenta Φ se $|D(i)| \geq K$.
(iv) Um push não-saturador diminui Φ em pelo menos 1 se $|D(i)| < K$.

Prova: Para todo vértice j da rede, temos que $|D(j)| \geq 0$, ou seja, temos que $\Phi(j) \leq K$. Como também sabemos que $\Phi(j) \geq 0$, então podemos concluir que K é um limitante superior para o quanto $\Phi(j)$ pode aumentar.

Temos por definição que, dado um vértice j, o valor de $\Phi(j)$ pode aumentar quando seu número de descendentes diminui. Logo, o valor da função potencial Φ pode aumentar somente quando o número de descendentes de um vértice diminui ou quando um vértice se torna ativo maximal.

Suponha que na dada iteração ocorre um relabel de i. Após esse relabel, alguns arcos de $\delta(i)$ podem ter sido adicionados a F e alguns arcos de $\delta(i)$ podem ter sido removidos de F. Primeiramente, podemos observar que a adição de arcos a F não pode diminuir o número de descendentes de nenhum vértice. Ademais, como o excesso dos vértices não se altera após um relabel, a adição também não pode fazer com que algum vértice se torne ativo maximal. Portanto, a adição de arcos não faz Φ aumentar.

Podemos observar também que, pelas condições necessárias para a execução de um relabel, não existem arcos de $\delta^-(i)$ em F antes do relabel. Portanto, precisamos apenas analisar o caso em que arcos de $\delta^+(i)$ são removidos de F. Como verificamos que não existem arcos de $\delta^-(i)$ em F antes do relabel, temos que somente o número de descendentes de i pode ser alterado pelo relabel. Logo, como o valor de $\Phi(i)$ pode aumentar em no máximo K, temos que um relabel aumenta Φ em no máximo K.

Suponha agora que na dada iteração ocorre um push sobre um arco a. Sabemos pela descrição do algoritmo que a está em F, $a = (i, j)$ e $i \neq j$.

Se o push é saturador, o arco a deixa de ser um arco admissível e, portanto, a operação remove o arco a de F. Logo, algum vértice que era ancestral de i antes do push pode se tornar um vértice ativo maximal após o push. Por definição, sabemos que um vértice é ponta inicial de no máximo um arco corrente. Portanto, qualquer vértice que era ancestral de i antes do push é ancestral de j após o push. Isso implica, pelo Lema 12.2, que no máximo um vértice pode se tornar ativo maximal. Como verificamos anteriormente que $\Phi(k) \leq K$ para qualquer vértice k da rede, podemos concluir que um push saturador é capaz de aumentar Φ em no máximo K.

Se o push é não-saturador, sabemos que a rede F não se altera, pois o arco a permanece na rede residual e, portanto, em F. Ademais, o excesso de i torna-se nulo, o que implica que i deixa de ser um vértice ativo. Como i é um vértice ativo de maior rótulo antes do push, temos pelo Lema 12.1 que i deixa de ser um vértice ativo maximal. Logo, algum ancestral de i pode se tornar ativo maximal após o push. Mas, novamente pelo Lema 12.2, isso pode acontecer com no máximo um vértice, que denotaremos por k. Ademais, como i é descendente de k, temos que $|D(i)| < |D(k)|$.

Portanto, se $\Phi(i) > 0$ então $\Phi(i) > \Phi(k)$. Por definição, temos que $\Phi(i) > 0$ se $|D(i)| < K$. Logo, podemos concluir que um push saturador diminui Φ em pelo menos 1 se $|D(i)| < K$. Por outro lado, se $\Phi(i) = 0$ então $\Phi(i) = \Phi(k)$. Por definição, temos que $\Phi(i) = 0$ se $|D(i)| \geq K$. Logo, podemos concluir que um push saturador não aumenta Φ se $|D(i)| \geq K$. □

Para apresentar o resultado a seguir, mais algumas notações serão necessárias. Considere uma **fase** do algoritmo como sendo uma seqüência de iterações durante a qual não ocorrem relabels. Diz-se que uma fase é **barata** se durante suas iterações são executados no máximo $2n/K$ pushes não-saturadores e diz-se que uma fase é **cara** caso contrário, ou seja, se durante suas iterações são executados pelo menos $2n/K$ pushes não-saturadores.

Proposição 12.4. *São executados $O(n^2\sqrt{m} + n^3/\sqrt{m})$ pushes não-saturadores durante todo o algoritmo dos vértices de maior rótulo.*

Prova: Por definição, o número de fases é limitado pelo número total de relabels. Pela Proposição 10.9, o algoritmo executa no máximo $2n^2$ relabels, ou seja, o número de fases é $O(n^2)$. Logo, podemos concluir por definição que $O(n^3/K)$ pushes não-saturadores são executados em fases baratas.

Como o algoritmo sempre escolhe os vértices ativos de maior rótulo, temos que se um vértice escolhido i torna-se inativo durante uma fase, então i permanece inativo até o final dessa fase, pois a maximalidade do rótulo de i dentre os vértices ativos e a ausência de relabels garantem que até o final da fase não existe nenhum arco admissível (j, i) tal que j é vértice ativo.

Por definição, temos que em uma fase cara são executados pelo menos $2n/K$ pushes não-saturadores. Pelo Lema 12.1, temos que todos esses pushes são executados sobre arcos cuja ponta inicial é um vértice ativo maximal. Finalmente, pelo Lema 12.2, temos que o número de vértices ativos maximais da rede com pelo menos K descendentes é no máximo n/K.

Como um push não-saturador torna um vértice inativo e vimos que esse vértice permanece inativo até o final da fase, podemos concluir que, durante uma fase, $O(n/K)$ pushes não-saturadores são executados sobre arcos cuja ponta inicial é um vértice ativo maximal i tal que $|D(i)| \geq K$. Resta encontrar um limitante para o caso em que i é tal que $|D(i)| < K$.

Pelo Lema 12.3, sabemos que se um push não-saturador é executado sobre um arco cuja ponta inicial é um vértice i, então esse push reduz Φ em pelo menos 1 se $|D(i)| < K$ e não aumenta Φ se $|D(i)| \geq K$. Pelo mesmo lema, temos que um push saturador aumenta Φ em no máximo K e um relabel aumenta Φ em no máximo K. Finalmente, pela Proposição 10.11, temos que o algoritmo executa $O(nm)$ pushes saturadores e verificamos anteriormente que o algoritmo executa $O(n^2)$ relabels. Logo, temos que as fases caras executam $O(n/K + nmK + n^2K)$ pushes não-saturadores.

Para concluir a demonstração, devemos atribuir um valor positivo adequado à constante K. Note que o valor n^3/K diminui conforme K aumenta e o valor $n/K + nmK + n^2K$ aumenta conforme K aumenta. K deve ser escolhido de forma a balancear satisfatoriamente as duas funções.

Uma escolha adequada para uma boa análise é $K = n/\sqrt{m}$. Temos nesse caso que $O(n^3/K) = O(n^2\sqrt{m})$ pushes não-saturadores são executados em fases baratas e $O(n/K + nmK + n^2K) = O(\sqrt{m} + n^2\sqrt{m} + n^3/\sqrt{m})$ pushes não-saturadores são executados em fases caras. Portanto, temos que o algoritmo executa $O(n^2\sqrt{m}+n^3/\sqrt{m})$ pushes não-saturadores no total. □

Proposição 12.5. *O algoritmo dos vértices ativos de maior rótulo encontra um fluxo máximo em $O(nm + n^2\sqrt{m} + n^3/\sqrt{m})$ iterações.*

Prova: Pelas Proposições 10.9, 10.11 e 12.4, temos que o algoritmo pára após $O(n^2 + nm + n^2\sqrt{m} + n^3/\sqrt{m}) = O(nm + n^2\sqrt{m} + n^3/\sqrt{m})$ iterações. □

Pela Proposição 10.14, sabemos que o tempo total gasto por relabels é $O(nm)$ e pela Proposição 10.16 sabemos que o tempo total gasto na escolha das operações durante as iterações do algoritmo é $O(nm)$. Ademais, Ahuja, Magnanti e Orlin [1] mostraram que é possível utilizar uma estrutura de dados especial para que as buscas pelos vértices ativos de maior rótulo consumam tempo $O(nm)$ no total. Disso segue o resultado abaixo.

Proposição 12.6. *O algoritmo dos vértices ativos de maior rótulo pode ser implementado de modo a executar em tempo $O(nm + n^2\sqrt{m} + n^3/\sqrt{m})$.*

Capítulo 13

Excess scaling

Scaling é uma idéia simples e poderosa que pode ser aplicada a diferentes algoritmos. Em capítulos anteriores vimos como essa idéia pode ser utilizada para obter uma implementação polinomial para o método dos caminhos de aumento. Vamos estudar agora um outro algoritmo obtido através da aplicação da mesma técnica, desta vez ao método do pré-fluxo.

O algoritmo **excess scaling**, idealizado por Ahuja e Orlin [2], baseia-se na idéia de priorizar vértices ativos com excesso alto. Neste capítulo apresentamos alguns conceitos necessários para a compreensão, descrevemos o algoritmo e provamos que seu consumo de tempo é polinomial.

13.1 Pushes restritos

Seja x um fluxo em uma rede G, seja i um vértice ativo e seja $a = (i, j)$ um arco admissível em $G(x)$ com relação a uma função-distância.

Dado um valor qualquer Δ, um push **restrito por** Δ, ou simplesmente **Δ-push**, em a é o processo de obtenção de um fluxo x' em G, a partir de x e a, da seguinte maneira: para cada arco b em A_G,

$$x'_b = \begin{cases} x_b + \min\{r_a, e_x(i), \Delta - e_x(j)\} & \text{se } b = a, \\ x_b - \min\{r_a, e_x(i), \Delta - e_x(j)\} & \text{se } b = \overline{a}, \\ x_b & \text{caso contrário.} \end{cases}$$

Um Δ-push é **saturador** se $r_a = \min\{r_a, e_x(i), \Delta - e_x(j)\}$ e é dito **não-saturador** caso contrário. Um arco a é dito **saturado** se ocorre um Δ-push saturador em a. Podemos observar que se a é saturado, então sua capacidade residual em relação a x' é nula, ou seja, a não está na rede residual $G(x')$.

A única diferença entre um push normal e um Δ-push é o fato de que um push restrito não permite que o excesso de um vértice ultrapasse o valor de Δ. Esta restrição visa evitar que vértices recebam fluxo excessivo e obriguem o algoritmo a desperdiçar iterações posteriores só para enviá-lo de volta.

13.2 Descrição

A idéia do algoritmo excess scaling é modificar a versão genérica do método do pré-fluxo de tal forma que, em cada iteração, exista um limitante inferior para o excesso do vértice ativo escolhido pelo algoritmo. Esse limitante é gradualmente reduzido durante a execução do algoritmo, de forma que a correção é garantida pelo caso em que ele é mínimo.

Para garantir que o número de iterações do algoritmo é limitado polinomialmente, ainda são necessárias duas modificações. Primeiramente, pelos motivos já citados, o algoritmo excess scaling utiliza Δ-pushes ao invés de pushes normais. Além disso, se existem vários vértices com excesso suficientemente grande, o vértice escolhido é aquele com menor rótulo.

O algoritmo iterativo abaixo recebe uma rede capacitada G e dois vértices s e t dessa rede e devolve um st-fluxo de intensidade máxima e um st-separador de capacidade mínima em G. Cada iteração mantém um fluxo x, uma função-distância d e um limitante Δ. No início da primeira iteração, $x_a = u_a$ para todo arco a em $\delta^-(s)$, $x_a = 0$ para todo arco a em $N_G \setminus \delta^-(s)$, $d(s) = n$, $d(i) = 0$ para todo vértice i em $N_G \setminus \{s\}$ e $\Delta = 2^{\lfloor \log mU \rfloor}$.

Caso 1: $\Delta < 1$.
Seja k um inteiro entre 1 e $n-1$ tal que $d(i) \neq k$, para todo vértice i.
Seja S o conjunto de todos os vértices em G tais que $d(i) > k$.
Devolva x e S e pare.

Caso 2: $\Delta \geq 1$ e não existe um vértice em N_G tal que $e(i) > \Delta/2$.
Comece nova iteração com $\Delta/2$ no papel de Δ.

Caso 3: $\Delta \geq 1$ e existe um vértice em N_G tal que $e(i) > \Delta/2$.
Seja i um vértice em N_G tal que $e(i) > \Delta/2$ e tal que $d(i)$ é mínimo.

Caso 3A Existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja a um arco admissível em $\delta^-_{G(x)}(i)$.
Seja x' o fluxo obtido após um Δ-push em a.
Comece nova iteração com x' no papel de x.

Caso 3B: Não existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja d' a função-distância obtida após um relabel de i.
Comece nova iteração com d' no papel de d.

De maneira análoga ao algoritmo capacity scaling, o algoritmo excess scaling executa $O(\log mU)$ fases de scaling e sua correção só é garantida se executarmos um pré-processamento e um pós-processamento que nos permitam assumir sem perda de generalidade que as capacidades são inteiras.

13.3 Desempenho

Os resultados abaixo baseiam-se em invariantes para provar que o número de iterações do algoritmo excess scaling é limitado polinomialmente.

Lema 13.1. *Seja i o vértice ativo escolhido em uma iteração do algoritmo excess scaling e seja $a = (i, j)$ um arco admissível nessa mesma iteração. Então, um Δ-push não-saturador em a diminui o excesso de i em ϵ e aumenta o excesso de j em ϵ, onde ϵ é um valor tal que $\epsilon \geq \Delta/2$.*

Prova: Pela descrição do algoritmo, sabemos que i é um vértice tal que $e(i) > \Delta/2$ e tal que $d(i)$ é mínimo. Como a é um arco admissível, temos que $d(i) = d(j)+1$, ou seja, $d(i) > d(j)$. Portanto, pela minimalidade de $d(i)$, podemos concluir que $e(j) \leq \Delta/2$ e isso implica que $\Delta - e(j) \geq \Delta/2$.

Como o Δ-push em a é não-saturador, temos $\min\{r_a, e(i), \Delta - e(j)\} = \min\{e(i), \Delta - e(j)\} \geq \Delta/2$ e, portanto, o resultado segue diretamente. □

Proposição 13.2. *São executados $O(n^2 \log mU)$ pushes não-saturadores durante todo o algoritmo excess scaling.*

Prova: Primeiramente, podemos observar que, durante uma fase de scaling, temos que o excesso de qualquer vértice é limitado superiormente por Δ. De fato, isso é verdade na primeira fase, pois o valor inicial de Δ é $2^{\lfloor \log mU \rfloor}$, e as condições de ocorrência do Caso 2 e a definição de Δ-push garantem que a propriedade também continua satisfeita para as fases seguintes.

