Progettazione di algoritmi

Reti di flusso (2)

- Correttezza e complessità dell’algoritmo di Ford-Fulkerson

- Il teorema del massimo flusso-minimo taglio

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

FOR **ogni arco** e **del grafo** G **DO** $f(e) = 0$ **/*Si parte con un flusso** f **di valore nullo*/**
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 di** G **è stato incrementato di** $bottleneck(P)$ **unità*/**
    Costruisci il nuovo grafo residuo G_f
ENDWHILE
RETURN f

- dove la procedura **AUMENTA**(f, P) lavora come segue:

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'

- **Complessità: (sotto l'assunzione che tutte le capacità $c(e)$ del grafo G hanno valore intero)**
- **I flussi f via via prodotti dalle varie iterazioni del WHILE dell'algoritmo hanno tutti valore intero.**
  - **Diciamo che un flusso f di G è a valori interi se $f(e)$ è un intero per ogni arco e. Il flusso nullo da cui parte l'algoritmo è a valori interi. Mostreremo che un'iterazione del WHILE dell'algoritmo di Ford e Fulkerson che parte da un flusso a valori interi produce sempre un nuovo flusso a valori interi. Questo, unito al fatto che un flusso a valori interi ha valore intero, prova che i flussi prodotti dall'algoritmo di Ford e fulkerson sono tutti a valore intero.**
    Sia f il flusso di valori interi di G, per costruzione $c_f(e)$ è un intero per ogni arco e di G_f (ricordando che $c(e)$ è intero). Di conseguenza per ogni cammino P in G_f il bottleneck b di P è un intero e quindi si avrà $f'(e) \in \{f(e), f(e) + b, f(e) - b\}$. Quindi per ogni arco e il valore di $f'(e)$ sarà intero.
- **L'algoritmo termina in al più $v(f^*)$ iterazioni dove f^* è il valore del flusso massimo della rete G (infatti il flusso ad ogni iterazione si incrementa almeno una unità).**
- **Ciascuna iterazione del WHILE richiede tempo $O(m)$.**
  - **il tempo per controllare se esiste un cammino P da s a t in $G_f(V, E_f)$ è pari al tempo per eseguire una visita di G_f che richiede tempo $O(|V| + |E_f|) = O(m)$ (perché $|E_f| \leq 2m$ e $m \geq |V| - 1$).**
  - **il tempo per calcolare il nuovo flusso è $O(n) = O(m)$ (perché P ha al più n archi).**
  - **il tempo per calcolare il nuovo grafo residuo G_f è $O(m)$.**
- **Il tempo totale dell'algoritmo è $O(m \cdot v(f^*))$.**

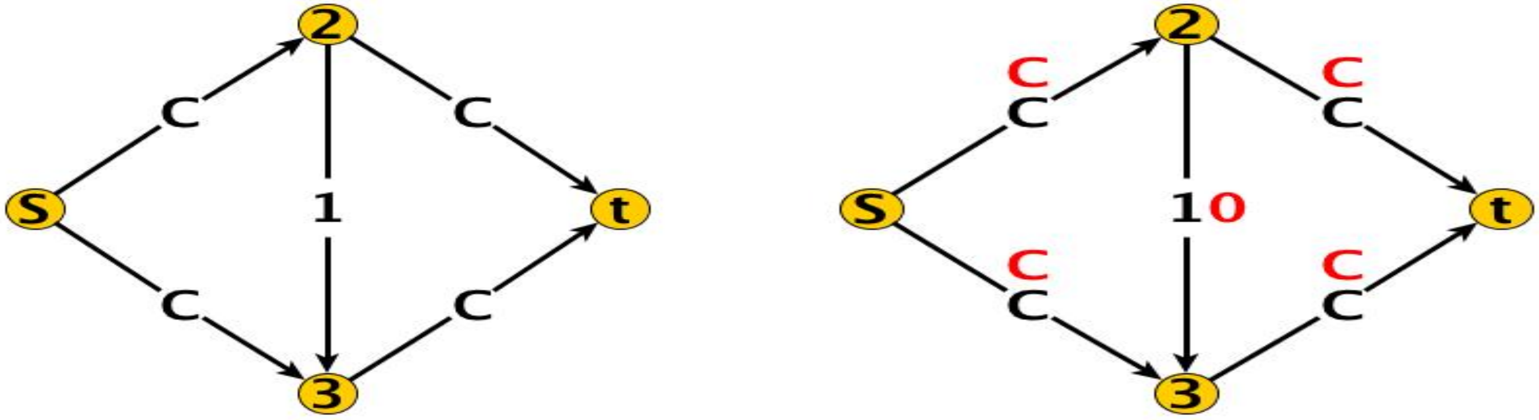
L'algoritmo di Ford e Fulkerson è molto efficiente in certe situazioni, ad esempio nel caso in cui per ogni arco e del grafo risulta $c(e) = 1$. In tal caso l'algoritmo ha complessità

