1. Il problema del massimo flusso

Nel problema del massimo flusso si considera un grafo, in cui due nodi s e t sono contraddistinti l'uno come *sorgente* e l'altro come *pozzo* e ad ogni arco $e = (i, j)$ è assegnato un valore positivo c_{ij} (oppure c_e) detto *capacità*. Ad ogni arco (i, j) viene inoltre associata una quantità non negativa x_{ij}, detta *flusso*, specificando inoltre se il flusso nell'arco è orientato da i a j oppure da j a i. Nel primo caso si dice che il flusso esce da i ed entra in j, nel secondo caso si dice che il flusso esce da j ed entra in i.

Un flusso è ammissibile se soddisfa i due seguenti vincoli

- in ogni arco il flusso non supera la capacità,
- in ogni nodo, tranne la sorgente e il pozzo, tutto il flusso entrante deve essere uguale al flusso uscente.

Si vuole trovare il massimo flusso ammissibile che può essere fatto uscire dalla sorgente (e che sarà necessariamente uguale a quello entrante nel pozzo)

Un concetto fondamentale nel problema del massimo flusso è costituito dalla *capacità di taglio*. Un taglio in un grafo è definito a partire da un sottoinsieme proprio S di nodi. Si definisce come taglio indotto da S l'insieme di archi

$$\delta(S) := \{e = (i, j) \in E : (i \in S \wedge j \notin S) \vee (j \in S \wedge i \notin S)\}$$

Il termine 'taglio' sta ad indicare che, rimuovendo gli archi del taglio, il grafo diventa sconnesso e non c'è modo di raggiungere i nodi non in S a partire da quelli in S. Ci possiamo chiedere qual è il massimo flusso che può passare sul taglio da S a $N \setminus S$, indipendentemente da ogni altra condizione. Se il flusso in ogni arco è limitato dalla capacità allora la massima quantità è data dall'espressione

$$c(S) := \sum_{e \in \delta(S)} c_e \tag{1}$$

La quantità $c(S)$ prende il nome di capacità di taglio. Se consideriamo un generico taglio che divide la sorgente dal pozzo, cioè un taglio indotto da S con $s \in S$ e $t \notin S$, possiamo notare come, dato un qualsiasi flusso ammissibile, la quantità $x(s)$ in uscita da s deve necessariamente attraversare il taglio $\delta(S)$ per raggiungere il pozzo, e quindi deve valere

$$x(s) \leq c(S)$$

Tale relazione deve essere vera per ogni flusso ammissibile e ogni taglio che separa la sorgente dal pozzo. Quindi possiamo scrivere

$$\max_x x(s) \leq \min_S c(S) \tag{2}$$

dove il minimo va inteso fra tutti i tagli che separano s da t e il massimo fra tutti i flussi ammissibili. Il teorema fondamentale del probema del massimo flusso è che la relazione precedente vale sempre con il segno di uguaglianza

$$\max_x x(s) = \min_S c(S) \tag{3}$$

Per dimostrare (3) serve il concetto di *cammino aumentante*. Data una soluzione ammissibile x, in un generico arco (i, j) il flusso, che supponiamo orientato da i a j, può essere aumentato al massimo della quantità positiva $c_e - x_e$ se $x_e < c_e$ e può essere diminuito al massimo della quantità positiva $c_e + x_e$ (cambiando quindi orientazione). Su un cammino P da s a t il flusso può essere aumentato se per ogni arco in cui il flusso è orientato con il cammino si ha $x_e < c_e$. La massima quantità di flusso che può essere fatta transitare sul cammino P è data da

$$\min_{(e \in P^+} c_e - x_e$$

dove P^+ sono gli archi in cui il flusso è orientato con il cammino.

Dato un flusso, se esiste un cammino aumentante, allora la soluzione corrente può essere aumentata e quindi non può essere massima. Supponiamo allora che non esistano cammini aumentanti. Definiamo come raggiungibili quei nodi i per i quali esiste un cammino aumentante da s ad i. Sia S questo inieme. Se

assumiamo l'ipotesi che non esiste un cammino aumentante da s a t allora $t \notin S$. Consideriamo gli archi del taglio $\delta(S)$. Per ogni arco $e = (i, j) \in \delta(S)$ con $i \in S$ e $j \notin S$ deve essere $x_e = c_e$ e il flusso orientato da i a j altrimenti esisterebbe un cammino aumentante da s a j contro l'ipotesi che $j \notin S$. Quindi la quantità di flusso che attraversa il taglio è esattamente uguale alla capacità di taglio, e valendo la relazione (2) il flusso non può che essere quello massimo e il taglio quello di capacità minima e quindi vale (3).

