Algoritmi e strutture di dati

Corso di Laurea in Informatica

Dispense

a.a. 2012-2013

Giorgio Gambosi

UNIVERSITÀ DI ROMA ``TOR VERGATA''

Revisione per l'a.a. 2013/14 a cura di Miriam Di Ianni (aggiornate al 06/05/2014)

Indice

Capitolo 1

Problemi di flusso su reti

Definizioni

Una rete N è una quadrupla $\langle G, s, t, c\rangle$, dove

- $G = (V, E)$ è un grafo orientato simmetrico, ossia, per ogni coppia di nodi u e v, se $(u, v) \in E$ allora $(v, u) \in E$,
- $s \in V$ è un nodo di V, detto sorgente (source),
- $t \in V$ è un nodo di V, detto destinazione (target),
- $c : E \mapsto \mathbb{R}^+$ associa ad ogni arco $e = (u, v) \in E$ una capacità $c(e)$, o $c(u, v)$

Un flusso su una rete N è una funzione $f : E \mapsto \mathbb{R}^+$. In flusso è ammissibile se sono verificate le condizioni seguenti:

1. Limite per la capacità: il flusso su un arco è al più pari alla sua capacità

   $$\forall e \in E, 0 \leq f(e) \leq c(e)$$

   (conseguenza di questa condizione è che un arco avente capacità nulla non può trasportare flusso)

2. Conservazione del flusso: ad eccezione di s e t, il flusso entrante in un nodo è pari a quello uscente

   $$\forall v \in V - \{s, t\}, \sum_{(v,x)\in E} f(v, x) = \sum_{(x,v)\in E} f(x, v)$$

Il valore di un flusso f è definito come il flusso uscente dalla sorgente

$$v(f) = \sum_{(s,x)\in E} f(s, x) - \sum_{(s,x)\in E} f(x, s)$$

Obiettivo di questa dispensa è il progetto di algoritmi che calcolino il flusso di valore massimo in una rete. Prima di procedere, osserviamo che ogni rete ammette un flusso f_0 di valore $v(f_0) = 0$; infatti, è sufficiente porre $f_0(u, v) = 0$ per ogni $(u, v) \in E$ e verificare che è un flusso ammissibile. Pertanto, essendo interessati ad individuare flussi di valore massimo, d'ora in avanti considereremo soltano flussi aventi valore non negativo.

Sia f un flusso ammissibile per la rete N. Un nodo $u \in V - \{s, t\}$ è attraversato dal flusso se esiste un arco $(v, u) \in E$ tale che $f(v, u) > 0$. In virtù della condizione di conservazione del flusso, ogni nodo attraversato da f è parte di (almeno) un percorso da s a t costituito da archi aventi capacità positiva. In quanto segue, con il termine cammino ci riferiremo sempre ad un percorso costituito da soli archi aventi capacità positiva. Diremo anche che un nodo u è raggiungibile da un nodo v se esiste in N un cammino da v ad u.

Per comodità di notazione, nel seguito, per ogni sottoinsieme $U \subseteq V$ di nodi, indicheremo con $\mathcal{I}(U)$ ed $\mathcal{O}(U)$ gli insiemi di archi entranti in U e uscenti da esso. Più formalmente, $\mathcal{I}(U) = \{(u,v) \in E : u \in V - U, v \in U\}$ e $\mathcal{O}(U) = \{(u,v) \in E : u \in U, v \in V - U\}$. Inoltre, nel caso particolare in cui $U = \{u\}$, scriveremo $\mathcal{I}(u), \mathcal{O}(u)$ per indicare l'insieme degli archi entranti ed uscenti da u.

Osserviamo che, in virtù della condizione di Conservazione del flusso, possiamo trasformare ogni flusso ammissibile $\overline{f}$ tale che $\overline{f}(x,y) \geq \overline{f}(y,x) > 0$, per qualche arco (x,y), in un flusso f che ha valore nullo sul"arco (y,x) e tale che $v(f) = v(\overline{f})$. Infatti, per ogni $(u,v) \in E$, è sufficiente porre

$$f(u,v) = \begin{cases} \overline{f}(x,y) - \overline{f}(y,x) & \text{se } u = x \text{ e } v = y, \\ 0 & \text{se } u = y \text{ e } v = x, \\ \overline{f}(u,v) & \text{altrimenti} \end{cases}$$

e verificare che f è ancora un flusso ammissibile e che $v(f) = v(\overline{f})$.
Analogamente, possiamo mostrare che ogni flusso ammissibile $\overline{f}$ tale che $v(\overline{f}) \geq 0$ e $\overline{f}(x,s) > 0$, per qualche nodo x adiacente alla sorgente, può essere trasformato in un flusso f che ha valore nullo su ogni arco entrante nella sorgente e tale che $v(f) = v(\overline{f})$: infatti, in questo caso, è sufficiente porre

$$f(u,v) = \begin{cases} \overline{f}(u,v) & \text{se } u \neq s \text{ e } v \neq s \\ \overline{f}(s,v) - \overline{f}(v,s) & \text{se } u = s \\ 0 & \text{se } v = s, \end{cases}$$

e verificare che f è ancora un flusso ammissibile e che $v(f) = v(\overline{f})$.
Infine, in modo analogo, possiamo trasformare ogni flusso ammissibile $\overline{f}$ tale che $v(\overline{f}) \geq 0$ e $\overline{f}(t,x) > 0$, per qualche nodo x adiacente alla destinazione, in un flusso f che ha valore nullo su ogni arco uscente dalla destinazione e tale che $v(f) = v(\overline{f})$.
Da ora in avanti, ci limiteremo dunque a considerare flussi f aventi valore non negativo e tali che, per ogni arco $(u,v) \in E$, $f(u,v) = 0$ oppure $f(v,u) = 0$ e che, inoltre, hanno valore nullo sugli archi entranti nella sorgente e sugli archi uscenti dalla destinazione; in tal caso, possiamo allora scrivere

$$v(f) = \sum_{e \in \mathcal{O}(s)} f(e)$$

Dato che tutti i nodi interni (diversi da s,t) conservano il flusso, dovrà necessariamente aversi che il flusso uscente dalla sorgente deve essere pari al flusso entrante nella destinazione.

Teorema 1.1 Per ogni flusso ammissibile f,

$$v(f) = \sum_{e \in \mathcal{O}(s)} f(e) = \sum_{e \in \mathcal{I}(t)} f(e)$$

Dimostrazione: Per il vincolo di conservazione del flusso,

$$\begin{aligned} v(f) &= \sum_{e \in \mathcal{O}(s)} f(e) \\ &= \sum_{e \in \mathcal{O}(s)} f(e) - \sum_{v \in V - \{s,t\}} \left(\sum_{e \in \mathcal{I}(v)} f(e) - \sum_{e \in \mathcal{O}(v)} f(e) \right) \end{aligned}$$

(per ogni nodo diverso da s e t flusso in entrata e in uscita sono uguali)

$$= \sum_{e \in E - \mathcal{I}(t)} (f(e) - f(e)) + \sum_{e \in \mathcal{I}(t)} f(e)$$

(per ogni arco (u,v) non entrante in t il suo flusso compare in uscita per u e in entrata per v)

$$= \sum_{e \in E - \mathcal{I}(t)} f(e)$$

□

Un taglio (cut) tra s e t è un sottoinsieme $A \subset V$ dei nodi della rete tale che $s \in A$ e $t \in V - A$. La capacità $c(A)$ del taglio A è definito come la somma delle capacità di tutti gli archi $e \in \mathcal{O}(A)$ uscenti da A (e quindi entranti in $V - A$, tali cioè che $e \in \mathcal{I}(V - A)$)

$$c(A) = \sum_{e \in \mathcal{O}(A)} c(e)$$

Generalizzando il teorema precedente, possiamo mostrare che un flusso ammissibile deve attraversare qualunque taglio della rete.

Teorema 1.2 Per ogni flusso ammissibile f e per ogni taglio A,

$$v(f) = \sum_{e \in \mathcal{O}(A)} f(e) - \sum_{e \in \mathcal{I}(A)} f(e)$$

Dimostrazione: La dimostrazione è del tutto simile a quella del Teorema 1.1: infatti, ancora per il vincolo di conservazione del flusso,

$$\begin{aligned} v(f) &= \sum_{e \in \mathcal{O}(s)} f(e) \\ &= \sum_{e \in \mathcal{O}(s)} f(e) - \sum_{v \in A - \{s\}} \left(\sum_{e \in \mathcal{I}(v)} f(e) - \sum_{e \in \mathcal{O}(v)} f(e) \right) \\ &= \sum_{e \in A^2} (f(e) - f(e)) + \sum_{e \in \mathcal{O}(A)} f(e) - \sum_{e \in \mathcal{I}(A)} f(e) \end{aligned}$$

(il flusso di un arco tra due nodi (v, v') in A compare due volte: sommato, quando uscente da v e sottratto, quando entrante in v'. Il flusso di un arco uscente da A compare soltanto sommato, quello di un arco entrante in A soltanto sottratto)

$$= \sum_{e \in \mathcal{O}(A)} f(e) - \sum_{e \in \mathcal{I}(A)} f(e)$$

□

Dato che un flusso ammissibile deve attraversare qualunque taglio, e per attraversarlo deve avere valore inferiore alla capacità del taglio stesso, ne deriva che un flusso ammissibile deve avere valore inferiore alla capacità di qualunque taglio nella rete.

Teorema 1.3 Per ogni flusso ammissibile f e per ogni taglio A,

$$v(f) \leq c(A)$$

Dimostrazione: L'enunciato deriva immediatamente osservando che

$$v(f) = \sum_{e \in \mathcal{O}(A)} f(e) - \sum_{e \in \mathcal{I}(A)} f(e) \leq \sum_{e \in \mathcal{O}(A)} f(e) \leq \sum_{e \in \mathcal{O}(A)} c(e) = c(A)$$

□

Quindi, per un qualunque flusso ammissibile f, la capacità di un qualunque taglio $s - t$ fornisce una delimitazione superiore di $v(f)$.

Data una rete N, il problema del massimo flusso (max-flow) chiede di trovare il flusso ammissibile f di valore massimo. Chiaramente, per il Teorema 1.3, abbiamo che

$$\max_{f \text{ ammissibile}} v(f) \leq \min_{A \subset V} c(A) \tag{1.1}$$

Una prima questione è allora se in generale, in questa relazione, vale l'uguaglianza.

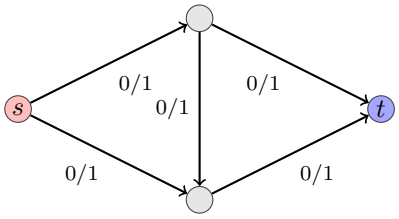


Figura 1.1: Esempio di rete: gli archi aventi capacità nulla non sono mostrati. Il numero alla destra del simbolo / nell'etichetta di un arco rappresenta la capacità di quell'arco; il numero alla sinistra di / rappresenta la quantità di flusso che attraversa quell'arco.

Algoritmo greedy per max-flow

Consideriamo un semplice algoritmo ``goloso" (greedy) per trovare il massimo flusso. L'algoritmo, iterativamente, verifica se esiste un cammino (orientato) da s a t i cui archi sono tutti non saturi (diciamo che una arco e è saturo se $f(e) = c(e)$, e quindi se il flusso su e non può aumentare): in questo caso, diciamo che il cammino è non saturo. Se tale cammino esiste, il flusso lungo esso viene incrementato per quanto possibile, in modo da renderlo saturo, altrimenti l'algoritmo termina.

Algorithm 1.1: Algoritmo greedy per maxflow

Input: $G = (V, E)$, $s, t \in V$, $c : E \mapsto \mathbb{R}^+$
Output: Flusso $f : E \mapsto \mathbb{R}^+$ da s a t

```
1 foreach e ∈ E do f(e) ← 0;
2 while esiste un cammino P da s a t con μ_P = min_{e∈P}(c(e) − f(e)) > 0 do
3     foreach e ∈ P do f(e) ← f(e) + μ_p;
4 return f
```

Possiamo facilmente verificare che l'Algoritmo 1.1 non restituisce necessariamente il flusso massimo. Infatti come vediamo, la sua applicazione alla rete in Figura 1.1 può portare a selezionare, come cammino da saturare, quello evidenziato in Figura 1.2, giungendo ad una situazione in cui non esistono altri cammini non saturi e si ha $v(f) = 1$.

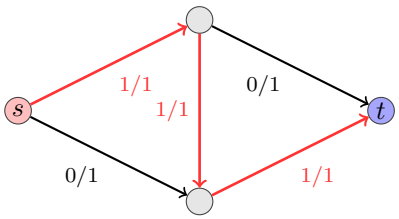


Figura 1.2: Possibile cammino saturato dall'algoritmo: $v(f) = 1$

Come si può osservare in Figura 1.3, esiste però una soluzione migliore (e ottima) con $v(f) = 2$.

Teorema di Ford e Fulkerson

Al fine di individuare un algoritmo che effettivamente restituisca sempre il flusso massimo, torniamo alla questione se la relazione tra valori del massimo flusso e del minimo taglio dell'Equazione 1.1 sia effettivamente un'uguaglianza. Possiamo mostrare, nel seguente classico teorema, che la risposta a tale domanda è affermativa.

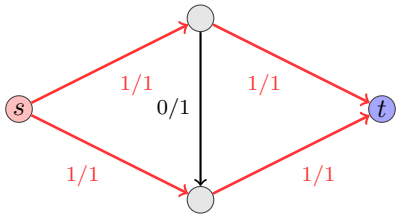


Figura 1.3: Soluzione con flusso massimo: $v(f) = 2$

Teorema 1.4 (Ford, Fulkerson) Per ogni rete N, il valore del massimo flusso è uguale a quello del taglio minimo.

$$\max_{f \text{ ammissibile}} v(f) = \min_{A \subset V} c(A)$$

Al fine di dimostrare il Teorema 1.4, introduciamo il concetto di rete residua.

Definizione 1.1 Data una rete N e un flusso ammissibile f su N, la rete residua N_f è la quadrupla $\langle G, s, t, c_f \rangle$, dove

- per ogni $(u, v) \in E_f$, $c_f(u, v) = c(u, v) - f(u, v) + f(v, u)$

Chiamiamo gli archi in N_f archi residui e capacità residue le relative capacità, definite da c_f. N_f rappresenta, con la sua struttura, come sia possibile modificare la distribuzione di flusso su N. Essa comprende, come definito sopra, i seguenti tipi di archi:

- archi di N che trasportano flusso e su cui è ancora possibile inviare flusso: un arco di questo tipo ha una capacità residua pari alla differenza tra la sua capacità e il flusso che lo attraversa
- archi opposti agli archi di N che trasportano flusso: un arco di questo tipo ha capacità residua pari alla somma fra la propria capacità e il flusso dell'arco di N a cui è opposto

Si noti che la somma delle capacità residue di una coppia di archi opposti di N_f è invariabilmente pari alla somma delle capacità dei corrispondenti archi opposti di N : la presenza di un flusso che attraversa un arco determina soltanto la ripartizione della capacità tra i due archi opposti di N_f.