$$O(m \cdot v(f^*)) = O(m \cdot n)$$

(dato che il valore $v(f^*)$ del massimo flusso f^* è sempre limitato superiormente da n).

Tuttavia ci sono casi in cui l'algoritmo può risultare anche molto inefficiente e richiedere tempi esponenziali nella dimensione dell'input ……

- Se C è la massima capacità per gli archi della rete allora l'algoritmo appena descritto può richiedere anche più di C iterazioni.



- **Il flusso massimo del grafo G (a sinistra) vale $2C$ (grafo a destra).**

- **Sulla rete G l'algoritmo di Ford e Fulkerson può richiedere anche $2C$ iterazioni prima di fermarsi. Questo accade se i cammini aumentanti scelti in G_f di volta in volta hanno tutti bottleneck 1. Questi cammini esistono come mostra la lista di seguito:**
  $S \rightarrow 2 \rightarrow 3 \rightarrow t$
  $S \rightarrow 3 \rightarrow 2 \rightarrow t$
  $S \rightarrow 2 \rightarrow 3 \rightarrow t$
  $S \rightarrow 3 \rightarrow 2 \rightarrow t$
  ......

- **Se non si introduce un qualche criterio con cui scegliere tra i vari cammini P presenti in G_f l'algoritmo non è polinomiale.**

- **L'esempio appena visto suggerisce che la scelta del cammino P tra i diversi cammini aumentanti presenti in G_f può influire fortemente sul numero di iterazioni dell'algoritmo.**

- **serve dunque un criterio di scelta che permetta di limitare il numero di iterazioni possibili.**

- **Un possibile criterio consiste nello scegliere ogni volta in G_f il cammino aumentante che presenta il bottleneck massimo (visto che poi il bottleneck corrispondende all'incremento di flusso della singola iterazione).**

  - **Si può dimostrare che scegliendo i cammini in base a questo criterio il numero di iterazioni è $O(\log C)$ dove C è la capacità massima sugli archi di G.**

  - **Un criterio che si è rivelato vincente richiede di scegliere il cammino in G_f da s a t che attraversa il minimo numero di archi.**

    * **implementare questo criterio non altera la complessità $O(m)$ della singola iterazione (basta infatti scegliere il primo cammino da s a t che si ottiene mediante una visita in ampiezza di G_f a partire da s).**

    * **si può mostrare che, scegliendo il cammino ad ogni iterazione con questo criterio allora il numero di iterazioni possibili è $O(nm)$.**

  - **Il metodo di Ford-Fulkerson modificato in modo da scegliere sempre il cammino aumentante più corto prende il nome di algoritmo di Edmonds-Karp e richiede tempo $O(nm^2)$.**

- Resta da dimostrare che il flusso prodotto dall'algoritmo di Ford-Fulkerson è effettivamente un flusso di valore massimo.
  Basterà dimostrare che:

Un flusso f è di valore massimo in G se e solo se in G_f non ci sono cammini aumentanti.

