Progettazione di algoritmi

RETI DI FLUSSO 1

- Reti di flusso
- Il problema della ricerca del flusso massimo
- L’algoritmo di Ford-Fulkerson

- I grafi possono essere naturalmente utilizzati per modellare *reti di trasporto*, eccone alcuni esempi:

  – Reti idriche: gli archi rappresentano condutture attraverso cui fluisce acqua o altri liquidi e i nodi rappresentano giunzioni tra le diverse condutture.

  – Reti di calcolatori: gli archi rappresentano connessioni di rete su cui viaggiano pacchetti di dati e i nodi rappresentano switches.

  – Reti autostradali: gli archi rappresentano strade su cui viaggiano autoveicoli e i nodi rappresentano svincoli.

  tutte queste reti hanno in comune certe caratteristiche:

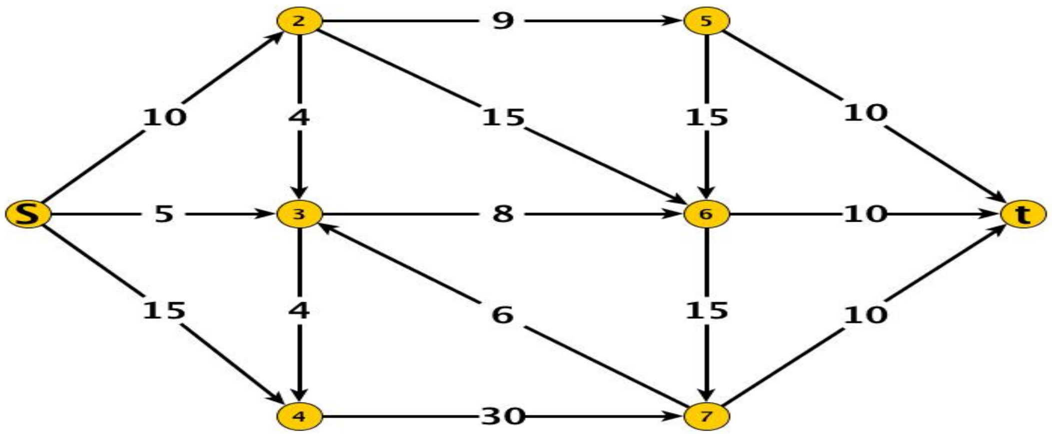
  – capacità associate agli archi: ad indicare quanto ciascun arco può trasportare

  – nodi sorgente: vale a dire nodi che generano traffico

  – nodi destinazione: vale a dire nodi che assorbono traffico

  – il flusso: il traffico che transita sugli archi.

- Formalmente, un *grafo di flusso* è un grafo diretto e pesato con le seguenti caratteristiche:
  - in G vi è un solo nodo senza archi entranti, lo indicheremo con s, ed è chiamato sorgente.
  - in G vi è un solo nodo senza archi uscenti, lo indicheremo con t, ed è chiamato destinazione.
  - tutti gli altri nodi del grafo appartengono a cammini da s a t.
  - il peso $c(e) \geq 0$ associato ad ogni arco e del grafo prende il nome di capacità.
- Ecco un un esempio di grafo di flusso:



- Nota che in un grafo di flusso vale sempre $m \geq n - 1$

- Dato un grafo di flusso un flusso del grafo è una quantità di traffico effettivamente circolante sulla rete.

- Formalmente un flusso per il grafo di flusso $G = (V, E)$ è una funzione f che associa ad ogni arco $e \in E$ un numero reale $f(e) \geq 0$ che rappresenta la quantità di flusso trasportata dall'arco e.

- Un flusso f deve soddisfare le seguenti due condizioni:

  1. Vincoli sulle capacità: per ogni $e \in E$ risulta $f(e) \leq c(e)$
     (il flusso su di un arco non può superare la capacità dell'arco).
  2. Vincoli di conservazione del flusso: per ogni $v \in V$ con $v \neq s$ and $v \neq t$ deve valere:

$$\sum_{e \text{ entranti in } v} f(e) = \sum_{e \text{ uscenti da } v} f(e)$$

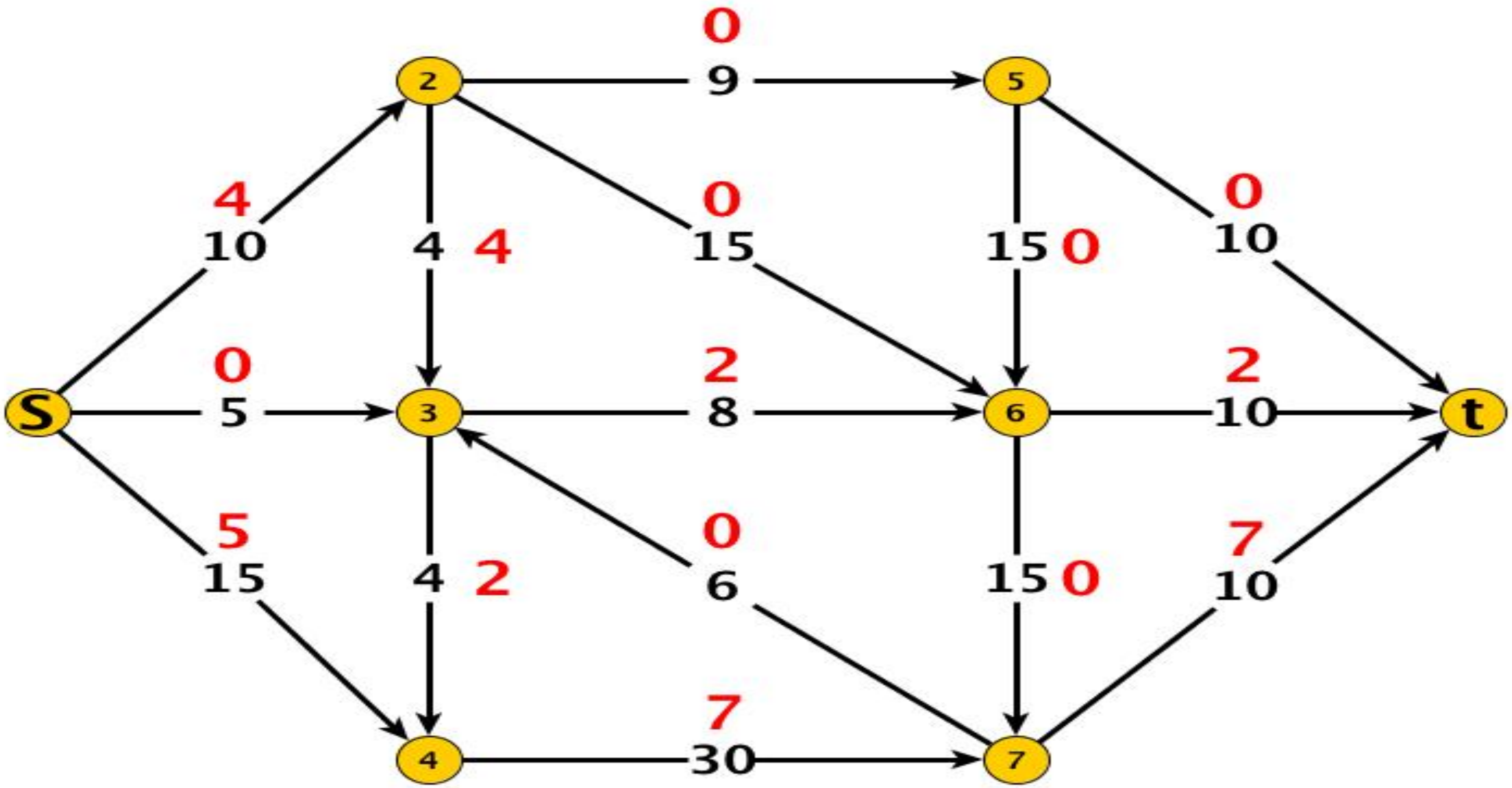
     (per ogni nodo diverso da sorgente e destinazione il flusso entrante nel nodo è pari a quello uscente)