Il ragionamento fatto contiene in sé anche un'idea algoritmica per trovare il massimo flusso: si itera generando cammini aumentanti e si termina quando non esistono più cammini aumentanti. Tuttavia, se attuata ingenuamente, quest'idea porta ad un algoritmo solo pseudopolinomiale. Esaminiamo dapprima come effettuare la ricerca di un cammino aumentante.

I nodi vengono ripartiti in tre insiemi: nodi raggiunti e processati S, nodi raggiunti e non ancora processati R, nodi non ancora raggiunti T. Inizialmente $S = \emptyset$, $R = \{s\}$ e $T = N \setminus \{s\}$. Un generico passo d'iterazione consiste nel prendere un nodo k in R valutare gli archi incidenti in k con altro estremo in T e vedere se possono appartenere ad un cammino aumentante. In particolare se il flusso è diretto da k a $j \in T$ deve essere $x_{kj} < c_{kj}$; se invece il flusso è diretto da j a k l'arco può essere aumentante. Se l'arco può essere aumentante allora il nodo j passa da T a R e viene memorizzato un puntatore da j a k per ricostruire alla fine il cammino. Terminato l'esame degli archi incidenti in k, il nodo k passa da R in S.

La procedura termina non appena $t \in R$ oppure quando $R = \emptyset$. Nel primo caso si è trovato un cammino aumentante che viene determinato usando ricorsivamente i puntatori a partire da t. In questa fase si determina anche la quantità di flusso che può essere inviata sul cammino. Nel secondo si è determinato che non esiste nessun cammino aumentante e quindi si è trovato il massimo flusso e un taglio di minima capacità indotto da S.

La procedura di ricerca di un cammino aumentante prende in esame al più un arco alla volta e quindi la sua complessità è $O(|E|)$. In generale però non ci sono garanzie che il numero globale di iterazioni a partire dalla soluzione nulla sia polinomiale nei dati del problema. Infatti tutto quello che si può dire è che, in presenza di dati di capacità interi, il flusso aumenta almeno di una unità per ogni cammino aumentante, ma questo porta ad un numero di iterazioni pseudopolinomiale.

Si può tuttavia dimostrare che se la ricerca del cammino aumentante viene eseguita in larghezza, cioè i nodi vengono scelti da R secondo la regola first-in-first-out, allora il numero di iterazioni è polinomiale, in particolare $O(n^2\, m)$. Con ricerche di cammini aumentanti non necessariamente ad albero si ottiene una complessità $O(n^3)$. Vi sono infine algoritmi di massimo flusso molto complicati con complessità $O(m\, n\, \log(n^2/m))$, che per grafi densi ($m = \Theta(n^2)$) è comunque $O(n^3)$ e per grafi sparsi ($m = O(n)$) invece si abbassa a $O(n^2\, \log n)$.

Esempio. Si riconsideri la rete in figura 1. Scegliendo i cammini aumentanti con una ricerca in larghezza i nodi vengono raggiunti (ad esempio scegliendo i nodi da raggiungere in T in ordine di indice crescente) secondo l'ordine (indicando l'arco con cui il nodo viene raggiunto) (1,2), (1,4), (1,5), (2,3), (4,7), (4,8), (5,6), (5,9). Quindi si è trovato il cammino aumentante $1 \to 5 \to 9$ sul quale possono essere inviate due unità di flusso. Ripetendo la ricerca del cammino aumentante i nodi vengono raggiunti secondo l'ordine (1,2), (1,4), (1,5), (2,3), (4,7), (4,8), (5,6), (8,9). Si è trovato il cammino aumentante $1 \to 4 \to 8 \to 9$ sul quale può essere inviata una unità di flusso. Nella terza iterazione i nodi vengono raggiunti secondo l'ordine (1,2), (1,4), (1,5), (2,3), (4,7), (5,6), (7,8), (6,9). Si è trovato il cammino aumentante $1 \to 5 \to 6 \to 9$ sul quale può essere inviata una unità di flusso. In totale finora il flusso in uscita dalla sorgente è di 4 unità, rappresentato in figura 2