Novamente, vamos utilizar uma função potencial para obter uma análise amortizada. Considere a função $\Phi = \sum_{i \in N} e(i)d(i)/\Delta$. Como vimos que Δ limita superiormente o excesso de qualquer vértice em uma fase de scaling e sabemos pelo Lema 10.8 que $d(i)$ é no máximo $2n$ para qualquer vértice i da rede, temos que $\Phi = O(n^2)$ no início de qualquer fase de scaling.

Se durante uma fase executa-se um relabel de um vértice i, o rótulo de i aumenta em ϵ. Como sabemos que $e(i) \leq \Delta$, isso implica que Φ aumenta em no máximo ϵ. Logo, novamente pelo Lema 10.8, temos que a execução de relabels durante uma fase de scaling pode aumentar Φ em $O(n^2)$.

Se durante uma fase executa-se um push em um arco $a = (i, j)$, temos pelo Lema 13.1 que o excesso de i diminui em ϵ e o excesso de j aumenta em ϵ, onde $\epsilon \geq \Delta/2$. Ademais, como a é um arco admissível, temos que

$d(i) = d(j) + 1$, ou seja, $d(i) - d(j) = 1$. Logo, podemos observar que:

$$\begin{aligned}
\frac{(e(i)-\epsilon)d(i) + (e(j)+\epsilon)d(j)}{\Delta} &= \frac{e(i)d(i) - \epsilon d(i) + e(j)d(j) + \epsilon d(j)}{\Delta} \\
&= \frac{e(i)d(i) + e(j)d(j) - \epsilon d(i) + \epsilon d(j)}{\Delta} \\
&= \frac{e(i)d(i) + e(j)d(j) - \epsilon(d(i) - d(j))}{\Delta} \\
&= \frac{e(i)d(i) + e(j)d(j)}{\Delta} - \frac{\epsilon(d(i) - d(j))}{\Delta} \\
&= \frac{e(i)d(i) + e(j)d(j)}{\Delta} - \frac{\epsilon}{\Delta} \\
&\leq \frac{e(i)d(i) + e(j)d(j)}{\Delta} - \frac{\Delta/2}{\Delta} \\
&= \frac{e(i)d(i) + e(j)d(j)}{\Delta} - \frac{1}{2}
\end{aligned}$$

Portanto, a execução de um push diminui Φ em pelo menos $1/2$. Como verificamos anteriormente que o valor de Φ no início de uma fase é $O(n^2)$ e esse valor aumenta em $O(n^2)$ devido a relabels, podemos concluir que o número de pushes não-saturadores durante uma fase de scaling é $O(n^2)$. Como o número total de fases é $O(\log mU)$, podemos finalmente concluir que o algoritmo executa $O(n^2 \log mU)$ pushes não-saturadores. □

Proposição 13.3. *O algoritmo excess scaling encontra um fluxo máximo em $O(nm + n^2 \log mU)$ iterações.*

Prova: Pelas Proposições 10.9, 10.11 e 13.2, temos que o algoritmo pára após $O(n^2 + nm + n^2 \log mU) = O(nm + n^2 \log mU)$ iterações. □

Pela Proposição 10.14, sabemos que o tempo total gasto por relabels é $O(nm)$ e pela Proposição 10.16 sabemos que o tempo total gasto na escolha das operações durante as iterações do algoritmo é $O(nm)$. Ahuja, Magnanti e Orlin [1] mostraram que as buscas por vértices de menor rótulo podem consumir tempo $O(nm)$ através do uso de uma estrutura de dados especial completamente análoga àquela utilizada no algoritmo dos vértices ativos de maior rótulo. Disse fato podemos inferir o resultado abaixo.

Proposição 13.4. *O algoritmo excess scaling pode ser implementado de modo a executar em tempo $O(nm + n^2 \log mU)$.*

Capítulo 14

Fluxos de custo mínimo

Suponha que uma empresa administra um conjunto de estabelecimentos formado por fábricas e depósitos de um determinado produto. Cada fábrica produz uma certa quantidade do produto e cada depósito deve armazenar uma certa quantidade. Os estabelecimentos são interconectados por uma rede de vias de transporte, como por exemplo estradas, sobre a qual são definidos custos de transporte e limites para a quantidade transportada. O desejo da empresa é estabelecer uma distribuição do produto sobre essa rede de forma a satisfazer as ofertas e demandas dos estabelecimentos, respeitar os limites das estradas e minimizar o custo total de transporte.

Considerando os estabelecimentos como sendo vértices com demanda e as estradas como sendo arcos com custo e capacidade, este problema resume-se a encontrar um fluxo viável de custo mínimo na rede definida pelos estabelecimentos e estradas. Este é o *problema do fluxo de custo mínimo*.

Este capítulo apresenta a definição formal desse problema e de conceitos relacionados, bem como a demonstração de propriedades envolvidas.

14.1 Problema

Um fluxo viável x em uma rede tem **custo mínimo** se não existe nenhum fluxo viável na mesma rede com custo menor que o de x.

O **problema do fluxo de custo mínimo** pode ser assim definido:

Problema MinCostFlow(G, u, c, b): *Dada uma rede G com capacidade u, custo c e demanda b, encontrar um fluxo viável de custo mínimo nessa rede.*

O problema MinCostFlow(G, u, c, b) pode ser expresso como o seguinte programa linear: encontrar um vetor x indexado por A_G tal que:

$$
\begin{array}{lrcll}
\text{minimize} & \sum_{a \in A_G} c_a x_a & & & \\
\text{sujeito a} & e(i) & = & b(i) & \text{para cada } i \text{ em } N_G; \\
& x_a & \leq & u_a & \text{para cada } a \text{ em } A_G; \\
& x_a & \geq & 0 & \text{para cada } a \text{ em } A_G.
\end{array}
$$

Como no problema do fluxo máximo, vários dos resultados que apresentamos posteriormente são baseados em resultados de programação linear.

14.2 Viabilidade

O problema do fluxo de custo mínimo apresenta uma diferença fundamental em relação ao problema do fluxo máximo: existem instâncias para as quais o problema não é viável, ou seja, existem redes sobre as quais não é possível atribuir um fluxo viável e, portanto, não há solução.

Entretanto, alguns dos métodos e algoritmos que serão apresentados para a resolução do problema muitas vezes admitem a existência de um fluxo viável qualquer e iniciam seu processamento a partir dele, buscando transformá-lo de forma a minimizar seu custo até atingir o valor ótimo.

Felizmente, é possível demonstrar que o problema de encontrar um fluxo viável qualquer para o problema do fluxo de custo mínimo, ou verificar que o mesmo não existe, equivale a resolver um problema de fluxo máximo, o que pode ser feito em tempo polinomial, conforme vimos anteriormente.

Seja G uma rede com custo, capacidade u e demanda b. Vamos considerar uma rede G' obtida da seguinte maneira: a partir da rede original G, adicionam-se dois novos vértices s e t, um arco (s, i) para cada vértice i que é produtor em G e um arco (i, t) para cada vértice i que é consumidor em G. Os novos arcos são chamados de **arcos artificiais** e os vértices s e t são respectivamente a fonte e o sorvedouro do problema de fluxo máximo.

A capacidade u' de G' é definida da seguinte forma: se a é um arco da rede original G, então $u'_a = u_a$. Se a é um arco artificial tal que $a = (s, i)$, para algum vértice i de G, então $u'_a = -b(i)$. Se a é um arco artificial tal que $a = (i, t)$, para algum vértice i de G, então $u'_a = b(i)$. Como só existem esses dois tipos de arcos artificiais, temos que a capacidade u' está bem definida para G'. Ademais, a construção também garante sua não-negatividade.

A rede G' com capacidade u' é chamada de **rede auxiliar** de G.

Proposição 14.1. *Seja G uma rede com custo, demanda e capacidade u. Então G admite um fluxo viável se, e somente se, todo fluxo máximo x^* sobre a rede auxiliar G' de G é tal que $x^*_a = u'_a$ para todo arco artificial a.*

Prova: Primeiramente, vamos mostrar que se um st-fluxo máximo x^* sobre a rede auxiliar G' é tal que $x^*_a = u'_a$ para todo arco artificial a, então G admite

um fluxo viável. Seja x o fluxo em G definido da seguinte forma: para cada arco a de G, $x_a = x^*_a$. Como o conjunto de arcos de G está contido no conjunto de arcos de G', a função definida por x está bem definida. Ademais, como $u'_a = u_a$ para todo arco a de G e x^* respeita u', temos que x é de fato um fluxo. Resta mostrar agora que x é um fluxo viável em G.

Seja i um vértice produtor de G. Pela construção dos arcos artificiais, sabemos que existe exatamente um arco artificial a em $\delta^+_{G'}(i)$ e nenhum arco artificial em $\delta^-_{G'}(i)$. Portanto, podemos concluir, pela definição de x^*, que vale a igualdade $e_x(i) = e_{x^*}(i) - x^*_a = 0 - u'_a = -(-b(i)) = b(i)$.

Seja i um vértice consumidor de G. Pela construção dos arcos artificiais, sabemos que existe exatamente um arco artificial a em $\delta^-_{G'}(i)$ e nenhum arco artificial em $\delta^+_{G'}(i)$. Portanto, podemos concluir, analogamente pela definição de x^*, que vale a igualdade $e_x(i) = e_{x^*}(i) + x^*_a = 0 + u'_a = b(i)$.

Seja i um vértice de G tal que $b(i) = 0$. Pela construção dos arcos artificiais, sabemos que não existe nenhum arco artificial que incide em i. Logo, $e_x(i) = e_{x^*}(i) = 0$ e concluímos que todas as demandas estão satisfeitas.

Vamos mostrar agora que se um st-fluxo máximo x^* em G' é tal que existe um arco artificial a para o qual $x^*_a < u'_a$, então G não admite um fluxo viável. Suponha por contradição que existe um fluxo viável x em G. Considere então o st-fluxo x' em G' definido da seguinte forma: para cada arco a de G', $x'_a = x_a$ se a é um arco de G, $x'_a = -b(i)$ se $a = (s,i)$ e $x'_a = b(i)$ se $a = (i,t)$. Como todo arco artificial é obrigatoriamente da forma (s,i) ou (i,t) por definição, temos que a função definida por x' está bem definida.

Seja i um vértice produtor de G. Pela construção dos arcos artificiais, sabemos que existe exatamente um arco artificial a em $\delta^+_{G'}(i)$ e nenhum arco artificial em $\delta^-_{G'}(i)$. Portanto, podemos concluir pela definição de x' que vale a igualdade $e_{x'}(i) = e_x(i) + x'_a = b(i) + (-b(i)) = b(i) - b(i) = 0$.

Seja i um vértice consumidor de G. Pela construção dos arcos artificiais, sabemos que existe exatamente um arco artificial a em $\delta^-_{G'}(i)$ e nenhum arco artificial em $\delta^+_{G'}(i)$. Portanto, podemos concluir analogamente pela definição de x' que vale a igualdade $e_{x'}(i) = e_x(i) - x'_a = b(i) - b(i) = 0$.

Seja i um vértice de G tal que $b(i) = 0$. Pela construção dos arcos artificiais, sabemos que não existe nenhum arco artificial que incide em i. Logo, $e_{x'}(i) = e_x(i) = 0$ e concluímos que $e_{x'}(i) = 0$ para todo vértice de G' diferente de s e t. Portanto, temos que x' é de fato um st-fluxo em G'.

Finalmente, pela definição de G', temos que $\mathrm{val}(x') = e_{x'}(t) = x'(\delta^+_{G'}(t))$. Ademais, pelo Lema 3.2, temos também que $\mathrm{val}(x') = -e_{x'}(s) = x'(\delta^-_{G'}(s))$. Mas como pela hipótese existe um arco artificial a para o qual $x^*_a < u'_a$ e temos que $x'_a = u'_a$ para todo arco a de G, então $x'(\delta^+_{G'}(t)) > x^*(\delta^+_{G'}(t))$ ou $x'(\delta^-_{G'}(s)) > x^*(\delta^-_{G'}(s))$. Em ambos os casos, temos que $\mathrm{val}(x') > \mathrm{val}(x^*)$, o que é uma contradição com a maximalidade da intensidade de x^*. □

Podemos observar que a construção utilizada na demonstração anterior

não só permite determinar se existe um fluxo viável sobre a rede do problema original, como também permite encontrar algoritmicamente tal fluxo.

14.3 Conexidade

Como mencionamos anteriormente, alguns algoritmos e resultados relacionados ao problema do fluxo de custo mínimo são válidos apenas diante da suposição de que a rede considerada admite um fluxo viável.

Paralelamente, existem também algoritmos e resultados que exigem uma suposição mais forte: a **conexidade** da rede residual. Dizemos que uma rede é **conexa** se existe nela um caminho entre qualquer par de vértices.

A proposição abaixo demonstra que quando queremos resolver um problema de fluxo de custo mínimo, assumir que a rede residual resultante de qualquer fluxo é sempre conexa não altera a solução do problema.

Proposição 14.2. *Seja G uma rede com custo c, capacidade u e demanda b sobre a qual queremos encontrar um fluxo viável de custo mínimo. Então podemos assumir sem perda de generalidade que, para qualquer fluxo x em G e qualquer par de vértices i e j, existe um caminho P entre i e j na rede residual $G(x)$ tal que todos os arcos de P têm capacidade residual infinita.*

Prova: Pela Proposição 14.1, podemos assumir sem perda de generalidade que existe um fluxo viável em G, pois caso isso não seja verdade é possível executar um pré-processamento de tempo polinomial que detecta esse fato independente de G satisfazer ou não a propriedade da conexidade.

Suponha então que G admite um fluxo viável mas não satisfaz a propriedade descrita. Vamos utilizar novamente o conceito de arcos artificiais para construir uma nova rede G' com capacidade u', custo c' e com a mesma demanda da rede original G, de forma que G' satisfaça a propriedade e seja tal que um fluxo x é viável e de custo mínimo em G' se, e somente se, o fluxo x restrito aos arcos de G corresponde a um fluxo viável de custo mínimo em G.