Per ogni arco $(u, v) \in E$ non saturo (per cui cioè $f(u, v) < c(u, v)$) la capacità residua $c_f(u, v)$ è la quantità di flusso che è ancora possibile inviare su (u, v), mentre l'aumento di capacità residua dell'arco opposto $c_f(v, u) - f(u, v)$ è la quantità di flusso che è possibile sottrarre da (u, v).

Definizione 1.2 Data una rete N e un flusso f, un cammino aumentante è un cammino da s a t in N_f.

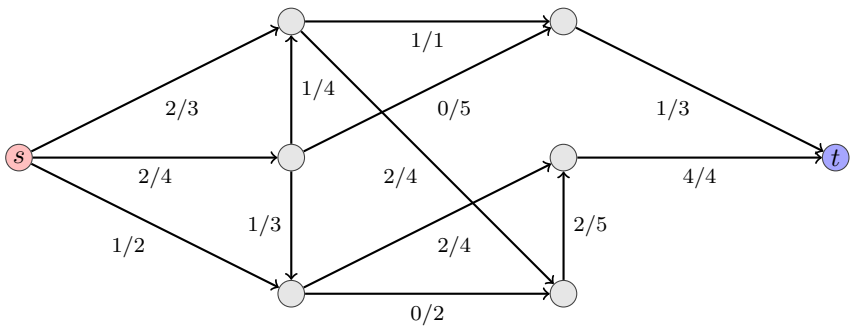


Figura 1.4: Esempio di flusso su rete

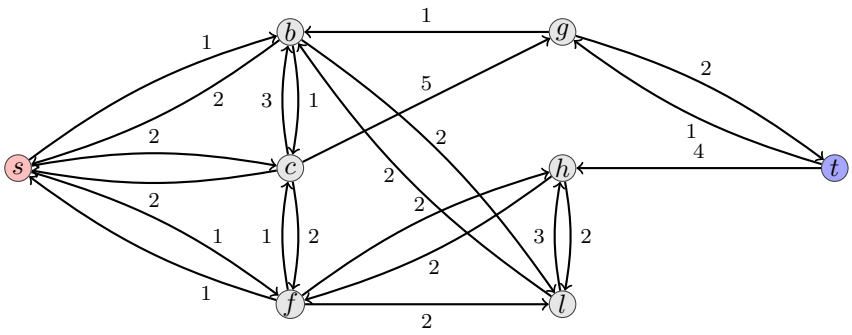


Figura 1.5: Rete residua

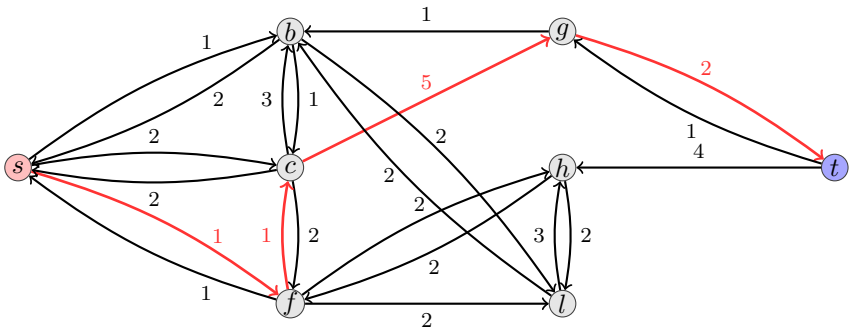


Figura 1.6: Cammino aumentante

Data una rete N e un flusso f, assumiamo che esista un cammino aumentante in N_f e sia $\mu_P = \min_{e \in P} c_f(e)$ la minima capacità residua sugli archi di P.

Definiamo un nuovo flusso f' su E nel modo seguente: per ogni $(u, v) \in P$,

1. $f'(u, v) = f(u, v) + \mu_p$ se $(u, v) \in P$
2. $f'(u, v) = f(u, v)$ altrimenti

Chiaramente, f' è un flusso ammissibile: infatti, la condizione sulla capacità dei nodi è soddisfatta in quanto, per ogni arco $(u, v) \in P$ dove viene incrementato, il flusso non supera, per definizione di μ_P la capacità dell'arco. Infine, la conservazione del flusso è mantenuta, in quanto in ogni nodo interessato (quelli sul cammino P) si ha il medesimo incremento del flusso in entrata e in uscita.
Inoltre, è evidente che $v(f') = v(f) + \mu_P$.

Osserviamo che f' potrebbe avere valore positivo su una coppia di archi opposti $(x, y), (y, x) \in E$; in tal caso, sarebbe sufficiente applicare il procedimento illustrato in precedenza ed ottenere un nuovo flusso $\overline{f}$ tale che $v(\overline{f}) = v(f') > v(f)$ e, per ogni arco $(x, y) \in E$, $\overline{f}(x, y) = 0$ o $\overline{f}(y, x) = 0$.

Possiamo allora dimostrare il seguente teorema.

Teorema 1.5 Data una rete N e un flusso ammissibile f, le seguenti condizioni sono equivalenti:

1. f è un flusso massimale
2. Non esistono cammini aumentanti in N_f
3. Esiste un taglio A tale che $v(f) = c(A)$

Dimostrazione: Mostriamo l'equivalenza dimostrando che 1⇒2⇒3⇒1.

1. $1 \Rightarrow 2$: infatti abbiamo già mostrato poco sopra l'implicazione equivalente $\neg 2 \Rightarrow \neg 1$, in cui dalla presenza di cammini aumentanti deriva che il flusso non è massimale

2. $2 \Rightarrow 3$: chiamiamo S_f l'insieme dei nodi raggiungibili da s in G_f. Sicuramente $t \notin S_f$ perché altrimenti esisterebbe un cammino aumentante. Possiamo osservare che per ogni $(u, v) \in E$ per cui $u \in S_f$ e $v \notin S_f$, si ha certamente $f(u, v) = c(u, v)$, in quanto altrimenti l'arco (u, v) sarebbe presente in G_f con capacità residua $c_f(u, v) = c(u, v) - f(u, v) > 0$, per cui v sarebbe raggiungibile da s, contraddicendo l'ipotesi che $v \notin S_f$. Inoltre, per ogni $(u, v) \in E$ per cui $u \notin S_f$ e $v \in S_f$ deve essere $f(u, v) = 0$, in quanto altrimenti in G_f sarebbe presente l'arco (v, u) con capacità residua $f(u, v) > 0$, per cui u sarebbe raggiungibile da s, contraddicendo ancora l'ipotesi che $u \notin S_f$.

   Per il Teorema 1.2, abbiamo che

   $$v(f) = \sum_{e \in \mathcal{O}(S_f)} f(e) - \sum_{e \in \mathcal{I}(S_f)} f(e)$$

   ma, dato che $f(e) = c(e)$ per ogni arco e che collega un nodo di S_f ad uno di $V - S_f$, ne deriva che $\sum_{e \in \mathcal{O}(S_f)} f(e) = \sum_{e \in \mathcal{O}(S_f)} c(e)$; inoltre, per ogni arco e' che collega un nodo di $V - S_f$ ad un nodo di S_f, abbiamo che $f(e') = 0$, per cui $\sum_{e \in \mathcal{I}(S_f)} f(e) = 0$ e quindi, in definitiva,

   $$v(f) = \sum_{e \in \mathcal{O}(S_f)} c(e) = c(S_f)$$

3. $3 \Rightarrow 1$: per il Teorema 1.3, $v(f)$ non può essere maggiore della capacità di un qualunque taglio su V. Dato che, per ipotesi, $v(f) = c(S_f)$, ne deriva che $v(f)$ è il valore massimo possibile per un flusso ammissibile su N.

□ La dimostrazione del Teorema 1.4 deriva immediatamente osservando che S_f deve essere necessariamente un taglio di capacità minima, in quanto altrimenti, detto M il taglio di capacità minima $c(M) < c(S_f)$ il flusso massimo f avrebbe valore $v(f) > c(M)$, contraddicendo il Teorema 1.3.

Algoritmo di Ford e Fulkerson

Le osservazioni precedenti permettono di definire un algoritmo per il calcolo del massimo flusso in una rete.

Algorithm 1.2: Algoritmo di Ford e Fulkerson per il massimo flusso

Input: $G = (V, E)$, $s, t \in V$, $c : E \mapsto \mathbb{R}^+$
Output: Flusso $f : E \mapsto \mathbb{R}^+$ da s a t

```
1 foreach e ∈ E do f(e) ← 0;
2 Deriva N_f;
3 while esiste un cammino aumentante da s a t in N_f do
4     Prendi un qualunque cammino aumentante P;
5     foreach (u, v) ∈ P do
6         f(e) ← f(e) + min_{e∈P} c_f(e);
7     Aggiorna N_f
8 return f
```

In Figura 1.7 viene mostrata la rete residua derivante da quella in Figura 1.5 aumentando il flusso lungo il cammino aumentante in Figura 1.6. In Figura 1.8 viene rappresentato il flusso corrispondente: si può osservare come il valore del flusso da s a t sia aumentato da 5 a 6.

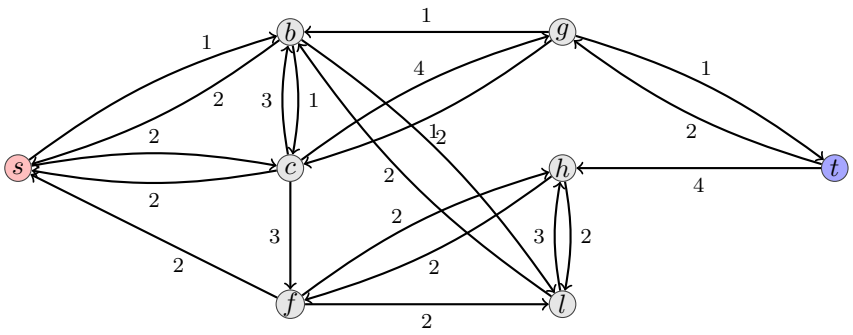


Figura 1.7: Rete residua successiva

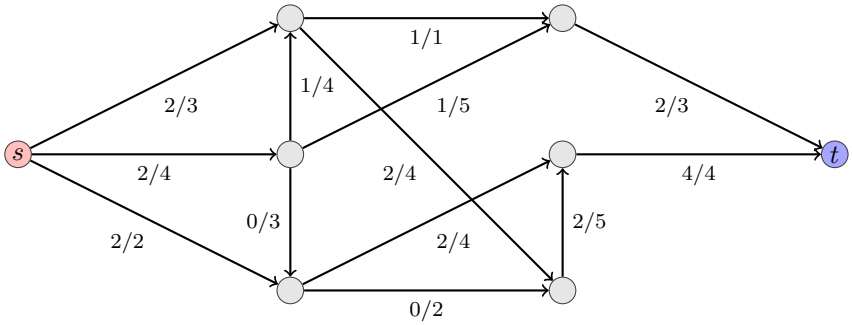


Figura 1.8: Flusso corrispondente alla rete residua di Figura 1.7

Per il Teorema 1.5, se l'algoritmo si ferma, non trovando cammini aumentanti, il flusso ottenuto è massimale. Si osservi però che, in presenza di flussi e capacità definite sui reali, l'incremento di flusso ad ogni singolo passo, pur necessariamente positivo, può tendere a 0 e l'algoritmo può non terminare, approssimando asintoticamente, all'infinito, il valore del flusso massimale.

Tale situazione può verificarsi, ad esempio, per la rete in Figura 1.9, in cui con r abbiamo indicato la soluzione positiva dell'equazione $x^2 + x - 1 = 0$, vale a dire

$$r = \frac{\sqrt{5}-1}{2} \simeq 0.618$$

(incidentalmente, tale valore è l'inverso della sezione aurea $\phi = \frac{1+\sqrt{5}}{2}$). Si noti che, per definizione, $r^2 = 1 - r$ e che, in generale, $r^{n+2} = r^n - r^{n+1}$ per $n \geq 1$. Si noti che il flusso massimo ha valore $v(f^*) = 1 + r + r^2 = 2$.

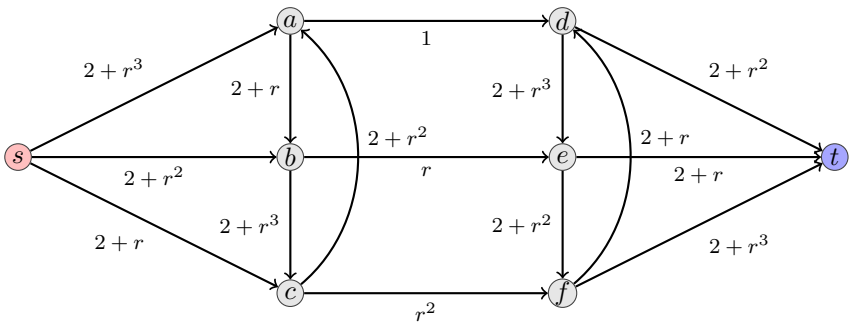


Figura 1.9: Esempio di rete su cui l'algoritmo di Ford e Fulkerson converge all'ottimo in infiniti passi

Assumiamo che il primo cammino aumentante selezionato dall'algoritmo corrisponda a $s-a-d-t$: assegnare agli archi su tale cammino la capacità residua minima su di essi (pari a 1), porta alla situazione in Figura 1.10, in cui ad ogni arco è associata la capacità residua, e ad un flusso di valore $v(f) = 1$.

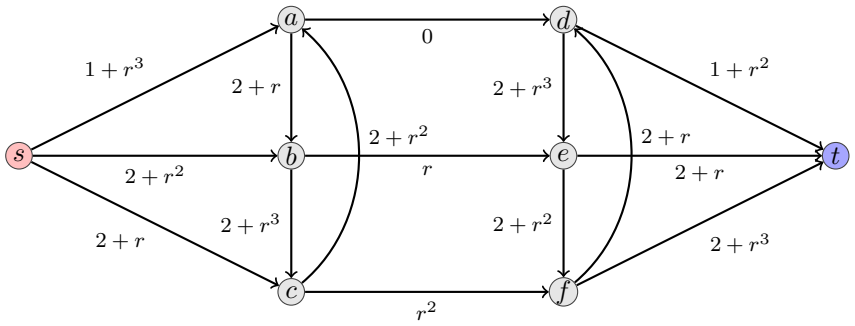


Figura 1.10: Rete precedente con flusso assegnato dopo un passo dell'algoritmo di Ford e Fulkerson

Assumiamo ora che il nuovo cammino aumentante selezionato dall'algoritmo sia $s-c-f-d-a-b-e-t$: con capacità residua minima pari a r^2: ciò porta alla situazione in Figura 1.11, in cui ad ogni arco è associata la capacità residua (ricordando che $r - r^2 = r^3$), e ad un flusso di valore $v(f) = 1 + r^2$.

