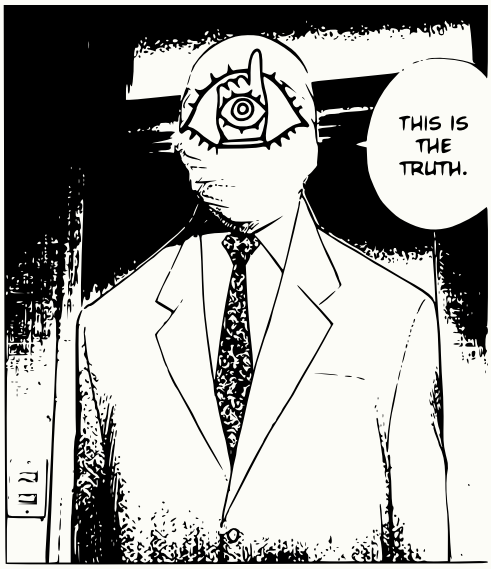
Marco Casu

Ottimizzazione

Sapienza
Università di Roma

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Dipartimento di Informatica

Questo documento è distribuito sotto la licenza GNU, è un resoconto degli appunti (eventualmente integrati con libri di testo) tratti dalle lezioni del corso di Ottimizzazione per la laurea triennale in Informatica. Se dovessi notare errori, ti prego di segnalarmeli.



INDICE

CAPITOLO

1

FLUSSI NEI GRAFI

1.1 Definizione e Grafo Residuo

Definizione 1 *Una **network** o **rete** $G = (V, E, c, s, t)$ è un particolare grafo diretto, in cui V ed E sono i vertici e gli archi, tali per cui è soddisfatta la condizione*

$$\forall (u, v) \in E(G), \quad \exists (v, u) \in E(G)$$

*$c : E(G) \to \mathbb{R}^+$ è una funzione detta **capacità**, s e t sono due particolari vertici in $V(G)$ denominati **source** e **sink**.*

Definizione 2 *Data una network $G = (V, E, c, s, t)$, un **flusso** per G è una funzione $f : E(G) \to \mathbb{R}$ tale per cui valgono le seguenti*

1. ***skew-simmetria:*** $f(u, v) = -f(v, u), \quad \forall (u, v) \in E(G)$
2. ***capacità rispettata:*** $f(u, v) \leq c(u, v), \quad \forall (u, v) \in E(G)$
3. ***conservatività del flusso:*** $\sum_{(u,v) \in E(G)} f(u, v) = 0, \quad \forall v \in V(G) \backslash \{s, t\}$

Denominiamo flusso uscente dal vertice v la somma del flusso (positivo) valutato su tutti gli archi che hanno v come primo membro (che collegano v ad un'altro vertice). Analogamente (ma in maniera opposta) si definisce il flusso entrante. Dato un flusso f per una network G si definisce il **valore del flusso** la somma del flusso uscente da s

$$\text{val}(f) = \sum_{(s,u) \in E(G)} f(s.u)$$

La terza proprietà, di conservazione del flusso, asserisce che il flusso uscente da un nodo deve essere identico al flusso entrante, sia x un vertice fissato in $V(G)$

$$\sum_{\substack{(u,x) \in E(G) \\ f(u,x) > 0}} f(u, x) = -\left(\sum_{\substack{(x,u) \in E(G) \\ f(x,u) < 0}} f(x, u) \right)$$

Definizione 3 *Sia $G = (V, E, c, s, t)$ una network e f un flusso per G, il **grafo residuo** è il grafo diretto G' definito come segue*

- $\forall v \in V(G), \quad v \in V(G')$
- $(u,v) \in E(G) \wedge f(u,v) < c(u,v) \implies (u,v) \in E(G')$