- Il valore di un flusso f, indicato con $v(f)$, sarà

$$\sum_{e \text{ uscenti da } s} f(e)$$

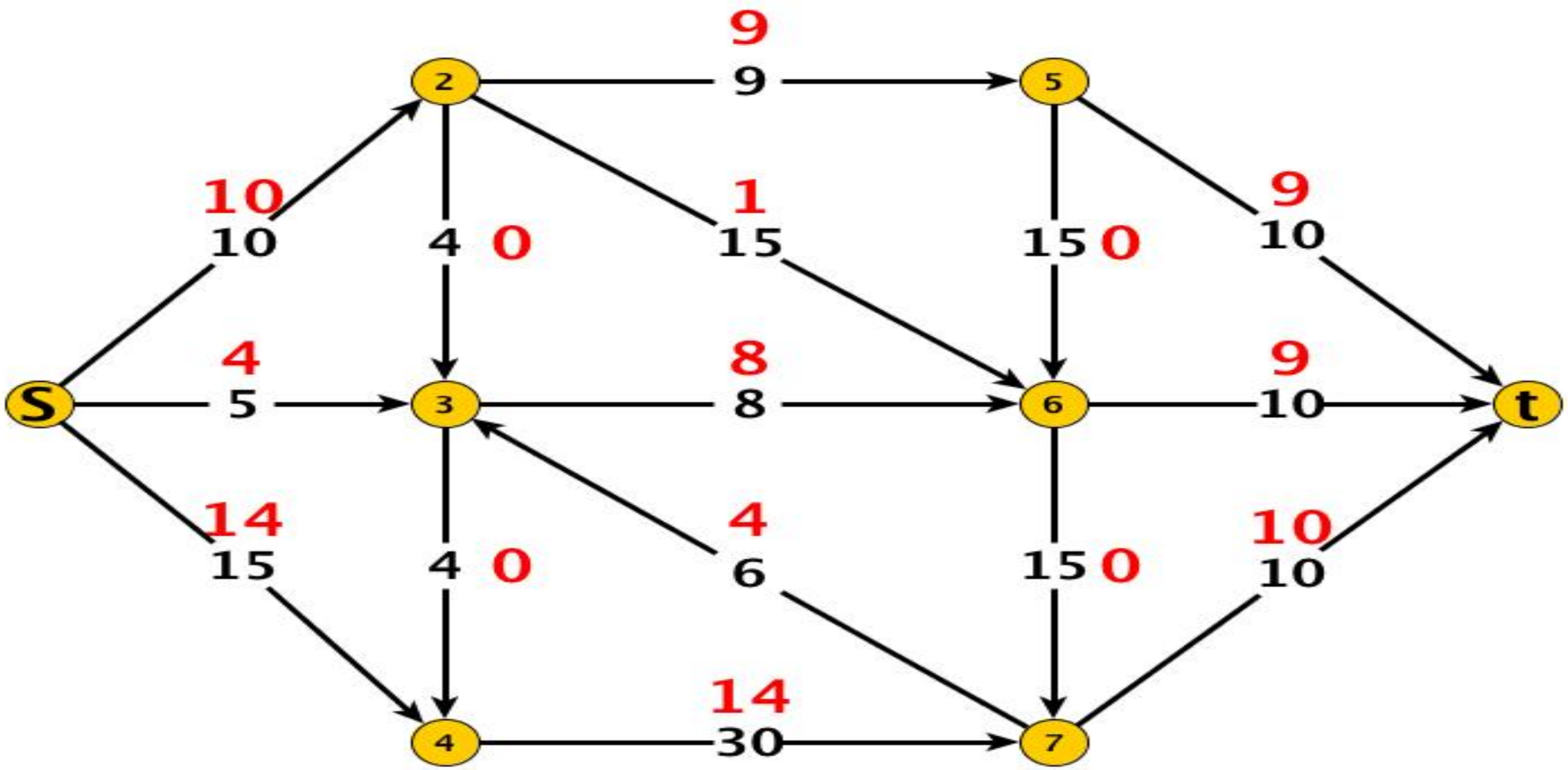
- Esempio di una rete di flusso con evidenziato in rosso un suo possibile flusso f.

- Questo particolare flusso ha valore $v(f) = 4 + 0 + 5 = 9$.



- La stessa rete di flusso dell'esempio precedente con evidenziato in rosso un altro suo possibile flusso f.

- Questo nuovo flusso ha valore $v(f) = 10 + 4 + 14 = 28$.