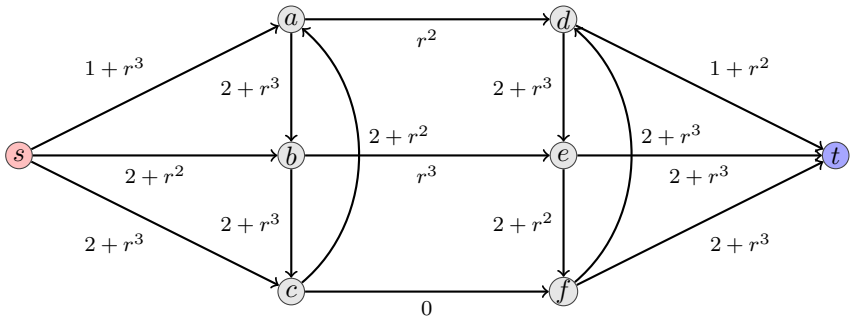


Figura 1.11: Rete precedente con flusso assegnato dopo due passi dell'algoritmo di Ford e Fulkerson

Assumiamo ora che il nuovo cammino aumentante selezionato dall'algoritmo sia $s-b-e-f-c-a-d-t$: con capacità residua minima pari a r^3: ciò porta alla situazione in Figura 1.12, in cui ad ogni arco è associata la capacità residua (ricordando che $r^2 - r^3 = r^4$), e ad un flusso di valore $v(f) = 1 + r^2 + r^3$.

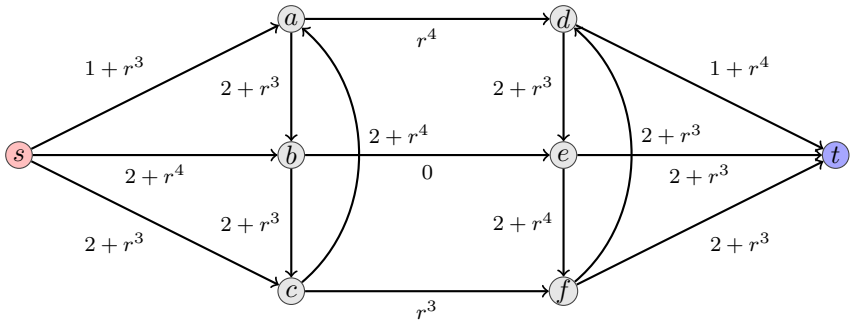


Figura 1.12: Rete precedente con flusso assegnato dopo tre passi dell'algoritmo di Ford e Fulkerson

Assumiamo ora che il nuovo cammino aumentante selezionato dall'algoritmo sia $s-a-d-e-b-c-f-t$: con capacità residua minima pari a r^4: ciò porta alla situazione in Figura 1.13, in cui ad ogni arco è associata la capacità residua (ricordando che $r^3 - r^4 = r^5$), e ad un flusso di valore $v(f) = 1 + r^2 + r^3 + r^4$.

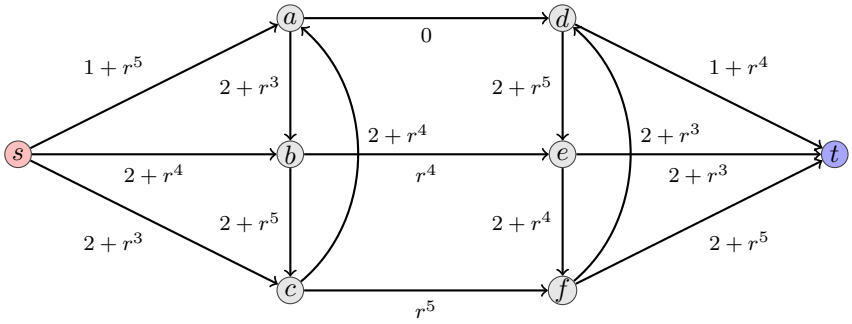


Figura 1.13: Rete precedente con flusso assegnato dopo quattro passi dell'algoritmo di Ford e Fulkerson

Proseguendo allo stesso modo, il flusso aggiunto dal k-esimo cammino aumentante ha valore r^k. Quindi, dopo k iterazioni, il flusso ha valore

$$v(f_k) = 1 + \sum_{i=2}^{k} r^i = -r + \sum_{i=0}^{k} r^i = \frac{1 - r^{k+1}}{1 - r} - r$$

Per k che tende ad infinito, abbiamo allora che

$$\lim_{k\to\infty} v(f_k) = \frac{1}{1-r} - r = \frac{1 - r + r^2}{1 - r} = \frac{2(1-r)}{1-r} = 2$$

per cui il valore del flusso tende, in un numero infinito di passi, al valore del flusso massimo.

La possibilità di una convergenza all'ottimo soltanto in un numero infinito di passi viene esclusa se ci si limita a considerare capacità definite su insiemi discreti, come ad esempio gli interi.

Teorema 1.6 Se $c(e) \in \mathbb{N}$ per tutti gli archi $e \in E$, allora l'Algoritmo 1.2 termina dopo un numero finito di passi.

Dimostrazione: Dato che le capacità degli archi di G sono intere, sono interi anche i valori dei flussi e le capacità degli archi di N_f. Quindi, se esiste un cammino aumentante, il suo arco di capacità minima avrà capacità intera, e quindi il flusso di tutti gli archi o rimane immutato (se gli archi non appartengono al cammino aumentante) o viene incrementato di un valore intero (se sono sul cammino).

Dato che $v(f) \leq \sum_{e\in\mathcal{O}(s)} c(e)$, ne consegue che $v(f)$ può essere incrementato al più $\sum_{e\in\mathcal{O}(s)} c(e) < \infty$ volte. □

Trovare un cammino aumentante su N_f richiede una visita in ampiezza del grafo, e quindi tempo $O(|E_f|) = O(|E|)$. Di conseguenza, l'Algoritmo 1.2 ha complessità $O(v(f^*) \cdot |E|)$, dove f^* è il flusso massimo. Si noti che l'algoritmo non è quindi polinomiale nella dimensione dell'input (che ha dimensione $\Theta(|E|(\log|V| + \log c_{\text{max}}))$), dove c_{max} è il valore della massima capacità di un arco in N.

Un esempio di rete con capacità intere per la quale l'Algoritmo 1.2 si comporta in modo inefficiente è mostrata in Figura 1.14.

Supponiamo che il primo cammino aumentante selezionato sia $s - a - b - t$, attraverso il quale è possibile inviare un flusso di valore unitario. La rete residua derivante è mostrata in Figura 1.15 , in cui ad ogni arco è associata la capacità residua. Il valore del flusso è $v(f) = 1$.

Se il successivo cammino aumentante selezionato è $s-b-a-t$, attraverso il quale è possibile inviare un flusso di valore unitario, ne deriva la rete rete residua in Figura 1.16 , in cui ad ogni arco è associata la capacità residua. Il valore del flusso è $v(f) = 2$. Iterando i passi precedenti, abbiamo che ad ogni iterazione il flusso aumenta di 1 unità. Dato che il flusso massimo è chiaramente pari a $2M$ (il valore del taglio minimo), ne deriva che l'algoritmo di Ford e Fulkerson esegue $2M$ iterazioni.

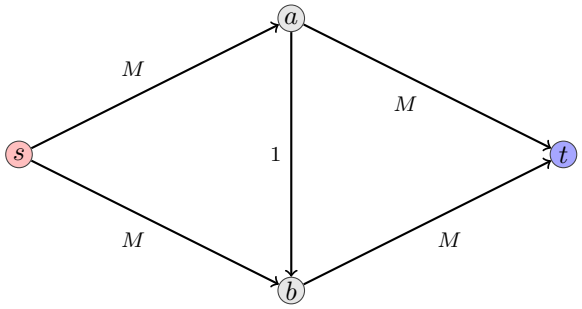


Figura 1.14: Esempio di rete su cui l'algoritmo di Ford e Fulkerson richiede tempo esponenziale

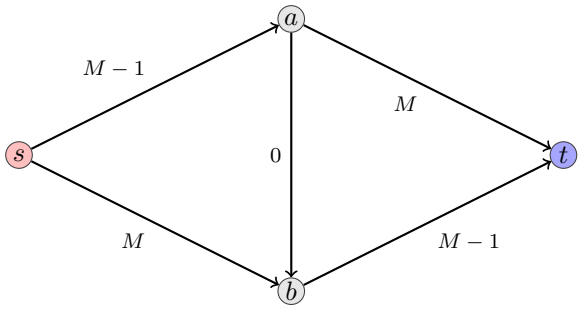


Figura 1.15: Rete precedente con flusso assegnato dopo un passo dell'algoritmo di Ford e Fulkerson

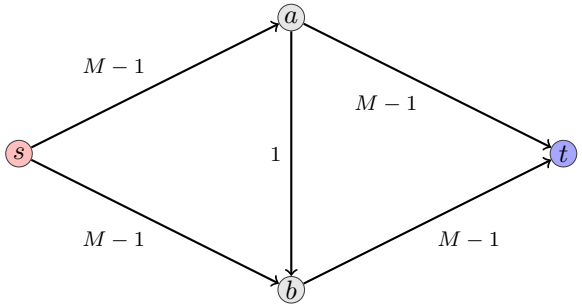


Figura 1.16: Rete precedente con flusso assegnato dopo due passi dell'algoritmo di Ford e Fulkerson

Decomposizione del flusso

Al fine di definire algoritmi polinomiali per il problema del massimo flusso, è necessario selezionare il cammino aumentante in modo oculato e al tempo stesso efficiente. Prima di considerare vari modi di far questo, mostriamo che una scelta sufficientemente oculata può consentire di avere ad ogni iterazione un aumento significativo del flusso, nel senso che esiste sempre un cammino aumentante la cui selezione porterebbe a tale incremento.

Per mostrare ciò, introduciamo per prima cosa il teorema seguente, che ci dice che un flusso ammissibile può essere decomposto su un insieme opportuno di cicli e cammini $s - t$.

Teorema 1.7 (Flow decomposition) Dati una rete N ed un suo flusso ammissibile f, esiste un insieme di cammini $\mathcal{P}$, un insieme di cicli $\mathcal{C}$ e una assegnazione di pesi $w : \mathcal{P} \cup \mathcal{C} \mapsto \mathbb{R}^+$, che interpretiamo come flussi lungo cammini e cicli, tali che:

1. $\forall e \in E, f(e) = \sum_{p \in \mathcal{PC}(e)} w(p)$, dove $\mathcal{PC}(e) = \{p \in \mathcal{P} \cup \mathcal{C} : e \in p\}$: in altri termini, il flusso su ogni arco può essere decomposto nella somma dei flussi associati a tutti i cammini e i cicli che includono l'arco stesso

2. $v(f) = \sum_{p \in \mathcal{P}} w(p)$: il flusso totale da s a t è decomposto nella somma dei flussi lungo tutti i cammini
3. $|\mathcal{P}| + |\mathcal{C}| \leq |E_>|$, dove $E_> = \{e \in E : f(e) > 0\}$ è l'insieme degli archi aventi flusso positivo: vale a dire, il numero di cicli e cammini non supera quello degli archi che trasportano flusso

Dimostrazione: La dimostrazione è effettuata per induzione sul numero di archi $e \in E_>$.

Caso base. $f(e) = 0$ per ogni $e \in E$, per cui $E_> = \emptyset$. Allora il lemma è banalmente verificato per $\mathcal{P}, \mathcal{C} = \emptyset$.

Passo induttivo. Assumiamo che le condizioni del teorema siano verificate per ogni flusso avente valore positivo su k archi. Consideriamo un qualunque flusso f avente valore positivo su $k+1$ archi. Sia $(u, v) \in E$ un arco con $f(u, v) > 0$. Se $v \neq t$, per il vincolo di conservazione del flusso al nodo v esiste un nodo x tale che $f(v, x) > 0$. Simmetricamente, se $u \neq s$ esiste $y \in V$ tale che $f(y, u) > 0$. Iterando la stessa considerazione per x e y, otteniamo o un cammino da s a t o un ciclo, che in entrambi i casi indichiamo come $\overline{p}$. Posto $w(\overline{p}) = \min_{e \in \overline{p}} f(e)$, consideriamo il flusso f' definito nel modo seguente:

$$f'(i,j) = \begin{cases} f(i,j) - w(\overline{p}) & (i,j) \in \overline{p} \\ f(i,j) & \text{altrimenti} \end{cases}$$

$f'(i, j)$ è quindi il flusso ottenuto riducendo al massimo quanto trasportato lungo $\overline{p}$. Questo insieme esiste e soddisfa le tre proprietà dell'enunciato, in quanto f' ha almeno un arco con flusso positivo in meno di f (quello di flusso minimo in $\overline{p}$). Indichiamo con $\mathcal{PC}'$ l'insieme di cicli e cammini associato a f' e con w' la relativa funzione di peso.

Definendo $\mathcal{PC} = \mathcal{PC}' \cup \{\overline{p}\}$ e $w(p) = w'(p)$ per ogni $p \in \mathcal{PC}'$, possiamo osservare quanto segue.

- La proprietà 1 vale anche per f: infatti, per ogni $e \notin \overline{p}$ si ha che $f(e) = f'(e)$ per definizione di f' e $\mathcal{PC}(e) = \mathcal{PC}'(e)$, mentre se $e \in \overline{p}$ allora $f(e) = f'(e) + w(\overline{p})$ e $\mathcal{PC}(e) = \mathcal{PC}'(e) \cup \{\overline{p}\}$.
- La proprietà 2 vale per f. Infatti, se $\overline{p} \in \mathcal{P}$ allora $\mathcal{P} = \mathcal{P}' \cup \{\overline{p}\}$ e

  $$v(f) = v(f') + w(\overline{p}) = \sum_{p \in \mathcal{P}'} w(p) + w(\overline{p}) = \sum_{p \in \mathcal{P}} w(p).$$

  Se invece $\overline{p} \in \mathcal{C}$ allora $\mathcal{P}' = \mathcal{P}$ e $v(f) = v(f')$: iInfatti, ricordiamo che $v(f) = \sum_{e \in \mathcal{O}(s)} f(e)$ e, poiché abbiamo assunto sempre nullo il flusso sugli archi entranti nella sogente e poiché $\overline{p}$ è un ciclo, nessun arco in $\mathcal{O}(s)$ è contenuto in $\overline{p}$.
- Infine, per quanto riguarda la proprietà 3, basta osservare che, nel passare da f' a f, il numero complessivo di cammini e cicli è aumentato di 1 ($\overline{p}$), mentre il numero di archi con flusso positivo è aumentato di almeno 1 (l'arco di flusso minimo in $\overline{p}$).

□

Possiamo allora derivare immediatamente il teorema seguente, che ci dice che esiste sempre un cammino da s a t sul quale viene trasportata una frazione significativa del flusso massimo.

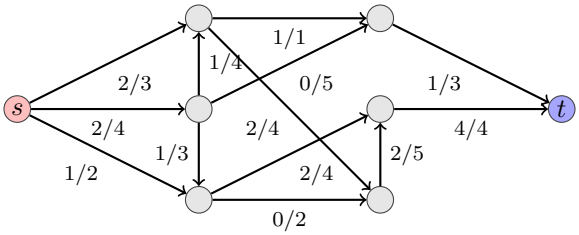
Teorema 1.8 Data una rete N di flusso massimo f^* esiste un cammino da s a t i cui archi hanno assegnato ciascuno un flusso di valore almeno $\dfrac{v(f^*)}{|E|}$.