A rede G' é obtida através de um procedimento simples de adição de arcos artificiais a G: dado um vértice qualquer i de G, adiciona-se a G os arcos (i, j) e (j, i), para todo vértice j diferente de i. Por essa construção, temos que, para qualquer par j e k de vértices, existe sempre um caminho P entre j e k em G': basta tomar o caminho $P = \langle j, (j, i), i, (i, k), k\rangle$.

Para garantir que tais caminhos também existem na rede residual $G(x)$ para qualquer fluxo x, basta definir u' de forma que os arcos artificiais tenham capacidade alta o suficiente para que $x_a < u_a$, para todo arco artificial a. Logo, é suficiente definir que a capacidade u' é tal que $u'_a = u_a$, para todo arco a da rede G, e $u'_a = \infty$, para todo arco artificial a.

Considere agora o custo c' definido da seguinte forma: $c'_a = c_a$, para todo arco a da rede original G, e $c'_a = \infty$, para todo arco artificial a.

Como todos os arcos de G estão em G' com a mesma capacidade e como os vértices de ambas as redes têm a mesma demanda, então qualquer fluxo

viável em G corresponde a um fluxo viável em G', bastando definir que os arcos artificiais têm fluxo nulo. Além disso, qualquer fluxo viável em G tem custo menor do que qualquer fluxo viável em G' que não seja nulo em algum dos arcos artificiais, uma vez que tais arcos possuem custo muito grande. Assim, como existe um fluxo viável em G, então qualquer fluxo viável de custo mínimo em G' é nulo em todos os arcos artificiais e, portanto, o problema de obter um fluxo de custo mínimo é equivalente em ambas as redes, de onde obtemos o resultado. □

14.4 Custos inteiros

Além das hipóteses de viabilidade e conexidade, existem também resultado e algoritmos para a resolução do problema do fluxo de custo mínimo que se baseiam no fato de que o custo de todos os arcos da rede sobre a qual o problema está definido é inteiro.

Seja G uma rede capacitada com custo e demanda. Vamos mostrar que o problema de encontrar um fluxo viável de custo mínimo em G pode ser reduzido ao problema de encontrar um fluxo viável de custo mínimo em uma rede cujos arcos possuem todos custo inteiro.

Para tanto, vamos fazer algumas definições. Seja c o custo da rede G. Como os custos são todos racionais, existe um inteiro positivo K tal que Kc_a é um valor inteiro para todo arco a da rede G. Consideremos, então, uma rede $G(K)$, com mesmo conjunto de vértices e arcos de G, mesma capacidade e mesma demanda. O custo c' de $G(K)$, entretanto, é definido da seguinte maneira: para cada arco a de $G(K)$, $c'_a = Kc_a$. Conforme podemos observar, $G(K)$ possui custo inteiro. A rede $G(K)$ é dita a **rede inteira** de G.

Estabelecemos a seguir um resultado que mostra que os problema de encontrar um fluxo de custo mínimo em G ou $G(K)$ são equivalentes.

Proposição 14.3. *Seja G uma rede capacitada, com custo e demanda e seja $G(K)$ sua rede inteira. Então x é um fluxo viável de custo mínimo em G se, e somente se, x é um fluxo viável de custo mínimo em $G(K)$.*

Prova: Primeiramente, notemos que um fluxo é viável em G se, e somente se, tal fluxo é viável em $G(K)$, uma vez que as capacidades e demandas de ambas as redes são as mesmas. Podemos, portanto, adotar a seguinte notação: se x é um fluxo viável em G, então $\text{custo}_G(x)$ denota o seu custo na rede G e $\text{custo}_{G(K)}(x)$ denota o seu custo na rede $G(K)$. Além disso, denotemos por c o custo da rede G e por c' o custo da rede $G(K)$.

Suponha, então, que x seja um fluxo viável de custo mínimo em G e não o seja em $G(K)$. Seja x^* um fluxo viável de custo mínimo em $G(K)$. Temos, então, que $\text{custo}_{G(K)}(x^*) < \text{custo}_{G(K)}(x)$. Agora, por definição, vale que:

$$\text{custo}_{G(K)}(x) = \sum_{a \in A_{G(K)}} c'_a x_a = \sum_{a \in A_{G(K)}} Kc_a x_a = K \sum_{a \in A_G} c_a x_a = K\text{custo}_G(x).$$

De forma análoga, podemos concluir que $\text{custo}_{G(K)}(x^*) = K\text{custo}_G(x^*)$.

Mas, então, concluímos que:

$$\text{custo}_G(x^*) = \frac{1}{K}\text{custo}_{G(K)}(x^*) < \frac{1}{K}\text{custo}_{G(K)}(x) = \text{custo}_G(x),$$

o que é uma contradição com a minimalidade do custo de x em G.

Assim, a primeira implicação está provada. A prova da segunda implicação é completamente análoga. □

Sendo assim, o problema de encontrar um fluxo de custo mínimo em uma rede qualquer pode ser resolvido por um algoritmo que tenha seu funcionamento baseado na hipótese de custos inteiros. Para tanto, basta que a rede sobre a qual o problema esteja definido passe pelo pré-processamento polinomial que transforma seu custo no custo de sua rede inteira.

14.5 Circuitos negativos

Vamos apresentar alguns resultados fundamentais que nos permitem concluir que a ausência de circuitos negativos na rede residual é uma condição necessária e suficiente para que um fluxo viável tenha custo mínimo.

Seja x um fluxo em uma rede G e seja W um circuito na rede residual $G(x)$. O fluxo-circuito x^W em $G(x)$ é definido da seguinte maneira:

$$x_a^W = \begin{cases} \min\{r_b : \text{ b é arco de } W\} & \text{se } a \text{ é arco de } W; \\ 0 & \text{caso contrário.} \end{cases}$$

O fluxo $x + x^W$ é o **fluxo obtido a partir de x e W**.

Lema 14.4. *Seja x um fluxo viável em uma rede G com custo, seja W um circuito na rede residual $G(x)$ e seja $w := \min\{r_a : \text{ a é arco de } W\}$. Então o fluxo x', obtido a partir de x e W, é um fluxo viável em G tal que* $\text{custo}(x') = \text{custo}(x) + w \cdot \text{custo}(W)$.

Prova: Considere o fluxo-circuito x^W, conforme definido anteriormente, e seja $w = \min\{r_a : \text{ a é arco de } W\}$ o seu valor. Seja i um vértice de $G(x)$. Se i não está em W, então não há fluxo entrando nem saindo de i em relação a x^W, o que implica que $e_{x^W}(i) = 0$. Por outro lado, se i está em W, temos que exatamente um arco de W entra em i e exatamente um arco sai. Logo, também vale que $e_{x^W}(i) = 0$. Portanto, pelo Corolário 3.19, temos que o fluxo x', obtido a partir de x e x^W, é viável em G.

Seja c' o custo da rede residual $G(x)$. Temos, então, que $\text{custo}(x^W) = \sum_{a \in A_W} c'_a w = w \sum_{a \in A_W} c'_a = w \cdot \text{custo}(W)$. Logo, pelo Lema 3.20, vale que $\text{custo}(x + x^W) = \text{custo}(x) + \text{custo}(x^W) = \text{custo}(x) + w \cdot \text{custo}(W)$. □

Teorema 14.5. *Seja x um fluxo viável em uma rede G com custo. Então x tem custo mínimo se, e somente se, não existem circuitos negativos em $G(x)$.*

Prova: Primeiramente, suponha que exista um circuito negativo W em $G(x)$. Vamos mostrar que o fluxo x não pode ter custo mínimo em G.

Considere o fluxo-circuito x^W e seja w o seu valor. Pelo Lema 14.4, sabemos o fluxo x', obtido a partir de x e W, é um fluxo viável em G tal que $\text{custo}(x') = \text{custo}(x) + w \cdot \text{custo}(W)$.

Sabemos também que w é um valor positivo por definição. Ademais, como W é um circuito negativo, então $\text{custo}(W) < 0$, de onde concluímos que $\text{custo}(x') < \text{custo}(x)$. Ou seja, o custo de x não pode ser mínimo.

Suponha agora que não existam circuitos negativos em $G(x)$. Vamos provar que x tem custo mínimo. Seja y um fluxo viável qualquer em G.

Considere o fluxo-diferença $y - x$ em $G(x)$. Sabemos pelo Lema 3.12 que vale a igualdade $x + (y - x) = y$. Logo, podemos utilizar o Lema 3.20 e concluir que $\text{custo}(y) = \text{custo}(x) + \text{custo}(y - x)$. Além disso, pelo Corolário 3.15, temos que $e_{y-x}(i) = 0$, para todo vértice i. Assim, pelo Corolário 3.10, sabemos que existe uma decomposição X de $y - x$ que só contém fluxos-circuito. Ademais, pelo Lema 3.11 temos que $\text{custo}(y - x) = \sum_{x' \in X} \text{custo}(x')$.

Por hipótese, todos os fluxos-circuitos em X correspondem a circuitos não-negativos. Ademais, se x' é um fluxo-circuito associado ao circuito W e de valor w, então $\text{custo}(x') = w \cdot \text{custo}(W)$, conforme já argumentamos na prova do Lema 14.4. Logo, temos que $\text{custo}(x') \geq 0$, para qualquer fluxo x' de X. Portanto, $\text{custo}(y) = \text{custo}(x) + \text{custo}(y - x) = \text{custo}(x) + \sum_{x' \in X} \text{custo}(x')$ e podemos concluir que $\text{custo}(y) \geq \text{custo}(x)$.

Assim, pela escolha arbitrária do fluxo y, concluímos que x é um fluxo viável de custo mínimo, o que conclui a prova. □

14.6 Custos reduzidos

Seja G uma rede com custo c e considere uma função π de N_G em $\mathbb{Z}_{\geq}$ qualquer. A função π é chamada de **função-potencial** sobre G.

O **custo reduzido** de um arco $a = (i, j)$ da rede G em relação a uma função-potencial π qualquer é definido como $c_a^\pi = c_a - \pi(i) + \pi(j)$.

Os resultados que são apresentados a seguir utilizam custos reduzidos como uma ferramenta auxiliar fundamental para se obter outras condições necessárias e suficientes para que um fluxo viável tenha custo mínimo.

Lema 14.6. *Seja π uma função-potencial definida sobre uma rede G com custo c. Então todo circuito W em G é tal que $\sum_{a \in A_W} c_a^\pi = \sum_{a \in A_W} c_a$ e todo caminho P em G entre i e j é tal que $\sum_{a \in A_P} c_a^\pi = \sum_{a \in A_P} c_a - \pi(i) + \pi(j)$.*

Prova: Seja W um circuito $\langle i_0, a_1, i_1, a_2, i_2, \ldots, a_{n-1}, i_{n-1}, a_n, i_0 \rangle$ em G. Temos por definição que vale a igualdade $\sum_{a \in W} c_a^\pi = \sum_{a \in W} c_a + (-\pi(i_0) + \pi(i_1) - \pi(i_1) + \pi(i_2) - \cdots - \pi(i_{n-1}) + \pi(i_0)) = \sum_{a \in W} c_a + 0 = \sum_{a \in W} c_a$.

Analogamente, seja P um caminho $\langle i, a_1, i_1, a_2, i_2, \ldots, a_{n-1}, i_{n-1}, a_n, j \rangle$ em G. Podemos observar que, por definição, $\sum_{a \in P} c_a^\pi = \sum_{a \in W} c_a + (-\pi(i) + \pi(i_1) - \pi(i_1) + \pi(i_2) - \cdots - \pi(i_{n-1}) + \pi(j)) = \sum_{a \in W} c_a + \pi(i) - \pi(j)$. □

Teorema 14.7. *Seja x um fluxo viável em uma rede G com custo c. Então x tem custo mínimo se, e somente se, existe uma função-potencial π sobre G tal que $c_a^\pi \geq 0$, para todo arco a da rede residual $G(x)$.*

Prova: Suponha que exista uma função-potencial π tal que $c_a^\pi \geq 0$, para todo arco a de $G(x)$. Então, pelo Lema 14.6, temos $\sum_{a \in A_W} c_a = \sum_{a \in A_W} c_a^\pi \geq 0$ para todo circuito W da rede residual. Portanto, temos que $G(x)$ não contém circuito negativo e, pelo Teorema 14.5, temos que x tem custo mínimo.

Suponha agora que x tenha custo mínimo. Seja s um vértice qualquer da rede e denote por $c(s,j)$ o custo de um caminho de custo mínimo entre s e j em $G(x)$, para cada vértice j da rede. Se não existe caminho entre s e j em $G(x)$, definimos que $c(s,j) = \infty$.

Como x é um fluxo de custo mínimo, então pelo Teorema 14.5 temos que $G(x)$ não possui circuitos negativos. Logo, podemos utilizar o Lema 2.5 e concluir que $c(s,i) + c_a \geq c(s,j)$, para todo arco $a = (i,j)$ de $G(x)$. Isso implica que $c_a \geq c(s,j) - c(s,i)$, para todo arco $a = (i,j)$ da rede $G(x)$.

Considere agora uma função-potencial π tal que $\pi(i) = -c(s,i)$ para todo vértice i da rede G. Como para todo arco $a = (i,j)$ da rede $G(x)$ temos que $c_a \geq c(s,j) - c(s,i)$, temos que $c_a \geq \pi(i) - \pi(j)$. Portanto, vale que $c_a^\pi = c_a - \pi(i) + \pi(j) \geq 0$, para todo arco a da rede residual $G(x)$. □

Lema 14.8. *Seja π uma função-potencial definida sobre uma rede G com custo c. Então temos que $c_a^\pi = -c_{\overline{a}}^\pi$, para todo arco a de $G(x)$.*

Prova: Seja $a = (i,j)$ um arco de G. Então vale que $c_a^\pi = c_a - \pi(i) + \pi(j) = -(-c_a + \pi(i) - \pi(j)) = -(c_{\overline{a}} - \pi(j) + \pi(i)) = -c_{\overline{a}}^\pi$. □

Teorema 14.9 (folgas complementares). *Seja x um fluxo viável em uma rede G com custo c. Então x tem custo mínimo se, e somente se, existe uma função-potencial π tal que, para todo arco a de G, vale que:*
(i) Se $c_a^\pi > 0$, então $x_a = 0$.
(ii) Se $0 < x_a < u_a$, então $c_a^\pi = 0$.
(iii) Se $c_a^\pi < 0$, então $x_a = u_a$.

Prova: Vamos provar que uma função-potencial π satisfaz as três propriedades acima se, e somente se, $c_a^\pi \geq 0$, para todo arco a da rede residual $G(x)$, o que permite inferir o resultado diretamente a partir do Teorema 14.7.