Si pone quindi il seguente problema:

Data una rete di flusso G, determinare il flusso f di valore

$$v(f) = \sum_{e \text{ uscente da } s} f(e)$$

massimo.

- Un possibile algoritmo greedy per risolvere il problema lavora come segue:

FOR ogni arco e del grafo **DO** $f(e) = 0$ /***Parti con un flusso** f **tale che** $v(f) = 0$*/

WHILE in G **c'è un cammino** P **da** s **a** t **con** $f(e) < c(e)$ **per ogni arco** e **in** P **DO**

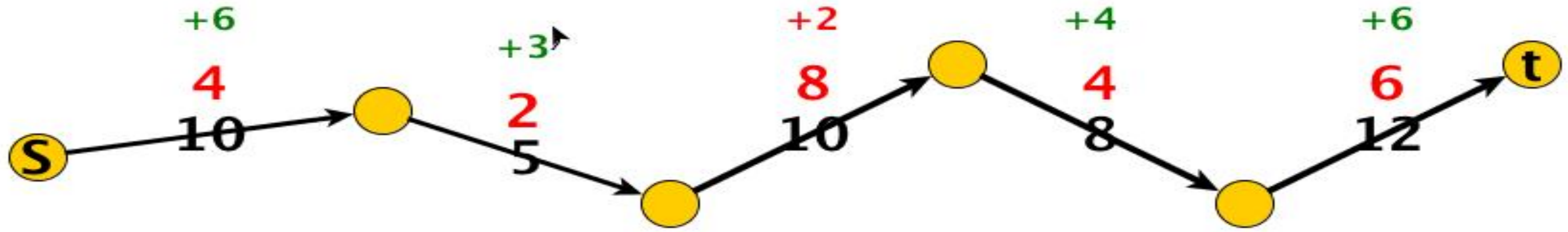
$\quad bottleneck(P) \leftarrow \min_{e \in P}\{c(e) - f(e)\}$

$\quad$ **FOR ogni arco** e **in** P **DO**

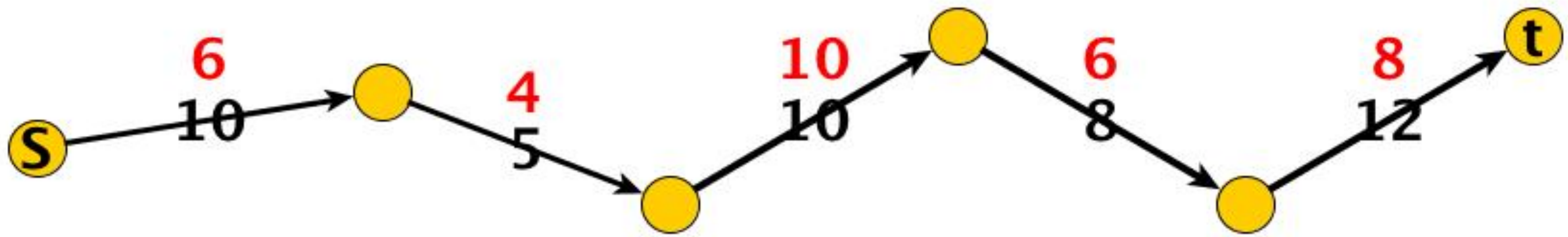
$\quad\quad f(e) = f(e) + bottleneck(P)$

$\quad$ **ENDFOR** /***il flusso del grafo è stato incrementato di** $bottleneck(P)$ **unità***/

ENDWHILE

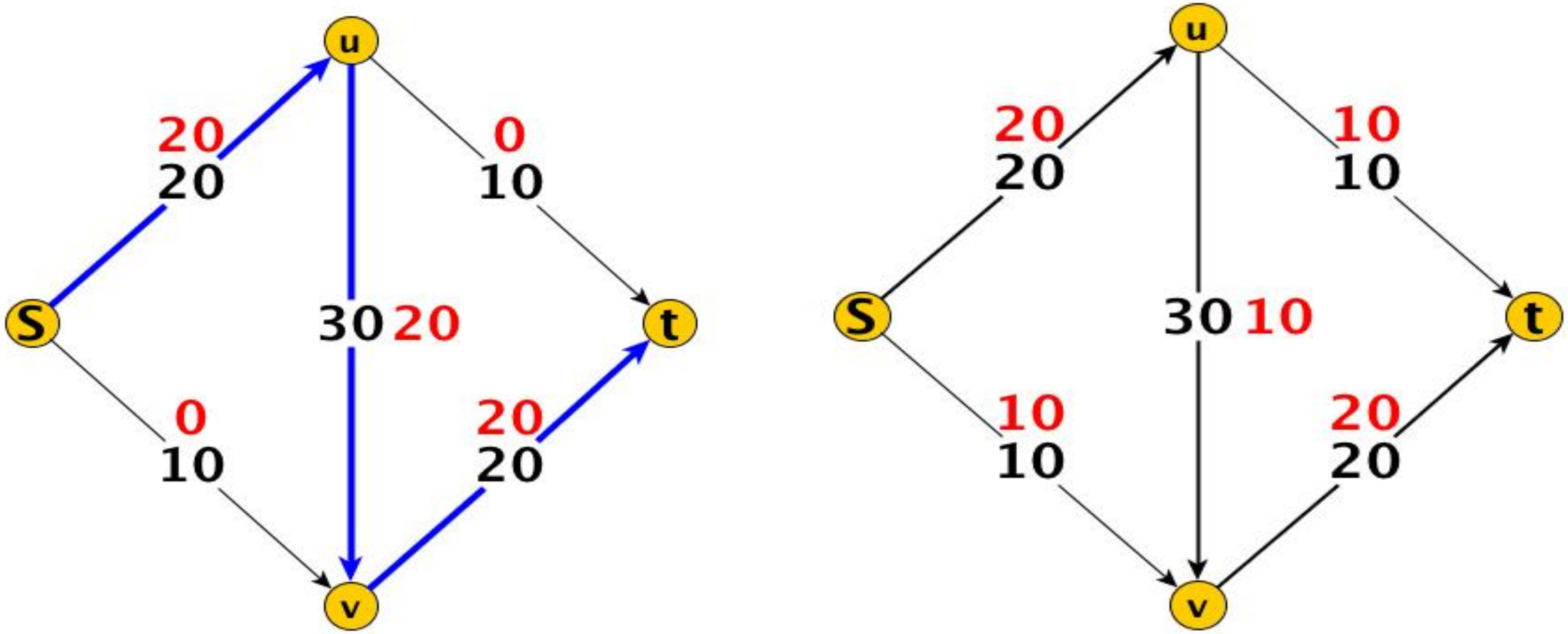


Nell'esempio in figura un cammino P in cui possiamo incrementare il flusso. Il suo bottleneck è 2.
Nella figura in basso il flusso del cammino incrementato dall'algoritmo greedy



L'incremento di flusso del cammino ha determinato un incremento di 2 del valore del flusso del grafo.