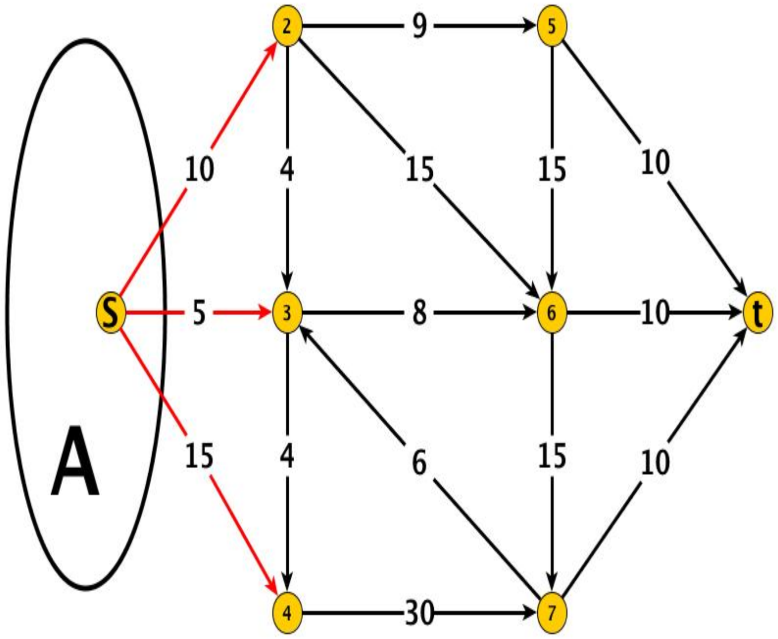
Per provare questo risultato è utile introdurre un nuovo concetto: il **taglio** della rete.

- Data una rete di flusso $G = (V, E)$, con sorgente s e destinazione t, un taglio in G è una partizione (A, B) dei vertici V in sottoinsiemi disgiunti A e B tali che $s \in A$ e $t \in B$.

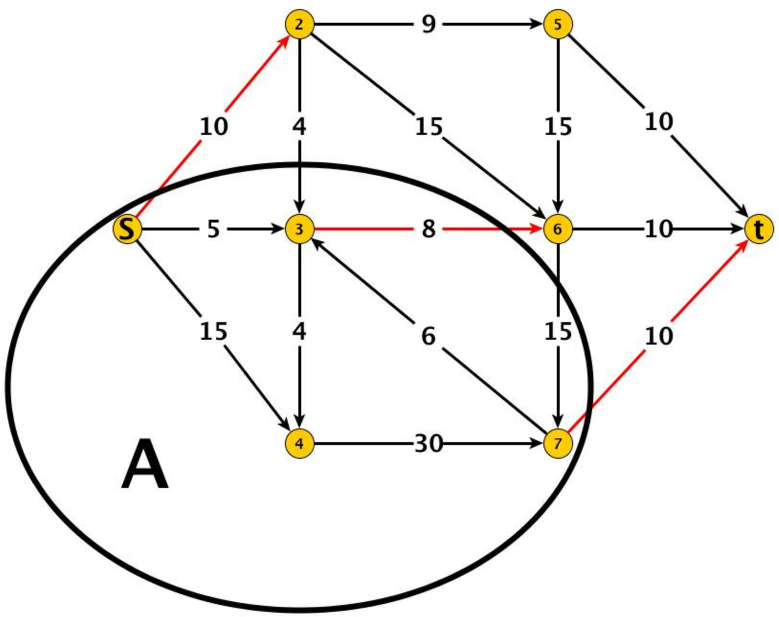
- La capacità del taglio (A, B) è data da

$$c(A,B) = \sum_{e \text{ uscente da } A} c(e) = \sum_{e=(a,b) \text{ con } a \in A \text{ e } b \in B} c(e)$$

- A destra un taglio di una rete con $A = \{s\}$.
  Vale $c(A, B) = 10 + 5 + 15 = 30$



- A destra un taglio di una rete con $A = \{s, 3, 4, 7\}$.
  Vale $c(A, B) = 10 + 8 + 10 = 28$



I tagli di un grafo danno limiti superiori al valore del flusso massimo:

- **Sia f un qualsiasi flusso di G e sia (A, B) un qualsiasi taglio in G. Vale che il valore del flusso $v(f)$ è al più pari alla capacità $C(A, B)$.**

$$
\begin{aligned}
v(f) &= \sum_{e \text{ uscente da } s} f(e) \\
&= \sum_{u \in A} \left(\sum_{e \text{ uscente da } u} f(e) - \sum_{e \text{ entrante in } u} f(e) \right) \\
&\quad \text{(per la conservazione del flusso, tutti i termini all'interno della parentesi} \\
&\quad \text{sono } = 0 \text{ tranne che per } u = s\text{)} \\
&= \sum_{e \text{ uscente da } A} f(e) - \sum_{e \text{ entrante in } A} f(e) \\
&\leq \sum_{e \text{ uscente da } A} f(e) \\
&\leq \sum_{e \text{ uscente da } A} c(e) \\
&= C(A, B)
\end{aligned}
$$

- Vogliamo dimostrare che

Un flusso f è di valore massimo in G se e solo se in G_f non ci sono cammini aumentanti.

Proveremo questo risultato mostrando che le seguenti tre affermazioni sono equivalenti (ovvero ognuna implica le altre):

1. esiste un taglio (A, B) tale che $v(f) = C(A, B)$
2. il flusso f ha valore $v(f)$ massimo
3. G_f non ha cammini aumentanti.

1) Esiste un taglio (A,B) tale che v(f)=C(A,B)

2) Il flusso f ha valore v(f) massimo

3) G_f non ha cammini alternanti

1) Esiste un taglio (A,B) tale che v(f)=C(A,B)

2) Il flusso f ha valore v(f) massimo

prova (per assurdo):

- Assumiamo che f non sia un flusso massimo,
- esiste dunque un flusso f' con $v(f') > v(f)$
- abbiamo quindi $v(f') > C(A, B)$. ASSURDO! (perché sappiamo che la capacità di un taglio rappresenta un limite superiore al valore dei flussi)

2) Il flusso f ha valore v(f) massimo

3) G_f non ha cammini aumentanti

prova (per assurdo):

- Assumiamo che in G_f ci sia un cammino P aumentante,

- grazie a P con un iterazione dell'algoritmo di Ford-Fulkerson possiamo ottenere per G un nuovo flusso f' con $v(f') > v(f)$. Assurdo! (perché f è flusso massimo in G).

3) G_f non ha cammini aumentanti

1) Esiste un taglio (A,B) tale che v(f)=C(A,B)

prova (costruttiva):

- considera il seguente insieme di nodi:

$$A = \{u : u \text{ è raggiungibile in } G_f \text{ a partire da } s\ \}.$$

- poiché G_f non ha cammini aumentanti deve aversi $t \in B = V - A$.
- Quindi (A, B) è un taglio. Faremo vedere che $v(f) = C(A, B)$
  - per ogni arco $e = (u, v)$ in G uscente da A deve aversi $f(e) = c(e)$
    - in caso contrario in G_f avremmo un arco (u, v) (con capacità residua $c_f(e) = c(e) - f(e)$). Il nodo v sarebbe dunque raggiungibile in G_f a partire da s (passando per il nodo u) e dunque l'arco e non sarebbe un arco uscente da A.
  - per ogni arco $e = (u, v)$ in G entrante in A deve aversi $f(e) = 0$
    - in caso contrario in G_f avremmo un arco $e' = (v, u)$ all'indietro (con capacità residua $c(e') = f(e)$). Il nodo u sarebbe dunque raggiungibile in G_f a partire da s (passando per il nodo v) e dunque l'arco e non sarebbe un arco entrante in A.
  - quindi:

$$\begin{aligned} v(f) &= \sum_{e \text{ uscente da } A} f(e) - \sum_{e \text{ entrante in } A} f(e) \\ &= \sum_{e \text{ uscente da } A} c(e) - \sum_{e \text{ entrante in } A} 0 \\ &= C(A, B) \end{aligned}$$

Abbiamo dunque dimostrato che le seguenti tre affermazioni sono equivalenti (ovvero ognuna implica le altre):

1. esiste un taglio (A, B) tale che $v(f) = C(A, B)$
2. il flusso f ha valore $v(f)$ massimo
3. G_f non ha cammini aumentanti.