Seja π uma função-potencial em G e suponha que $c_a^\pi \geq 0$ para todo arco a de $G(x)$. Vamos provar que as três propriedades são válidas.

Seja a um arco de G. Se $c_a^\pi > 0$, então pelo Lema 14.8, temos que $c_{\overline{a}}^\pi < 0$. Logo, o arco $\overline{a}$ não está em $G(x)$ e podemos concluir que $x_a = 0$.

Se $0 < x_a < u_a$, então ambos os arcos a e $\overline{a}$ estão na rede residual $G(x)$. Isso implica que $c_a^\pi \geq 0$ e $c_{\overline{a}}^\pi \geq 0$. Como o Lema 14.8 diz que $c_{\overline{a}}^\pi = -c_a^\pi$, temos portanto que $c_a^\pi \geq 0$ e $c_a^\pi \leq 0$ e podemos concluir que $c_a^\pi = 0$.

Se $c_a^\pi < 0$, então a não está em $G(x)$ e, portanto, temos $x_a = u_a$.

Suponha agora que π satisfaz as três propriedades descritas no enunciado e seja a um arco da rede $G(x)$. Suponha por contradição que $c_a^\pi < 0$.

Se a é um arco de G, temos que $x_a < u_a$, pois caso contrário a não poderia ser um arco de $G(x)$ por definição. Mas como temos que $c_a^\pi < 0$, então a validade da propriedade (iii) implica que deveria valer $x_a = u_a$.

Se $\overline{a}$ é um arco de G, temos que $x_{\overline{a}} > 0$. Mas como temos que $c_a^\pi < 0$, então $c_{\overline{a}}^\pi > 0$ pelo Lema 14.8 e, pela propriedade (i), deveria valer $x_{\overline{a}} = 0$.

Portanto, provamos por contradição que se as três propriedades são válidas, então $c_a^\pi \geq 0$, para todo arco a da rede residual $G(x)$. □

As três condições apresentadas no enunciado do Teorema 14.9 são ditas **condições de otimalidade** para o fluxo x e a função-potencial π.

Capítulo 15

Método do cancelamento de circuitos

No capítulo anterior, vimos que se não existem circuitos negativos na rede residual relativa a um fluxo viável, então tal fluxo possui custo mínimo.

Partindo dessa observação, Goldberg e Tarjan [15] desenvolveram o **método do cancelamento de circuitos**, para a resolução do problema do fluxo de custo mínimo. O método é também uma prova algoritmica para o **teorema da integralidade**, em sua versão para o problema em questão.

Este capítulo apresenta o método de forma genérica e sugere uma implementação baseada em uma idéia de Bellman [4] e Ford [13].

15.1 Descrição

O método do cancelamento de circuitos, descrito abaixo, é um algoritmo iterativo que recebe uma rede capacitada G, com custo e demanda, e devolve um fluxo viável de custo mínimo nessa rede. Cada iteração começa com um fluxo viável x. Sendo assim, um fluxo viável deve ser estabelecido na rede antes do início da primeira iteração.

A idéia do método é reduzir o custo do fluxo viável inicial através de circuitos negativos na rede residual, de forma a não comprometer sua viabilidade. A obtenção de um novo fluxo a partir do circuito negativo encontrado deve ocorrer conforme descrito na Seção 14.5.

Caso 1: Não existem circuitos negativos em $G(x)$.
Devolva x e pare.

Caso 2: Existe um circuito negativo em $G(x)$.
Seja W um circuito negativo em $G(x)$.
Seja x' o fluxo obtido a partir de x e W.
Comece nova iteração com x' no papel de x.

A figura 15.1 mostra uma simulação completa do método do cancelamento de circuitos, a partir de um fluxo viável pré-estabelecido.

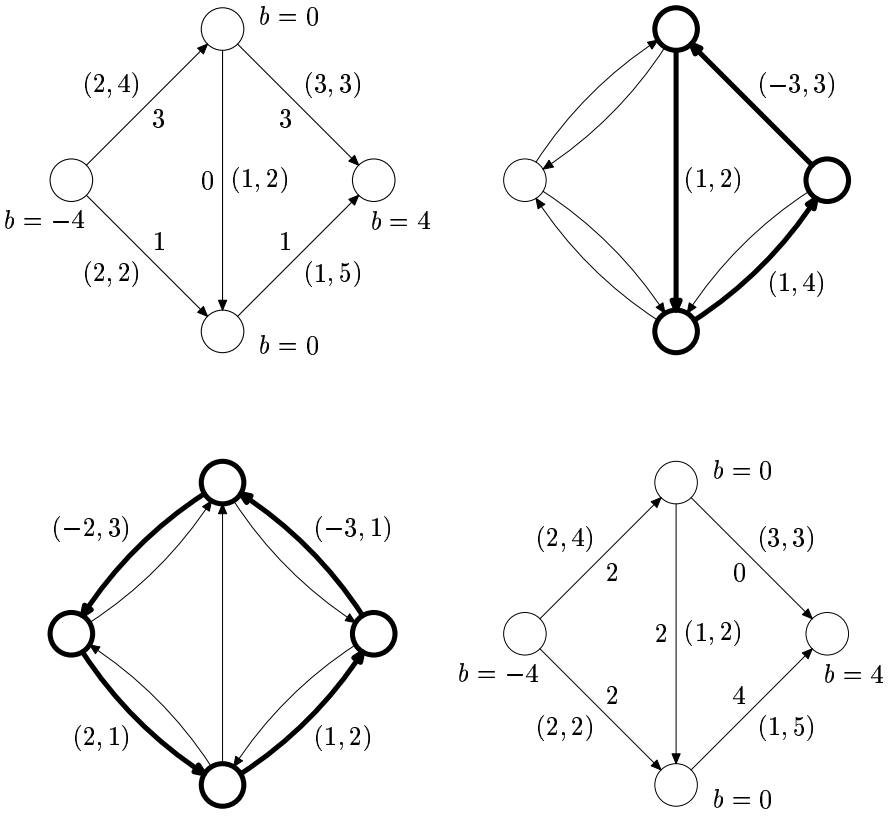

Figura 15.1: Simulação do método do cancelamento de circuitos. Em cada diagrama, os números entre parênteses próximos aos arcos representam, respectivamente, seu custo e sua capacidade residual. O primeiro diagrama representa o fluxo viável inicial e o último, o fluxo de custo mínimo encontrado. Os demais diagramas indicam os circuitos negativos na rede residual, através dos quais o custo do fluxo foi reduzido.

Conforme mencionado, para que o método do cancelamento de circuitos possa ser aplicado sobre determinada rede, é preciso que um fluxo viável exista e seja determinado através de um pré-processamento. Sabemos, pela Seção 14.2, que é possível encontrar um fluxo viável em uma rede ou ainda detectar que o mesmo não existe em tempo polinomial. Portanto, assumiremos que o método só é executado sobre redes para as quais já existe um fluxo viável pré-estabelecido.

15.2 Correção e desempenho

Se o método do cancelamento de circuitos termina, então ele devolve um fluxo viável. De fato, o método sempre começa com um fluxo viável. Além disso, pelo Lema 14.4, se x é um fluxo viável em uma rede G, então o fluxo obtido a partir de x e W, para qualquer circuito W em $G(x)$, também o é. Assim, o resultado segue por indução no número de iterações.

Notemos também que, se o Caso 1 ocorre, então, pelo Teorema 14.5, o fluxo viável devolvido pelo método do cancelamento de circuitos tem custo mínimo, uma vez que sua rede residual não contém circuitos negativos.

Resta, portanto, verificarmos que o método pára. Para tanto, vamos estabelecer um limitante para o seu número de iterações.

Proposição 15.1. *O método do cancelamento de circuitos encontra um fluxo viável de custo mínimo em $O(mUC)$ iterações.*

Prova: Seja u a capacidade e seja c o custo da rede G recebida como parâmetro pelo método do cancelamento de circuitos. Como as capacidades e os custos dos arcos são números racionais, então existem inteiros positivos K_1 e K_2 tais que $K_1 u_a$ e $K_2 c_a$ são inteiros para todo arco a da rede G.

Além disso, pelo Lema 14.4, se x é um fluxo viável e W é um circuito em $G(x)$, então o custo do fluxo obtido a partir de x e W é custo(x) + w.custo(W), em que w é a menor capacidade residual de um arco de W.

Sendo assim, e como os circuitos selecionados durante as ocorrências do Caso 2 são todos negativos, concluímos que o custo do fluxo mantido pelo método diminui em pelo menos $1/(K_1 K_2)$ em cada iteração.

Notemos, então, que o custo de qualquer fluxo viável na rede é sempre limitado superiormente por mUC e inferiormente por $-mUC$, uma vez que $-C \leq c_a \leq C$ e $0 \leq x_a \leq U$, para todo arco a. Concluímos, portanto, que o método devolve um fluxo de custo mínimo após $O(mUC)$ iterações. □

Um fluxo x em uma rede é dito **inteiro** se o valor x_a é inteiro para todo arco a dessa rede. O seguinte teorema é uma conseqüência do método do cancelamento de circuitos.

Teorema 15.2 (integralidade). *Seja G uma rede cujas capacidades dos arcos e demandas dos vértices são todas inteiras. Se G admite um fluxo viável, então existe um fluxo viável de custo mínimo inteiro em G.*

Prova: Como as capacidades e demandas são todas inteiras, então todas as capacidades da rede auxiliar de G, obtida como na Seção 14.2, são inteiras. Assim, pelo Teorema 5.7, existe um fluxo máximo x inteiro na rede auxiliar de G. Como G admite um fluxo viável, então a prova da Proposição 14.1 nos diz que o fluxo x restrito aos arcos de G é um fluxo viável nessa rede. Concluímos, portanto, que G admite um fluxo viável inteiro.

Podemos assumir, portanto, que o método do cancelamento de circuitos pode ser aplicado sobre a rede G, partindo de um fluxo viável inteiro.

Notemos, então, que se no início de determinada iteração do método o fluxo x é inteiro e se todas as capacidades são inteiras, então todas as capacidades residuais também são inteiras nessa iteração. Logo, o fluxo x', obtido a partir de x e W, também é um fluxo inteiro ao final da iteração.

Assim, por indução no número de iterações, temos que o fluxo devolvido pelo método do cancelamento de circuitos, quando aplicado sobre a rede G, é inteiro. O resultado segue da Proposição 15.1. □

O conhecido algoritmo de Bellman, Ford e Moore [4, 13] para o problema dos caminhos mínimos pode ser utilizado para encontrar um circuito negativo na rede residual ou mesmo detectar sua ausência. Uma implementação deste algoritmo com consumo de tempo $O(nm)$ pode ser encontrada em Cormen, Leiserson, Rivest e Stein [6]. Ademais, a redução do custo do fluxo através de um circuito negativo pode ser realizada em tempo $O(n)$, pois exige apenas a visita de cada um dos arcos do circuito ou de seus irmãos no máximo uma vez. Com isso, concluímos que cada iteração do método do cancelamento de circuitos pode ser implementada de forma a consumir tempo $O(nm)$.

Feitas essas observações e utilizando também a Proposição 15.1, podemos estabelecer o seguinte resultado.

Proposição 15.3. *O método do cancelamento de circuitos pode ser implementado de modo a executar em tempo* $O(nm^2UC)$.

Por fim, observamos que o limitante estabelecido para a complexidade do método não é polinomial. Essa característica pode torná-lo inviável na prática quando aplicado sobre determinados tipos de instâncias.

Entretanto, notamos que melhores resultados teóricos e práticos podem ser obtidos de acordo com a maneira através da qual os circuitos negativos são detectados ou a partir do estabelecimento de regras para a ordem em que tais circuitos são processados pelo algoritmo.

O algoritmo simplex para redes, proposto por Dantzig [7], por exemplo, é uma implementação do método do cancelamento de circuitos que mantém estruturas de dados que tornam possível a identificação de circuitos negativos em tempo $O(m)$. Apesar de também não possuir complexidade teórica polinomial, tal algoritmo apresenta um excelente desempenho na prática.

Capítulo 16

Método dos caminhos de viabilidade

Edmonds e Karp [10] propuseram um método para resolver o problema do fluxo viável de custo mínimo que, de maneira análoga ao método dos caminhos de aumento para o problema do fluxo máximo, consiste na idéia simples de aumentar o fluxo ao longo de um caminho na rede residual. Essa idéia é o princípio básico do **método dos caminhos de viabilidade**.

Neste capítulo são apresentados conceitos e resultados preliminares, seguidos da descrição do método e da demonstração de sua complexidade.

16.1 Caminhos de viabilidade

Seja G uma rede com demanda b e seja x um fluxo sobre G. Como definimos anteriormente, um caminho alternante é um caminho na rede residual $G(x)$. Um **caminho de viabilidade** é um caminho alternante cuja origem i é tal que $e(i) > b(i)$ e cujo destino j é tal que $e(j) < b(j)$.

Considere agora a seguinte notação: se P é um caminho de viabilidade em $G(x)$, então o fluxo x^P é um fluxo-caminho na rede $G(x)$ tal que $x^P_a = \min\{e(i) - b(i), b(j) - e(j), \min\{r_a : a \in P\}\}$ se a é um arco de P e $x^P_a = 0$ se a não é um arco de P. As definições de capacidade residual e de caminho de viabilidade garantem que x^P é de fato um fluxo-caminho em $G(x)$.

O fluxo $x + x^P$ é o **fluxo obtido a partir de x e P**.

Vamos mostrar a seguir alguns resultados que demonstram que a ausência de caminhos de viabilidade é uma condição necessária e suficiente para a viabilidade de um fluxo, supondo que a rede residual é sempre conexa.

Lema 16.1. *Considere um fluxo em uma rede G com demanda b que admite um fluxo viável. Então existe um vértice i tal que $e(i) > b(i)$ se, e somente se, existe um vértice j tal que $e(j) < b(j)$.*

Prova: Como a rede G admite um fluxo viável, temos que $\sum_{i \in N_G} b(i) = 0$.

Seja i um vértice de G tal que $e(i) > b(i)$. Suponha por contradição que todo vértice j é tal que $e(j) \geq b(j)$. Então, $\sum_{i \in N_G} e(i) > \sum_{i \in N_G} b(i) = 0$. Mas isso é um absurdo, pois pelo Lema 3.1 temos que $\sum_{i \in N_G} e(i) = 0$.