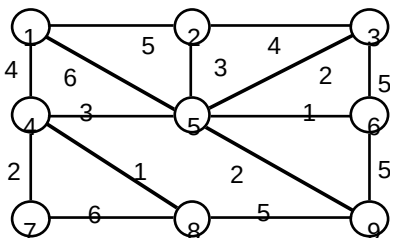


Figura 1 - capacità

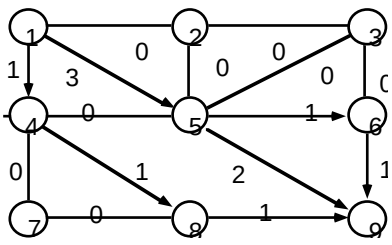


Figura 2

Nella quarta iterazione i nodi vengono raggiunti secondo l'ordine (1,2), (1,4), (1,5), (2,3), (4,7), (3,6), (7,8), (6,9), generando il cammino $1 \to 2 \to 3 \to 6 \to 9$ sul quale si possono inviare 4 unità di flusso. La nuova soluzione è in figura 3. Nella quinta iterazione i nodi vengono raggiunti secondo l'ordine (1,2), (1,4), (1,5), (4,7), (5,3), (7,8), (3,6), (8,9), generando il cammino $1 \to 4 \to 7 \to 8 \to 9$ sul quale si possono inviare 2 unità di flusso. La nuova soluzione è in figura 4.

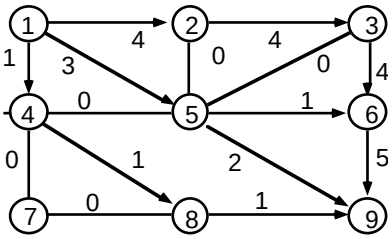


Figura 3

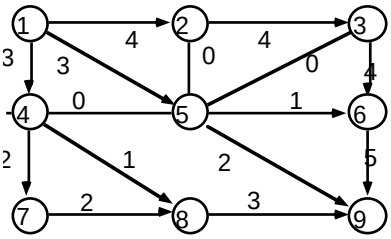


Figura 4

Nella sesta iterazione i nodi vengono raggiunti secondo l'ordine (1,2), (1,4), (1,5), (5,3), (3,6). Altri nodi non vengono raggiunti in quanto gli archi $(2,3)$, $(4,7)$, $(4,8)$, $(5,6)$, $(5,9)$ e $(6,9)$ sono saturi. Quindi la soluzione corrente è quella di massimo flusso e il taglio di minima capacità è quello indotto dall'insieme $S = \{1,2,3,4,5,6\}$ con valore uguale a 10. ∎

Se in particolare le capacità sono unitarie il problema del massimo flusso diventa il problema di trovare il massimo numero di cammini disgiunti negli archi che connettono due nodi dati. Il teorema del massimo flusso-minima capacità può pertanto essere riformulato come: il massimo numero di cammini disgiunti negli archi fra due nodi dati è uguale al minimo numero di archi necessario a sconnettere i nodi dati.

Piccolo aneddoto storico sul problema del massimo flusso: quanto segue è stato portato alla luce da Schrijver e raccontato ad un recente congresso. Il problema del massimo flusso che apparentemente fu studiato per la prima volta negli Stati Uniti da Ford e Fulkerson negli anni 50, fu in realtà oggetto d'indagine in anni anteriori in Unione Sovietica come risulta da un articolo (in russo), che si trova nella Biblioteca Lenin di Mosca e in cui si descrive il problema di trasportare tramite ferrovia il massimo numero di truppe dalle frontiere europee al Pacifico (o viceversa). Gli autori dell'articolo, pur non usando i concetti poi sviluppati da Ford e Fulkerson, trovano una soluzione che, come ha verificato Schrijver, risulta corretta. Nella sua ricerca storica Schrijver si è poi imbattuto in una citazione contenuta nei lavori di Ford a Fulkerson in merito ad una particolare applicazione del problema. L'articolo citato però era ancora classificato. Inoltrata e ottenuta la richiesta di declassificazione al Pentagono, si è scoperto che anche gli americani erano interessati ad una applicazione ferroviaria del problema, solo che questa riguardava il sistema ferroviario ... dell'Unione Sovietica. Evidentemente, mentre i russi erano interessati al massimo flusso, gli americani erano interessati al minimo taglio...

2. Tagli di capacità minima

Spesso viene richiesto di trovare in un grafo non orientato un taglio di capacità minima. Si noti che il taglio indotto da un insieme S ha la stessa capacità di quello indotto dall'insieme $N \setminus S$ data la simmetria dei valori di capacità su ogni arco in entrambe le direzioni.