Dimostrazione: Per il Teorema 1.7, esiste in N un insieme $\mathcal{P}$ di cammini da s a t che si ripartiscono l'intero f^*.

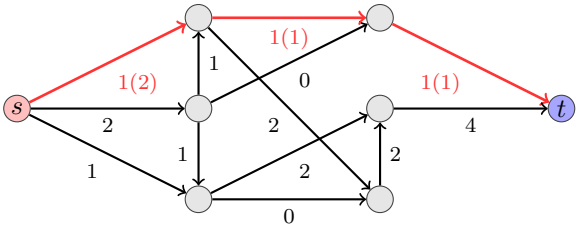
Quindi, esiste un cammino $p_i \in \mathcal{P}$ con assegnato un flusso f_i tale che

$$v(f_i) \geq \frac{v(f^*)}{|\mathcal{P}|} \geq \frac{v(f^*)}{|E|} \qquad \text{dato che } |E| \geq |E_{f^*}| \geq |\mathcal{P}| + |\mathcal{C}| \geq |\mathcal{P}|$$

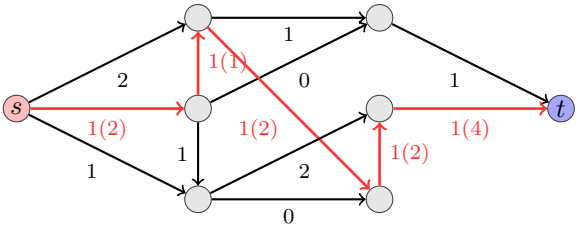
Dato che ogni arco su un cammino che porta flusso f deve necessariamente avere un flusso di valore almeno pari a $v(f)$, ne deriva l'enunciato. □



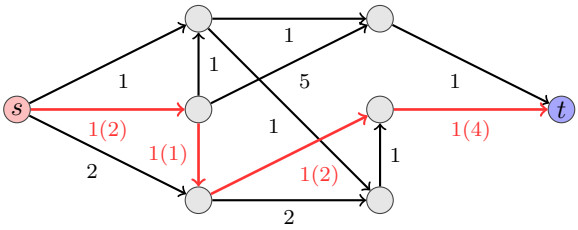
(a) Esempio di flusso



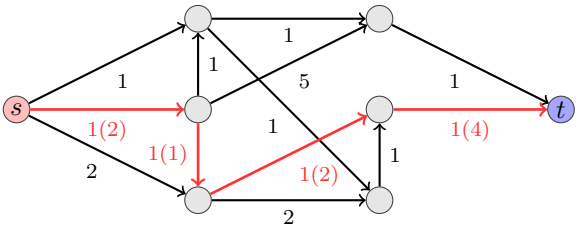
(b) Cammino 1



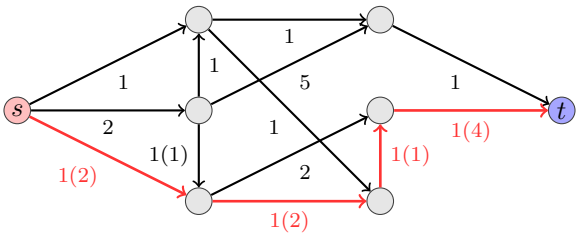
(c) Cammino 2



(d) Cammino 3



(e) Cammino 4



(f) Cammino 5

Figura 1.17: Esempio di decomposizione di flusso in cammini

Il Teorema 1.7 ci dice che qualunque flusso può essere decomposto in un insieme di cammini da s a t e un insieme di cicli (al più tanti quanti sono gli archi del grafo), ognuno con flusso associato w, in modo tale che il flusso su un arco è dato dalla somma dei flussi sui cammini e sui cicli cui l'arco appartiene.

Il teorema seguente ci dice che il problema del massimo flusso si riduce a sé stesso quando da una rete si sottrae un flusso.

Teorema 1.9 Sia data una rete N di flusso massimo f^* e sia f un flusso su essa. Allora, il flusso massimo sulla rete N' ottenuta ponendo per ogni arco e la relativa capacità come $c'(e) = c(e) - f(e)$ ha valore $v(f^*) - v(f)$.

Dimostrazione: Come prima cosa, mostriamo che ogni flusso in N' ha valore al più pari a $v(f^*) - v(f)$. A tal fine, consideriamo un qualunque flusso f' in N' e definiamo il flusso $\overline{f}$ come:

$$\overline{f}(e) = f(e) + f'(e) \qquad \forall e \in E$$

Il flusso $\overline{f}$ è quindi ottenuto a partire da f', ripristinando il flusso f originariamente sottratto da N: chiaramente, $\overline{f}$ è un flusso su N, in quanto le condizioni di continuità sui nodi e di rispetto delle capacità degli archi continuano a valere. Inoltre, si ha che $v(\overline{f}) = v(f) + v(f')$ per definizione e che $v(\overline{f}) \leq v(f^*)$ per la massimalità di f^*: da ciò deriva che $v(f) + v(f') \leq v(f^*)$ e quindi $v(f') \leq v(f^*) - v(f)$

Definiamo ora il flusso $\hat{f}$ come:

$$\hat{f}(e) = f^*(e) - f(e) \qquad \forall e \in E$$

Osserviamo che $\hat{f}$ è un flusso in N', in quanto $\hat{f}(e) = f^*(e) - f(e) \leq c(e) - f(e)$. Inoltre, dato che $v(\hat{f}) = v(f^*) - v(f)$ per definizione, $\hat{f}$ è un flusso massimo in N'. □

Chiaramente, per definizione di rete residua, $v(f^*) - v(f)$ è anche il massimo flusso nella rete residua N_f corrispondente.

Algoritmi polinomiali per max-flow

Al fine di rendere più efficiente l'Algoritmo 1.2, dobbiamo effettuare, ad ogni iterazione, una scelta oculata del cammino aumentante, selezionando cammini che consentano un incremento significativo del flusso.

Cammino di capacità massima

Una scelta ragionevole appare quella di selezionare il cammino aumentante di capacità massima, ottenendo l'Algoritmo 1.3

Algorithm 1.3: Algoritmo di cammino aumentante di massima capacità

Input: $G = (V, E)$, $s, t \in V$, $c : E \mapsto \mathbb{R}^+$
Output: Flusso $f : E \mapsto \mathbb{R}^+$ da s a t

```
1 foreach e ∈ E do f(e) ← 0;
2 Deriva N_f;
3 while esiste un cammino aumentante da s a t in N_f do
4     Prendi il cammino aumentante P tale che min_{e∈P}(c_f(e)) è massimo;
5     foreach (u, v) ∈ P do
6         if (u, v) ∈ E then f(e) ← f(e) + min_{e∈P} c_f(e);
7         else f(e) ← f(e) − min_{e∈P} c_f(e)
8     Aggiorna N_f
9 return f
```

Al fine di mostrare che l'algoritmo opera in tempo polinomiale, dimostriamo il seguente teorema.

Teorema 1.10 L'Algoritmo 1.3 calcola il flusso massimo f^* effettuando $O(|E| \log v(f^*))$ iterazioni.

Dimostrazione: Indichiamo con f_i il flusso totale trovato dopo i iterazioni, e quindi dopo aver trovato i cammini aumentanti (e aver aumentato il flusso per i volte). Mostriamo che nella rete residua ottenuta N_{f_i} il flusso massimo ha valore al più pari a

$$v(f^*)\left(1 - \frac{1}{|E|}\right)^i$$

Notiamo che in generale, per il Teorema 1.9, il flusso complessivo ancora disponibile nella rete N_{f_i} è $v(f^*) - v(f_i)$ e, per il Teorema 1.8, esiste uno di tali cammini (che chiamiamo p_i^*) di capacità almeno pari al rapporto tra $v(f^*) - v(f_i)$ e il numero di archi nella rete, che in una rete residua è sempre $|E|$.

Quindi, il cammino aumentante in N_{f_i} corrispondente a p_i^* ha capacità almeno $\frac{v(f^*) - v(f_i)}{|E|}$. L'Algoritmo 1.3, selezionando proprio tale cammino, fa sì che

$$v(f_{i+1}) \geq v(f_i) + \frac{v(f^*) - v(f_i)}{|E|}$$

Il flusso ancora disponibile dopo l'$(i+1)$-esima iterazione è allora

$$\begin{aligned} v(f^*) - v(f_{i+1}) &\leq v(f^*) - \left(v(f_i) - \frac{v(f^*) + v(f_i)}{|E|} \right) = (v(f^*) - v(f_i)) \left(1 - \frac{1}{|E|} \right) \\ &\leq (v(f^*) - v(f_{i-1})) \left(1 - \frac{1}{|E|} \right)^2 \leq \dots \\ &\leq (v(f^*) - v(f_{i-k})) \left(1 - \frac{1}{|E|} \right)^{k+1} \end{aligned}$$

Dato che inizialmente $v(f_0) = 0$, abbiamo che

$$v(f^*) - v(f_{i+1}) \leq v(f^*) \left(1 - \frac{1}{|E|} \right)^{i+1}$$

In generale, si ha che $1 - x \leq e^{-x}$ se $x \geq 0$,per cui

$$v(f^*) - v(f_{i+1}) \leq v(f^*) \left(1 - \frac{1}{2|E|} \right)^{i+1} \leq v(f^*) e^{-\frac{i+1}{2|E|}} = e^{\log v(f^*) - \frac{i+1}{2|E|}}$$

se $i > 2|E| \log v(f^*) - 1$ ne consegue che $e^{\log v(f^*) - \frac{i+1}{2|E|}} < 1$, quindi $v(f^*) - v(f_{i+1}) < 1$ e di conseguenza $v(f^*) - v(f_{i+1}) = 0$ necessariamente, in quanto le capacità e i flussi sono interi per ipotesi.

In definitiva, abbiamo mostrato che l'Algoritmo 1.3 in al più $1 + |E| \log f(v^*)$ iterazioni raggiunge una situazione in cui la rete residua non ha più cammini aumentanti, e quindi il flusso trovato ha valore massimo $v(f^*)$. □

La complessità totale dell'Algoritmo 1.3 dipende inoltre dal costo della ricerca di un cammino di capacità massima, operazione effettuata ad ogni iterazione. Mostriamo ora che questa ricerca può essere effettuata mediante una semplice modifica dell'algoritmo di Dijkstra per la ricerca del cammino minimo tra s e t, riportato come Algoritmo 1.4.

Come detto, l'algoritmo di Dijkstra può essere modificato in modo semplice per trovare il cammino di capacità massima in un grafo. A tal fine osserviamo che:

1. nella ricerca di un cammino minimo il valore associato ad un cammino comprendente gli archi $E_P = \{e_1, \dots, e_k\}$ è dato da $\sum_{e \in E_P} l(e)$; nella ricerca di un cammino di capacità massima, il valore del cammino è $\min_{e \in E_P} c(e)$
2. nel primo caso cerchiamo un minimo, nel secondo un massimo

L'Algoritmo 1.4 è basato su alcune caratteristiche del problema del cammino minimo, che sono valide ``mutatis mutandis" anche per il problema del cammino di capacità massima.

- il costo di un cammino è monotono all'estendere del cammino stesso: ad esempio, per il cammino minimo, il costo di un cammino $E_P = \{e_1, \dots, e_k\}$ è almeno pari al costo del cammino $E_{P'} = \{e_1, \dots, e_t\}$, con $t < k$ (per il cammino di capacità massima, è al più pari a tale valore)
- è possibile determinare il costo di un cammino $E_P = \{e_1, \dots, e_k\}$ a partire dal costo del cammino $E_{P''} = \{e_1, \dots, e_{k-1}\}$ e da $l(e_k)$ (o $c(e_k)$)

Da queste considerazioni, deriva l'Algoritmo 1.5 per la ricerca del cammino di capacità massima.

L'Algoritmo 1.5 ha la stessa struttura e la stessa complessità $O((|V| + |E|) \log |V|)$ dell'Algoritmo 1.4, da cui deriva che l'Algoritmo 1.3 ha complessità totale $O((|V| + |E|)|E| \log |V| \log v(f^*))$.

Algorithm 1.4: Algoritmo di Dijkstra per la ricerca di un cammino minimo

Input: $G = (V, E)$, $s, t \in V$, $l : E \mapsto \mathbb{R}^+$;
Output: Cammino minimo da s a t;

```
1  foreach v ∈ V − {s} do d(v) ← ∞;
2  d(s) ← 0;
3  foreach v ∈ V do
4      Inserisci v in una coda di priorità PQ con chiave d(v);
5      π(v) ←nil;
6  while PQ ≠ ∅ do
7      Estrai da PQ il nodo u con d(u) minimo;
8      foreach v tale che (u, v) ∈ E do
9          if d(v) > d(u) + l(u, v) then
10             d(v) ← d(u) + l(u, v);
11             π(v) ← u;
12             Decrementa a d(v) la chiave di v in PQ;
13 Sia P il cammino da t a s definito dai puntatori π;
14 return il cammino inverso di P
```

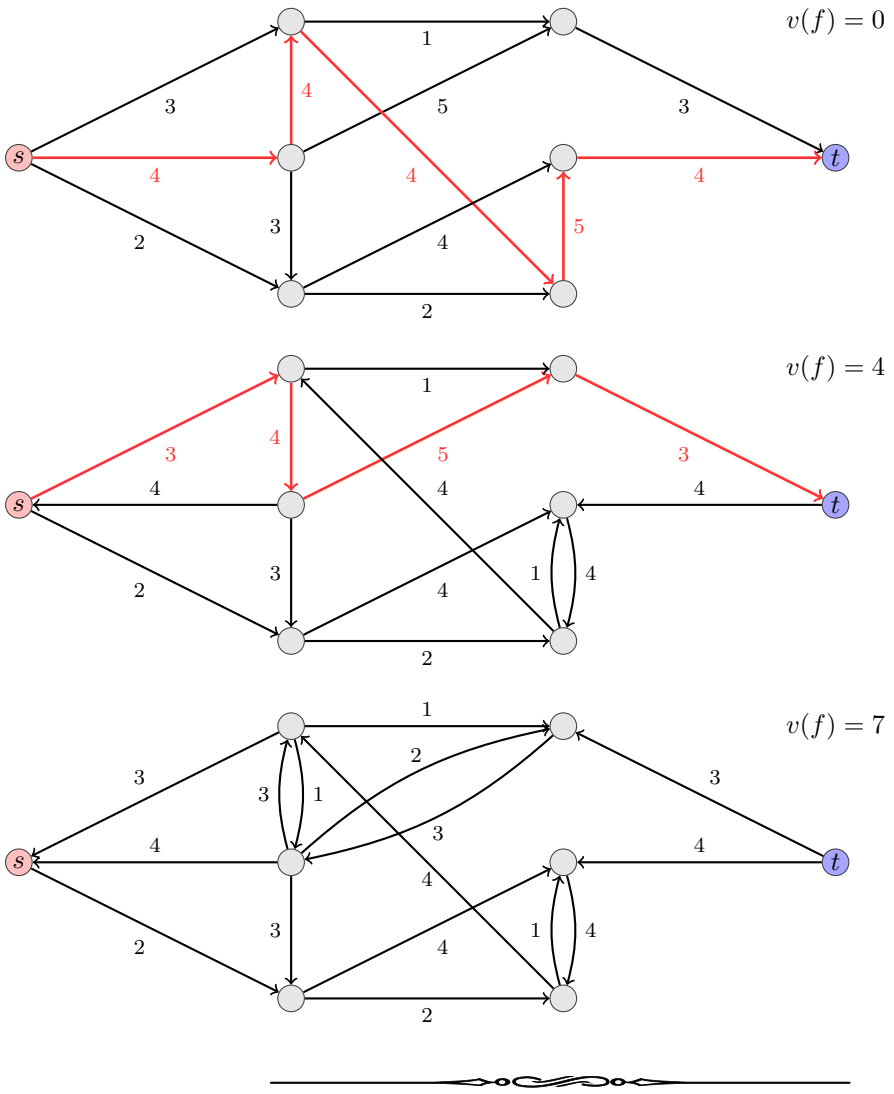
Algorithm 1.5: Algoritmo di ricerca di un cammino aumentante di massima capacità