Analogamente, seja i um vértice de G tal que $e(i) < b(i)$. Suponha por contradição que todo vértice j é tal que $e(j) \leq b(j)$. Então, $\sum_{i \in N_G} e(i) < \sum_{i \in N_G} b(i) = 0$. Mas isso é um absurdo, pois temos que $\sum_{i \in N_G} e(i) = 0$. □

Corolário 16.2. *Seja x um fluxo em uma rede G com demanda b tal que G admite um fluxo viável e a rede residual $G(y)$ é conexa para qualquer fluxo y. Então x é viável se, e somente se, $G(x)$ não tem caminhos de viabilidade.*

Prova: Primeiramente, suponha que $G(x)$ não tem caminhos de viabilidade. Como por hipótese a rede $G(x)$ é conexa, isso implica pela definição de caminho de viabilidade que não existe um vértice i tal que $e(i) < b(i)$ ou não existe um vértice i tal que $e(i) > b(i)$. Logo, pelo Lema 16.1, temos que $e(i) = b(i)$ para todo vértice i da rede G, ou seja, x é um fluxo viável.

Suponha agora que x seja um fluxo viável. Isso implica por definição que $e(i) = b(i)$ para todo vértice i da rede G. Logo, podemos concluir por definição que não podem existir caminhos de viabilidade em $G(x)$. □

16.2 Atualizações de potencial

Seja x um fluxo em uma rede G com custo c e seja π uma função-potencial sobre G. Considere a seguinte notação: dados dois vértices i e j de G, o valor $c_x^\pi(i,j)$ é o custo de um caminho de custo mínimo entre i e j na rede residual $G(x)$, onde o custo dos arcos de $G(x)$ é o custo reduzido c^π.

Dado um vértice i de G, uma **atualização de π em relação a i e x** é uma função-potencial π' tal que $\pi'(j) = \pi(j) - c_x^\pi(i,j)$ para todo vértice j.

Os resultados a seguir demonstram alguns invariantes fundamentais em atualizações de potencial que são explorados posteriormente pelo método.

Lema 16.3. *Seja x um fluxo em uma rede G com custo c, seja i um vértice de G e seja π uma função-potencial tal que $c_a^\pi \geq 0$ para todo arco a de $G(x)$. Seja π' a atualização de π em relação a i e x e seja P um caminho em $G(x)$ entre i e um vértice j de tal que P é um caminho correspondente ao custo $c_x^\pi(i,j)$. Então temos que são válidas as seguintes afirmações:*
(i) $c_a^{\pi'} \geq 0$ para todo arco a de $G(x)$.
(ii) $c_a^\pi = 0$ para todo arco a de P.

Prova: Como $c_a^\pi \geq 0$ para todo arco a de $G(x)$, não há circuitos negativos em $G(x)$ com relação a c^π. Portanto, pelo Lema 2.5 podemos garantir que $c_x^\pi(i,k) - c_x^\pi(i,j) \leq c_a^\pi$ para todo arco $a = (j,k)$ de $G(x)$. Logo, temos que $c_x^\pi(i,k) - c_x^\pi(i,j) \leq c_a - \pi(j) + \pi(k)$ por definição e podemos concluir que $c_a - (\pi(j) - c_x^\pi(i,j)) + (\pi(k) - c_x^\pi(i,k)) \geq 0$, ou seja, que $c_a^{\pi'} \geq 0$.

Ademais, também pela ausência de circuitos negativos, o Lema 2.8 garante que $c_x^\pi(i,k) - c_x^\pi(i,j) = c_a^\pi$ para todo arco $a = (j,k)$ de P. Logo, temos que $c_x^\pi(i,k) - c_x^\pi(i,j) = c_a - \pi(j) + \pi(k)$ por definição e podemos concluir que $c_a - (\pi(j) - c_x^\pi(i,j)) + (\pi(k) - c_x^\pi(i,k)) = 0$, ou seja, que $c_a^{\pi'} = 0$. □

Corolário 16.4. *Seja x um fluxo em uma rede G com custo c e seja π uma função-potencial tal que $c_a^\pi \geq 0$ para todo arco a de $G(x)$. Seja P um caminho de viabilidade em $G(x)$ e seja x' o fluxo obtido através de x e P. Seja i um vértice de G e seja π' a atualização de π em relação a i e x. Então, temos que $c_a^{\pi'} \geq 0$ para todo arco a de $G(x')$.*

Prova: Como a propriedade (i) do Lema 16.3 garante que $c_a^{\pi'} \geq 0$ para todo arco a de $G(x)$, temos apenas que provar que $c_a^{\pi'} \geq 0$ para qualquer arco a que está em $G(x')$ e não está em $G(x)$. Mas temos por definição que se a está em $G(x')$ e não está em $G(x)$, então $\overline{a}$ está em P. Logo, pela propriedade (ii) do Lema 16.3, temos que $c_{\overline{a}}^{\pi'} = 0$. Portanto, o Lema 14.8 nos permite concluir que $c_a^{\pi'} = 0$ e obtemos o resultado. □

16.3 Descrição

Como vimos anteriormente, o método do cancelamento de circuitos mantém um fluxo viável e busca, em cada iteração, minimizar o custo desse fluxo. Quando o algoritmo pára, certas propriedades garantem que o custo do fluxo viável mantido não pode mais ser reduzido, ou seja, o custo é mínimo.

O método dos caminhos de viabilidade, em contrapartida, mantém um fluxo não necessariamente viável e satisfaz desde o início certas propriedades que garantem que, se o fluxo mantido for viável, ele tem custo mínimo. Em cada iteração, o método busca viabilizar o fluxo mantido, através do aumento de fluxo em caminhos de viabilidade, sempre fazendo todas as atualizações necessárias para garantir as propriedades que minimizam o custo.

Sob o ponto de vista de programação linear, pode-se dizer que o método do cancelamento de circuitos mantém viabilidade primal e busca viabilidade dual, enquanto o método dos caminhos de viabilidade faz o contrário.

O algoritmo iterativo abaixo recebe uma rede G com capacidade u, custo c e demanda b e devolve um fluxo viável de custo mínimo e uma função-potencial que certifica a minimalidade do custo desse fluxo. Cada iteração começa com um fluxo x e uma função-potencial π. No início da primeira iteração, temos que $x_a = 0$ para todo arco a tal que $c_a \geq 0$, $x_a = u_a$ para todo arco a tal que $c_a < 0$ e $\pi(i) = 0$ para todo vértice i.

Caso 1: Não existe vértice i tal que $e(i) > b(i)$.
Devolva x e π e pare.

Caso 2: Existe vértice i tal que $e(i) > b(i)$.
Seja j um vértice tal que $e(j) < b(i)$.

Seja P um caminho de viabilidade entre o vértice i e o vértice j tal que P é um caminho na rede $G(x)$ correspondente ao custo $c_x^\pi(i,j)$.
Seja x' o fluxo obtido a partir de x e P.
Seja π' a atualização de π em relação a i e x.
Comece nova iteração com x' no papel de x e π' no papel de π.

Intuitivamente, o método dos caminhos de viabilidade consiste simplesmente em enviar fluxo de vértices com excesso maior do que a demanda para vértices com excesso menor do que a demanda, de forma a eliminar sucessivamente o desequilíbrio de todos os vértices da rede. A figura 16.1 mostra uma simulação completa do método dos caminhos de viabilidade.

Cabe observar que a correção do método só é garantida se a rede G admite um fluxo viável e a rede residual $G(x)$ é conexa para qualquer fluxo x. As Proposições 14.1 e 14.2 garantem que essas duas propriedades podem ser obtidas com pré-processamentos polinomiais. Portanto, para os resultados a seguir, vamos assumir sem perda de generalidade que elas são válidas.

16.4 Correção e desempenho

Vamos demonstrar a seguir resultados que provam que o fluxo x devolvido pelo algoritmo é um fluxo viável de custo mínimo.

Lema 16.5. *No início de qualquer iteração do método dos caminhos de viabilidade, temos que $c_a^\pi \geq 0$ para todo arco a de $G(x)$.*

Prova: Primeiramente, considere a primeira iteração do algoritmo. Temos que $\pi(i) = 0$ para todo vértice i. Seja $a = (j,k)$ um arco de $G(x)$. Se a está em G, então temos que $c_a \geq 0$, o que implica que $c_a^\pi = c_a - \pi(j) + \pi(k) = c_a - 0 + 0 = c_a \geq 0$. Analogamente, se a não está em G, então temos $c_{\overline{a}} < 0$, o que implica que $c_a^\pi = -c_{\overline{a}} - \pi(k) + \pi(j) = -c_{\overline{a}} - 0 + 0 = -c_{\overline{a}} \geq 0$.

Portanto, temos que a propriedade desejada é válida na primeira iteração. A validade nas iterações seguintes segue diretamente do Corolário 16.4. □

Lema 16.6. *Seja x o fluxo devolvido pelo método dos caminhos de viabilidade. Então x é um fluxo viável.*

Prova: Como a condição do Caso 1 garante que não existe um vértice i na rede tal que $e(i) > b(i)$, então por definição não existem caminhos de viabilidade na rede $G(x)$. Logo, o resultado segue do Corolário 16.2. □

Corolário 16.7. *Seja x o fluxo devolvido pelo método dos caminhos de viabilidade. Então x é um fluxo viável de custo mínimo.*

Prova: O resultado segue dos Lemas 16.5 e 16.6 e do Teorema 14.7. □

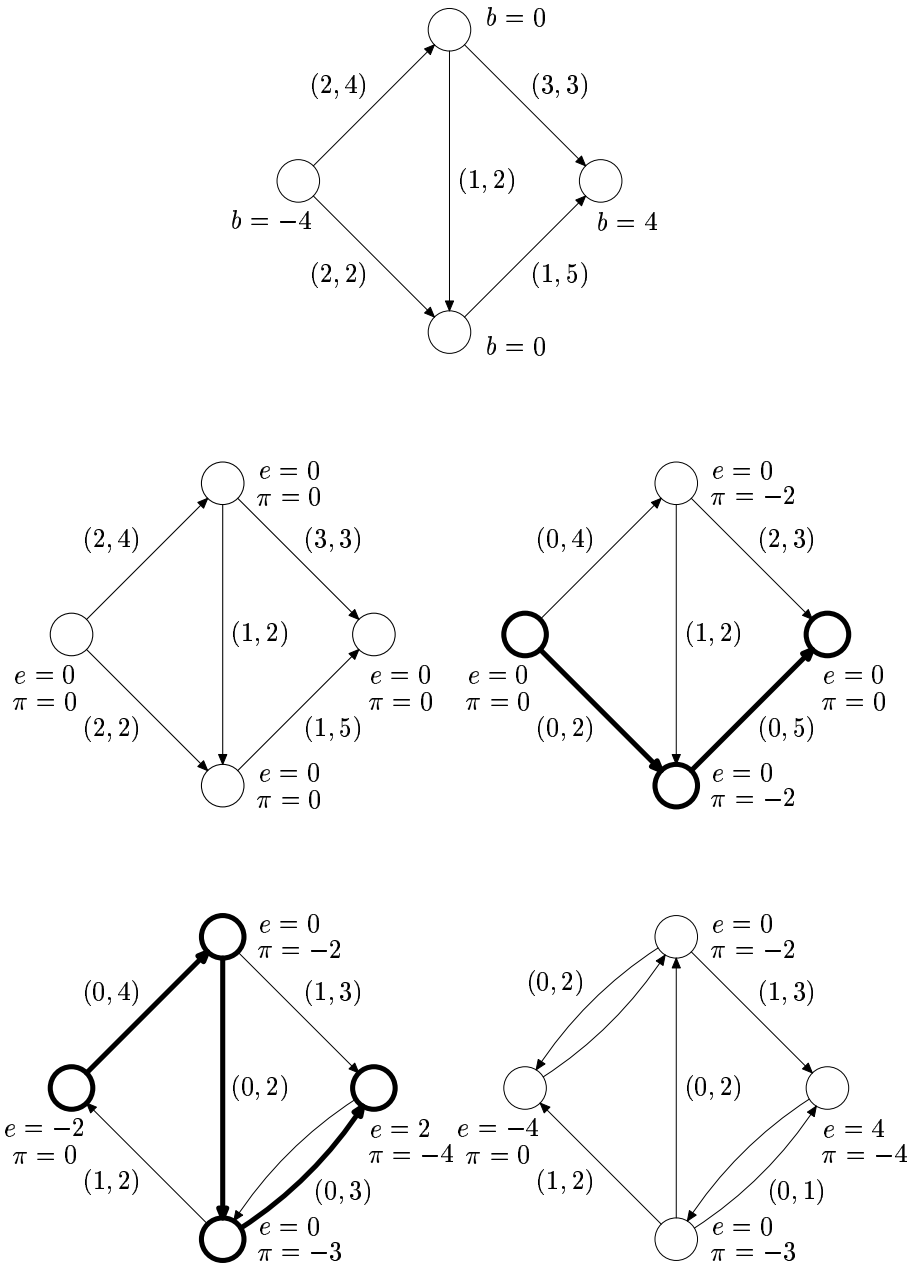

Figura 16.1: Simulação do método dos caminhos de viabilidade. Em cada diagrama, os números entre parênteses próximos aos arcos representam, respectivamente, seu custo reduzido e sua capacidade residual

Os resultados obtidos acima demonstram que se o método pára, então o fluxo devolvido com certeza é um fluxo viável de custo mínimo. Para concluir a prova da correção, resta mostrar que o método pára após um número finito de iterações, o que é feito através do resultado abaixo.

Proposição 16.8. *O método dos caminhos de viabilidade encontra um fluxo viável de custo mínimo em $O(nmU)$ iterações.*

Prova: Seja x o fluxo mantido pelo método dos caminhos de viabilidade e considere a função $\Phi(y) = \sum_{i \in N_G} |b(i) - e_y(i)|$ para um dado fluxo y. Pelas condições do Caso 1 e pelo Lema 16.1, temos que o método dos caminhos de viabilidade pára quando $\Phi(x) = 0$. Ademais, o excesso de um vértice é no máximo mU, portanto o valor inicial de $\Phi(x)$ é no máximo nmU.

Seja x' um fluxo obtido a partir de x e um caminho de viabilidade P em $G(x)$. Sabemos por definição que a origem i de P é tal que $e_x(i) > b(i)$ e $e_{x'}(i) < e_x(i)$ e o destino j de P é tal que $e_x(j) < b(j)$ e $e_{x'}(j) > e_x(j)$. Portanto, podemos concluir que $\Phi(x') < \Phi(x)$. Ademais, também podemos observar que o valor de $\Phi(x)$ nunca aumenta ao longo do método.