Un caso particolarmente rilevante consiste nel trovare il taglio con il minor numero di archi che sconnette il grafo. Se il grafo rappresenta una rete di comunicazione (vedi per esempio il caso sopra citato delle ferrovie russe) è importante sapere qual è il minimo numero di archi necessari a sconnettere il grafo. Tanto più piccolo sarà questo numero tanto meno affidabile sarà la rete. Per questo problema, come detto precedentemente, basta porre capacità unitarie su ogni arco.

Un modo per risolvere il problema consiste nel risolvere ripetutamente un problema di massimo flusso. Si sceglie arbitrariamente un nodo come sorgente e poi si risolvono $n-1$ problemi di massimo flusso assumendo come destinazione a turno uno degli altri nodi e prendendo il minimo dei tagli di minima capacità trovati.

Nessun taglio viene trascurato da questa procedura. Infatti assegnato un taglio arbitrario indotto da un insieme S di nodi (si può sempre scegliere S in modo che $s \in S$, data la simmetria esposta sopra) esiste almeno un nodo $k \notin S$. Questo taglio viene considerato quando si risolve il problema del massimo flusso da s a k. La complessità di questo metodo dipende dall'algoritmo di massimo flusso usato. Usando un algoritmo particolarmente veloce si ottiene una complessità $O(m\, n^2\, \log(n^2/m))$.

Esistono comunque anche algoritmi diretti che non usano concetti di flusso. Presentiamo due algoritmi, il secondo dei quali è stocastico e permette di trovare la soluzione solo con probabilità prefissata.

L'idea del primo algoritmo si basa sull'osservazione che, dati due nodi qualsiasi, o il taglio minimo separa i due nodi oppure non li separa. In questo secondo caso i due nodi possono essere fusi in uno fondendo eventualmente archi incidenti in entrambi i nodi e sommandone le capacità, e il taglio minimo del grafo ridotto è uguale a quello del grafo originario. Quindi, se si è in grado di trovare due nodi di cui si conosce il minimo taglio separatore, basta confrontare questo valore con quello del minimo taglio del grafo ridotto. Ricorsivamente nel grafo ridotto si determinano due nodi di cui si conosce il minimo taglio separatore e si prosegue finché il grafo è ridotto a due soli nodi. A questo punto basta confrontare tutti i tagli generati e prendere il migliore.

La parte difficile consiste ovviamente nel determinare due nodi e il minimo taglio che li separa. Naturalmente non è pensabile definire a priori i due nodi, altrimenti dovremmo risolvere un problema di massimo flusso per calcolare il minimo taglio. La procedura per generare due nodi di cui si conosca anche il minimo taglio separatore è abbastanza semplice, anche se la dimostrazione di correttezza non è banale.

Per descrivere l'algoritmo è necessaria la seguente definizione: dati due insiemi A e B sia

$$c(A : B) := \sum_{\substack{i \in A \\ j \in B}} c_{ij}$$

L'algoritmo procede nel seguente modo: si sceglie arbitrariamente un nodo come nodo iniziale v_1 e si pone $S := \{v_1\}$. Ad esempio sia $v_1 := 1$. Poi si sceglie un nodo v_2 come

$$v_2 := \operatorname*{argmax}_{k \notin S} \; c(S : \{k\})$$

e si aggiorna $S := S \cup \{v_2\}$. La procedura viene ripetuta fino a definire il nodo v_n. A questo punto si calcola la capacità del taglio indotto da v_n, cioè $c(\{v_n\}) =: C_n$.

Terminata questa fase, i nodi v_{n-1} e v_n vengono fusi generando il grafo G'. La procedura viene ripetuta sul grafo G' generando una successione di nodi (possibilmente diversa dalla precedente) $v'_1, \ldots, v'_{n-1}$ e una capacità di taglio $C_{n-1} := c(\{v'_{n-1}\})$.

La procedura viene ripetuta fino ad avere un grafo di due nodi in cui il valore C_2 è la capacità dell'unico arco del grafo e che corrisponde al taglio indotto da $v_1 = 1$ sul grafo originale (per costruzione il nodo 1 non viene mai fuso con altri nodi). Infine si sceglie il minimo fra i valori C_i. Questo individua il taglio di minima capacità.