Inoltre è definita una funzione $r : E(G') \to \mathbb{R}^+$ *detta* ***capacità residua*** *definita come segue*

$$r(u,v) = c(u,v) - f(u,v)$$

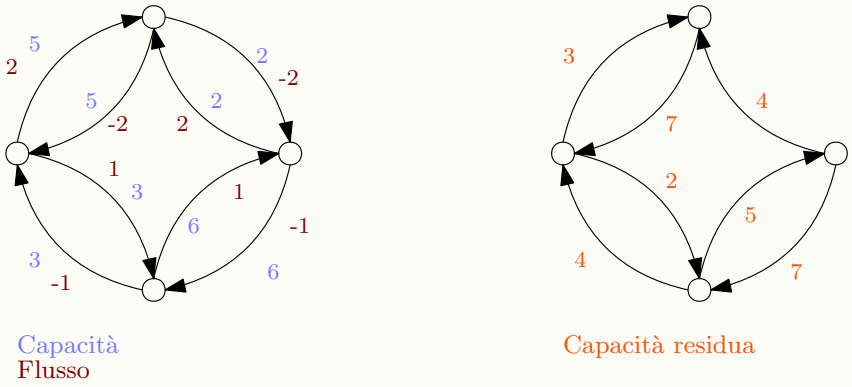


Figura 1.1: Capacità residua del flusso (evidenziato in rosso)

Si assuma che esiste un cammino P in G' da s a t, si consideri il residuo minimo valutato sugli archi contenuti nel cammino

$$\alpha = \min_{(u,v) \in E(P)} r(u,v)$$

Si definisce una funzione $f' : E(G) \to \mathbb{R}$ come segue

$$f'(u,v) = \begin{cases} f(u,v) + \alpha & \text{se} \quad (u,v) \in E(P) \\ f(u,v) - \alpha & \text{se} \quad (v,u) \in E(P) \\ f(u,v) & \text{altrimenti} \end{cases}$$

Proposizione 1 *f' è un flusso per G.*

Dimostrazione : Sia (u,v) un arco in G, se $(u,v) \notin E(P)$, allora $f'(u,v) = f(u,v)$ e conseguentemente $f'(v,u) = f'(v,u)$, quindi la proprietà di skew simmetria è preservata. Differentemente, se $(u,v) \in E(P)$ si avrebbe che $f'(u,v) = f(u,v) + \alpha$ e $f'(v,u) = f(v,u) - \alpha = -f(u,v) - \alpha = -(f(u,v) + \alpha)$, quindi il nuovo flusso rispetta la proprietà di skew-simmetria.

Per ogni arco $(u,v) \in E(P)$ si ha che $f'(u,v) = f(u,v) + \alpha$, α è (per definizione) minore o uguale a $r(u,v)$ quindi

$$f'(u,v) \leq f(u,v) + r(u,v)$$

Ma essendo che $f(u,v) + r(u,v) = c(u,v)$, f' rispetta la capacità.

Se $x \notin V(P)$ si avrebbe che $f'(x,u) = f(x,u)$ per ogni u adiacente ad x, allora

$$\sum_{(x,u) \in E(G)} f(x,u) = 0$$

Assumendo che $x \in V(P)$, vi è un arco uscente da x il cui flusso è aumentato di α, vi è quindi (per definizione di f') un'arco entrante in x il cui flusso è diminuito di α, quindi è ancora vero che

$$\sum_{\substack{(u,x) \in E(G) \\ f'(u,x) > 0}} f'(u,x) = -\left(\sum_{\substack{(x,u) \in E(G) \\ f'(x,u) < 0}} f'(x,u) \right)$$

la proprietà di conservazione del flusso è rispettata. ■

Il valore del nuovo flusso è uguale al valore del flusso di partenza aumentato di α

$$\text{val}(f') = \text{val}(f) + \alpha$$

Algorithm 1 Ford–Fulkerson

Require: network $G = (V, E, c, s, t)$
  si definisce un flusso f tale che $f(u, v) = 0$, $\forall (u, v) \in E(G)$
  si definisce il grafo residuo G' dato il flusso f
  while Esiste un cammino P in G' da s a t **do**
    si definisce la funzione delle capacità residue $r : E(G') \to \mathbb{R}$
    $\alpha = \min_{(u,v) \in E(P)} r(u, v)$
    Si definisce un flusso $f' = f$
    for $(u, v) \in E(P)$ **do**
      $f'(u, v) = f(u, v) + \alpha$
      $f'(v, u) = f(v, u) - \alpha$
    end for
  end while

Dato che un singolo arco (s, u) per qualche u è necessariamente presente nel cammino P da s a t, ed il valore di f' su (s, u) è stato aumentato di α. La proposizione 1 delinea una procedura per la ricerca di un flusso ottimale (di valore massimo) per una network. Alla fine dell'esecuzione, il flusso f' sarà ottimale per la network data.