Input: $G = (V, E)$, $s, t \in V$, $c : E \mapsto \mathbb{R}^+$;
Output: Cammino di capacità massima P da s a t;

```
1  foreach v ∈ V − {s} do m(v) ← 0;
2  m(s) ← ∞;
3  foreach v ∈ V do
4      Inserisci v in una coda di priorità PQ con chiave m(v);
5      π(v) ←nil;
6  while PQ ≠ ∅ do
7      Estrai da PQ il nodo u con m(u) massimo;
8      foreach v tale che (u, v) ∈ E do
9          if m(v) < min{m(u), c(u, v)} then
10             m(v) ← min{m(u), c(u, v)};
11             π(v) ← u;
12             Incrementa a m(v) la chiave di v in PQ;
13 Sia P il cammino da t a s definito dai puntatori π;
14 return il cammino inverso di P
```

La complessità ottenuta, come si può vedere, è funzione del valore del flusso ottimo, che non è un dato in input del problema. Possiamo però osservare, ad esempio, che $v(f^*) \leq \sum_{e \in \mathcal{O}(s)} c(e)$ e ottenere così una ulteriore delimitazione superiore $O((|V| + |E|)|E| \log |V| \log \sum_{e \in \mathcal{O}(s)} c(e))$ di tale complessità.

Di seguito, viene illustrato l'effetto sulla rete di Figura 1.4 dell'esecuzione dell'algoritmo basato sui cammini di massima capacità. In rosso è mostrato, di volta in volta, il cammino di capacità massima da s a t. Si noti che il massimo viene trovato utilizzando due cammini aumentanti, in quanto dopo due iterazioni non esistono più cammini da s a t.



Cammino δ-aumentante

Una variante della ricerca del cammino aumentante di massima capacità è costituita dal limitarsi a selezionare un cammino di capacità ``sufficientemente grande", in modo tale da incrementare comunque il flusso di una quantità significativa, anche se non la massima possibile. Se un cammino di questo tipo non viene trovato (nel senso che tutti i cammini disponibili hanno capacità limitata), allora possiamo verificare che il valore del flusso attuale è vicino al massimo.

Definizione 1.3 Un cammino P da s a t viene detto δ-aumentante se è aumentante e tutti i suoi archi hanno capacità almeno δ: cioè, se $\forall e \in P, c(e) \geq \delta$.

L' Algoritmo 1.6 di capacity scaling opera cercando iterativamente cammini δ-aumentanti, facendo tendere δ a 0.

Ad ogni iterazione, l'algoritmo verifica la presenza di un cammino aumentante di capacità almeno δ (inizialmente, δ è circa pari alla capacità dell'arco di capacità massima). Se tale cammino esiste, viene selezionato e flusso e rete residua vengono aggiornati, altrimenti, il valore di δ viene dimezzato. L'algoritmo termina quando non esistono più cammini aumentanti.

Algorithm 1.6: Algoritmo di cammino δ-aumentante

Input: $G = (V, E)$, $s, t \in V$, $c : E \mapsto \mathbb{R}^+$
Output: Flusso $f : E \mapsto \mathbb{R}^+$ da s a t

```
1  C ← max_{e∈E} c(e);
2  δ ← 2^⌊log2 C⌋;
3  foreach e ∈ E do f(e) ← 0;
4  Deriva N_f;
5  while esiste un cammino aumentante da s a t in N_f do
6      if esiste un cammino δ-aumentante P then
7          foreach (u, v) ∈ P do
8              if (u, v) ∈ E then f(e) ← f(e) + δ;
9              else f(e) ← f(e) − δ
10     else δ ← δ/2;
11     Aggiorna N_f
12 return f
```

La ricerca, dato δ, di un cammino aumentante di capacità almeno δ può essere effettuata mediante una semplice modifica dell'algoritmo di visita di un grafo in cui si considerano i soli archi di capacità almeno δ.

Teorema 1.11 L'Algoritmo 1.6 determina il flusso ottimo di una rete N in tempo $O((|V| + |E|)|E| \log C)$, dove $C = \max_{e \in E} c(e)$.

Dimostrazione: Chiaramente l'algoritmo, procedendo fino a che esistono cammini aumentanti, trova il flusso ottimo.

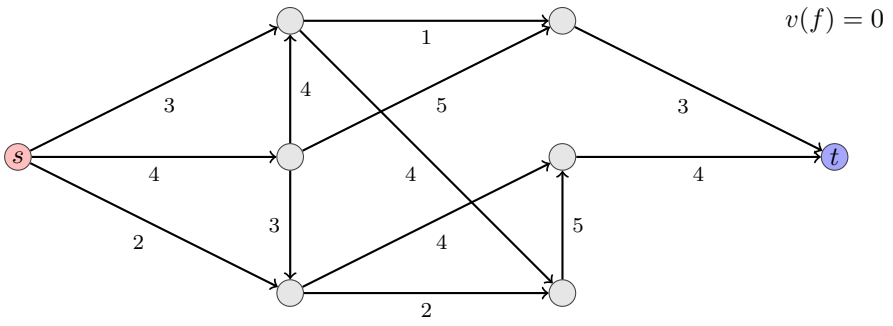
Per quanto riguarda la complessità, assumiamo, per induzione, che, nel momento in cui l'algoritmo cerca cammini di capacità almeno δ, non ne esistano di capacità $\geq 2\delta$. Ciò è banalmente verificato all'inizio, quando $\delta = 2^{\lfloor \log_2 C \rfloor} > 2^{\log_2 C - 1} = \dfrac{C}{2}$.

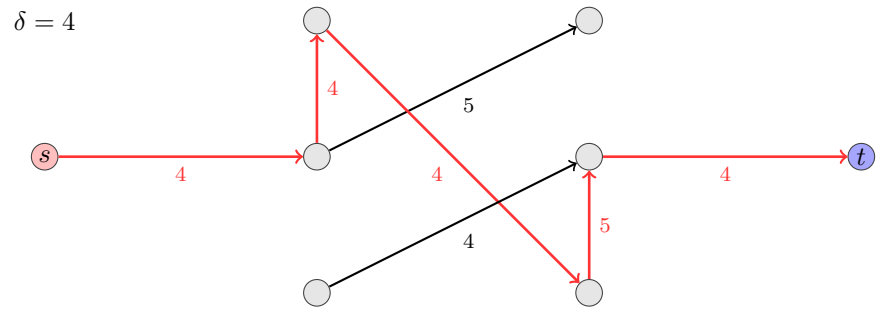
Dato che, per ogni cammino di capacità pari almeno a δ esiste almeno un arco con tale capacità, ne consegue che il numero di cammini individuati per ogni δ non può essere superiore a $2|E|$ (si ricordi che N_f può avere fino a $2|E|$ archi). Le capacità di tutti i cammini sono almeno dimezzate, dato che le loro capacità erano, per l'ipotesi induttiva, minori di 2δ.

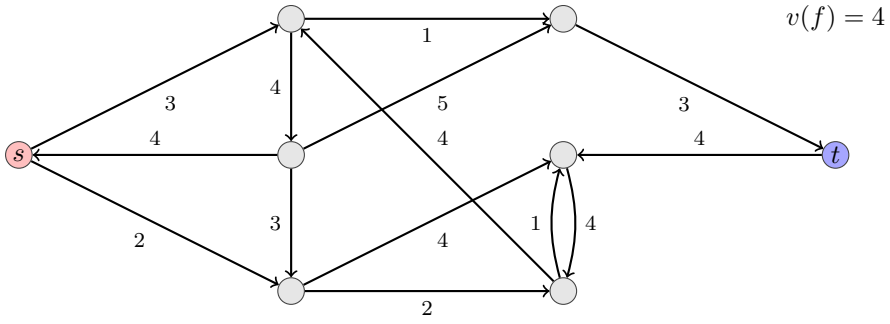
Per ogni δ, possono essere quindi effettuate al più $2|E|$ ricerche di cammini di capacità almeno δ. Ognuna di tali ricerche può essere effettuata per mezzo di una visita in ampiezza di N_f, e quindi in tempo $O(|V| + |E|)$.

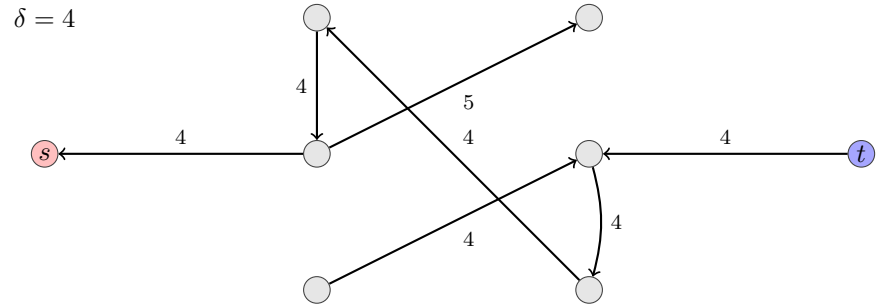
Il numero di valori δ considerati è al massimo $O(\log C)$, per cui la complessità totale risulta in effetti $O((|V| + |E|)|E| \log C)$ e, assumendo che la rete sia connessa, $O(|E|^2 \log C)$. □

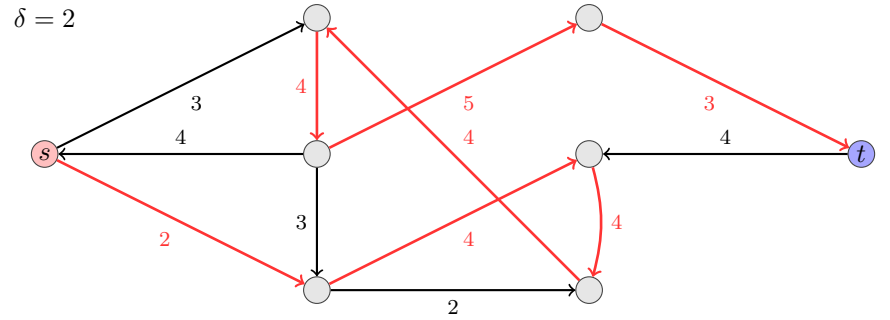
Di seguito, viene illustrato l'effetto sulla rete di Figura 1.4 dell'esecuzione dell'algoritmo basato sui cammini δ-aumentanti. In rosso è mostrato, di volta in volta, il cammino δ-aumentante selezionato, sulla rete ottenuta eliminando tutti gli archi di capacità inferiore a δ.

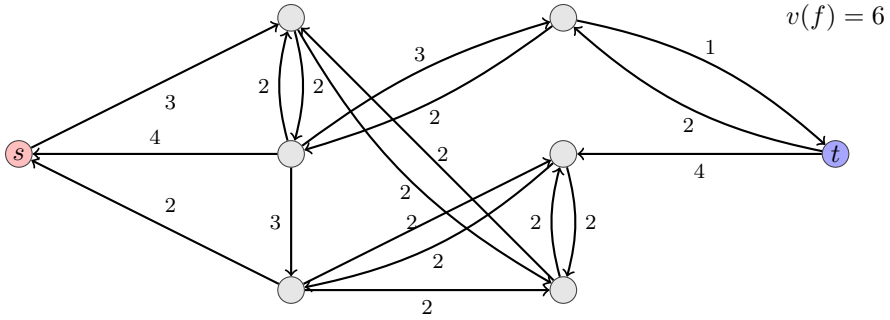


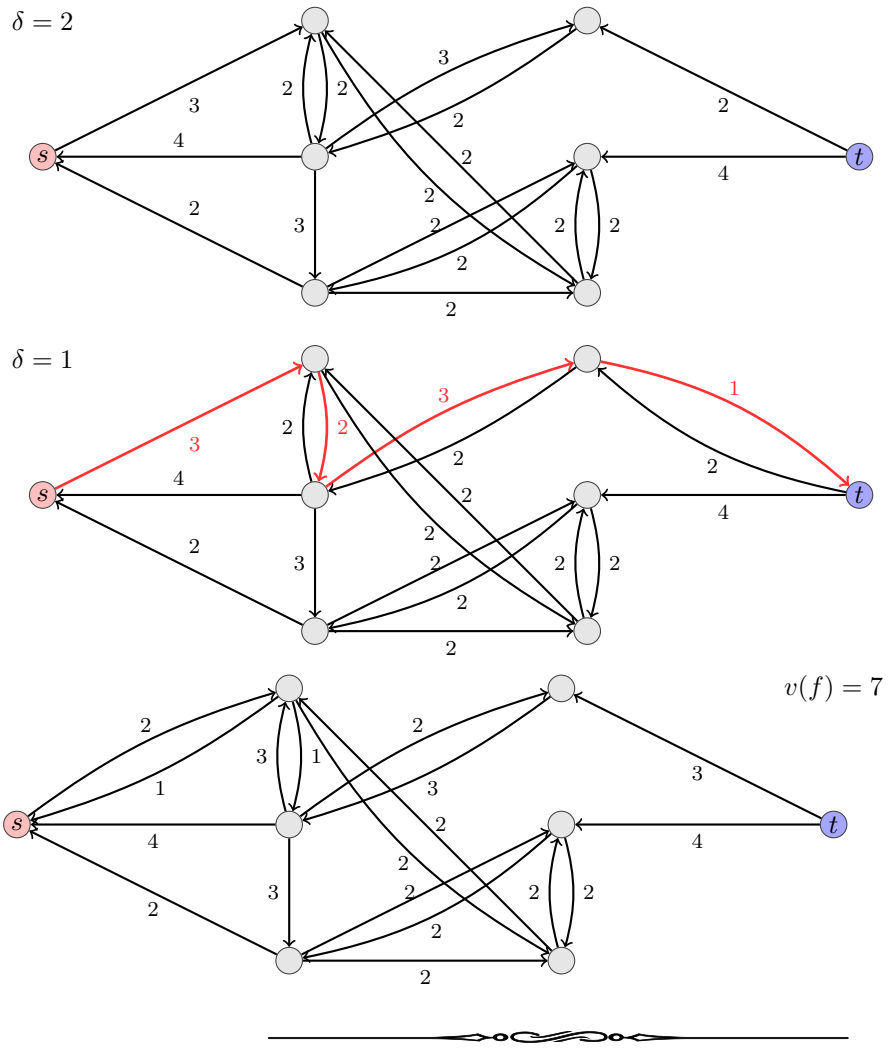

$\delta = 4$
s
t
4
5
4
4
4
4
5


$v(f) = 4$
s
t
1
3
4
5
3
4
4
4
3
1
4
2
4
2


$\delta = 4$
s
t
4
5
4
4
4
4
4


$\delta = 2$
s
t
4
3
5
4
3
4
4
2
3
4
2


$v(f) = 6$
s
t
3
2
2
3
2
1
2
4
2
2
2
2
2
2
2
2
2



Cammino di lunghezza minima

Si può osservare che sia la complessità sia dell'Algoritmo 1.3 che quella dell'Algoritmo 1.6 dipendono dai valori numerici (le capacità degli archi) in input.