Como as capacidades dos arcos são números racionais, então existe um inteiro positivo K tal que Ku_a é um inteiro para todo arco a da rede. Logo, temos que em cada iteração do método, a capacidade residual do caminho de viabilidade escolhido é pelo menos $1/K$. Isso implica que o valor de $\Phi(x)$ diminui em pelo menos $1/K$ a cada iteração do método, o que nos permite finalmente concluir que o número de iterações do método é $O(nmU)$. □

Em cada iteração do método dos caminhos de viabilidade, é preciso calcular os custos de caminhos de custo mínimo entre um determinado vértice e todos os outros da rede. O conhecido algoritmo de Dijkstra [8] executa essa operação em tempo $O(n + m \log n)$ segundo a implementação de Cormen, Leiserson, Rivest e Stein [6]. Todas as outras operações necessárias, como o aumento do fluxo através do caminho de viabilidade e as atualizações, podem ser feitas em tempo $O(n + m)$. Disso inferimos o resultado abaixo.

Proposição 16.9. *O método dos caminhos de viabilidade pode ser implementado de modo a executar em tempo $O(nmU(n + m \log n))$.*

Cabe observar que demonstramos uma complexidade pseudo-polinomial para o método dos caminhos de viabilidade. No próximo capítulo apresentaremos uma implementação específica do método que permite resolver o problema do fluxo de custo mínimo com consumo de tempo polinomial.

Capítulo 17

Path scaling

Como o método dos caminhos de viabilidade para resolver o problema do fluxo de custo mínimo possui muitas similaridades com o método dos caminhos de aumento para resolver o problema do fluxo máximo, é natural esperar que certas estratégias e técnicas que permitem uma melhoria de desempenho em um deles também tenham o mesmo efeito no outro.

De fato, o método dos caminhos de viabilidade também pode ser implementado utilizando-se a mesma estratégia de *scaling* utilizada no algoritmo capacity scaling, ou seja, a escolha de caminhos de viabilidade com capacidade residual suficientemente grande. É nessa idéia recorrente que reside o **algoritmo path scaling**, idealizado originalmente por Orlin [18].

Este capítulo introduz conceitos necessários para desenvolver o algoritmo, descreve seu funcionamento e demonstra sua complexidade.

17.1 Correções de fluxo

Assim o algoritmo capacity scaling, o algoritmo path scaling utiliza o conceito de redes residuais restritas por uma constante Δ. Entretanto, resultados essenciais para a correção do método dos caminhos de viabilidade não são válidos para redes restritas. Particularmente, não é possível utilizar o Teorema 14.7, pois como a rede restrita pode não conter todos os arcos da rede residual original, as condições do teorema podem não estar satisfeitas.

Para manter a correção do algoritmo mesmo diante dessas circunstâncias, é necessária uma operação especial que altera o valor do fluxo mantido pelo algoritmo. Seja x um fluxo sobre uma rede G com custo c e seja π uma função-potencial sobre G. O **fluxo obtido após a correção de x em π** é um fluxo x', também sobre a rede G, que é definido da seguinte maneira:

$$x'_a = \begin{cases} u_a & \text{se } r_a \geq \Delta/2 \text{ e } c_a^\pi < 0, \\ x_a & \text{caso contrário.} \end{cases}$$

A correção de um fluxo visa eliminar da rede residual todos os arcos com custo reduzido negativo, de forma a satisfazer as condições do Teorema 14.7.

17.2 Descrição

A idéia do algoritmo path scaling é modificar a versão genérica do método dos caminhos de viabilidade de tal forma que, em cada iteração, exista um limitante inferior para a capacidade residual de um caminho de viabilidade. Esse limitante é gradualmente reduzido durante a execução do algoritmo, de forma que a correção é garantida pelo caso em que ele é mínimo.

Para que esse limitante de fato viabilize uma melhoria de complexidade, ele também é utilizado para limitar inferiormente o desequilíbrio dos vértices escolhidos como extremos, pois pela definição de fluxo obtido a partir de um caminho de viabilidade, uma capacidade residual alta só é relevante se os desequilíbrios dos extremos do caminho considerado também forem altos.

O algoritmo iterativo abaixo recebe uma rede G com capacidade u, custo c e demanda b e devolve um fluxo viável de custo mínimo e uma função-potencial que certifica a minimalidade. Cada iteração começa com um fluxo x, uma função-potencial π e um limitante Δ. No início da primeira iteração, temos que $x_a = 0$ para todo arco a tal que $c_a \geq 0$, $x_a = u_a$ para todo arco a tal que $c_a < 0$, $\pi(i) = 0$ para todo vértice i e $\Delta = 2^{\lfloor \log U \rfloor}$. Para este algoritmo, o valor $c_x^\pi(i,j)$ refere-se ao custo de um caminho de custo mínimo entre i e j na rede restrita $G(x,\Delta)$, não na rede residual $G(x)$.

Caso 1: $\Delta < 1$

Devolva x e π e pare.

Caso 2: $\Delta \geq 1$

Caso 2A: Não existe um vértice i tal que $e(i) - b(i) \geq \Delta$ ou
não existe um vértice j tal que $e(j) - b(j) \leq -\Delta$.

Seja x' o fluxo obtido após a correção de x em π.

Comece nova iteração com x' no papel de x e $\Delta/2$ no papel de Δ.

Caso 2B: Existe um vértice i tal que $e(i) - b(i) \geq \Delta$ e
existe um vértice j tal que $e(j) - b(j) \leq -\Delta$.

Seja P um caminho de viabilidade entre o vértice i e o vértice j que é um caminho em $G(x,\Delta)$ correspondente ao custo $c_x^\pi(i,j)$.

Seja x' o fluxo obtido a partir de x e P.

Seja π' a atualização de π em relação a i e x.

Comece nova iteração com x' no papel de x e π' no papel de π.

De maneira análoga aos algoritmos de scaling anteriores, o algoritmo path scaling executa $O(\log U)$ fases de scaling e sua correção só é garantida se executarmos um pré-processamento e um pós-processamento que nos permitam assumir sem perda de generalidade que as capacidades são inteiras.

17.3 Desempenho

O resultado abaixo fornece um limitante polinomial para o número total de iterações do algoritmo, uma melhoria em relação à versão genérica.

Proposição 17.1. *O algoritmo path scaling encontra um fluxo viável de custo mínimo em* $O(nm + (n+m)\log U)$ *iterações.*

Prova: Vamos considerar novamente a função $\Phi(y) = \sum_{i \in N_G} |b(i) - e_y(i)|$ para um dado fluxo y. No início da primeira fase de scaling do algoritmo, temos que $\Phi(x)$ é no máximo $nm\Delta$. Pela descrição do algoritmo, temos que o aumento de fluxo em um caminho de viabilidade reduz $\Phi(x)$ em pelo menos Δ. Logo, a primeira fase de scaling executa no máximo nm iterações.

Considere agora uma fase de scaling qualquer do algoritmo. Pelas condições do Caso 2A, temos que no final dessa fase é válido que $e(i) - b(i) < \Delta$ para todo vértice i tal que $e(i) - b(i) > 0$ ou $e(i) - b(i) > -\Delta$ para todo vértice i tal que $e(i) - b(i) < 0$. Logo, pelo Lema 3.1 podemos concluir que $\Phi(x) < n\Delta$ no final de uma fase de scaling qualquer do algoritmo.

Como temos que $c_a^\pi \geq 0$ para qualquer arco a da rede $G(x, \Delta)$, temos que se $c_a^\pi < 0$, então $r_a < \Delta$. Logo, podemos concluir que $\Phi(x)$ aumenta em no máximo $m\Delta$ durante uma correção de fluxo. Isso implica que, no início de qualquer fase de scaling do algoritmo temos que $\Phi(x)$ é no máximo $(n+m)\Delta$.

Portanto, podemos concluir que a primeira fase de scaling executa no máximo nm iterações e as fases seguintes executam no máximo $n+m$ iterações cada. Logo, o número de iterações é $O(nm + (n+m)\log U)$. □

Como vimos anteriormente que uma iteração do método dos caminhos de viabilidade pode ser implementada em tempo $O(n + m\log n)$, podemos concluir com o seguinte resultado.

Proposição 17.2. *O algoritmo path scaling pode ser implementado de modo a executar em tempo* $O((nm + (n+m)\log U)(n + m\log n))$.

Capítulo 18

Cost scaling

Existe um algoritmo polinomial para resolver o problema do fluxo viável de custo mínimo que não se baseia em nenhum dos dois métodos pseudopolinomiais apresentados anteriormente, e sim generaliza o método de pré-fluxo utilizado para resolver o problema do fluxo máximo.

O algoritmo **cost scaling**, proposto por Goldberg e Tarjan [16], baseia-se na idéia de manter uma solução *aproximada* para o problema do fluxo de custo mínimo e melhorar sucessivamente a aproximação através de pushes e relabels.

18.1 Otimalidade aproximada

Um fluxo x em uma rede com custo c é dito **ϵ-ótimo**, para algum $\epsilon \geq 0$, se existe uma função-potencial π tal que, para todo arco a da rede, vale que:
(i) Se $c_a^\pi > \epsilon$, então $x_a = 0$.
(ii) Se $-\epsilon \leq c_a^\pi \leq \epsilon$, então $0 \leq x_a \leq u_a$.
(iii) Se $c_a^\pi < -\epsilon$, então $x_a = u_a$.

As condições (i),(ii) e (iii) são ditas **condições de ϵ-otimalidade**.

Apresentamos a seguir alguns resultados relacionados à ϵ-otimalidade de fluxos, utilizados para a descrição e análise do algoritmo cost scaling.

Lema 18.1. *Seja x um fluxo em uma rede G com custo c. Então x é ϵ-ótimo em relação a uma função-potencial π se, e somente se, $c_a^\pi \geq -\epsilon$, para todo arco a da rede residual $G(x)$.*

Prova: Seja x um fluxo ϵ-ótimo em relação à função-potencial π. Suponha por contradição que $c_a^\pi < -\epsilon$, para algum arco a da rede residual $G(x)$.

Se a é um arco de G, então, pela condição (iii) de ϵ-otimalidade, temos que $x_a = u_a$. Mas isso é uma contradição, pois a é um arco da rede residual $G(x)$ não podendo, portanto, possuir capacidade residual nula. Por outro lado, se a não é um arco de G, então, como a está na rede residual, devemos ter $x_{\overline{a}} > 0$. Agora, pelo Lema 14.8, temos que $c_{\overline{a}}^\pi = -c_a^\pi > \epsilon$. Mas, então,

pela condição (i) de ϵ-otimalidade, vale que $x_{\overline{a}} = 0$, o que também é uma contradição. Assim, $c_a^\pi \geq -\epsilon$, para todo arco a em $G(x)$.

Considere agora um fluxo x e uma função-potencial π em G tais que que $c_a^\pi \geq -\epsilon$, para todo arco a de $G(x)$. Vamos mostrar que x é um fluxo ϵ-ótimo com relação à π. Para tanto, vamos verificar que as condições de ϵ-otimalidade são satisfeitas por qualquer arco de G.

Seja a um arco da rede G. Se $c_a^\pi > \epsilon$ então, novamente pelo Lema 14.8, temos que $c_{\overline{a}}^\pi < -\epsilon$, o que nos permite concluir que $\overline{a}$ não está em $G(x)$, ou seja, que $x_a = 0$. Assim, a condição (i) de ϵ-otimalidade é válida.

Como x é um fluxo, então a condição (ii) é trivialmente válida.

Por fim, se $c_a^\pi < -\epsilon$, então a não está na rede residual, o que implica que $x_a = u_a$. Ou seja, a condição (iii) também é válida.

Portanto, x é um fluxo ϵ-ótimo. □

Lema 18.2. *Seja G uma rede com demanda e custo c. Então qualquer fluxo em G é ϵ-ótimo se $\epsilon \geq C$.*

Prova: Seja x um fluxo qualquer em G e seja π uma função-potencial tal que $\pi(i) = 0$, para todo vértice i. Temos, então, que $c_a^\pi = c_a \geq -C$, para qualquer arco a na rede residual $G(x)$. Portanto, pelo Lema 18.1, concluímos que se $\epsilon \geq C$, então x é um fluxo ϵ-ótimo. □

Lema 18.3. *Seja G uma rede com demanda e custo inteiro c. Se $\epsilon < 1/n$, então qualquer fluxo viável ϵ-ótimo em G é um fluxo viável de custo mínimo nessa rede.*

Prova: Seja x um fluxo viável ϵ-ótimo, com $\epsilon < 1/n$. Seja W um circuito em $G(x)$. Pelo Lema 18.1, temos que $\sum_{a \in W} c_a^\pi \geq -\epsilon n > -1$. Como os custos são todos inteiros, isso implica que $\sum_{a \in W} c_a^\pi$ é não-negativo, para qualquer circuito W em $G(x)$. Agora, sabemos pelo Lema 14.6 que $\sum_{a \in W} c_a^\pi = \sum_{a \in W} c_a = \text{custo}(W)$. Concluímos, assim, que $G(x)$ não contém circuitos negativos e, pelo Teorema 14.5, que x é um fluxo viável de custo mínimo. □

18.2 Pushes e relabels aproximados

O algoritmo cost scaling mantém um fluxo ϵ-ótimo e busca transformá-lo num fluxo viável de custo mínimo através de variações dos procedimentos push e relabel, primeiramente apresentados durante a descrição do método do pré-fluxo para a resolução do problema do fluxo máximo.

Apesar de baseados em idéias semelhantes às dos procedimentos push e relabel já descritos, os procedimentos utilizados pelo cost scaling apresentam diferenças fundamentais em relação àqueles. Vamos, portanto, redefinir tais procedimentos, bem como os conceitos relacionados, no contexto do algoritmo que estamos descrevendo.

Seja G uma rede capacitada, com custo c e demanda b e seja x um fluxo qualquer em G. Considere também uma função-potencial π sobre essa rede e um valor positivo ϵ qualquer.

Um vértice i de G é dito **ativo** se $e_x(i) > b(i)$ e um arco a da rede residual $G(x)$ é dito **admissível** se $-\epsilon/2 \leq c_a^\pi < 0$.

Seja i um vértice ativo em G e seja a um arco admissível em $\delta^-_{G(x)}(i)$.

Um **relabel** de i é o processo de obtenção de uma função-potencial π', a partir de i e π, da seguinte maneira: para cada vértice j em N_G,

$$\pi'(j) = \begin{cases} \pi(j) & \text{se } j \neq i, \\ \pi(j) + \epsilon/2 & \text{caso contrário.} \end{cases}$$

A função π' é a **função-potencial obtida após um relabel de** i.