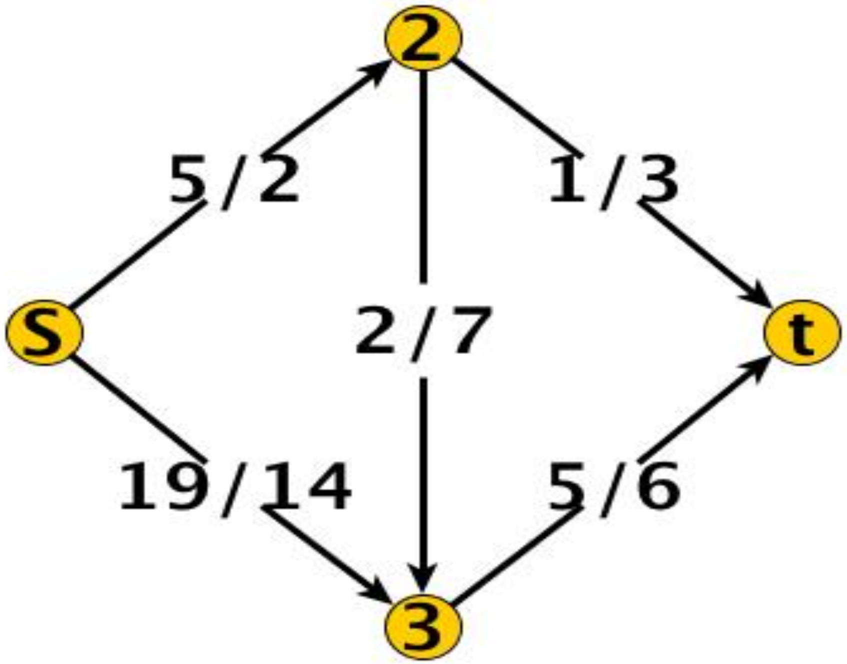
Da queste equivalenze discende:

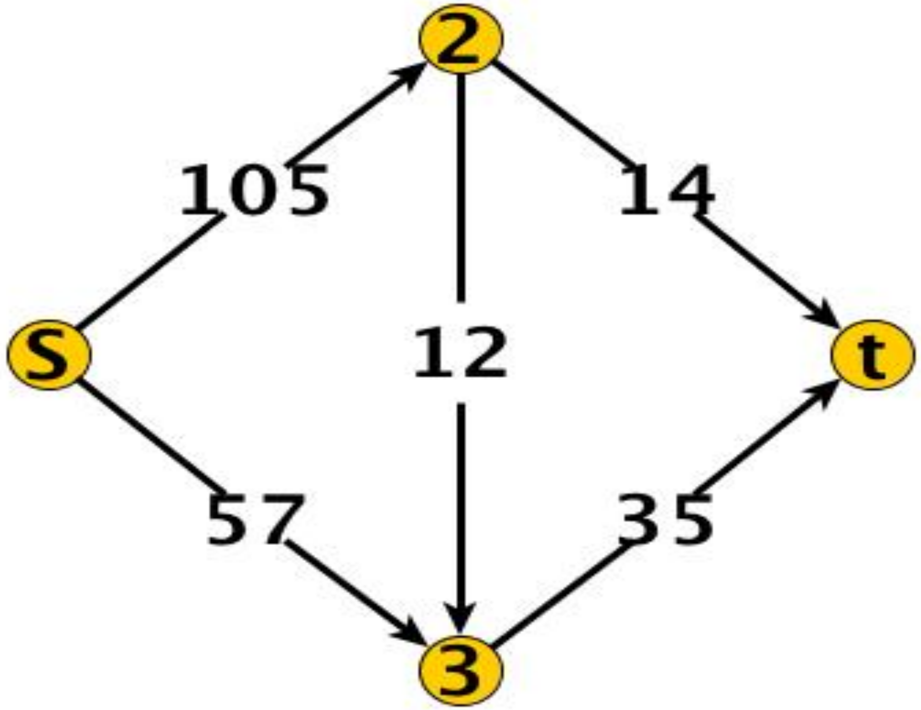
- correttezza dell'algoritmo di Ford-Fulkerson.
  - "dimenticando" la 1 si ha: il flusso f ha valore $v(f)$ massimo se e solo se G_f non ha cammini aumentanti.
  - il risultato segue ricordando che l'algoritmo si ferma solo quando si raggiunge un flusso per cui G_f non ha cammini aumentanti.
- **il teorema del massimo flusso-minimo taglio:**
  il valore $v(f)$ di un flusso di valore massimo è uguale alla capacità di un taglio di minima capacità.
  - "dimenticando" la 3 si ha: il flusso f ha valore $v(f)$ massimo se e solo se esiste un taglio (A, B) tale che $f(v) = C(A, B)$.
  - il risultato segue ricordando che per ogni flusso f il valore $v(f)$ non può superare la capacità del taglio minimo.