Possiamo però eliminare questa dipendenza, ottenendo un algoritmo fortemente polinomiale (strongly polynomial)[1] attraverso una semplice variante dell'algoritmo di Ford e Fulkerson (1.2), detta Algoritmo di Edmonds e Karp, in cui ad ogni iterazione, il cammino aumentante in N_f è ricercato mediante una visita in ampiezza. Quindi, l'algoritmo, riportato come Algoritmo 1.7, seleziona, ad ogni iterazione, il cammino aumentante che attraversa il minimo numero di archi.

Come si può vedere, l'algoritmo ha la struttura dell'Algoritmo 1.2, così come dell'Algoritmo 1.3: l'unica differenza consiste nelle modalità di scelta del prossimo cammino aumentante da considerare.

Essendo una specifica implementazione dell'algoritmo di Ford e Fulkerson, sappiamo che l'Algoritmo 1.7 fornisce il flusso massimo in N. Per quanto riguarda la sua complessità computazionale, vale il seguente teorema.

Teorema 1.12 La lunghezza del cammino minimo da s a t in N_f ha un andamento monotono non decrescente al susseguirsi delle iterazioni. Inoltre, non si possono avere più di $|E|$ iterazioni (necessariamente consecutive) in cui il valore di tale grandezza rimane immutato.

[1] Un algoritmo strongly polynomial ha complessità che non dipende dai valori numerici in input, assumendo che una operazione aritmetica richieda tempo costante.

Algorithm 1.7: Algoritmo di cammino aumentante di lunghezza minima

Input: $G = (V, E)$, $s, t \in V$, $c : E \mapsto \mathbb{R}^+$
Output: Flusso $f : E \mapsto \mathbb{R}^+$ da s a t

```
1 foreach e ∈ E do f(e) ← 0;
2 Deriva N_f;
3 while esiste un cammino aumentante da s a t in N_f do
4     Prendi il cammino aumentante P tale che |P| è minima;
5     foreach (u, v) ∈ P do
6         if (u, v) ∈ E then f(e) ← f(e) + min_{e∈P} c_f(e);
7         else f(e) ← f(e) − min_{e∈P} c_f(e)
8     Aggiorna N_f
9 return f
```

Dimostrazione: Assumiamo che, dopo T iterazioni, il cammino minimo nella rete residua N_f abbia lunghezza l. Effettuando una visita in ampiezza a partire da s, otteniamo una partizione di $V - \{s\}$ in l sottoinsiemi $V_1, V_2, \ldots, V_l$, dove V_i è l'insieme dei nodi a distanza i da s.

Si può osservare che, per le proprietà della visita in ampiezza, per ogni arco (u, v), se $u \in V_i$ e $V \in V_j$, allora necessariamente $j \leq i + 1$: chiamiamo arco forward un arco tale che $j = i + 1$, un arco quindi che fa incrementare la distanza da s.

Un cammino minimo da s a t deve evidentemente essere composto da soli archi forward (Figura 1.18) o, in modo equivalente, un cammino in cui compare un arco non forward non è minimo.

Nel corso del $T + 1$-esima iterazione un cammino di lunghezza l viene selezionato in N_f, e il flusso su tutti i suoi archi viene incrementato il massimo possibile, vale a dire della capacità residua minima lungo il cammino. Di conseguenza almeno un arco nel cammino viene ad avere capacità residua nulla e quindi non compare nella rete residua risultante. Inoltre, per definizione di N_f, per ogni arco (u, v) del cammino, che ha capacità residua $c(u, v) - f(u, v)$ esiste l'arco opposto (v, u), con capacità residua $f(u, v)$, se $f(u, v) > 0$.

Per ogni arco nella nuova rete residua continua ad essere verificata la proprietà di incrementare la distanza da s di al più 1, in quanto i nuovi archi che sono eventualmente stati introdotti sono necessariamente archi opposti a quelli del cammino: di conseguenza, la distanza da s a t non è certamente diminuita.

Osserviamo ora che, nel caso in cui la distanza sia immutata, il nuovo cammino minimo avrà la stessa lunghezza del cammino minimo precedente, per cui userà soltanto archi forward già presenti nella rete residua precedente (Figura 1.19).

Quindi, per tutte le iterazioni successive alla T-esima per le quali la distanza da s a t rimane immutata e pari ad l, esiste un cammino aumentante di lunghezza minima che era già presente al momento dell'iterazione T.

Dato che ad ogni iterazione esiste almeno un arco che ha assegnato un flusso che satura la sua capacità, e che quindi non può più far parte di cammini di lunghezza l, ne deriva che il numero di iterazioni possibili su reti residue di distanza l da s a t è al più pari a $|E|$. □

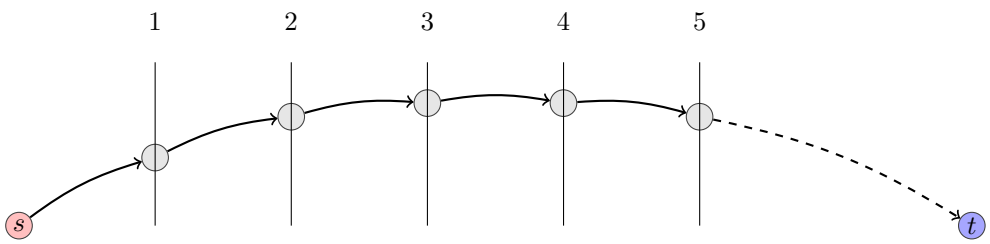


Figura 1.18: Cammino di lunghezza minima

La distanza da s a t è al più $|V| - 1$ quindi, per il Teorema 1.12, l'Algoritmo 1.7 esegue al più $|E|(|V| - 1)$ iterazioni, nell'ambito di ognuna delle quali viene eseguita una visita in ampiezza del grafo (in tempo $O(|V| + |E|)$).

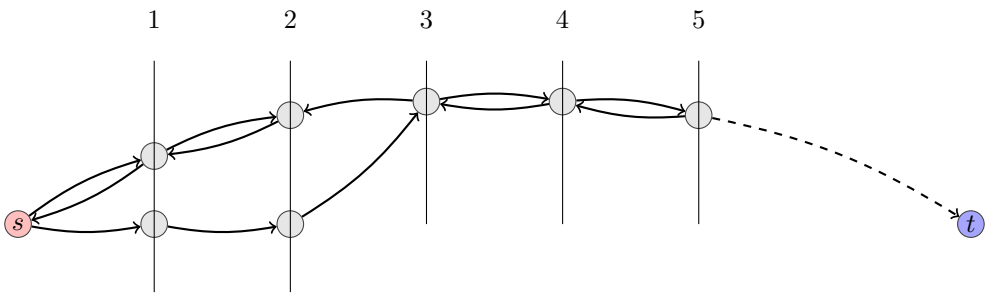
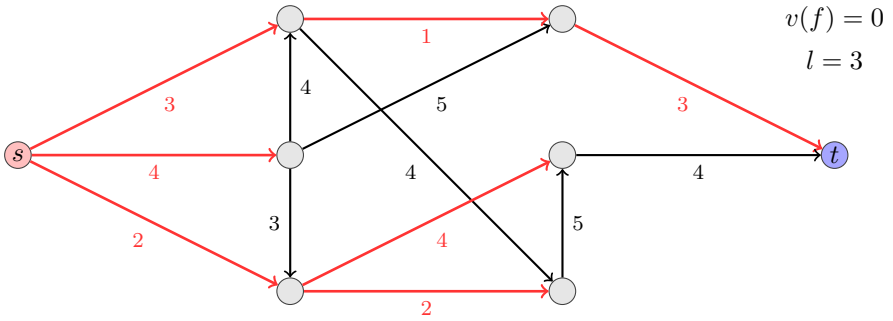
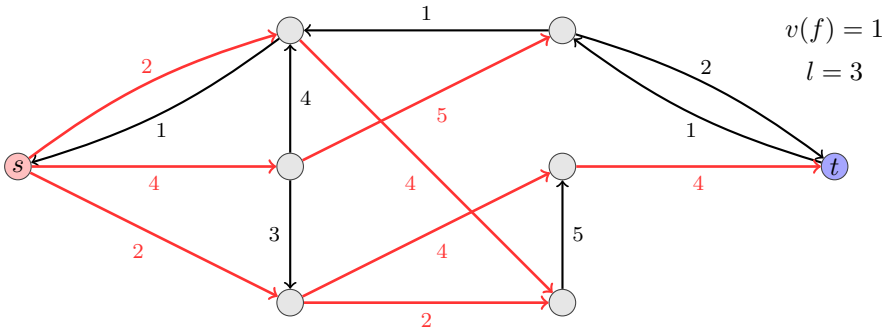


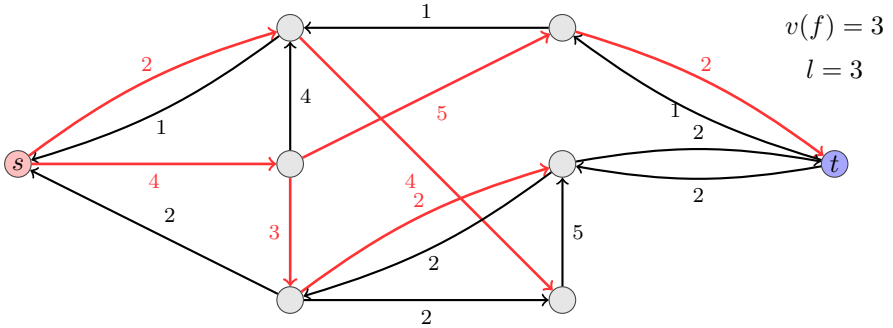
Figura 1.19: Caso in cui l'eliminazione del cammino minimo non aumenta la distanza

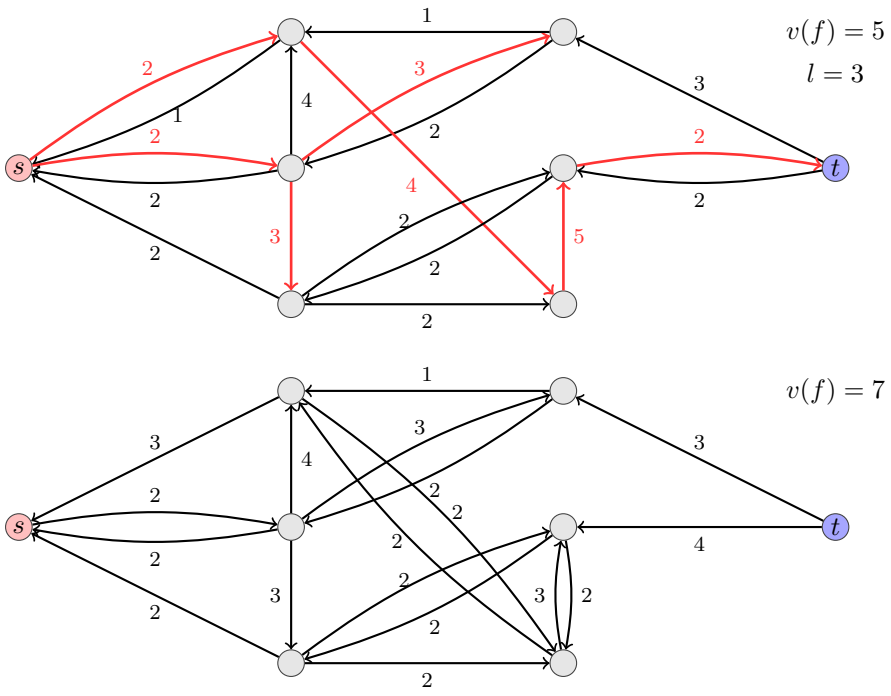
Di conseguenza, la complessità totale è $O(|E||V|(|E| + |V|))$ e quindi $O(|E|^2|V|)$ assumendo nuovamente che la rete sia connessa.

Di seguito, viene illustrato l'effetto sulla rete di Figura 1.4 dell'esecuzione dell'algoritmo basato sulla ricerca del cammino di lunghezza minima. In rosso è mostrato, di volta in volta, l'albero BFS individuato.









Applicazioni del massimo flusso

Car sharing

Supponiamo che n studenti decidano, per un periodo di m giorni, di utilizzare a turno le proprie automobili per andare a lezione. Supponiamo anche ogni auto sia sufficientemente capiente da poter trasportare tutti gli studenti.

Non tutti gli studenti devono andare a lezione negli stessi giorni: in altri termini, esiste una funzione booleana che associa ad una coppia studente-giorno il valore true se lo studente deve andare a lezione quel giorno e il valore false altrimenti.

Un esempio molto ridotto di tale problema, con $n = 4$ e $m = 5$ è riportato nella tabella sottostante, in cui per ogni giorno da Lunedì a Venerdì viene riportato se lo studente i-esimo (assumiamo $i = 1, \ldots, 4$) deve andare a lezione.

	Lu	Ma	Me	Gi	Ve
1	X	X	X		
2		X		X	X
3	X				X
4	X		X	X	

Vogliamo trovare una assegnazione ``equa'' dell'utilizzo delle automobili, nel senso intuitivo che chi deve andare più spesso a lezione dovrà anche mettere più spesso a disposizione la propria automobile.

Possiamo assegnare ad ogni studente un valore che misura la sua quota di utilizzo del car sharing nel modo seguente:

1. per ogni giorno, se k sono i viaggiatori, ognuno di essi riceve un valore pari a $1/k$
2. il valore totale T_i assegnato ad uno studente è pari alla somma dei valori ricevuti ogni gionro

La tabella seguente mostra i valori nell'esempio considerato, l'ultima colonna specifica la quota di utilizzo per ogni viaggiatore.

	Lu	Ma	Me	Gi	Ve	T
1	$1/3$	$1/2$	$1/2$			$4/3$
2		$1/2$		$1/2$	$1/2$	$3/2$
3	$1/3$				$1/2$	$5/6$
4	$1/3$		$1/2$	$1/2$		$4/3$

L'assegnazione che cerchiamo dovrà far s√¨ che il viaggiatore i non usi la propria automobile per più di $\lceil T_i \rceil$ giorni, sugli m complessivi.