Um **push** em a é o processo de obtenção de um fluxo x' em G, a partir de x e a, da seguinte maneira: para cada arco b em A_G,

$$x'_b = \begin{cases} x_b + \min\{r_a, e_x(i) - b(i)\} & \text{se } b = a, \\ x_b - \min\{r_a, e_x(i) - b(i)\} & \text{se } b = \overline{a}, \\ x_b & \text{caso contrário.} \end{cases}$$

Podemos observar que a definição do push garante que x' é de fato um fluxo. O fluxo x' é dito o **fluxo obtido após um push em** a.

Também de forma análoga ao já definido, um push é **saturador** se $r_a = \min\{r_a, e_x(i) - b(i)\}$ e é dito **não-saturador** caso contrário.

18.3 Melhoria de Aproximação

Apresentamos aqui um algoritmo de **melhoria de aproximação** que recebe uma rede G, um fluxo viável x' e uma função potencial π' em G, tais que x' é ϵ-ótimo com relação à π', e devolve um fluxo viável x e uma função-potencial π em G, de forma que x é $\frac{\epsilon}{2}$-ótimo com relação a π. Para realizar a melhoria de aproximação, o algoritmo utiliza os procedimentos push e relabel, bem como os conceitos relacionados, definidos na seção anterior.

O algoritmo, descrito abaixo, é iterativo e mantém em cada iteração um fluxo x e uma função-potencial π. Na primeira iteração, $\pi = \pi'$ e x é definido da seguinte forma: $x_a = 0$, para todo arco a tal que $c_a^\pi > 0$, $x_a = u_a$, para todo arco a tal que $c_a^\pi < 0$, e $x_a = x'_a$, para todo arco a tal que $c_a^\pi = 0$.

Caso 1: Não existe um vértice ativo em N_G.
Devolva x e π e pare.

Caso 2: Existe um vértice ativo em N_G.
Seja i um vértice ativo em N_G.

Caso 2A: Existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja a um arco admissível em $\delta^-_{G(x)}(i)$.
Seja x'' o fluxo obtido após um push em a.
Comece nova iteração com x'' no papel de x.

Caso 2B: Não existe um arco admissível em $\delta^-_{G(x)}(i)$.
Seja π'' a função-potencial obtida após um relabel de i.
Comece nova iteração com π'' no papel de π.

O fluxo x e a função-potencial π devolvidos pelo algoritmo de melhoria de aproximação são ditos **fluxo e função-potencial obtidos após uma melhoria de aproximação sobre x e π.**

O algoritmo cost scaling utiliza o algoritmo de melhoria de aproximação como subrotina. Mais adiante, após a descrição completa do cost scaling, demonstramos que o algoritmo de melhoria de aproximação sempre termina, devolvendo um fluxo viável $\frac{\epsilon}{2}$-ótimo e limitamos seu número de iterações.

18.4 Descrição

O algoritmo cost scaling é um algoritmo iterativo que recebe uma rede capacitada G, com custo inteiro e demanda, e devolve um fluxo viável de custo mínimo nessa rede e uma função-potencial que certifica a minimalidade do custo do fluxo devolvido. Durante toda a sua execução, ele mantém um fluxo viável ϵ-ótimo. A idéia é, então, reduzir iterativamente o valor de ϵ, através da utilização do já descrito algoritmo de melhoria de aproximação.

Em cada iteração, o algoritmo cost scaling mantém um fluxo x, uma função potencial π e um limitante ϵ. Na primeira iteração, x é um fluxo viável qualquer, $\pi(i) = 0$, para todo vértice i da rede e $\epsilon = C$. A descrição do algoritmo está abaixo.

Caso 1: $\epsilon < 1/n$.
Devolva x e π e pare.

Caso 2: $\epsilon \geq 1/n$.
Sejam x' e π' o fluxo e a função-potencial obtidos após uma melhoria de aproximação sobre x e π.
Comece nova iteração com x' no papel de x, π' no papel de π e $\epsilon/2$ no papel de ϵ.

Conforme mencionado, o cost scaling supõe que os custos dos arcos da rede recebida como parâmetro são todos inteiros. Pela Seção 14.4, sabemos que o problema de encontrar um fluxo de custo mínimo em uma rede com custo racional pode ser reduzido ao problema de encontrar um fluxo de custo mínimo em uma rede com custo inteiro. Além disso, tal redução pode facilmente ser realizada em tempo polinomial.

Assim, assumiremos, sem perda de generalidade, que o algoritmo cost scaling sempre recebe uma rede com custo inteiro e utilizaremos esse fato para prova de sua correção e análise de seu desempenho.

Assumiremos também que o algoritmo cost scaling só é aplicado sobre redes que admitam um fluxo viável e sobre as quais já exista um fluxo viável pré-estabelecido. Tal fluxo viável, ou a inexistência do mesmo, também pode ser determinado através de um pré-processamento polinomial, conforme apresenta a Seção 14.2.

18.5 Correção e desempenho

Começaremos a análise da correção e do desempenho do algoritmo cost scaling através da análise da correção e do desempenho do algoritmo de melhoria de aproximação. Para tanto, vamos limitar o seu número de iterações e verificar que ele sempre devolve um fluxo viável $\frac{\epsilon}{2}$-ótimo.

Lema 18.4. *O algoritmo de melhoria de aproximação sempre mantém um fluxo $\frac{\epsilon}{2}$-ótimo.*

Prova: Vamos fazer uma prova por indução no número de iterações.

Por construção, o fluxo x com o qual o algoritmo de melhoria de aproximação inicia sua primeira iteração é 0-ótimo. Logo, a $\frac{\epsilon}{2}$-otimalidade do fluxo mantido é claramente válida no início do algoritmo.

Vamos supor, então, que o fluxo mantido no início de determinada iteração do algoritmo é $\frac{\epsilon}{2}$-ótimo e analisar os efeitos dos possíveis pushes e relabels efetuados. Neste caso, pelo Lema 18.1, temos que $c_a^\pi \geq -\epsilon/2$, para todo arco a da rede residual, no início da iteração.

Se ocorre um push em um arco a, então o único arco que pode passar a pertencer à rede residual no final da iteração é o arco $\overline{a}$. Como o arco a é um arco admissível, vale que $-\epsilon/2 \leq c_a^\pi < 0$. Assim, pelo Lema 14.8, temos que $c_{\overline{a}}^\pi > 0 > -\epsilon/2$. Notemos também que um push não altera o potencial de nenhum vértice. Portanto, a propriedade $c_a^\pi \geq -\epsilon/2$ para todo arco a da rede residual é mantida ao final da iteração. Ou seja, a $\frac{\epsilon}{2}$-otimalidade do fluxo mantido é conservada.

Um relabel de um vértice ativo i ocorre quando não há arcos admissíveis em $\delta^-_{G(x)}(i)$, ou seja, quando todo arco a em $\delta^-_{G(x)}(i)$ é tal que $c_a^\pi \geq 0$ ou $c_a^\pi < -\epsilon/2$. Se a é um arco da rede residual no início da iteração, então $c_a^\pi \geq -\epsilon/2$, pela hipótese de indução. Portanto, temos que todo arco a em $\delta^-_{G(x)}(i)$ é tal que $c_a^\pi \geq 0$, quando ocorre um relabel de i.

Como após um relabel o potencial de i aumenta em $\epsilon/2$, temos que o custo reduzido de todos os arcos em $\delta^-_{G(x)}(i)$ reduz em $\epsilon/2$. Mas, como vimos que o custo reduzido de qualquer arco em $\delta^-_{G(x)}(i)$ é não-negativo, então, após o relabel, nenhum deles torna-se menor do que $-\epsilon/2$. Notemos também que, após um relabel de i, o custo reduzido dos arcos em $\delta^+_{G(x)}(i)$

aumenta e que i é o único vértice que tem seu potencial alterado. Assim, após uma iteração em que ocorre um relabel, a propriedade $c_a^\pi \geq -\epsilon/2$ é mantida para todo arco a da rede residual e, portanto, a $\frac{\epsilon}{2}$-otimalidade do fluxo mantido é conservada, o que conclui a indução. □

Lema 18.5. *São executados no máximo 3n relabels sobre cada vértice durante o algoritmo de melhoria de aproximação.*

Prova: Seja x um fluxo não-viável $\frac{\epsilon}{2}$-ótimo durante uma melhoria de aproximação e seja x' o fluxo viável ϵ-ótimo recebido pelo algoritmo como parâmetro.

Seja i um vértice qualquer tal que $e_x(i) > b(i)$ e considere o fluxo-diferença $x' - x$. Temos que existe um caminho P entre o vértice i e um vértice j em G tal que $e_x(j) < b(j)$ e $(x' - x)_a > 0$, para todo arco a de P, pois, caso contrário, x' não seria um fluxo viável.

Assim, temos que $P = \langle i_0, \ldots, i_k \rangle$ é um caminho em $G(x)$ e que o caminho reverso $\overline{P} = \langle i_k, \ldots, i_0 \rangle$, formado pelos arcos irmãos dos arcos de P, é um caminho em $G(x')$.

Sejam π e π' os potenciais correspondentes a x e x' respectivamente. Como x é $\frac{\epsilon}{2}$-ótimo, então, pelo Lema 18.1, temos que $\sum_{a \in P} c_a^\pi \geq -k(\epsilon/2)$. Portanto, pelo Lema 14.6, vale que $\sum_{a \in P} c_a - \pi(i) + \pi(j) \geq -k(\epsilon/2)$. Analogamente, como x' é ϵ-ótimo, temos que $\sum_{a \in \overline{P}} c_a^{\pi'} \geq -k\epsilon$ e, portanto, $\sum_{a \in \overline{P}} c_a - \pi'(j) + \pi'(i) \geq -k\epsilon$.

Como os arcos de P são os irmãos dos arcos de $\overline{P}$, temos que $\sum_{a \in P} c_a = -\sum_{a \in \overline{P}} c_a$ e podemos inferir que $\sum_{a \in P} c_a \leq k\epsilon - \pi'(j) + \pi'(i)$.

Temos, então, que $\sum_{a \in P} c_a \leq k\epsilon - \pi'(j) + \pi'(i)$ e $\sum_{a \in P} c_a \geq -k(\epsilon/2) + \pi(i) - \pi(j)$. Ou seja, $-k(\epsilon/2) + \pi(i) - \pi(j) \leq k\epsilon - \pi'(j) + \pi'(i)$ e, finalmente, $(\pi(i) - \pi'(i)) \leq (\pi(j) - \pi'(j)) + 3k\epsilon/2$.

Por fim, observamos que, como $e_x(j) < b(j)$, então o vértice j nunca foi vértice ativo e, portanto, não sofreu relabels. Isto é, $\pi(j) = \pi'(j)$. Ademais, temos que $k \leq n$. Assim, $(\pi(i) - \pi'(i)) \leq 3n\epsilon/2$ e, como cada relabel de i aumenta o potencial de i em $\epsilon/2$, concluímos que o vértice i não pode sofrer mais do que $3n$ relabels. □

Corolário 18.6. *São executados $O(n^2)$ relabels durante o algoritmo de melhoria de aproximação.*

Prova: O resultado segue diretamente do Lema 18.5. □

Lema 18.7. *São executados $O(nm)$ pushes saturadores durante o algoritmo de melhoria de aproximação.*

Prova: Denotemos por π_k o potencial mantido pelo algoritmo de melhoria de aproximação na iteração k.

Seja $a = (i, j)$ um arco que sofre um push saturador em determinada iteração α. Suponha que exista uma outra iteração γ, com $\gamma > \alpha$, em que

a sofra novamente um push saturador. Para que ocorra um push em a, tal arco precisa ser admissível. Temos, portanto, que $c_a^{\pi_\alpha} < 0$ e $c_a^{\pi_\gamma} < 0$.

Para que um arco seja admissível, ele precisa estar na rede residual. Entretanto, sabemos que o arco a não está na rede residual no início da iteração $\alpha+1$. Sendo assim, para que a esteja na rede residual na iteração γ, é preciso existir uma iteração β, com $\alpha < \beta < \gamma$, em que o fluxo no arco $\overline{a} = (j, i)$ foi aumentado. Ou seja, o arco $\overline{a}$ deve ter sofrido um push na iteração β. Mas então o arco $\overline{a}$ era um arco admissível nessa iteração e, portanto, $c_{\overline{a}}^{\pi_\beta} < 0$. Ou ainda, pelo Lema 14.8, $c_a^{\pi_\beta} > 0$.

Agora, como o custo reduzido de $a = (i, j)$ é negativo na iteração α e positivo na iteração β e como os potenciais de todos os vértices nunca diminuem durante a melhoria de aproximação, então o vértice j sofreu um relabel entre as iterações α e β. Analogamente, como o custo reduzido de a é positivo na iteração β e negativo na iteração γ, então o vértice i deve ter sofrido um relabel entre as iterações β e γ. Concluímos, portanto, que entre dois pushes saturadores de um arco suas duas pontas sempre sofrem um relabel.

Assim, como pelo Lema 18.5 cada vértice sofre $O(n)$ relabels, então cada arco sofre $O(n)$ pushes saturadores. Conseqüentemente, o número de pushes saturadores está limitado em $O(nm)$. □

Lema 18.8. *Durante o algoritmo de melhoria de aproximação, a rede formada pelos arcos admissíveis não contém circuitos.*

Prova: Vamos provar a propriedade por indução no número de iterações. Como o fluxo construído antes do início da primeira iteração do algoritmo de melhoria de aproximação é 0-ótimo, então não existem arcos admissíveis e, portanto, a propriedade é trivialmente válida.

Vamos supor, então, que a propriedade valha no início de determinada iteração e mostrar que ela continua válida ao seu final. Para tanto, vamos verificar o efeito dos pushes e relabels sobre a rede formada apenas pelos arcos admissíveis.

Se ocorre um push em um arco a, então o único arco que pode passar a pertencer à rede residual é o arco $\overline{a}$. Além disso, como o arco a sofre um push, então ele é um arco admissível e, portanto, $c_a^\pi < 0$. Mas, então, pelo Lema 14.8, temos que $c_{\overline{a}}^\pi > 0$. Ou seja, um push não adiciona arcos admissíveis à rede e, conseqüentemente, não há a formação de circuitos.