Osservazione 1 *Se le capacità della network sono numeri interi, l'algoritmo termina. Se invece le capacità sono numeri reali, l'algoritmo potrebbe non terminare.*

1.2 Tagli $s - t$

Data una network $G = (V, E, c, s, t)$, ed un flusso f per G, si consideri un'insieme $\mathcal{U} \subset V(G)$ tale che

- $s \in \mathcal{U}$
- $t \notin \mathcal{U}$

Tale insieme è detto **insieme di taglio**, si consideri ora il flusso uscente dai vertici presenti in $\mathcal{U}$

$$\sum_{\substack{(u,x) \in E(G) \\ \text{t.c. } u \in \mathcal{U}}} f(u, x)$$

Per la proprietà di conservazione del flusso si ha che il flusso uscente da ogni vertice diverso da s è nullo, ed il flusso uscente dal vertice s è il valore del flusso.

$$\sum_{\substack{(u,x) \in E(G) \\ \text{t.c. } u \in \mathcal{U}}} f(u, x) = \sum_{(s,x) \in E(G)} f(s, x) = \text{val}(f)$$

La sommatoria a sinistra può essere riscritta come la somma del flusso uscente dai vertici in $\mathcal{U}$ verso i vertici in $\mathcal{U}$, e del flusso uscente dai vertici in $\mathcal{U}$ verso i vertici che non sono contenuti in $\mathcal{U}$

$$\sum_{\substack{(u,x) \in E(G) \\ \text{t.c. } u \in \mathcal{U}}} f(u, x) = \sum_{\substack{(u,x) \in E(G) \\ \text{t.c. } u, x \in \mathcal{U}}} f(u, x) + \sum_{\substack{(u,x) \in E(G) \\ \text{t.c. } u \in \mathcal{U} \\ x \notin \mathcal{U}}} f(u, x)$$

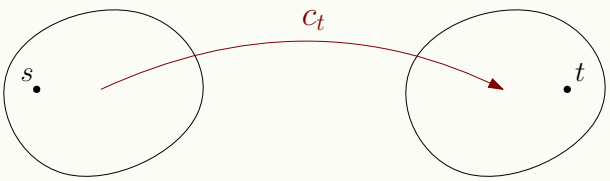
Per la proprietà di skew-simmetria il flusso uscente dai vertici in $\mathcal{U}$ verso i vertici in $\mathcal{U}$ è nullo

$$\sum_{\substack{(u,x) \in E(G) \\ \text{t.c. } u \in \mathcal{U}}} f(u, x) = \sum_{\substack{(u,x) \in E(G) \\ \text{t.c. } u \in \mathcal{U} \\ x \notin \mathcal{U}}} f(u, x)$$

Conclusione : Il valore di f è uguale alla somma dei flussi uscenti dai vertici in $\mathcal{U}$ verso i vertici non contenuti in $\mathcal{U}$. Questa proprietà è invariante rispetto la scelta di $\mathcal{U}$, a patto che rispetti le proprietà inizialmente elencate (deve contenere s ma non t).

Definizione 4 *si definisce* ***capacità di taglio*** *la somma delle capacità degli archi che collegano i vertici in* $\mathcal{U}$ *ai vertici in* $V(G)\backslash\mathcal{U}$

$$c_t = \sum_{\substack{(u,x) \in E(G), \\ u \in \mathcal{U}, \\ x \notin \mathcal{U}}} c(u,x)$$



Osservazione 2 *il valore massimale del flusso è limitato dalla capacità di taglio*

$$val(f) \leq c_t$$

Proposizione 2 *Data una network* G*, se esiste un flusso* f^* *ed un'insieme di taglio* $\mathcal{U}$ *tali che*

$$val(f) = \sum_{\substack{(u,x) \in E(G), \\ u \in \mathcal{U}, \\ x \notin \mathcal{U}}} c(u,x)$$

ossia, il valore del flusso è identico alla capacità di taglio, allora f^* *è un flusso ottimale.*

L'algoritmo di Ford-Fulkerson restituisce un flusso ottimale f^*, da questo è possibile individuare l'insieme di taglio $\mathcal{U}$ associato, in particolare, se G^* è il grafo residuo della network rispetto il flusso dato in output f^*, allora l'insieme di taglio sarà composto da tutti i nodi raggiungibili da s in G^*, chiaramente, fra questi non vi sarà t, data la definizione dell'algoritmo, che termina proprio quando non vi è un cammino da s a t.

