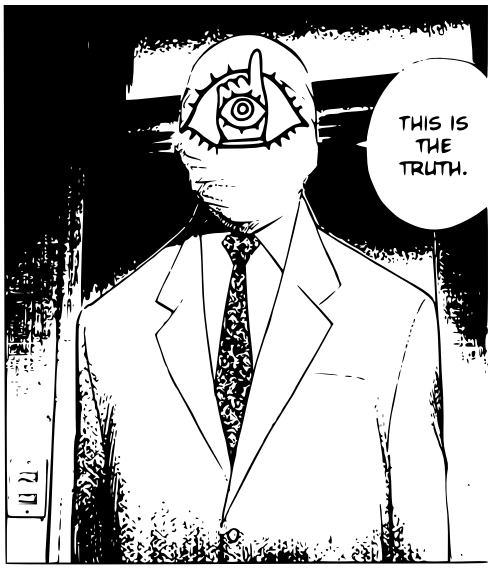
Marco Casu

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:*** $\displaystyle\sum_{(u,v) \in E(G)} f(u, v) = 0, \quad \forall v \in V(G) \backslash \{s, t\}$

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*

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

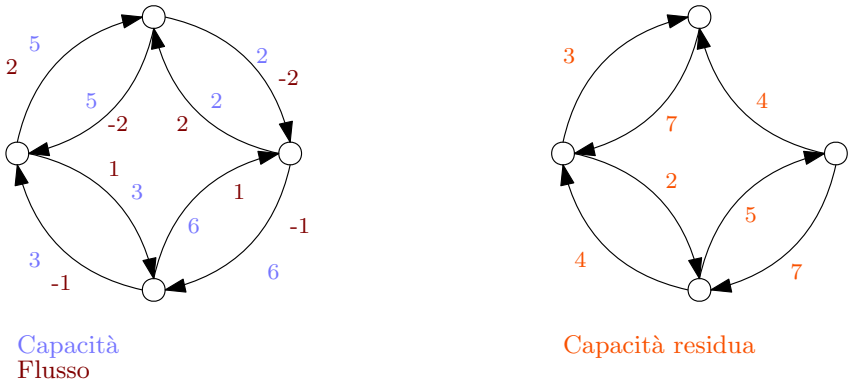


Figura 1.1: Capacità residua del flusso (evidenziato in rosso)

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 $\mathcal{U}$

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

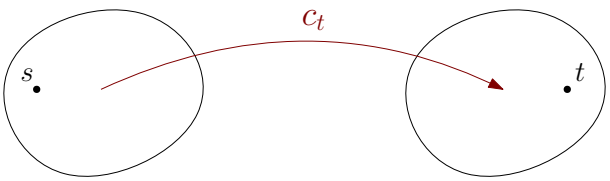
Per la proprietà di skew-simmetria il flusso uscente dai vertici in $\mathcal{U}$ verso i vertici in $\mathcal{U}$ è nullo

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

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. ■

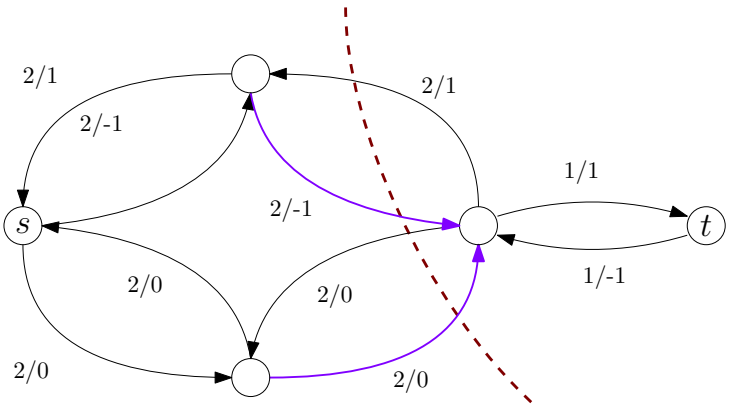


Figura 1.2: network con taglio sui vertici

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 flusso massimale ha valore $2M$,