L'algoritmo greedy non necessariamente trova il flusso massimo!

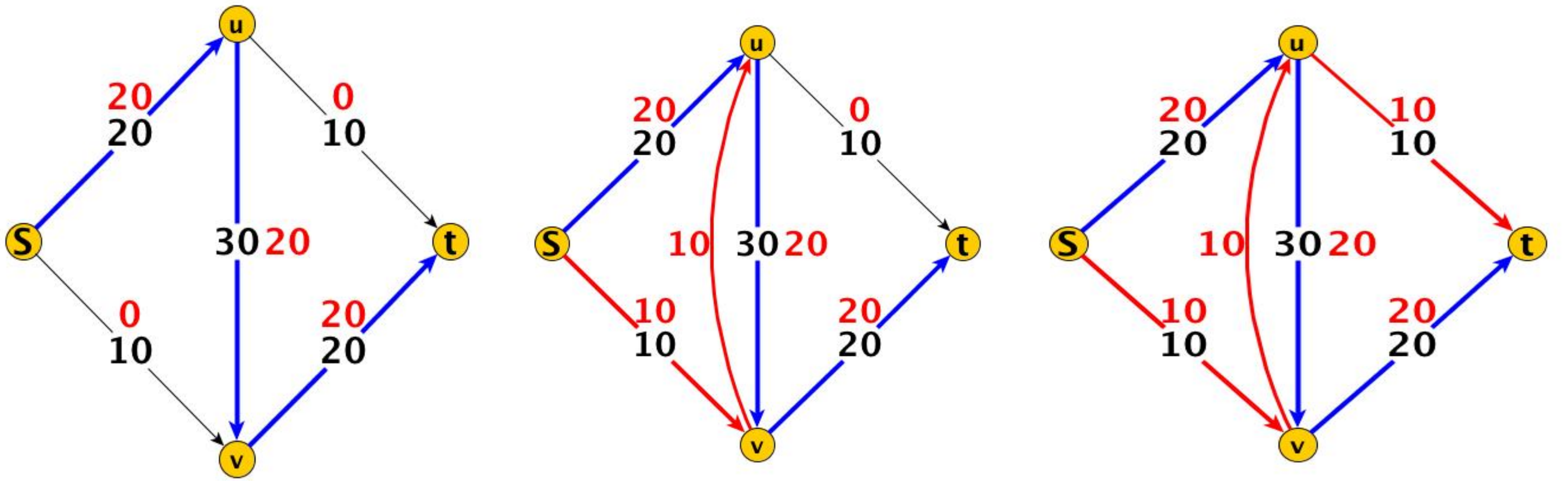


- Nella rete di sinistra, l'algoritmo termina dopo aver selezionato il cammino s, u, v, t e prodotto un flusso di valore $v(f) = 20{+}0 = 20$.

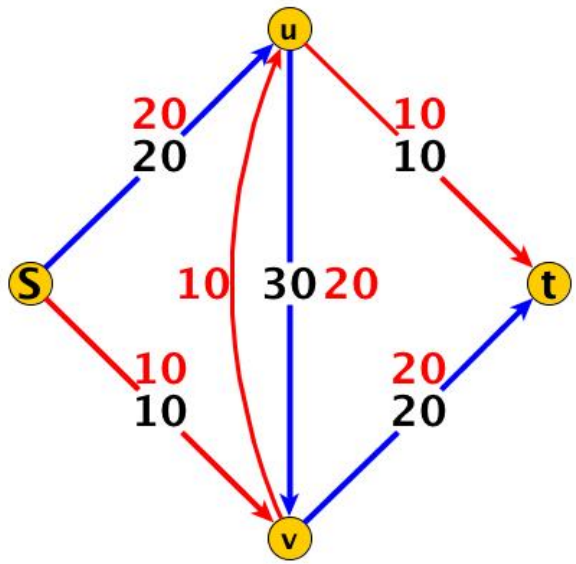
- Per la stessa rete, nella figura di destra, viene riportato un flusso di valore $v(f) = 20 + 10 = 30$.

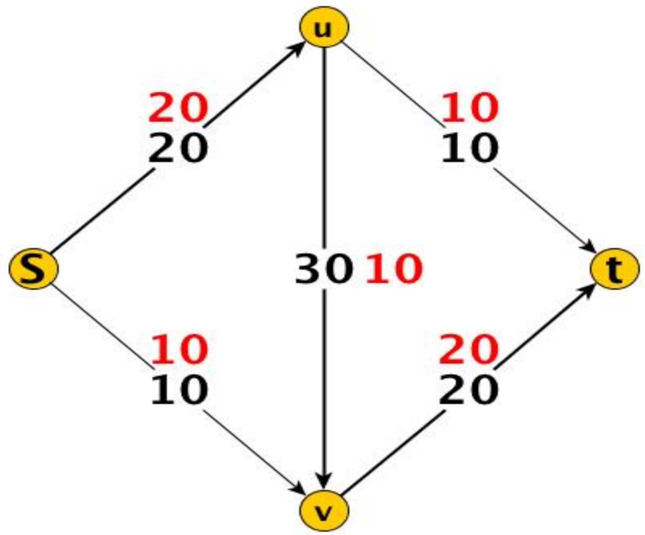
Partendo dalla soluzione prodotta dal greedy di valore 20 possiamo migliorarla e produrre il flusso da 30 come segue:

- **mandiamo 10 nuove unità di flusso sull'arco (s, v).**
- **ci troviamo un "eccesso" di 10 unità di flusso entranti nel nodo v.**
- **Per cavarcela, possiamo "spingere" tali 10 unità di flusso eccedenti in v sull'arco (u, v) in direzione contraria al flusso blu precedentemente assegnato,**
- **queste 10 unità di flusso che giungono ad u possono poi essere inviate a t attraverso l'arco (u, t).**
- **ottenendo così il flusso da 30.**