- Fin'ora abbiamo sempre assunto che tutte le capacità $c(e)$ degli archi di G sono numeri interi. Cosa succede se non lo sono?
  - Se le capacità $c(e)$ sono numeri razionali (ovvero, frazioni del tipo $a(e)/b(e)$ per $a(e)$ e $b(e)$ numeri interi) il problema non diventa più difficile:
    * possiamo moltiplicare ogni capacità $c(e) = a(e)/b(e)$ per il minimo comune multiplo x dei numeri $b(e)$, ottenendo così dei numeri $C(e)$ interi
    * risolviamo il problema nella rete di flusso con capacità $C(e)$ al fine di ottenere il valore f' del flusso massimo in tale rete.
    * Per conoscere il valore del flusso massimo nella rete di partenza con capacità $c(e)$, basterà dividere f' per z.

ESEMPIO:

- Per calcolare il flusso massimo della rete con pesi frazionari (in figura a sinistra) basta calcolare il flusso massimo della rete con pesi interi (in figura a destra) e poi dividerlo per 42.
  Infatti: $\frac{19}{14} = \frac{57}{42}$, $\frac{5}{2} = \frac{105}{42}$, $\frac{2}{7} = \frac{12}{42}$, $\frac{1}{3} = \frac{14}{42}$, $\frac{5}{6} = \frac{35}{42}$





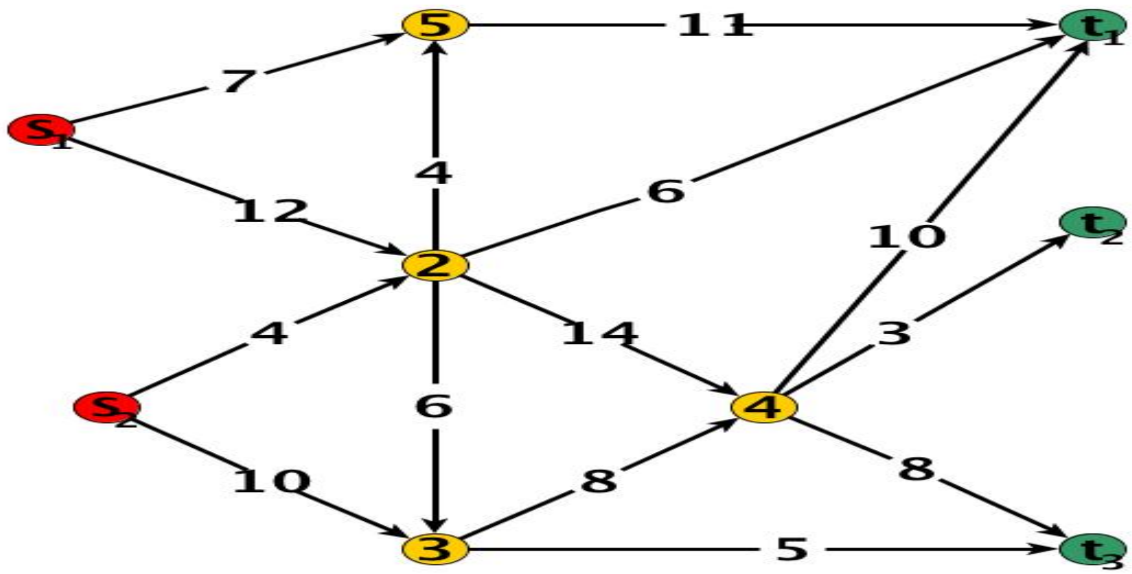
Reti con sorgenti e pozzi multipli. Un problema di flusso massimo su una rete G può avere diverse sorgenti $\{s_1, s_2, \ldots, s_m\}$, e diversi pozzi $\{t_1, t_2, \ldots, t_n\}$.
Si può ridurre il problema di determinare il flusso massimo in una rete con sorgenti multiple e pozzi multipli ad un problema di flusso massimo ordinario. Questo viene fatto trasformando la rete G in una rete di flusso ordinaria $\overline{G}$ con singola sorgente e singolo pozzo nel modo seguente