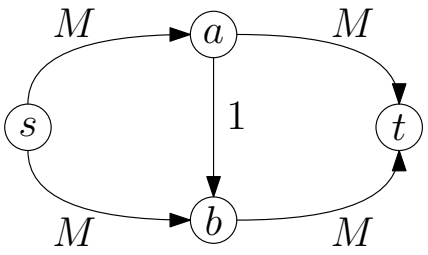


Figura 1.3: Sugli archi sono indicate le capacità

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

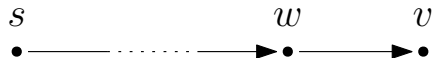
Osservazione 5 *Se G è un grafo diretto e P è il percorso più breve fra due vertici x ed y, allora $\forall z \in V(P)$, si ha che il sotto cammino $x \to z$ in P è anch'esso un percorso più breve.*

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. ■

Teorema 1 *Nell'algoritmo di Edmonds-Karp, applicato su una network G, il numero totale di incrementi del valore del flusso è al più $n \cdot m$, con $n = |V(G)|$ e $m = |E(G)|$. Tale affermazione è valida anche se le capacità sugli archi sono numeri reali.*

Dimostrazione : Sia G_i il grafo residuo nell'i-esima iterazione dell'algoritmo 2, analogamente, sia f_i il flusso valutato anch'esso durante l'i-esima iterazione. Chiaramente $G_0 = G$ e $f_0(e) = 0,\ \forall e$.

Definizione 5 *Durante l'esecuzione dell'algoritmo 2, un'arco (u, v) è detto **critico** in i se*

- $(u, v) \in G_i$
- $(u, v) \notin G_{i+1}$

Se P_i è il percorso minimo da s a t considerato nell'i-esima iterazione, e (u, v) è critico in i, per definizione dell'algoritmo si ha che $(u, v) \in E(P_i)$.

> **Lemma 1** *Sia (u, v) un'arco di una network G, durante l'esecuzione dell'algoritmo 2, l'arco (u, v) può essere considerato critico al più $\frac{n}{2}$ volte.*
>
> *Dimostrazione Lemma* : Siano
>
> $$\pi(1) < \pi(2) < \cdots < \pi(L)$$
>
> gli indici delle iterazioni in cui (u, v) è critico, con $L \leq \frac{n}{2}$, chiaramente
>
> $$(u, v) \in E(P_{\pi(i)})$$
>
> per qualche $1 \leq i \leq L$. Chiaramente
>
> $$\text{dist}_{G_{\pi(i)}}(s, v) = \text{dist}_{G_{\pi(i)}}(s, u) + 1$$
>
> Se (u, v) è critico in $\pi(i)$ ed in $\pi(i+1)$, allora deve esistere un iterazione i' compresa fra queste
>
> $$\pi(i) < i' < \pi(i + 1)$$
>
> In cui l'arco (u, v) è stato re-inserito nel grafo residuo, quindi il flusso su (u, v) in tale iterazione è diminuito, necessariamente (per skew-simmetria) il flusso su (v, u) è aumentato, quindi quest'ultimo arco si trovava sul percorso da s a t nell'iterazione i'.
>
> $$(v, u) \in E(P_{i'})$$

Date le precedenti osservazioni, si deducono le seguenti disuguaglianze

$$\text{dist}_{G_{i'}}(s,u) = \text{dist}_{G_{i'}}(s,v) + 1 \tag{1.3}$$
$$\text{dist}_{G_{i'}}(s,v) + 1 \geq \text{dist}_{G_{\pi(i)}}(s,v) + 1 \tag{1.4}$$
$$\text{dist}_{G_{\pi(i)}}(s,v) + 1 = \text{dist}_{G_{\pi(i)}}(s,u) + 2 \tag{1.5}$$
$$\Downarrow \tag{1.6}$$
$$\text{dist}_{G_{\pi(i+1)}}(s,u) \geq \text{dist}_{G_{\pi(i)}}(s,u) + 2 \tag{1.7}$$