Si consideri adesso una network G, di cui f^* è il flusso ottimale trovato tramite l'algoritmo 1. Sia $\mathcal{U}$ l'insieme di taglio dato dai nodi raggiungibili da s nel grafo residuo G^*.

Osservazione 3 *Per ogni arco* $(x,y) \in E(G)$ *con* $x \in \mathcal{U}$ *e* $y \notin \mathcal{U}$*, si avrà che*

$$f^*(x,y) = c(x,y)$$

Il valore del flusso è uguale alla somma delle capacità degli archi che collegano i vertici in $\mathcal{U}$ a quelli fuori da $\mathcal{U}$

$$\sum_{\substack{(x,y) \in E(G) \\ x \in \mathcal{U} \\ y \notin \mathcal{U}}} f^*(x,y) = \sum_{\substack{(x,y) \in E(G) \\ x \in \mathcal{U} \\ y \notin \mathcal{U}}} c(x,y) = \text{val}(f^*)$$

La proposizione 2 non implica che non ci possa essere una network il cui flusso ottimale a valore strettamente minore della capacità di taglio di uno specifico insieme $\mathcal{U}$, si consideri l'immagine in figura 1.2, in cui è applicata la notazione sugli archi *capacità/flusso*, la capacità di taglio è data dalla somma delle capacità sugli archi evidenziati, ed è uguale a 4, nonostante questo, il flusso ottimale per la network in questione ha valore 1.

Nonostante ciò, esiste sempre un insieme $\mathcal{U}$ contenente s e non t la cui capacità di taglio è uguale al valore del flusso ottimale per la network data, tale insieme può essere trovato adoperando l'algoritmo di Ford-Fulkerson nella procedura precedentemente elencata.

Osservazione 4 *L'algoritmo di Ford-Fulkerson, termina sempre se le capacità della network sono numeri in* $\mathbb{Q}$*.*

Dimostrazione : Se le capacità c_i sono numeri razionali allora esiste esiste un numero naturale $N \in \mathbb{N}$ tale che ogni capacità è della forma $c_i = \frac{a_i}{N}$, ad ogni iterazione dell'algoritmo il valore del flusso aumenta di almeno $\frac{1}{N}$, quindi in un numero finito di passi raggiungerà il valore ottimale. ■

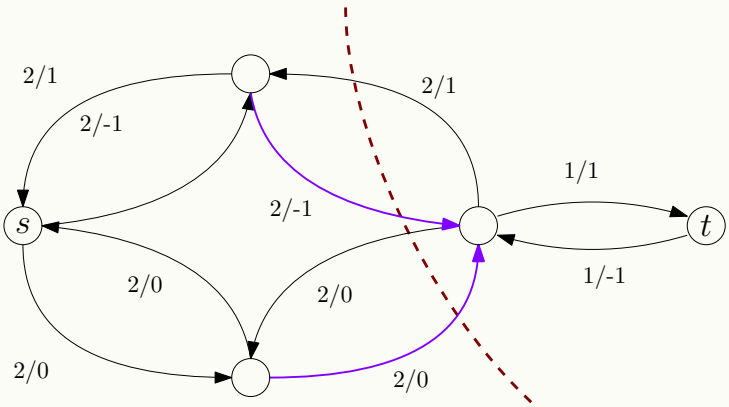


Figura 1.2: network con taglio sui vertici

1.3 Percorso Minimo nell'Aumento del Flusso

Durante la computazione dell'algoritmo di Ford-Fulkerson, viene scelto un qualsiasi percorso che connetta s a t nel grafo residuo, tale scelta comporta un aumento del valore del flusso, ma una scelta differente di percorso potrebbe far si che l'aumento in quella iterazione sia maggiore, e che il numero finale di iterazioni per trovare il flusso ottimale sia minore. Il seguente esempio mostra l'inefficienza dell'algoritmo 1, si consideri la network in figura 1.3 (alcuni archi sono stati omessi). Il flow massimale ha valore $2M$,