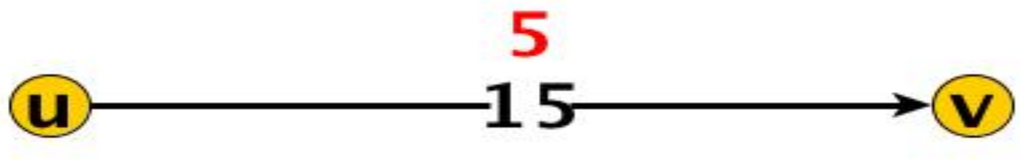
- Osserviamo ora che le assegnazioni di flusso nella figura di sinistra sono equivalenti a quelle di destra, in quanto il flusso rosso "controcorrente" di valore 10 sull'arco (v, u) equivale a diminuire di 10 il flusso blu di 20 precedentemente assegnato all'arco (u, v).





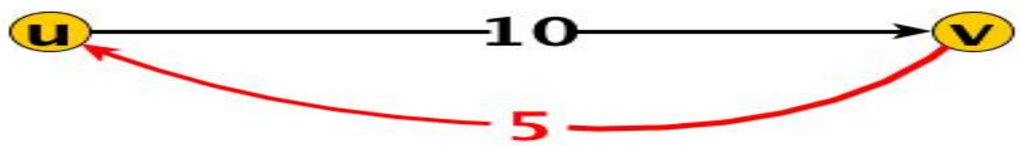
- Occorre un modo per formalizzare la possibilità di usare flusso "controcorrente" al fine di migliorare soluzioni greedy.

- Consideriamo un arco $e = (u, v) \in E$, con capacità $c(e)$ e flusso assegnato $f(e)$.



- Su quest'arco, c'è possibilità di aumentare il flusso da u a v di 10 unità, ma anche di diminuire il flusso da u a v di 5 unità (vale a dire di aumentare il flusso da v verso u di 5 unità).

- Quindi, dopo aver assegnato 5 unità di flusso da u verso v, dal punto di vista concettuale la capacità (residua) dell'arco da u a v è adesso solo di 10 (possiamo infatti mandare su questo arco fino ad altre 10 unità di flusso) ma si è anche creato un nuovo arco da v verso u di capacità 5 (su cui appunto possiamo mandare fino a 5 unità di flusso).
  In conclusione la nuova situazione è



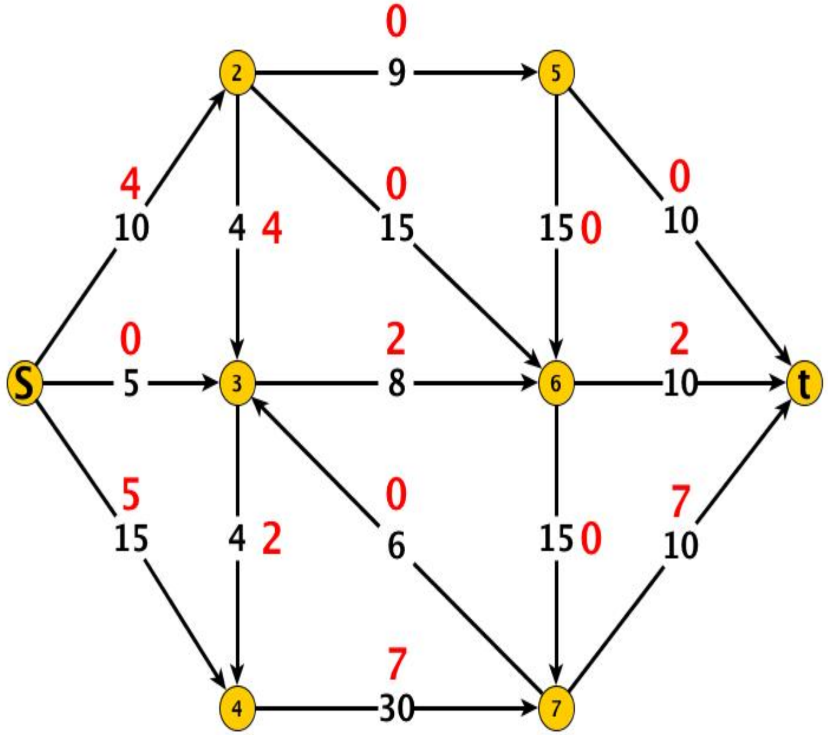
Formalmente:

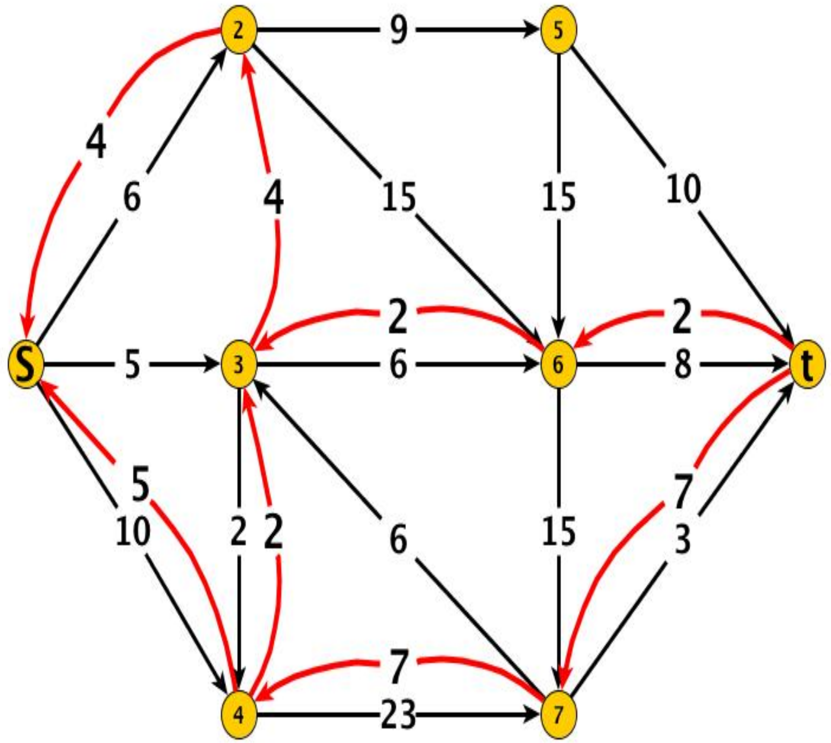
- **Data una rete di flusso $G = (V, E)$ con capacità sugli archi $c(e)$ e flusso assegnato $f(e)$, definiamo rete residua di G rispetto al flusso f il grafo $G_f(V, E_f)$ con capacità sugli archi $c_f(e)$ così definita:**