Essendo che la distanza fra due vertici è limitata da $n = |V(G)|$, si ha che

$$\text{dist}_{G_{\pi(i)}}(s,u) \leq n - 1$$

- la distanza fra s ed u è al più $n-1$
- la distanza fra s ed u aumenta almeno di due in due iterazioni differenti in cui (u,v) è critico

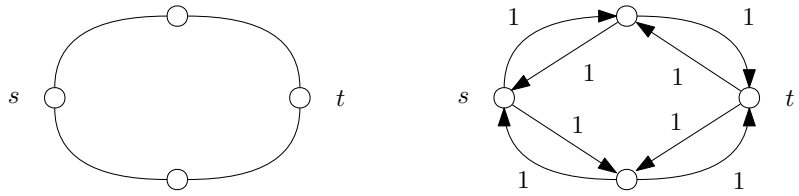
La conclusione è che non possono esistere più di $\frac{n}{2}$ indici $\pi(i)$ in cui (u,v) è critico. □

La dimostrazione del teorema 1 segue in maniera naturale, ad ogni iterazione un'arco è critico, essendo che ci sono m archi ed ognuno può essere critico al più $\frac{n}{2}$ volte, il numero totale di aumenti del flusso è al più $\frac{1}{2}nm$. ■

1.4 Cammini Edge-Disjoint in un Grafo

In questa sezione verrà esposta un'applicazione dell'algoritmo di ricerca del flusso massimo. Sia G un grafo non diretto, si definisce la *network associata* a G, il grafo diretto $\vec{G}$ tale che

- $u \in V(G) \implies u \in V(\vec{G})$
- $(u,v) \in E(G) \implies \begin{cases} (u,v) \in E(\vec{G}) \\ (v,u) \in E(\vec{G}) \end{cases}$
- $\forall (u,v) \in \vec{G} \quad , c(u,v) = 1$



Si può anche risalire in maniera naturale ad un grafo non diretto associato ad una network.

Definizione 6 *Dato un grafo non diretto G e due nodi s,t, un'insieme di cammini da s a t è **edge-disjoint** se non condividono alcun arco.*

Verrà mostrato come, dato un grafo G, il numero massimo di cammini edge-disjoint è uguale al valore del flusso ottimale nella network associata.

Definizione 7 *Dato un flusso f per una network $\vec{G}$, si definisce **supporto del flusso** l'insieme*

$$W = \{(u,v) \in E(\vec{G}) \text{ t.c. } f(u,v) \neq 0\}$$

Proposizione 4 *Se nel supporto W di un flusso f per una network $\vec{G}$ gli archi compongono k cammini diretti da s a t, allora ci sono k cammini edge-disjoint da s a t nel grafo non diretto associato.*

Dimostrazione : TODO : continuare

CAPITOLO

2

PROGRAMMAZIONE LINEARE

2.1 Insiemi Convessi

La programmazione lineare consiste nella ricerca di un vettore (ingresso di una funzione lineare) in cui tale funzione assume il valore massimo, all'interno di un dominio definito da un'insieme di vincoli lineari. La funzione da massimizzare è detta **funzione obiettivo**, un'esempio di programma lineare può essere il seguente

$$x_1 + x_2$$

soggetto ai vincoli

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_2 - x_1 &\leq 1 \\ x_1 + 6x_2 &\leq 15 \\ 4x_1 - x_2 &\leq 10 \end{aligned}$$

Un punto è **ammissibile** se soddisfa tutti i vincoli lineari. L'insieme di punti ammissibili si può rappresentare su un piano in questo caso, essendo un sotto-insieme di $\mathbb{R}^2$. La funzione obiettivo essendo lineare