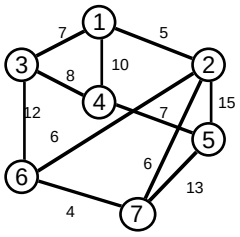


Figura 5

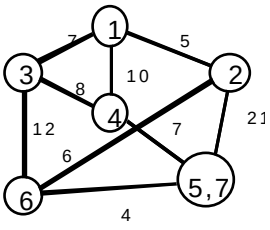


Figura 6

Ad esempio dato il grafo in figura 5, la sequenza di nodi nella prima fase è $v_1 = 1$, $v_2 = 4$, $v_3 = 3$, $v_4 = 6$, $v_5 = 2$, $v_6 = 5$, $v_7 = 7$. Quindi $C_7 = c(\{7\}) = 23$. Si fondono i nodi 5 e 7 ottenendo il grafo in figura 6. Su questo grafo si ripete la procedura ottenendo il seguente ordine di nodi $v_1 = 1$, $v_2 = 4$, $v_3 = 3$, $v_4 = 6$, $v_5 = (5, 7)$, $v_6 = 2$. Quindi $C_6 = c(\{2\}) = 32$. Si fondono i nodi 2 e (5,7) ottenendo il grafo in figura 7.

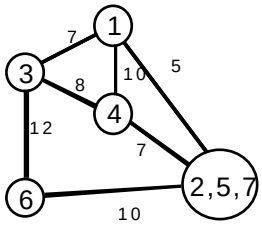


Figura 7

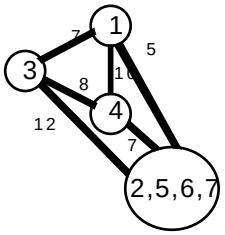
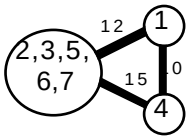


Figura 8

Si ottiene il seguente ordine di nodi $v_1 = 1$, $v_2 = 4$, $v_3 = 3$, $v_4 = (2, 5, 7)$, $v_5 = 6$. Quindi $C_3 = c(\{6\}) = 22$. Si fondono i nodi 6 e (2,5,7) ottenendo il grafo in figura 8. Si ottiene il seguente ordine di nodi $v_1 = 1$, $v_2 = 4$, $v_3 = 3$, $v_4 = (2, 5, 6, 7)$. Quindi $C_2 = c(\{(2, 5, 6, 7)\}) = 24$. Si fondono i nodi 3 e (2,5,6,7) ottenendo il grafo in figura 9.



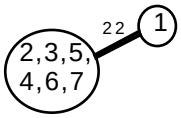
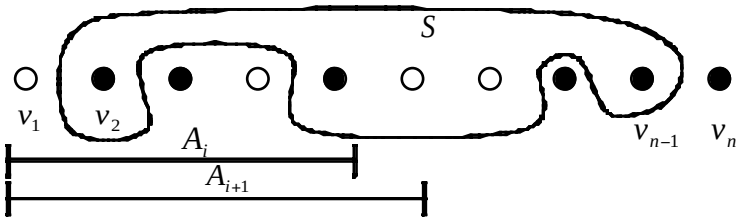


Figura 9

Si ottiene il seguente ordine di nodi $v_1 = 1$, $v_2 = (2, 3, 5, 6, 7)$, $v_3 = 4$. Quindi $C_3 = c(\{4\}) = 25$. Si fondono i nodi 4 e (2,3,5,6,7) ottenendo il grafo in figura. Infine $C_2 = c(\{1\}) = 22$. Confrontando i valori $C_2, \ldots, C_n$, il taglio minimo è $C_2 = 22$, indotto da $\{1\}$.

Per dimostrare la correttezza dell'algoritmo, basta dimostrare che $c(\delta(\{v_n\}))$ è il taglio di minima capacità fra tutti i tagli che separano v_{n-1} da v_n. Come detto precedentemente, se il taglio minimo separa v_{n-1} da v_n deve essere quello indotto da $\{v_n\}$, altrimenti ha i nodi v_{n-1} e v_n dalla stessa parte e quindi questi possono essere fusi e il taglio minimo si cercherà nel grafo ridotto per fusione dei nodi.

Definiamo $A_i := \{v_1, \ldots, v_i\}$. Quindi $c(A_{n-1} : \{v_n\}) = c(\delta(\{v_n\}))$. Sia S un qualsiasi insieme tale che $v_{n-1} \in S$ e $v_n \notin S$



Definiamo critici i nodi v_i tali che $v_i \in S$ e $v_{i-1} \notin S$ oppure $v_i \notin S$ e $v_{i-1} \in S$. In figura i nodi critici sono in nero. In particolare v_n è critico per ogni S che separa v_{n-1} da v_n. Sia $S_i := A_i \cap S$ e $\bar{S}_i := A_i \setminus S_i$. Quindi S_i e $\bar{S}_i$ sono una partizione di A_i.