  1. **ad ogni $(u, v) \in E$ con $f(u, v) < c(u, v)$ corrisponde in G_f l'arco (u, v) con capacità $c_f(u, v) = c(u, v) - f(u, v)$**
  2. **ad ogni $(u, v) \in E$ con $f(u, v) > 0$ corrisponde in G_f l'arco (v, u) con capacità $c_f(v, u) = f(u, v)$**

- **Gli archi di G_f di tipo 1 li chiameremo archi in avanti, gli archi di tipo 2 li chiameremo archi all'indietro. Le capacità degli archi le chiameremo capacità residue.**
  Nota che per ogni arco e di G_f risulta $c_f(e) > 0$.

- A sinistra una rete di flusso G con il suo flusso f (evidenziato in rosso). A destra il grafo residuo G_f (con gli archi all'indietro evidenziati in rosso):





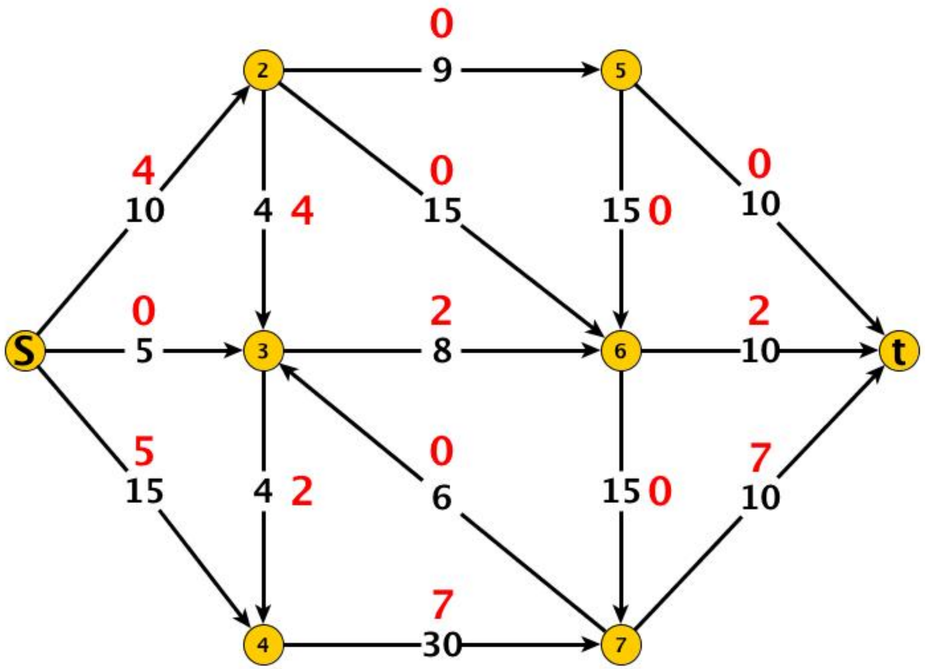
- Sia f un flusso nella rete G. Un cammino semplice P da s a t nel grafo residuo G_f è detto cammino aumentante.
  Il nome deriva dal fatto che la presenza del cammino P in G_f rende possibile aumentare il flusso f.
  di G grazie al seguente algoritmo:
  Definiamo

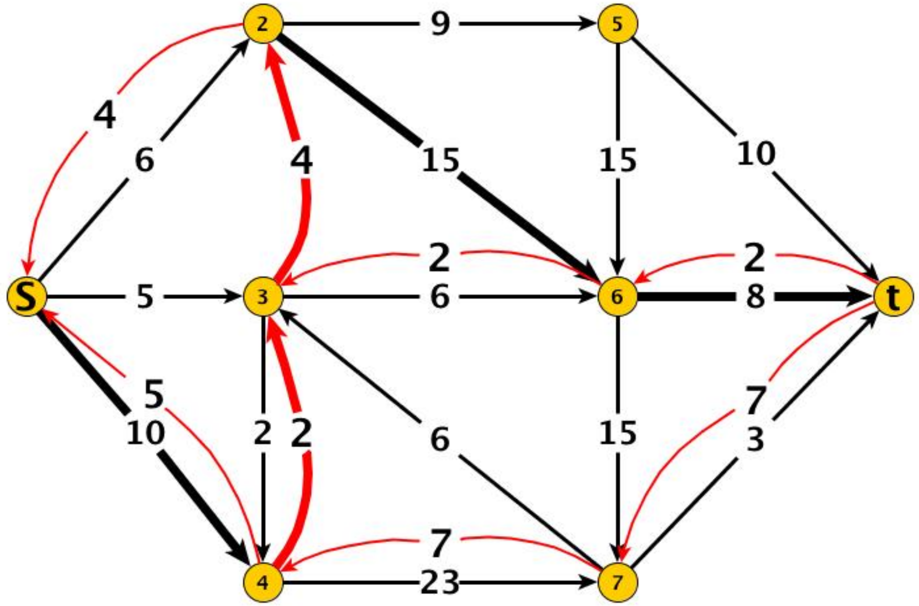
$$bottleneck(P) = \min_{e \in P}\{c_f(e)\}$$

  e consideriamo il seguente algoritmo:

FOR ogni arco $e \notin P$ DO $f'(e) \leftarrow f(e)$
Sia $b = bottleneck(P)$
FOR ogni arco e nel cammino P DO
        IF (in G_f l'arco e è un arco in avanti) THEN $f'(e) \leftarrow f(e)+b$
                ELSE $f'(e) \leftarrow f(e)-b$
ENDFOR
RETURN f'

Esempio (1)