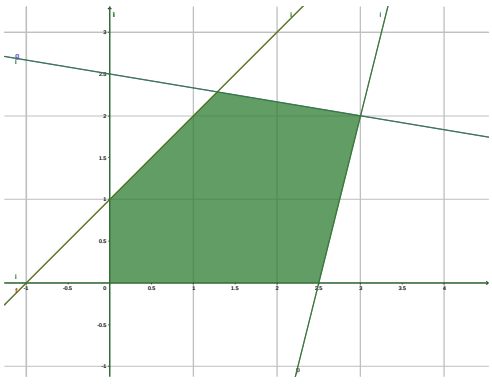


Figura 2.1: Insieme dei punti ammissibili

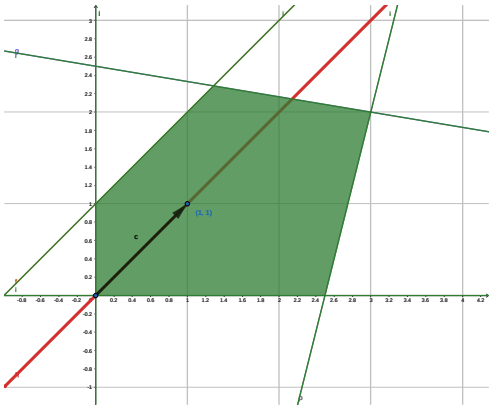
si può rappresentare come prodotto scalare fra due vettori $\mathbf{c}$ e $\mathbf{x}$

$$\mathbf{c}^T\mathbf{x} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 + x_2$$

Può risultare utile rappresentare sul piano anche il vettore $\mathbf{c}$ e la retta equivalente al sottospazio

$$\text{span}(\mathbf{c}) = \{\alpha\mathbf{c} \mid \alpha \in \mathbb{R}\}$$

Si consideri una retta y' perpendicolare alla retta definita da span($\mathbf{c}$), i punti di y' che intersecano



l'insieme delle soluzioni ammissibili condividono la stessa immagine se valutati sulla funzione obiettivo. Un'interpretazione geometrica del problema può essere la seguente

> Massimizzare la funzione obiettivo equivale a trovare il massimo β tale che l'iperpiano definito da $\mathbf{c}^T\mathbf{x} = \beta$ perpendicolare alla retta $\alpha\mathbf{c}$ interseca l'insieme dei punti ammissibili.

Si definisce **soluzione ottimale** ogni soluzione ammissibile che massimizza la funzione obiettivo. Il

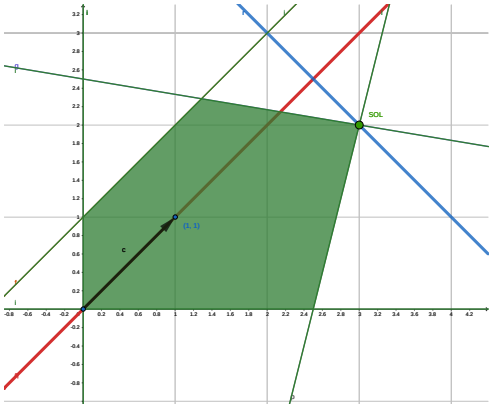


Figura 2.2: Il punto $(3, 2)$ è una soluzione ottimale per il programma lineare

numero di soluzioni di un programma lineare può variare fra i seguenti casi

1. Vi è un'unica soluzione ottimale.
2. Vi sono infinite soluzioni ottimale.
3. Non ci sono soluzioni ammissibili, nessun punto soddisfa tutti i vincoli lineari.
4. Non ci sono soluzioni ottimali perché il problema non è limitato, ciò avviene se l'insieme dei punti ammissibili non è chiuso.

Generalmente la funzione obiettivo $f : \mathbb{R}^n \to \mathbb{R}$ è del tipo

$$c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

e gli m vincoli sono della forma

$$\begin{aligned} a_{11}x_1 + a_{21}x_2 + \cdots + a_{n1}x_2 &\leq b_1 \\ a_{12}x_1 + a_{22}x_2 + \cdots + a_{n2}x_2 &\leq b_2 \\ &\vdots \\ a_{1m}x_1 + a_{2m}x_2 + \cdots + a_{nm}x_2 &\leq b_m \end{aligned}$$

Osservazione 6 *Massimizzare* $\mathbf{c}^T\mathbf{x}$ *equivale a minimizzare* $-\mathbf{c}^T\mathbf{x}$.