Vogliamo dimostrare che se v_i è critico allora

$$c(A_{i-1} : \{v_i\}) \leq c(S_i : \bar{S}_i) \tag{4}$$

Siccome v_n è critico, la diseguaglianza (4) implica che $\delta(\{v_n\})$ è minimo fra i tagli che separano v_{n-1} da v_n. La dimostrazione di (4) verrà fatta per induzione sui nodi critici. Se v_h è il primo nodo critico, consideriamo i due casi $v_h \in S$ e $v_h \notin S$. Se $v_h \in S$ si ha $S_h = \{v_h\}$ e $\bar{S}_h = A_{h-1}$ e quindi (4) diventa

$$c(A_{h-1} : \{v_h\}) = c(S_h : \bar{S}_h) \tag{5}$$

Se invece $v_h \notin S$, allora $A_{h-1} \subset S$ e $S_h = A_{h-1}$ e $\bar{S}_h = \{v_h\}$ e ritroviamo (5).

Ora si supponga vera la relazione (4) per il nodo critico v_i. Dobbiamo dimostrare che la relazione è vera per il successivo nodo critico v_j. Allora si ha

$$\begin{aligned} c(A_{j-1} : \{v_j\}) &= c(A_{i-1} : \{v_j\}) + c(A_{j-1} \setminus A_{i-1} : \{v_j\}) \\ &\leq c(A_{i-1} : \{v_i\}) + c(A_{j-1} \setminus A_{i-1} : \{v_j\}) \\ &\leq c(S_i : \bar{S}_i) + c(A_{j-1} \setminus A_{i-1} : \{v_j\}) \\ &\leq c(S_j : \bar{S}_j) \end{aligned} \tag{6}$$

dove l'uguaglianza deriva dalla definizione additiva della capacità di taglio, la prima diseguaglianza dalla definizione stessa dei nodi v_i e la seconda diseguaglianza dall'ipotesi induttiva. Per quel che riguarda la terza diseguaglianza si supponga $v_i \in S$ e $v_j \notin S$. Allora $S_{j-1} = S_j$. Siccome v_i e v_j sono due nodi critici successivi si ha

$$A_{j-1} \setminus A_{i-1} = \{v_i, v_{i+1}, \ldots, v_{j-1}\} \subset S$$

da cui

$$A_{j-1} \setminus A_{i-1} = S_{j-1} \setminus S_{i-1} = S_j \setminus S_{i-1}$$

Allora

$$c(S_i : \bar{S}_i) + c(A_{j-1} \setminus A_{i-1} : \{v_j\}) = c(S_i : \bar{S}_i) + c(S_j \setminus S_{i-1} : \{v_j\}) \tag{7}$$

Da $\bar{S}_j = \bar{S}_i \cup \{v_j\}$ e dalla definizione di $c(A : B)$ abbiamo

$$c(S_j : \bar{S}_j) = c(S_j : \bar{S}_i \cup \{v_j\}) = c(S_j : \bar{S}_i) + c(S_j : \{v_j\}) \tag{8}$$

e siccome $S_j \supset S_i \supset S_{i-1}$, abbiamo

$$c(S_j : \bar{S}_i) \geq c(S_i : \bar{S}_i), \quad c(S_j : \{v_j\}) \geq c(S_j \setminus S_{i-1} : \{v_j\}) \tag{9}$$

Quindi, combinando (7), (8) e (9) si dimostra la terza diseguaglianza di (6) per il caso $v_i \in S$ e $v_j \notin S$.

Per il caso opposto $v_i \notin S$ e $v_j \in S$ si ha $\bar{S}_j = \bar{S}_{j-1}$ e $A_{j-1} \setminus A_{i-1} \cap S = \emptyset$, da cui

$$A_{j-1} \setminus A_{i-1} = \bar{S}_{j-1} \setminus \bar{S}_{i-1} = \bar{S}_j \setminus \bar{S}_{i-1}$$

Allora

$$c(S_i : \bar{S}_i) + c(A_{j-1} \setminus A_{i-1} : \{v_j\}) = c(S_i : \bar{S}_i) + c(\bar{S}_j \setminus \bar{S}_{i-1} : \{v_j\}) \tag{10}$$