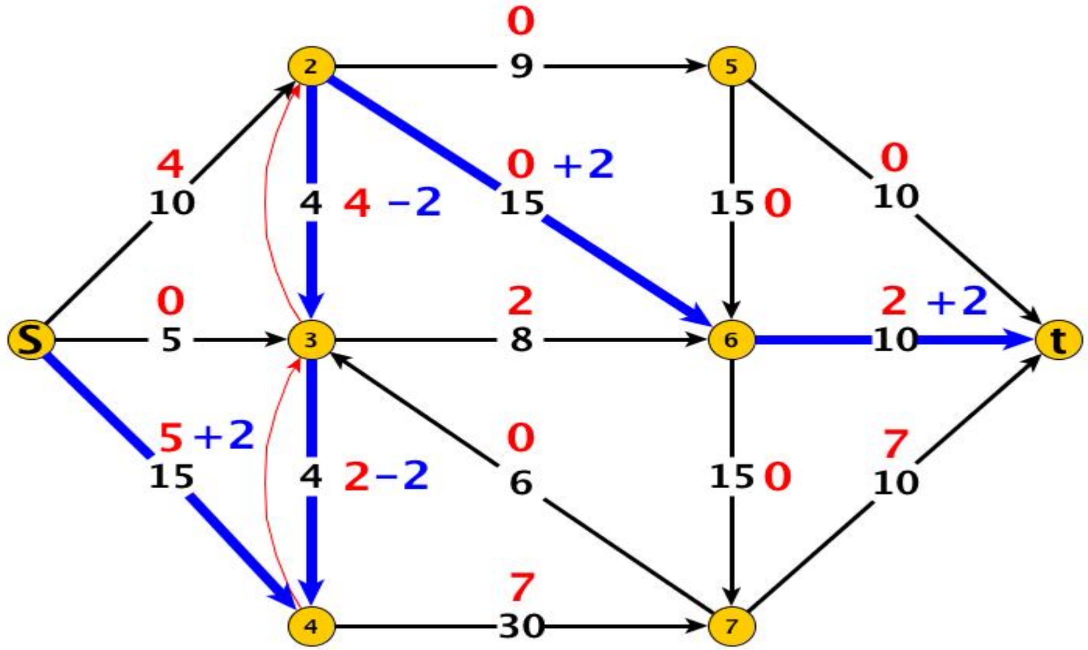


- Dal grafo G (riportato a sinistra) con il suo flusso (in rosso) viene costruito il grafo residuo G_f (riportato a destra).

- Dal grafo G_f viene selezionato un cammino semplice P da s a t. In questo esempio si è scelto il cammino $P = s, 4, 3, 2, 6, t$ (evidenziato in grassetto nel grafo G_f).

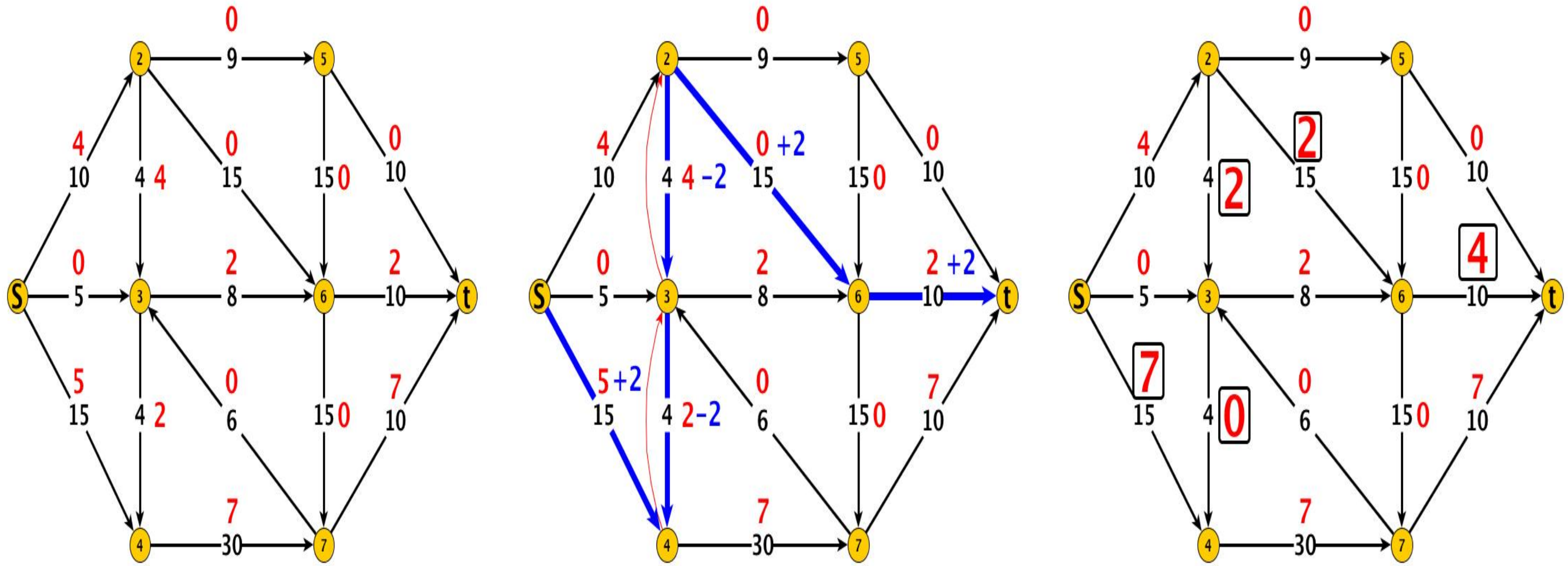
- per il cammino P scelto vale $bottleneck(P) = \min_{e \in P}\{c_f(e)\} = 2$.

Esempio (2)



- Il cammino P di G_f non corrisponde ad un vero e proprio cammino nel grafo G.
- In G la direzione dell'arco risulta invertita quando in G_f si è utilizzato un arco all'indietro (nella figura a destra è evidenziato in blu il "cammino P in G).
- i flussi f degli archi di G coinvolti dal cammino P ricevono un incremento o un decremento pari al bottleneck di P (che nel nostro caso è 2).
- Dalla figura si evince che i nuovi flussi f' soddisfano i vincoli di capacità (vale a dire $0 \leq f'(e) \leq c(e)$ per ogni arco e).
- Dalla figura si evince che i nuovi flussi f' soddisfano anche i vincoli di conservazione del flusso.