Si può quindi assumere che un generico problema di programmazione lineare riguardi la massimizzazione di una funzione del tipo $\mathbf{c}^T\mathbf{x}$ per un fissato $\mathbf{c} \in \mathbb{R}^n$. Inoltre, ogni vincolo del tipo

$$a_{1i}x_1 + a_{2i}x_2 + \cdots + a_{ni}x_n \leq b_i \tag{2.1}$$

è soddisfatto dagli stessi punti che soddisfano

$$-a_{1i}x_1 - a_{2i}x_2 - \cdots - a_{ni}x_2 \geq b_i$$

Quindi si può assumere che ogni vincolo sia scritto nella forma 2.1. Inoltre ogni vincolo di uguaglianza equivale a due disuguaglianze. Date le precedenti osservazioni, si può definire un'insieme di m vincoli in maniera compatta tramite una matrice $m \times n$ ed un vettore $\mathbf{b} \in \mathbb{R}^m$.

$$A\mathbf{x} \leq \mathbf{b}$$

Inoltre ogni vincolo di positività del tipo $x_i \geq 0$ equivale ad il vincolo $-x_i \leq 0$. Se in un programma lineare una variabile x_i può assumere qualsiasi valore in $\mathbb{R}$, si può sostituire con la differenza di due nuove variabili

$$x_i = z_i - z_i'$$

ed imporre i vincoli

$$z_i, z_i' \geq 0$$

in tal modo è possibile, per ogni variabile del programma lineare, imporre la positività. Ciò è utile per definire una *forma standard* per un programma lineare.

Definizione 8 *Un programma lineare in* ***forma standard*** *è un problema di ottimizzazione del tipo*

$$\begin{gathered} \max \ \mathbf{c}^T\mathbf{x} \\ A\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq 0 \end{gathered}$$

Dove

$$\begin{gathered} \mathbf{x}, \mathbf{c} \in \mathbb{R}^n \\ A \in \mathrm{Mat}(m \times n) \\ \mathbf{b} \in \mathbb{R}^m \end{gathered}$$

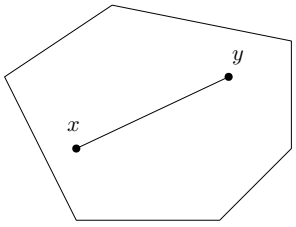
L'esistenza di soluzioni ammissibili dipende esclusivamente dalla matrice A. È impossibile che un programma lineare abbia un numero finito di soluzioni diverso da 0 e 1. In seguito verrà dimostrato che se un programma lineare ha due punti ammissibili che sono soluzione, allora ha infinite soluzioni.

Definizione 9 *Dati due punti* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$*, si definisce* ***segmento di linea*** *fra i punti l'insieme*

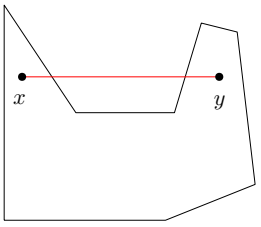
$$\{\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \text{ t.c. } \alpha \in [0,1]\}$$

Definizione 10 *Un sotto-insieme* $X \subseteq \mathbb{R}^n$ *è* ***convesso*** *se la seguente è verificata*

$$\forall \mathbf{x}, \mathbf{y} \in X, \quad \{\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \text{ t.c. } \alpha \in [0,1]\} \subseteq X$$



sotto-insieme di $\mathbb{R}^2$ convesso



sotto-insieme di $\mathbb{R}^2$ non convesso

Proposizione 5 *L'insieme dei punti ammissibili di un programma lineare è convesso.*