Da $S_j = S_i \cup \{v_j\}$, come nel caso precedente, possiamo scrivere

$$c(S_j : \bar{S}_j) = c(S_i \cup \{v_j\} : \bar{S}_j) = c(S_i : \bar{S}_j) + c(\{v_j\} : \bar{S}_j) \tag{11}$$

e siccome $\bar{S}_j \supset \bar{S}_i \supset \bar{S}_{i-1}$, si ha

$$c(S_j : \bar{S}_j) \geq c(S_i : \bar{S}_i), \quad c(\bar{S}_j : \{v_j\}) \geq c(\bar{S}_j \setminus \bar{S}_{i-1} : \{v_j\}) \tag{12}$$

e combinando (10), (11) e (12) si dimostra la terza diseguaglianza di (6) per il caso $v_i \notin S$ e $v_j \in S$.

Per quel che riguarda la complessità computazionale l'algoritmo esegue n iterazioni. All'interno di ogni iterazione bisogna calcolare i valori $c(S : \{k\})$ e valutarne il massimo. Questo si può realizzare con delle strutture ad heap. All'interno di ogni iterazione bisogna quindi al più scandire tutti gli archi e per ogni scansione aggiornare dei valori su uno heap. Questo costa $O(m \log n)$. Le operazioni di fusione dei

nodi possono anche essere eseguite riscrivendo il grafo con complessità $O(m)$, che è comunque dominata da $O(m \log n)$ (si possono anche usare strutture Union-Find senza dover riscrivere ogni volta tutto il grafo, visto che l'aggiornamento riguarda solo due nodi e gli archi incidenti). In totale quindi si ha una complessità $O(m\,n \log n)$. Usando heap di Fibonacci la complessità può essere abbassata a $O(m\,n + n^2 \log n)$

Il secondo algoritmo è probabilistico e permette di trovare il taglio di capacità minima con probabilità arbitrariamente bassa.

Prima di descrivere l'algoritmo è utile dire cosa sia un algoritmo probabilistico. In un algoritmo probabilistico alcune scelte sono lasciate al caso secondo un preciso meccanismo probabilistico. Questo fa sì che la soluzione desiderata sia ottenuta solo con una certa probabilità p. Se si ripete l'algoritmo la probabilità che in nessuna delle due iterazioni si sia trovata la soluzione desiderata è $(1-p)^2$. In generale la probabilità di non ottenere mai la soluzione in k iterazioni è $(1-p)^k$. Se si fissa a priori un valore ε e si vuole che $(1-p)^k \leq \varepsilon$, il numero di iterazioni deve essere almeno $\log \varepsilon / \log(1-p)$. Se $1/\log(1-p)$ è polinomiale allora si parla di un algoritmo probabilistico polinomiale.

Per descrivere l'algoritmo probabilistico per il minimo taglio si definisca preliminarmente

$$C^* = \text{capacità del minimo taglio}, \qquad C_\Sigma = \sum_{e \in E} c_e, \quad C_i = \sum_{e \in \delta(i)} c_e$$

L'algoritmo consiste dei seguenti passi: un arco $\hat{e} = (u, v)$ viene scelto con probabilità c_e/C_Σ, i nodi u e v vengono fusi, $\hat{e}$ viene rimosso e gli archi eventualmente incidenti in u e in v vengono fusi (con somma delle capacità). Se $\hat{e}$ non appartiene al taglio minimo, il nuovo grafo ha il medesimo taglio minimo con lo stesso valore (infatti ogni taglio del grafo originario che non contiene $\hat{e}$ è presente anche nel grafo contratto, mentre ogni taglio del grafo originario che contiene $\hat{e}$ non esiste nel grafo contratto, inoltre i tagli corrispondenti hanno le medesime capacità di taglio). La procedura continua finché rimangono solo due nodi che individuano il taglio finale.

Illustriamo la procedura con un esempio. Sia dato il grafo rappresentato in figura 10-a con i valori di capacità indicati accanto agli archi. Supponiamo che l'arco scelto a caso sia $(3-6)$. Dopo la fusione dei nodi 3 e 6 si ha il grafo in figura 11-b. Nelle figure 11-c, 11-d, 11-e, 11-f, si vedono le varie fasi dell'algoritmo. Gli archi scelti sono rispettivamente $(1-4)$, $((1,4)-5)$, $((1,4,5)-2)$, e $((1,4,5,2)-7)$. Questo identifica il taglio indotto da $S = \{1,4,5,2,7\}$ di capacità 25 (non si tratta del taglio minimo).