Esempio (3)



- Grazie al cammino P con bottleneck 2 del grafo residuo G_f si è ottenuto un nuovo flusso f' per G con $v(f') = v(f) + 2$.

FOR ogni arco $e \notin P$ DO $f'(e) \leftarrow f(e)$
Sia $b = bottleneck(P)$
FOR ogni arco e nel cammino P DO
IF (in G_f l'arco e è un arco in avanti) THEN $f'(e) \leftarrow f(e) + b$
ELSE $f'(e) \leftarrow f(e) - b$
ENDFOR
RETURN f'

- Bisogna dimostrare che i nuovi valori $f'(e)$ prodotti dall'algoritmo sono un flusso per G e che questo nuovo flusso ha un valore superiore al flusso f. In altre parole bisogna dimostrare che

1. f' soddisfa i vincoli sulle capacità, vale a dire
$$0 \leq f'(e) \leq c(e)$$
per ogni arco e.
2. f' soddisfa i vincoli di conservazione del flusso, vale a dire
$$\sum_{e \text{ entranti in } u} f'(e) = \sum_{e \text{ uscenti da } u} f'(e)$$
per ogni $u \notin \{s, t\}$
3. $v(f') > v(f)$.

FOR ogni arco $e \notin P$ DO $f'(e) \leftarrow f(e)$
Sia $b = bottleneck(P)$
FOR ogni arco e nel cammino P DO

IF (in G_f l'arco e è un arco in avanti) THEN $f'(e) \leftarrow f(e)+b$

ELSE $f'(e) \leftarrow f(e)-b$

ENDFOR
RETURN f'

1) valori di f' soddisfano i vincoli sulle capacità:

- per gli archi che non appartengono al cammino P vale $f = f'$ quindi i vincoli sono rispettati.
- consideriamo dunque gli archi e in P (ricordando che per questi archi vale $b \leq c_f(e)$)
  - se e è un arco in avanti allora $c_f(e) = c(e) - f(e)$ quindi $f'(e) = f(e) + b \leq f(e) + c_f(e) = c(e)$.
  - se e è un arco all'indietro allora $c_f(e) = f(e)$ quindi $f'(e) = f(e) - b \geq f(e) - c_f(e) = 0$.

FOR ogni arco $e \notin P$ DO $f'(e) \leftarrow f(e)$
Sia $b = bottleneck(P)$
FOR ogni arco e nel cammino P DO
IF (in G_f l'arco e è un arco in avanti) THEN $f'(e) \leftarrow f(e) + b$
ELSE $f'(e) \leftarrow f(e) - b$
ENDFOR
RETURN f'

2) f' soddisfa i vincoli di conservazione del flusso:

- basta provarlo per i nodi che sono nel cammino P visto che per gli altri nodi il flusso passante su di essi non cambia.
- Sia u un nodo nel cammino P, (x, u) l'arco attraverso il quale P entra in u e (u, y) l'arco attraverso cui P esce da u.
  La prova è per casi e vanno distinti 4 casi a seconda che gli archi (x, u) e (u, y) siano entrambi archi in avanti o entrambi archi all'indietro o uno dei due arco in avanti e l'altro arco all'indietro.
  Noi consideriamo un solo caso (per gli altri la prova è simile).
- (x, u) e (u, y) sono entrambi archi in avanti.
  - in questo caso $f'(x, u) = f(x, u) + b$ e $f'(u, y) = f(u, y) + b$, ne segue:

$$\begin{aligned}
\sum_{e \text{ entranti in } u} f'(e) &= \sum_{e \text{ entranti in } u \text{ con } e \neq (x,u)} f'(e) + f'(x, u) \\
&= \sum_{e \text{ entranti in } u \text{ con } e \neq (x,u)} f'(e) + f(x, u) + b \\
&= \sum_{e \text{ uscenti da } u \text{ con } e \neq (u,y)} f(e) + f(u, y) + b \\
&= \sum_{e \text{ uscenti da } u \text{ con } e \neq (u,y)} f'(e) + f'(u, y) \\
&= \sum_{e \text{ uscenti da } u} f'(e)
\end{aligned}$$

FOR ogni arco $e \notin P$ DO $f'(e) \leftarrow f(e)$
Sia $b = bottleneck(P)$
FOR ogni arco e nel cammino P DO
IF (in G_f l'arco e è un arco in avanti) THEN $f'(e) \leftarrow f(e) + b$
ELSE $f'(e) \leftarrow f(e) - b$
ENDFOR
RETURN f'

3) **Il flusso f' ha valore superiore al flusso f:**

- **c'è un unico arco uscente da s che appartiene a P (è il primo arco del cammino P)**
- **quest'arco, che indicheremo con e', è un arco in avanti in G_f si avrà quindi $f'(e') = f(e') + b$**
- **inoltre, poiché' $c_f(e) > 0$ per ogni arco e allora il bottleneck b del cammino P risulterà strettamente positivo.**
- **Ricordando che gli unici archi che cambiano flusso sono quelli che appartengono a P otteniamo:**

$$v(f') = \sum_{e \text{ uscenti da } s} f'(e) = f(e') + b + \sum_{e \text{ uscenti da } s,\ e \neq e'} f(e) = v(f) + b > v(f)$$

- L'algoritmo per il calcolo del massimo flusso, noto come algoritmo di Ford-Fulkerson, lavora come segue:

FOR ogni arco e del grafo G DO $f(e) = 0$ /*Parti con un flusso f tale che $v(f) = 0$*/
Costruisci il grafo residuo G_f
WHILE in G_f c'è un cammino P da s a t DO
  $f \leftarrow AUMENTA(f, P)$ /*il flusso del grafo è stato incrementato di $bottleneck(P)$ unità*/
  Costruisci il nuovo grafo residuo G_f
ENDWHILE
RETURN f

- L'algoritmo di Ford-Fulkerson è più flessibile dell'algoritmo greedy in quanto su certi archi può anche diminuire il flusso precedentemente assegnato (mandando flusso "controcorrente") al fine di utilizzare cammini da s a t altrimenti inutilizzabili.

- Resta da analizzare la complessità di tale algoritmo e soprattutto dimostrare che il flusso prodotto dall'algoritmo è effettivamente il flusso massimo.