Modelliamo il problema sotto forma di flusso su rete nel modo seguente:

- per l'insieme dei nodi si ha $V = \{s, r, t\} \cup \{v_i, i = 1, \ldots, n\} \cup \{d_j, j = 1, \ldots, m\}$
- per l'insieme degli archi si ha che E include
  1. (s, r) di capacità m
  2. per $i = 1, \ldots, m$, (r, t_i) di capacità $\lceil T_i \rceil$
  3. per $i = 1, \ldots, m$, $j = 1, \ldots, n$ (t_i, d_j) di capacità 1
  4. per $j = 1, \ldots, n$, (d_j, t) di capacità 1

La rete relativa all'esempio considerato è riportata in Figura 1.20. Gli archi senza valori associati hanno capacità 1.

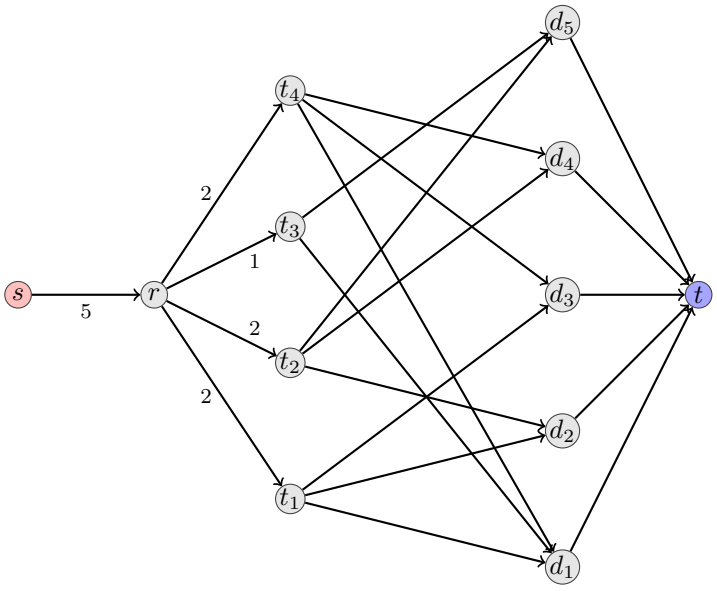


Figura 1.20: Rete per l'esempio di car sharing

Dato che le capacità degli archi della rete sono intere, il flusso massimo sarà intero. Inoltre, per ogni flusso ammissibile:

- per ogni giorno, dato che in uscita da d_i c'è capacità totale 1, al più un arco da un qualche viaggiatore t_j avrà flusso non negativo; tale flusso sarà al più 1 (la capacità di (t_j, d_i)) e quindi, per il flusso massimo, esattamente 1. Quindi, nella assegnazione corrispondente al flusso massimo, per ogni giorno c'è al più un viaggiatore associato (quello che utilizza la sua auto)
- per ogni viaggiatore t_i, si potranno avere, nel flusso massimo, al più tanti archi uscenti con flusso 1 quanta è la capacità dell'arco (d, t_i). Quindi, nella assegnazione corrispondente al flusso massimo, ogni viaggiatore utilizza la sua auto per un numero di giorni non superiore al sua quota di utilizzo del car sharing.

In definitiva, il flusso ottimo della rete fornisce una assegnazione di guidatori a giorni che soddisfa in parte le condizioni poste (un viaggiatore non guida per più della sua quota, per ogni giorno non ci sono più di un guidatore): dobbiamo però ancora mostrare che per ogni giorno c'è esattamente un guidatore assegnato. Ciò è equivalente a verificare che $v(f^*) = m$: dal Teorema 1.4, $v(f^*)$ è pari alla capacità del taglio minimo che separa s da t ed è facile rendersi conto che i tagli minimi nella rete, con capacità m, sono quello che taglia l'arco (s, r) e quello che taglia tutti gli archi (d_i, t).

Flusso ammissibile

In questo problema, consideriamo:

1. un grafo $G = (N, E)$ con archi di capacità $c : E \mapsto \mathbb{Z}$
2. una funzione $b : N \mapsto \mathbb{Z}$, che possiamo interpretare come offerta ($b(i) > 0$) o domanda ($b(i) < 0$) di una determinata merce, e per la quale assumiamo $\sum_{v \in N} b(v) = 0$, l'equilibrio complessivo quindi tra domanda e offerta.

Ci chiediamo se esiste una modalità di trasferimento della merce attraverso gli archi del grafo che soddisfi per ogni nodo il limite posto dalla sua capacità e inoltre faccia s√¨ che avvenga un trasferimento perfetto della merce, che soddisfi tutta la domanda. In termini più precisi, ci chiediamo se esiste un flusso f tale che

- per ogni nodo $v \in N$,

$$\sum_{e \in \mathcal{O}(v)} f(e) - \sum_{e \in \mathcal{I}(v)} f(e) = b(v)$$

- per ogni $e \in E$, $0 \leq f(e) \leq c(e)$

In Figura 1.4 viene mostrata una semplice istanza di esempio di questo problema: i valori $b(v)$ sono mostrati all'interno dei nodi corrispondenti. Si noti che domanda e offerta complessive sono uguali in modulo e pari a 9,

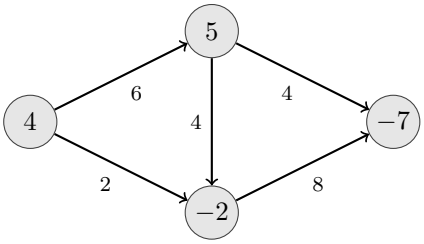


Figura 1.21: Esempio di istanza di feasible flow

Il problema può essere risolto in modo semplice attraverso la ricerca di un flusso massimo su una rete ottenuta da G introducendo i due nodi s e t e, per ogni nodo v, se $b(v) > 0$ inserendo un arco (s, v) con $c(s, v) = b(v)$, se $b(v) < 0$ inserendo un arco (v, t) con $c(v, t) = -b(v)$. La rete derivata da quella in Figura 1.4 è mostrata in Figura 1.22.

Possiamo osservare che:

- se esiste un flusso ammissibile f sul grafo, allora lo stesso flusso esteso ponendo $f(s, v) = b(v)$ per gli archi da s e $f(v, t) = -b(v)$ per gli archi verso t, è un flusso massimo, in quanto satura il taglio composto dal solo nodo s (e anche quello comprendente tutti i nodi eccetto t).
- se il flusso massimo nella rete satura tutti i nodi nel taglio comprendente il solo nodo s (e necessariamente anche quello comprendente tutti i nodi eccetto t), lo stesso flusso, applicato ai soli nodi in N soddisfa le condizioni di flusso ammissibile.

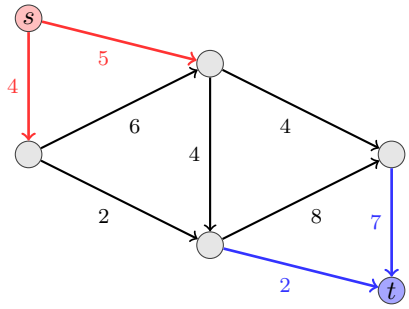


Figura 1.22: Istanza di max flow derivata

Quindi, per determinare se esiste un flusso ammissibile nel grafo G, è sufficiente verificare se il flusso massimo nella rete derivata satura il taglio composto dal solo nodo s: equivalentemente, potremmo dire che esiste un flusso ammissibile in G se e solo se il taglio composto dal solo nodo s è un taglio minimo che separa s da t. È facile verificare che il massimo flusso in Figura 1.22 ha valore pari a 9, e quindi satura il taglio, dal che consegue che nell'esempio in Figura 1.21 esiste un flusso ammissibile.

Al contrario, l'istanza in Figura 1.23 non ammette un flusso ammissibile in quanto la rete derivata, mostrata in Figura 1.21, ha valore del massimo flusso minore di 9, mentre esiste un taglio, mostrato in figura, che separa s da t e ha capacità pari a 8.

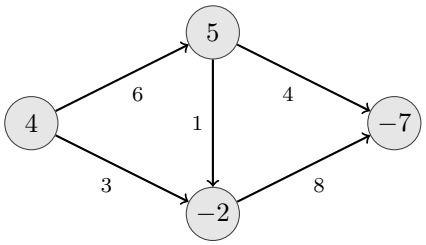


Figura 1.23: Esempio di istanza negativa di feasible flow

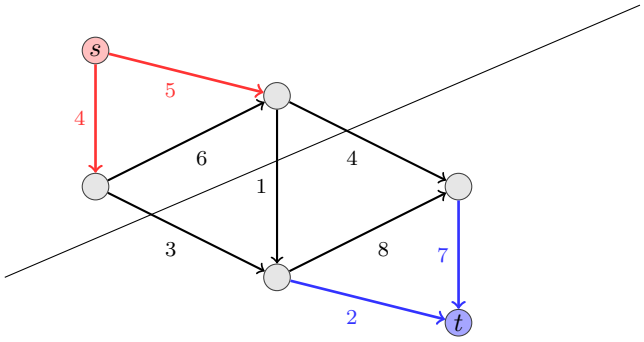


Figura 1.24: Istanza di max flow derivata da istanza negativa

Il problema dei rappresentanti

Una città ha r residenti $R_1, \ldots, R_r$, q associazioni $C_1, \ldots, C_q$ e p partiti politici $P_1, \ldots, P_p$. Ogni residente è membro di almeno un'associazione e di esattamente un partito. Ogni associazione deve nominare uno dei suoi membri come proprio rappresentante nel consiglio della città, in modo che, per ogni partito P_k, il numero di rappresentanti non sia superiore a un valore predefinito $u_k > 0$. Esiste una scelta dei rappresentanti possibile?

Data una istanza del problema, è possibile costruire una rete in cui $N = \{R_1, \ldots, R_r, C_1, \ldots, C_q, P_1, \ldots, P_p, s, t\}$ e l'insieme degli archi è definito come segue

- $(s, C_k) \in E$ per $k = 1, \ldots, q$, con $c = 1$
- $(C_i, R_j) \in E$ se R_j appartiene a C_i, con $c = 1$
- $(R_i, P_j) \in E$ se R_i appartiene a P_j, con $c = 1$
- $(P_k, t) \in E$ per $k = 1, \ldots, p$, con $c = u_k$

Ad esempio, consideriamo il caso in cui $r = 7$, $q = 4$, $p = 3$ e valgono le seguenti appartenenze, ad associazioni e partiti:

- $C_1 = \{R_1, R_2\}$, $C_2 = \{R_2, R_3, R_4\}$, $C_3 = \{R_4, R_5\}$, $C_4 = \{R_4, R_5, R_6, R_7\}$
- $P_1 = \{R_1, R_2\}$, $P_2 = \{R_3, R_4\}$, $P_3 = \{R_5, R_6, R_7\}$

e assumiamo $u_1 = 1$, $u_2 = 2$, $u_3 = 2$. Ne deriva la rete in Figura 1.25, in cui archi non etichettati vanno considerati di capacità unitaria.

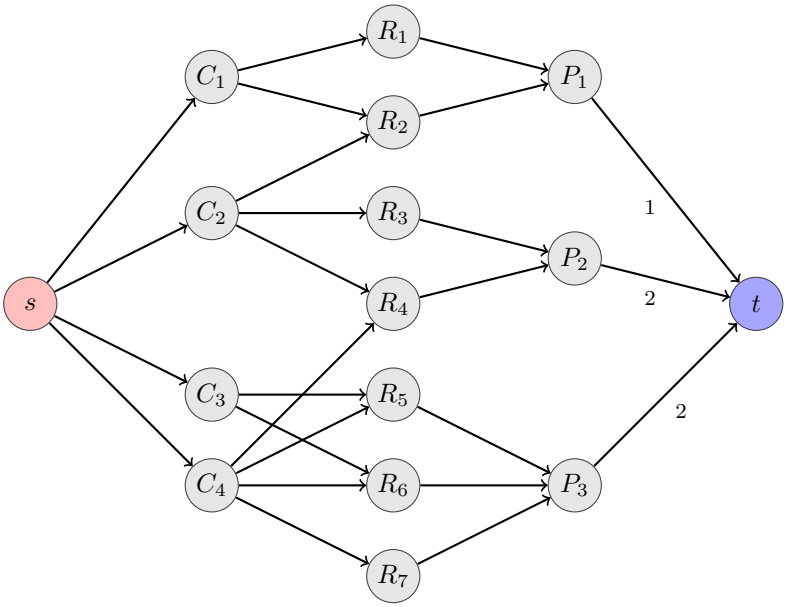


Figura 1.25: Rete derivata da istanza del problema dei rappresentanti

Possiamo mostrare che una scelta dei rappresentanti che soddisfi i vincoli è possibile se e solo se nella rete esiste un flusso di valore pari a q (si osservi che non può esistere un flusso di valore maggiore, per il taglio $\{s\}$, avente capacità q).

- se esiste una scelta, allora per ogni C_i esiste esattamente un residente R_j appartenente ad essa scelto: poniamo $f = 1$ per gli archi (s, C_i), (C_i, R_j), (R_j, P_k), dove P_k è il partito di appartenenza di R_j. Si noti che, in tal modo, gli archi in uscita da s hanno tutti flusso pari a 1, per ogni C_i esiste uno ed un solo arco uscente di flusso pari a 1, un R_j ha flusso entrante e uscente pari a 1 se e solo se è un residente scelto, il flusso entrante in P_k è pari al numero di residenti scelti appartenenti a quel partito, che per costruzione è inferiore alla capacità dell'arco (P_k, t). Ne deriva che il flusso è ammissibile e di valore pari a q.
- se esiste un flusso di valore pari a q, allora per ogni C_i c'è uno e un solo arco uscente (C_i, R_j) di flusso pari a 1. R_j è allora un residente scelto. Per il bilanciamento del flusso a R_j, l'unico arco uscente (R_j, P_k) ha

flusso pari a 1: quindi, per ogni P_k, il flusso entrante e uscente è pari al numero di residenti scelti appartenenti al k-esimo partito. Dato che il flusso sull'arco (P_k, t) deve essere non superiore alla sua capacità, ne deriva che il vincolo sul massimo numero di scelti da ogni partito è verificato. Quindi, in definitiva, esiste al meno un residente scelto per associazione (per l'ipotesi che il flusso sia pari a q), esiste al più un residente scelto per associazione (per il vincolo sulla capacità degli archi uscenti da s), ogni partito P_k è rappresentato da al più u_k appartenenti (dal vincolo sulla capacità degli archi entranti in t).

Problemi di matching

Un accoppiamento (o matching) in un grafo è un insieme di coppie di nodi tale che ogni nodo compare in al più una coppia. Più formalmente, dato un grafo non orientato $G = (V, E)$ un accoppiamento è un sottoinsieme degli archi $M \subseteq E$ tale che ogni nodo appare in al più un elemento di M. Se ogni nodo compare in esattamente un arco di M (e quindi tutti i nodi sono accoppiati) il matching è detto perfetto.

Il problema del massimo accoppiamento chiede, dato un grafo $G = (V, E)$, di trovare un matching di dimensione massima. In questa sede, consideriamo questo problema per una classe di grafi particolari: i grafi bipartiti.