Dimostrazione : Siano $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ due punti ammissibili, sia $\alpha \in [0,1]$ fissato, si considera il punto (appartenente al segmento)

$$\alpha\mathbf{x} + (1-\alpha)\mathbf{y}$$

si ha che

$$A(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) = \tag{2.2}$$
$$\alpha A\mathbf{x} + (1-\alpha)A\mathbf{y} = \tag{2.3}$$
$$\alpha\mathbf{b} + (1-\alpha)\mathbf{b} = \mathbf{b} \tag{2.4}$$

quindi Il punto soddisfa gli m vincoli

$$A(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \leq \mathbf{b}$$

Inoltre essendo che

$$\alpha \geq 0, \mathbf{x} \geq 0, \mathbf{y} \geq 0$$

si ha che

$$\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \geq 0$$

Quindi $\alpha\mathbf{x} + (1-\alpha)\mathbf{y}$ soddisfa tutti i vincoli del programma lineare, ed appartiene quindi ai punti ammissibili. ■

Proposizione 6 *Se per un programma lineare esistono due soluzioni ottimali distinte $\mathbf{x}^*, \mathbf{y}^* \in \mathbb{R}^n$, allora tale programma lineare ha infinite soluzioni ottimali.*

Dimostrazione : Sia $\mathbf{z}$ un punto sul segmento delineato da $\mathbf{x}^*, \mathbf{y}^*$

$$\mathbf{z} = \alpha\mathbf{x}^* + (1-\alpha)\mathbf{y}^* \quad \text{per qualche } \alpha \in [0,1]$$

si ha che

$$\mathbf{c}^T\mathbf{z} = \tag{2.5}$$
$$\mathbf{c}^T(\alpha\mathbf{x}^* + (1-\alpha)\mathbf{y}^*) = \tag{2.6}$$
$$\alpha\mathbf{c}^T\mathbf{x}^* + (1-\alpha)\mathbf{c}^T\mathbf{y}^* \tag{2.7}$$

Essendo che $\mathbf{x}^*$ e $\mathbf{y}^*$ sono entrambe soluzioni ottimali, si ha che $c^T\mathbf{x}^* = c^T\mathbf{y}^*$

$$\alpha\mathbf{c}^T\mathbf{x}^* + (1-\alpha)\mathbf{c}^T\mathbf{y}^* = \tag{2.8}$$
$$\alpha\mathbf{c}^T\mathbf{x}^* + (1-\alpha)\mathbf{c}^T\mathbf{x}^* = c^T\mathbf{x}^* \tag{2.9}$$

Quindi anche $\mathbf{z}$ è soluzione, essendo che quest'ultimo è un generico punto sul segmento, tutti i punti del segmento sono soluzioni. ■

2.2 Applicazioni della Programmazione Lineare

Un classico esempio di applicazione riguarda la scelta di una dieta che soddisfi dei vincoli nutrizionali minimizzando il costo degli alimenti. Vi è un'insieme di n alimenti $x_1 \ldots, x_n$ ed ognuno di questi ha un costo c_i, si vuole minimizzare

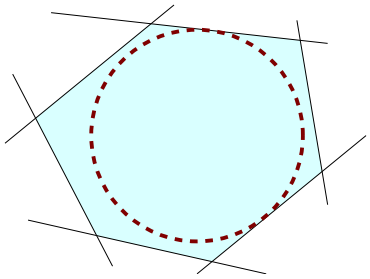
$$\sum_{i=1}^{n} c_i x_i$$

tenendo conto dei vincoli nutrizionali del tipo

$$a_{1i}x_1 + a_{2i}x_2 + \cdots + a_{ni}x_n \geq b_i$$

Un'altro esempio riguarda la ricerca di un flusso ottimale per una network $G = (V, E, c, s, t)$, per ogni arco (i,j) vi sarà una variabile x_{ij} che rappresenta il flusso su tale arco, si vuole massimizzare la somma del flusso uscente dal vertice source s

$$\sum_{(s,j)\in E(G)} x_{sj}$$



I vincoli riguardano la skew simmetria, la conservazione del flusso e il vincolo delle capacità, ad esempio:

$$x_{ij} \leq c_{ij}$$
$$x_{ij} = -x_{ji}$$

Un'esempio interessante di problema che si riconduce alla programmazione lineare riguarda la ricerca del cerchio di raggio massimo che può essere inscritto in un poligono. Si assume che i lati del poligono possono essere descritti da un'insieme di rette del tipo $y = mx + b$ con $m \neq 0$.