Se ocorre um relabel de um vértice i, então todos os arcos em $\delta_G^-(i)$ têm seu custo reduzido diminuído em $\epsilon/2$, podendo tornar-se admissíveis. Mais ainda, tais arcos são os únicos que podem tornar-se admissíveis após o relabel. Entretanto, notemos também que todos os arcos em $\delta_G^+(i)$ que eram admissíveis no início da iteração deixam de ser admissíveis após o relabel. De fato, como pelo Lema 18.4 o algoritmo de melhoria de aproximação sempre mantém um fluxo $\frac{\epsilon}{2}$-ótimo, então, pelo Teorema 18.1, temos que $c_a^\pi \geq -\epsilon/2$, para todo arco a da rede residual. Ademais, um relabel em i aumenta o custo reduzido de cada arco em $\delta_G^+(i)$ em $\epsilon/2$.

Assim, não há a formação de circuito com a adição de arcos admissíveis em $\delta_G^-(i)$ e a propriedade continua válida após a iteração. □

Lema 18.9. *São executados $O(n^2m)$ pushes não-saturadores durante o algoritmo de melhoria de aproximação.*

Prova: Para cada vértice i da rede recebida como parâmetro, defina $g(i)$ como o número de vértices acessíveis a partir de i na rede formada apenas pelos arcos admissíveis. Notemos que $g(i) \geq 1$, para todo vértice i, uma vez que todo vértice é sempre acessível a partir dele próprio em qualquer rede. Ademais, é claro que $g(i) \leq n$ durante todo o algoritmo.

Vamos estabelecer um limitante para o número de pushes não-saturadores com o auxílio da função $\Phi := \sum_{i \text{ é ativo}} g(i)$. Para tanto, vamos verificar como cada operação efetuada pelo algoritmo altera o valor de Φ.

Como o fluxo construído antes do início do algoritmo é 0-ótimo, então não existem arcos admissíveis antes da primeira iteração e, conseqüentemente, o valor inicial de Φ é no máximo n.

Se ocorre um push saturador em um arco $a = (i, j)$, então o vértice j pode se tornar ativo, sem que o vértice i deixe de ser. Com isso, o valor de Φ pode ser acrescido em no máximo n após o push. Como sabemos pelo Lema 18.7 que o número total de pushes saturadores é $O(nm)$, concluímos que o valor de Φ é acrescido em $O(n^2m)$ após todos os pushes saturadores.

Se ocorre um relabel de um vértice ativo i, então todos os arcos em $\delta_G^-(i)$ têm seu custo reduzido diminuído em $\epsilon/2$, podendo, portanto, passar a ser admissíveis. Sendo assim, após um relabel de i, $g(i)$ pode ser acrescido em no máximo n. Por outro lado, conforme já argumentado na prova do Lema 18.8, sabemos também que todos os arcos em $\delta_G^+(i)$ passam a ter custo reduzido não-negativo e que, portanto, após o relabel, não existem arcos admissíveis em $\delta_G^+(i)$. Ou seja, após um relabel de i, $g(k)$ não aumenta para nenhum vértice $k \neq i$. Logo, Φ é acrescida em no máximo n após um relabel. Como pelo Lema 18.6 ocorrem $O(n^2)$ relabels durante todo o algoritmo, então Φ é acrescida em $O(n^3)$ em função de todos os relabels.

Suponha agora que ocorra um push não-saturador em um arco $a = (i, j)$. Neste caso, o vértice i deixa de ser ativo e o vértice j pode tornar-se ativo. Notemos, então, que $g(i) > g(j)$. De fato, todo vértice acessível a partir de j através de arcos admissíveis também o é a partir de i, uma vez que o arco a é admissível no início da iteração. Entretanto, como pelo Lema 18.8 a rede formada apenas pelos arcos admissíveis é acíclica, então o vértice i não pode ser acessível a partir do vértice j, de onde concluímos que $g(i) \geq g(j) + 1$. Assim, temos que o valor de Φ decresce em pelo menos 1 após um push não-saturador.

Temos, então, que o valor inicial de Φ é $O(n)$ e que Φ recebe um acréscimo de $O(n^2m) + O(n^3)$ durante toda a execução do algoritmo. Além disso, sabemos que o valor final de Φ é 0, pois o algoritmo pára apenas quando

não existem vértices ativos, e que cada push não-saturador, e apenas eles, diminuem o valor da função em pelo menos uma unidade.

Disso concluímos que o número de pushes não-saturadores é $O(n) + O(n^2m) + O(n^3)$. Por fim, observando que $m \geq n+1$ sob a hipótese de conexidade, concluímos que o número de pushes não-saturadores é $O(n^2m)$. □

Proposição 18.10. *O algoritmo de melhoria de aproximação devolve um fluxo viável $\frac{\epsilon}{2}$-ótimo em $O(n^2m)$ iterações.*

Prova: Em cada iteração do algoritmo, ocorre um relabel, um push saturador ou um push não-saturador. Logo, como conseqüência dos Lemas 18.7 e 18.9 e do Corolário 18.6, o algoritmo pára em $O(n^2m)$ iterações.

Agora, pelo Lema 18.4, o algoritmo de melhoria de aproximação sempre mantém um fluxo $\frac{\epsilon}{2}$-ótimo. Ademais, no início da última iteração, não existem vértices ativos. Isto é, não existem vértices tais que $e(i) > b(i)$ e, em função do Lema 16.1, não existem vértices com excesso diferente da demanda. Portanto, o fluxo x é viável no início da última iteração, de onde segue o resultado. □

Podemos, agora, estabelecer a correção e a análise do desempenho do algoritmo cost scaling.

Lema 18.11. *O algoritmo cost scaling mantém um fluxo viável ϵ-ótimo durante toda a sua execução.*

Prova: O resultado pode ser estabelecido por indução no número de iterações. Pelo Lema 18.2, temos que o fluxo x definido antes do início do algoritmo é um fluxo viável ϵ-ótimo com relação à função potencial π.

Ademais, pela Proposição 18.10, uma melhoria de aproximação, que deve receber como parâmetros um fluxo viável x e uma função-potencial π tais que x é ϵ-ótimo com relação a π, sempre devolve um fluxo viável x' e uma função potencial π' tais que x' é $\frac{\epsilon}{2}$-ótimo com relação a π'.

Portanto, se determinada iteração começa com um fluxo viável ϵ-ótimo, então ela termina com um fluxo viável $\frac{\epsilon}{2}$-ótimo.

Para concluir o resultado, basta observarmos que o valor de ϵ é reduzido pela metade ao final de cada iteração. □

Proposição 18.12. *O algoritmo cost scaling encontra um fluxo viável de custo mínimo em $O(\log(nC))$ iterações.*

Prova: Pelo Lema 18.11, no início da última iteração do cost scaling, x é um fluxo viável ϵ-ótimo, para algum $\epsilon < 1/n$. Agora, como estamos assumindo que os custos são todos inteiros, então, pelo Lema 18.3, x é um fluxo viável de custo mínimo no início da última iteração do algoritmo.

Para concluir, basta observarmos que todas as iterações do algoritmo cost scaling estão bem definidas, em função da corretude do algoritmo de

melhoria de aproximação, e que ocorrem $O(\log(nC))$ iterações, uma vez que o valor inicial de ϵ é C, o algoritmo termina quando $\epsilon < 1/n$ e o valor de ϵ é reduzido pela metade a cada iteração. □

Por fim, vamos apresentar alguns resultados para que seja possível estimar o consumo de tempo de uma implementação do algoritmo.

Lema 18.13. *Se um arco $a = (i, j)$ não é admissível no final de uma iteração α do algoritmo de melhoria de aproximação, então ele é admissível no início de uma iteração γ, com $\alpha < \gamma$, somente se existe uma iteração β, com $\alpha < \beta < \gamma$, na qual i sofre um relabel.*

Prova: Denotemos por π_k a função-potencial mantida pelo algoritmo de melhoria de aproximação na iteração k.

Se um arco a não é admissível no final da iteração α, então a não está na rede residual no final dessa iteração ou $c_a^{\pi_\alpha} > 0$. O caso em que $c_a^{\pi_\alpha} < -\epsilon/2$ nunca ocorre, pois sabemos que o algoritmo de melhoria de aproximação sempre mantém um fluxo $\frac{\epsilon}{2}$-ótimo.

Se o primeiro caso ocorre, então, para que a seja admissível na iteração γ, ele precisa estar na rede residual no início dessa iteração. Mas então deve ter ocorrido um push sobre o arco $\overline{a}$ em alguma iteração α', com $\alpha < \alpha' < \gamma$. Neste caso, o arco $\overline{a}$ era admissível na iteração α' e, portanto, $-\epsilon/2 < c_{\overline{a}}^{\pi_{\alpha'}} < 0$. Assim, pelo Lema 14.8, $\epsilon/2 > c_a^{\pi_{\alpha'}} > 0$. Ou seja, podemos assumir, sem perda de generalidade, que o segundo caso ocorre. Isto é, podemos assumir que $c_a^{\pi_\alpha} > 0$.

Temos, então, que o custo reduzido do arco a é positivo na iteração α e negativo na iteração γ, uma vez que a é um arco admissível nesta última iteração. Ademais, sabemos que durante toda a execução do algoritmo os potenciais só aumentam e, portanto, o custo reduzido do arco $a = (i, j)$ só diminui quando o potencial do vértice i aumenta. Por fim, notamos que o potencial do vértice i aumenta apenas quando o mesmo sofre um relabel, de onde podemos concluir o resultado. □

Lema 18.14. *Durante a execução do algoritmo de melhoria de aproximação, o tempo total gasto para saber qual operação deve ser feita em cada iteração é $O(nm)$.*

Prova: Um vértice i deve sofrer relabel em uma iteração se nenhum arco de $\delta^-_{G(x)}(i)$ é admissível nessa iteração. Portanto, em uma análise de pior caso temos que o conjunto $\delta^-_{G(x)}(i)$ deve ser varrido em todas as iterações. Porém, o Lema 18.13 implica que se um arco foi verificado como não admissível, ele não precisa ser verificado novamente até o vértice i sofrer um relabel.

Logo, como o Lema 18.5 nos diz que um vértice sofre no máximo $3n$ relabels, temos que o tempo total gasto para descobrir se os vértices ativos devem sofrer push ou relabel é $O(3n \sum_{i \in N_G} |\delta^-_{G(x)}(i)|) = O(nm)$. □

Proposição 18.15. *O algoritmo de melhoria de aproximação pode ser implementado de modo a executar em tempo* $O(n^2m)$.

Prova: Cada operação push e cada operação relabel pode ser executada em tempo constante. Ademais, pelos Lemas 18.7 e 18.9 e pelo Corolário 18.6, sabemos que ocorrem $O(n^2m)$ operações no total. Além disso, conforme estabelecido pelo Lema 18.14, o tempo total gasto para decidir se uma operação push ou relabel deve ser executada no início de cada iteração é $O(nm)$.

Para concluir o resultado, notamos que é possível acessar um vértice ativo em tempo constante no início de determinada iteração, se a implementação do algoritmo utilizar, por exemplo, uma estrutura de dados FIFO - fila simples - para armazenar os vértices ativos. □

Proposição 18.16. *O algoritmo cost scaling pode ser implementado de modo a executar em tempo* $O(n^2m\log(nC))$.

Prova: Conforme já estabelecido, o algoritmo cost scaling executa $O(\log(nC))$ iterações sendo que, em cada iteração, a única operação que não consome tempo constante é a melhoria de aproximação. Ademais, pela Proposição 18.15, o procedimento de melhoria de aproximação pode ser implementado de forma a consumir tempo $O(n^2m)$. Assim, é possível implementar o algoritmo cost scaling de modo que ele execute em tempo $O(n^2m\log(nC))$. □

Referências Bibliográficas

[1] R.K. Ahuja, T.L. Magnanti, and J. Orlin, *Network flows: Theory, algorithms, and applications*, Practice Hall, 1993.

[2] R.K. Ahuja and J.B. Orlin, *A fast and simple algorithm for the maximum flow problem*, Operations Research (1989), no. 37, 748–759.

[3] ———, *Distance-directed augmenting path algorithms for maximum-flow and parametric maximum flow problems*, Naval Research Logistics Quarterly (1991), no. 38, 413–430.

[4] R. Bellman, *On a routing problem*, Quarterly of Applied Mathematics (1958), no. 16, 87–90.

[5] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver, *Combinatorial otimization*, Wiley-Interscience series in discrete mathematics and optimization, John Wiley & Son's, New York, 1998.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd. ed., The MIT Press and McGraw-Hill, 2001.

[7] G.B. Dantzig, *Application of the simplex method to a transportation problem*, Activity analysis of production and allocation (T.C. Koopmans, ed.), J. Wiley, New York, 1951, pp. 359–373.

[8] E.W. Dijkstra, *A note on two problems in connexion with graphs*, Numerische Mathematik **1** (1959), 269–271.

[9] E.A. Dinits, *Algorithm for solution of a problem of maximum flow in a network with power estimation*, Sov. Math. Dokl. **11** (1970), no. 5, 1277–1280.

[10] J. Edmonds and R.M. Karp, *Theoretical improvements in algorithmic efficiency for network flow problems*, J. ACM **19** (1972), no. 2, 248–264.

[11] P. Feofiloff, *Notas de aula de MAC 5781 otimização combinatória*, `http://www.ime.usp.br/~pf/`, 2002.

[12] L.R. Ford and D.R. Fulkerson, *Maximal flow through a network*, CJM **8** (1956), 399–404.

[13] ———, *Flows in networks*, Princeton University Press, Princeton, NJ, 1962.

[14] A.V. Goldberg and R.E. Tarjan, *A new approach to the maximum flow problem*, Journal of ACM (1988), no. 35, 921–940.

[15] ———, *Finding minimum-cost circulations by cancelling negative cycles*, Journal of ACM (1989), no. 36, 873–886.

[16] ———, *Solving minimum cost flow problem by successive approximation*, Mathematics of Operations Research (1990), no. 15, 430–466.

[17] J.C. Pina Jr., *Notas de aula de MAC 328 algoritmos em grafos*, `http://www.ime.usp.br/~coelho/grafos/`, 1999.

[18] J.B. Orlin, *A faster strongly polynomial minimum cost flow algorithm*, Operations Research (1988).

[19] A. Schrijver, *Combinatorial optimization: Polyhedra and efficiency*, Algorithms and Combinatorics, vol. 24, Springer, 2003.

[20] Y. Shiloach and U. Vishkin, *An* $o(n^2 \log n)$ *parallel max-flow algorithm*, Journal of Algorithms (1982), no. 3, 128–146.