Un grafo non orientato $G = (V, E)$ è detto bipartito se esiste una partizione (U, W) di V, per cui quindi $V = U \cup W$ e $U \cap W2 = \emptyset$, tale che per ogni $(u, v) \in E$ si ha $u \in U$ e $v \in W$ (o viceversa). Un esempio di grafo di questo tipo è dato in Figura 1.26.

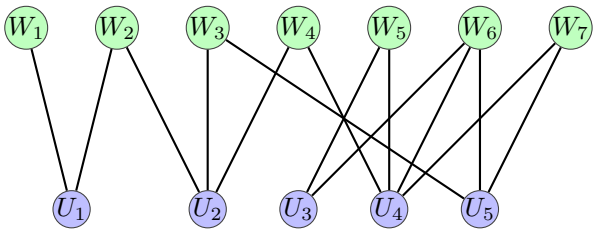


Figura 1.26: Esempio di grafo bipartito

Nel calcolo del matching massimo su un grafo bipartito vogliamo quindi selezionare il più grande insieme di coppie (u_i, w_i) tali che $u_i \in U, w_i \in W$ e $u_i \neq u_j$, $w_i \neq w_j$, per $i \neq j$. Un esempio di matching massimo sul grafo bipartito precedente è mostrato in Figura 1.27.

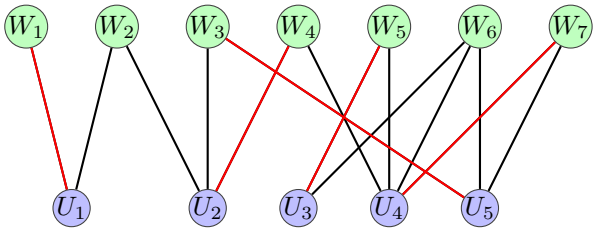


Figura 1.27: Esempio di matching massimo

Si noti che il problema del matching perfetto su un grafo bipartito può essere facilmente risolto se si sa risolvere il problema del matching massimo, verificando che $|U| = |W|$ e che il matching massimo includa $|U|$ elementi.

Il problema del matching massimo in un grafo bipartito può essere trovato mediante riduzione al problema del massimo flusso. Il concetto di riduzione è molto importante nell'ambito della teoria degli algoritmi e della teoria della complessità: sostanzialmente, esso ci descrive la possibilità di risolvere un problema P_1 se si sa risolvere un diverso problema P_2.

Una riduzione da P_1 a P_2 opera nel modo seguente. Assumiamo di avere una istanza x di P_1, allora:

1. x è trasformata in una opportuna istanza y di P_2

2. y è risolta per mezzo di un algoritmo per P_2, fornendo una soluzione $s(y)$
3. la soluzione $s(x)$ di x è derivata da $s(y)$

Nel caso che si sta considerando, un grafo bipartito, istanza del problema del matching massimo in un grafo bipartito, viene trasformato in una rete, istanza del problema del massimo flusso nel modo seguente: il grafo della rete N ha insieme dei nodi $V = U \cup W \cup \{s, t\}$ e insieme degli archi (orientati) $E = \{(u, w), u \in U, w \in W\} \cup \{(s, u), u \in U\} \cup \{(w, t), w \in W\}$. La capacità degli archi di N è posta uniformemente pari a 1.

La rete derivata dal grafo di esempio è data in Figura 1.28

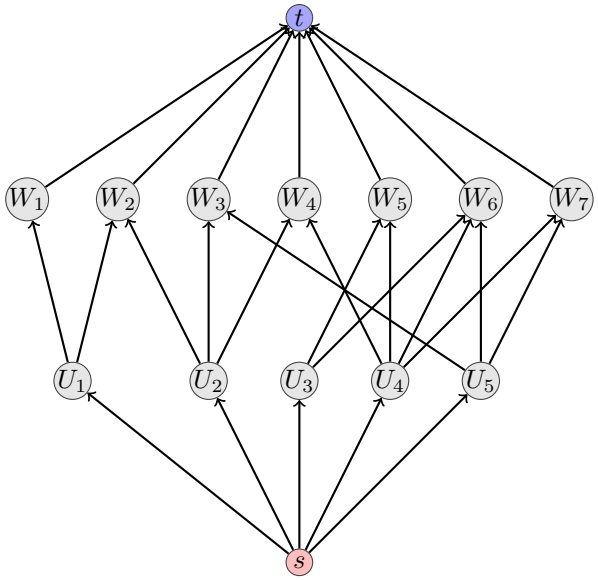


Figura 1.28: Istanza di maxflow derivata

L'applicazione di un algoritmo di maxflow sulla rete N fornirà un flusso massimo nel quale, necessariamente dato che le capacità sono tutte unitarie, il flusso su ogni arco è 0 o 1.

Mostriamo che il matching massimo è dato dall'insieme delle coppie $(u, w), u \in U, w \in W$ tali che $f(u, v) = 1$. A tal fine, mostriamo (1) che l'algoritmo fornisce un matching e (2) che il matching fornito è ottimo.

1. L'algoritmo fornisce un matching di dimensione pari al flusso massimo in N. Dato il flusso massimo f trovato, infatti, si ha che per ogni nodo $u \in U$

$$\sum_{(u,w)\in E} f(u, w) = f(s, u) \leq 1$$

quindi, al più uno degli archi tra u e i nodi di W ha flusso pari a 1 (ed è quindi incluso nel matching). Allo stesso modo, per ogni nodo $w \in W$

$$\sum_{(u,w)\in E} f(u, w) = f(w, t) \leq 1$$

quindi, al più uno degli archi tra w e i nodi di U ha flusso pari a 1 (ed è quindi incluso nel matching).

2. Non esistono matching più grandi di quello corrispondente al massimo flusso in N. Consideriamo infatti il matching massimo M^* e definiamo a partire da esso un flusso in N ponendo, per ogni $(u, w) \in M^*$, $f(s, u) = f(u, w) = f(w, t) = 1$ e ponendo a zero il flusso su tutti gli archi rimanenti. Il flusso è chiaramente ammissibile, in quanto soddisfa i vincoli sui nodi e sulle capacità, e ha valore pari al numero di archi nel matching, e quindi a M^*. Quindi, esiste un flusso ammissibile corrispondente al matching massimo, per cui non esistono matching più grandi del valore di tale flusso e, a maggior ragione, del valore del flusso massimo.

Per quanto riguarda la complessità dell'algoritmo, la costruzione di N richiede tempo $O(|E|)$, così come la derivazione del matching massimo dal massimo flusso. A tali costi va aggiunta la complessità di calcolo del maxflow su N: dato che il flusso massimo su N è certamente limitato superiormente da $\max(|U|, |W|)$, l'algoritmo di Ford e Fulkerson ha complessità $O(|E| \cdot \max(|U|, |W|)) = O(|E||V|)$.

Arrotondamento di valori in una matrice

Sia data una matrice $D^{p \times q}$ di reali d_{ij}, e siano specificati $p + q$ valori, corrispondenti a somme α_i sulle righe e β_j sulle colonne.

Vogliamo arrotondare ognuno dei d_{ij}, degli α_i e dei β_j ad un valore intero (quello immediatamente minore o quello immediatamente maggiore) in modo consistente, facendo sì che le somme sulle righe e sulle colonne rimangano verificate anche con i nuovi valori.

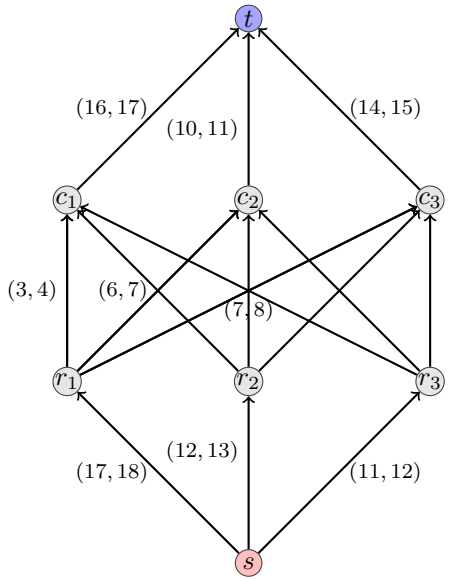
Possiamo rappresentare questo problema come un problema di massimo flusso in cui sono presenti dei vincoli sia superiori (capacità degli archi) che inferiori (minimo flusso) sui flussi negli archi. È possibile in generale (anche se non lo verifichiamo in questa sede) risolvere un problema con tali vincoli aggiuntivi riducendolo ad un problema di massimo flusso con soli vincoli superiori su una rete derivata in modo opportuno.

Data la matrice D, la rete corrispondente ha $p + q$ nodi $r_i, i = 1, \ldots, p$ e $c_j, j = 1, \ldots, q$, corrispondenti alle righe e alle colonne. Per quanto riguarda gli archi, per ogni coppia r_i, c_j esiste l'arco corrispondente, con capacità $c(r_i, c_j) = \lceil d_{ij} \rceil$ e flusso minimo $m(r_i, c_j) = \lfloor d_{ij} \rfloor$, inoltre per ogni r_i esiste l'arco (s, r_i) con capacità $c(s, r_i) = \lceil \alpha_i \rceil$ e flusso minimo $m(r_i, c_j) = \lfloor \alpha_i \rfloor$ e per ogni colonna c_j esiste l'arco (c_i, t) con capacità $c(c_j, t) = \lceil \beta_j \rceil$ e flusso minimo $m(c_j, t) = \lfloor \beta_j \rfloor$.

Data ad esempio la matrice

3.1	6.8	7.3	17.2
9.6	2.4	0.7	12.7
3.6	1.2	6.5	11.3
16.3	10.4	14.5	

La rete corrispondente è la seguente, in cui, per semplicità, sono indicati i valori associati ai soli archi da s e r_1 e verso t.



Non è difficile verificare che un flusso ammissibile nella rete (che necessariamente è intero) corrisponde a un arrotondamento consistente per D.

Selezione di progetti

Consideriamo un insieme P di progetti da intraprendere. Il progetto i, se attivato garantisce un profitto (differenza tra ritorno e investimento) associato p_i che può essere positivo, nullo o negativo. Inoltre, su P è definita una relazione di dipendenza, che specifica, per ogni progetto i, l'insieme π_i di altri progetti che devono essere necessariamente attivati per attivare i. Questa relazione può essere modellata per mezzo di un grafo orientato aciclico $G = (P, E)$.

Un insieme di progetti $A \subseteq P$ è ammissibile se per ogni progetto $i \in A$ si ha che ogni progetto in π_i è anch'esso incluso in A. Il profitto di A è dato da $\sum_{i \in A} p_i$.

Il problema che consideriamo è quello di determinare un insieme ammissibile di progetti di profitto massimo.

Possiamo costruire una istanza di maxflow nel modo seguente:

- l'insieme dei nodi è $P \cup \{s, t\}$
- l'insieme degli archi corrisponde alle relazioni di dipendenza tra progetti: in particolare, esiste l'arco (i, j) se i dipende da j. Inoltre, per ogni progetto i con $p_i > 0$ aggiungiamo un arco (s, i) e, per ogni progetto j con $p_j < 0$ aggiungiamo un arco (j, t)
- le capacità degli archi sono definite pari a p_i per gli archi (s, i) e pari a $-p_j$ per gli archi (j, t); tutti gli altri archi (quelli di E) hanno capacità $C + 1$, dove

  $$C = \sum_{i \in P; p_i > 0} p_i$$

  è la capacità totale degli archi da s che, necessariamente, è non superiore al taglio minimo e quindi al flusso massimo nella rete. Di conseguenza ogni arco in E ha capacità maggiore del massimo flusso, per cui non pone un vincolo effettivo sul flusso che lo attraversa.

Mostriamo ora che il minimo taglio nella rete corrisponde all'insieme ottimo di progetti.

1. Sia A un insieme ammissibile di progetti e consideriamo il taglio tra $A' = A \cup \{s\}$ e $B' = (P - A) \cup \{t\}$. Dato che A è ammissibile, non esistono archi da A a $P - A$. Tutti gli archi da A' a B' sono o uscenti da s o entranti in t. Per i primi abbiamo capacità complessiva

   $$\sum_{i \notin A; p_i > 0} p_i = C - \sum_{i \in A; p_i > 0}$$

   mentre per i secondi abbiamo

   $$\sum_{i \in A; p_i < 0} -p_i$$

   La capacità del taglio è data dalla somma di questi due termini

   $$C - \sum_{i \in A; p_i > 0} + \sum_{i \in A; p_i < 0} -p_i = C - \sum_{i \in A} p_i$$

2. Inoltre, dato un taglio $A' = A \cup \{s\}$ di capacità non superiore a C, tale taglio non può essere attraversato da archi corrispondenti a dipendenze, in quanto ognuno di tali archi a capacità maggiore di C. Di conseguenza, l'insieme di progetti A è ammissibile.

3. Quindi, esiste una corrispondenza biunivoca i tagli di capacità al più C e gli insiemi ammissibili di progetti. Inoltre, il profitto di un insieme ammissibile di progetti A è pari a $C - c(A')$, dove $c(A')$ è la capacità del taglio $A' = A \cup \{s\}$. Quindi, l'insieme ottimo di progetti corrisponde al taglio di capacità minima.

Cammini disgiunti sugli archi

Dato un grafo orientato $G = (V, E)$ e due nodi $s, t \in V$, ci chiediamo quale sia il massimo numero di cammini da s a t che siano disgiunti sugli archi, intendendo con ciò che non esiste alcun arco che compare in più di un cammino. Si noti che due cammini disgiunti sugli archi possono però attraversare uno stesso nodo.

Questo problema può essere risolto facilmente considerando una istanza di maxflow, derivata da G, in cui la capacità degli archi è unitaria. Infatti il flusso massimo nella rete risultante sarà necessariamente decomponibile nell'unione di un insieme di cammini da s a t e in un insieme di cicli. Per l'ipotesi sulla capacità, il flusso su tali cicli e cammini è necessariamente unitario, dal che deriva che non esistono archi appartenenti a più di un cammino (o ciclo), per cui i cammini e i cicli in questione sono disgiunti sugli archi. A maggior ragione lo sono quindi i soli cammini: infine, dato che il flusso da s a t è massimo e ogni cammino contribuisce per 1, ne consegue che è massimo il numero di cammini.

Al contrario, ogni insieme massimo di cammini disgiunti sugli da s a t fornisce chiaramente un flusso ottimo sulla rete, costituito assegnando flusso unitario agli archi dei cammini.

Cammini disgiunti sui nodi

Dato un grafo orientato $G = (V, E)$ e due nodi $s, t \in V$, ci chiediamo quale sia il massimo numero di cammini da s a t che siano disgiunti sui nodi, intendendo con ciò che non esiste alcun nodo che compare in più di un cammino.

Questo problema può essere risolto applicando lo stesso approccio precedente, considerando però il grafo $G' = (V', E')$ derivato da G introducendo, per ogni nodo $v \in V$, due nodi $v_{in}, v_{out} \in V'$. Ogni arco $(w, u) \in E$ è sostituito da un arco $(w, v_{in}) \in E'$ e ogni arco $(v, w) \in E$ è sostituito da un arco $(v_{out}, w) \in E'$. Si noti che un cammino è disgiunto sui nodi in G se e solo se è lo è, sugli archi, in G'.