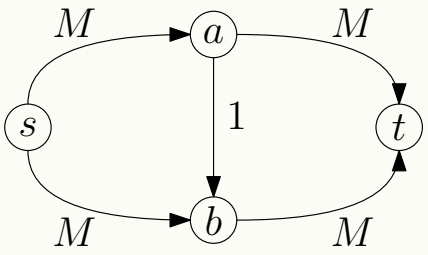


Figura 1.3: Sugli archi sono indicate le capacità

nonostante ciò, se ad ogni iterazione dell'algoritmo venisse selezionato il percorso $s \to a \to b \to t$, allora l'aumento del valore sarebbe uguale ad uno, e sarebbero necessarie $2M$ iterazioni, differentemente, la scelta del percorso $s \to a \to t$ implicherebbe già solo alla prima iterazione un'aumento pari ad M.

La complessità computazionale in questo caso dipende linearmente da M, tale valore è però codificato in binario (occupando $\log M$ spazio), quindi l'algoritmo di Ford-Fulkerson è esponenziale nelle dimensioni dell'input. È possibile considerare una rivisitazione dell'algoritmo 1, in cui ad ogni iterazione viene selezionato il percorso più breve (minor numero di archi) da s a t nel grafo residuo. Tale algoritmo rivisitato è noto con il nome di **Edmonds-Karp**.

Algorithm 2 Edmonds-Karp

Require: network $G = (V, E, c, s, t)$
  si definisce un flusso f tale che $f(u, v) = 0, \forall (u, v) \in E(G)$
  si definisce il grafo residuo G' dato il flusso f
  while Esiste un cammino P in G' da s a t **do**
    $P =$ cammino più breve da s a t in G'
    si definisce la funzione delle capacità residue $r : E(G') \to \mathbb{R}$
    $\alpha = \min_{(u,v) \in E(P)} r(u, v)$
    Si definisce un flusso $f' = f$
    for $(u, v) \in E(P)$ **do**
      $f'(u, v) = f(u, v) + \alpha$
      $f'(v, u) = f(v, u) - \alpha$
    end for
  end while

Osservazione 5 *Se G è un grafo diretto e P è il percorso più breve fra due vertici x ed y, allora $\forall z \in V(P)$, si ha che il sotto cammino $x \to z$ in P è anch'esso un percorso più breve.*

Proposizione 3 *Sia $G = (V, E, c, s, t)$ una network. Sia G_i il grafo residuo all'i-esima iterazione dell'algoritmo 2, e $G_{i'}$ il grafo residuo all'i'-esima iterazione, con $i' > i$, allora*

$$dist_{G_i}(s, u) \leq dist_{G_{i'}}(s, u) \tag{1.1}$$

La distanza dal vertice source s rispetto ogni altro vertice aumenta in maniera monotona ad ogni passo dell'algoritmo.

Dimostrazione : Supponiamo che esiste un nodo $v \in G$ tale che $\text{dist}_{G_i}(s, v) > \text{dist}_{G_{i'}}(s, v)$, si assume inoltre che la distanza $\text{dist}_{G_{i'}}(s, v)$ sia la più piccola possibile (v è il nodo più vicino ad s in $G_{i'}$). Sia w il penultimo vertice del cammino $P' = u_1, u_2 \dots u_k$ in $G_{i'}$, con $u_1 = s$ e $u_k = v$. Ne segue che

$$\text{dist}_{G_i}(s, v) > \text{dist}_{G_{i'}}(s, v) = \text{dist}_{G_{i'}}(s, w) + 1 \geq \text{dist}_{G_i}(s, w) + 1 \tag{1.2}$$

Nota : nella dimostrazione si sta assumendo che la proposizione non sia valida per il nodo v, ma che sia valida per il nodo w, da qui è verificata la disuguaglianza a destra nell'equazione 1.2.

Ciò implica che l'arco (w, v) è presente in $G_{i'}$ ma non in G_i, se così non fosse sarebbe vero che $\text{dist}_{G_{i'}}(s, w) \geq \text{dist}_{G_i}(s, w) + 1$, e quindi $w = u_i$ e $v = u_{i-1}$ per qualche i, ma questa è una contraddizione dato che v segue w nel cammino P', quindi l'asserto è verificato. ■

TODO : continuare e dimostrare che l'algoritmo esegue al più nm iterazioni