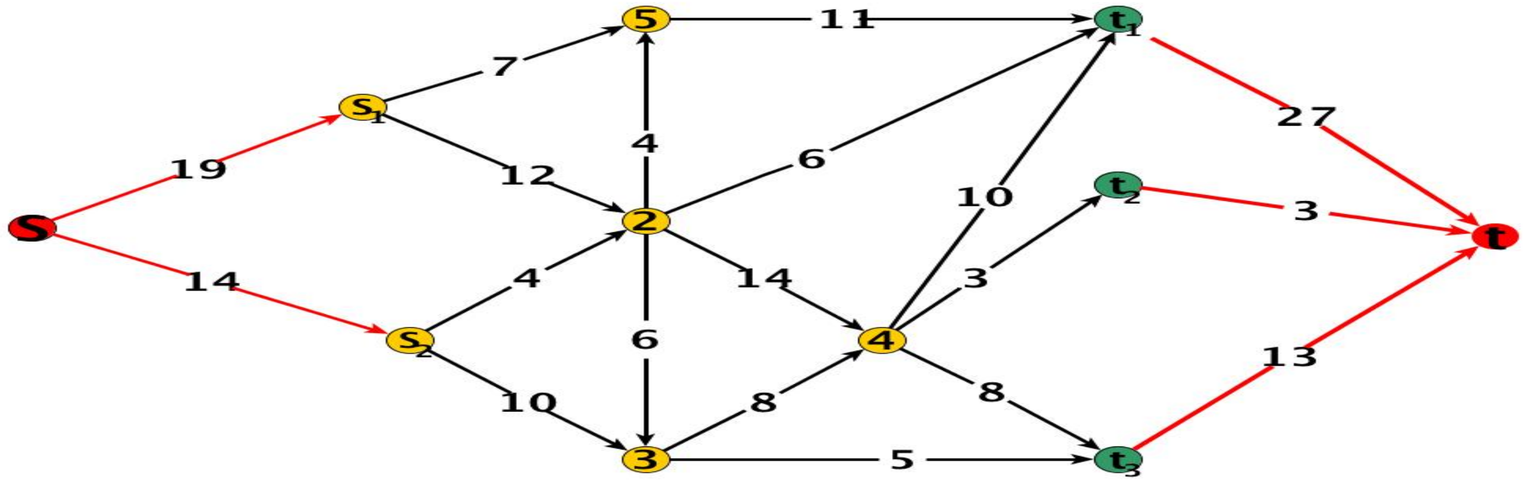
- si aggiunge una super-sorgente s ed un arco orientato (s, s_i), per ogni $i = 1, \ldots, m$, con capacità $c(s, s_i) = \sum_{e \text{ uscenti da } s_i} c(e)$

- si aggiunge un super-pozzo t ed un arco orientato (t_j, t), per ogni $j = 1, \ldots, n$, con capacità $c(t_j, t) = \sum_{e \text{ entranti in } t_j} c(e)$

Intuitivamente, ogni flusso nella rete $\overline{G}$ corrisponde ad un flusso nella rete G.

Esempio:





- In alto una rete con 2 sorgenti e 3 destinazioni.
- In basso la rete "equivalente" con 1 sorgente e 1 destinazione.