Si consideri una delle rette $y = mx + b$, si vuole trovare una retta perpendicolare a questa passante per un fissato punto (x_0, y_0). Questa ha equazione

$$y = \frac{y_0 - x}{m} + y_0$$

Il punto di intersezione è $P = (x', y')$ dove

$$x' = \frac{x_0 + my_0}{m^2 + 1} - mb \tag{2.10}$$
$$y' = m\frac{x_0 + my_0 - mb}{m^2 + 1} + b \tag{2.11}$$

La distanza fra P e (x_0, y_0), applicando la norma euclidea, risulta essere

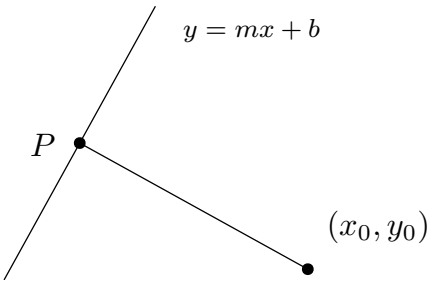


Figura 2.3: retta perpendicolare passante per (x_0, y_0)

- $\frac{b + mx_0 - y_0}{\sqrt{1 + m^2}}$ se (x_0, y_0) si trova sotto la retta
- $\frac{-b - mx_0 + y_0}{\sqrt{1 + m^2}}$ se (x_0, y_0) si trova sopra la retta

Il programma lineare riguarda la ricerca di un cerchio che massimizzi il raggio r (distanza fra (x_0, y_0) e P) variando fra il possibile centro (x_0, y_0) all'interno del poligono, rispettando i vincoli del tipo

- $\frac{b_l + mx_0 - y_0}{\sqrt{1+m^2}} \geq r, \quad \forall l \in 1, \ldots k$
- $\frac{-b_l - mx_0 + y_0}{\sqrt{1+m^2}} \geq r \quad \forall l \in k+1, \ldots n$

2.3 Il Metodo del Simplesso

Il metodo del simplesso è un'algoritmo per la risoluzione dei programmi lineari. Questi devono però assumere una forma differente da quella standard introdotta nella definizione 8. Un programma lineare è definito come segue

$$\begin{gathered} \max \ \mathbf{c}^T\mathbf{x} \\ A\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq 0 \end{gathered}$$

Per ogni generica disuguaglianza del tipo

$$a_{1i}x_1 + a_{2i}x_2 + \cdots + a_{ni}x_n \leq b_i$$

Si considera una nuova variabile $y_i \geq 0$ detta **slack**, e la disuguaglianza diventa un'uguaglianza come segue

$$a_{1i}x_1 + a_{2i}x_2 + \cdots + a_{ni}x_n + y_i = b_i$$

Vi sarà quindi un vettore $\mathbf{y} \in \mathbb{R}^m$ che viene introdotto nel problema, che assume la seguente forma

$$\begin{gathered} \max \ [\mathbf{c}^T \mid 0 \ \ldots \ 0] \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \\ [A \mid \mathrm{Id}_m] \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \mathbf{b} \\ \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \geq 0 \end{gathered}$$

Dove Id_m è la matrice identità $m \times m$. Un problema LP di questo tipo è detto in **forma di equazione**, ed il metodo del simplesso opera su un programma lineare di tale forma. In generale, si scrive

$$\begin{gathered} \max \ \mathbf{c}^T\mathbf{x} \\ A\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq 0 \end{gathered}$$

Assunzione : Il programma lineare considerato ha almeno una soluzione ammissibile.