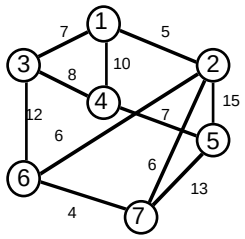


Figure 11-a

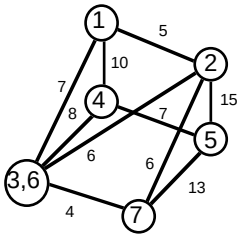


Figure 11-b

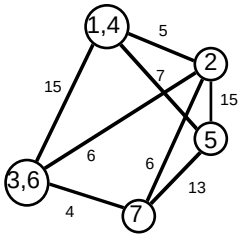


Figure 11-c

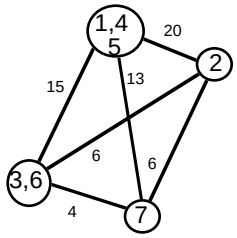


Figure 11-d

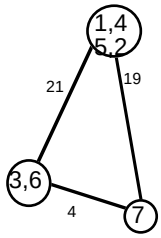


Figure 11-e

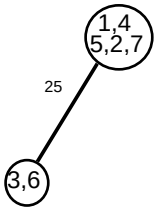


Figure 11-f

Analizziamo ora la probabilità con cui viene generato il taglio di capacità minima. Ovviamente si ha

$$C_i \geq C^*$$

e sommando su tutti i nodi si ottiene

$$\sum_i C_i \geq n\, C^*$$

E siccome

$$\sum_i C_i = 2\, C_\Sigma$$

si ottiene

$$\frac{C^*}{C_\Sigma} \leq \frac{2}{n}$$

Quindi la probabilità di non scegliere un arco del taglio minimo è almeno

$$1 - \frac{2}{n}$$

Nel grafo successivo con $(n-1)$ nodi la probabilità diventa

$$1 - \frac{2}{n-1}$$

L'ultimo grafo ha 3 nodi e quindi la probabilità è

$$1 - \frac{2}{3}$$

Quindi la probabilità di non scegliere mai un arco del taglio minimo (e quindi fornire come soluzione proprio il taglio minimo) è almeno

$$(1 - \frac{2}{n})\,(1 - \frac{2}{n-1})\,(1 - \frac{2}{n-2})\,\ldots\,(1 - \frac{2}{5})\,(1 - \frac{2}{4})\,(1 - \frac{2}{3}) = \frac{2}{n\,(n-1)} \geq \frac{2}{n^2}$$

e la probabilità di non trovare il taglio minimo è al più

$$1 - \frac{2}{n^2}$$

dopo k tentativi la probabilità di non trovare mai il taglio minimo è al più

$$(1 - \frac{2}{n^2})^k \approx e^{-2k/n^2}$$

quindi

$$e^{-2k/n^2} \leq \varepsilon \quad \Longrightarrow \quad -\frac{2\,k}{n^2} \leq \ln \varepsilon \quad \Longrightarrow \quad k \geq \frac{n^2\, \ln \varepsilon^{-1}}{2}$$

Per valutare la complessità bisogna considerare quanto costa ogni singola iterazione: ogni fusione realizzata con struttura Union-Find ha un costo $O(\log n)$. La scelta casuale di un arco richiede una ricerca binaria fra gli archi ancora da scegliere ed ha complessità $O(\log m) = O(\log n)$. Anche l'aggiornamento della struttura dati per la scelta casuale ha complessità $O(\log m)$. Il numero di iterazioni per la ricerca di un taglio è al più pari a $m-1$ (ad esempio due cliques connesse con un solo arco potrebbero richiedere la scelta di tutti gli archi delle due cliques). Quindi un taglio si trova con complessità $O(m \log n)$. Quindi per avere il taglio minimo con probabilità $1-\varepsilon$, la complessità è $O(m\, n^2 \log n \log \varepsilon^{-1})$.

La limitazione trovata alla probabilità di trovare il taglio minimo è stretta. Si consideri un circuito con n archi e capacità M per due archi e $M+1$ per gli altri $n-2$ archi. Allora $C^* = 2\,M$, $C_\Sigma = n\,M + n - 2$ e

$$\frac{C^*}{C_\Sigma} = \frac{2\,M}{n\,M + n - 2} \to \frac{2}{n} \quad \text{se} \quad M \to \infty$$

Quindi il rapporto può essere reso arbitrariamente vicino alla limitazione e questo fatto è vero anche per i successivi grafi ottenuti per fusione.