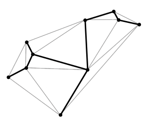
OTTIMIZZAZIONE COMBINATORIA

Prof. Marco Trubian
6 CFU

Luca Cappelletti

Lecture Notes
Year 2017/2018



Magistrale Informatica
Università di Milano
Italy
9 novembre 2018

Indice

1 Introduzione

L' **Ottimizzazione combinatoria** propone modelli di soluzioni ad innumerevole problemi, tra i quali vi sono:

Matching covers Consideriamo due insiemi A e B, di cardinalità n: ad ogni coppia di valori del prodotto cartesiano dei due insiemi è associato un valore positivo che descrive la compatibilità tra i due valori. Si vanno a scegliere n coppie, senza che gli elementi vengano ripetuti, in modo da massimizzare la compatibilità totale.

Set Covering Data una *matrice binaria* ed un vettore di costi associati alle colonne si va a realizzare il sottoinsieme di costo minimo che copra tutte le righe.

Set Packing Data una *matrice binaria* ed un vettore di valori associati alle colonne, si cerca il sottoinsieme di colonne di valore massimo tali che non coprino entrambe una stessa riga.

Set Partitioning Data una *matrice binaria* ed un vettore di cosi associati alle colonne, si cerca il sottoinsieme di colonne di costo minimo che copra tutte le righe senza conflitti.

Vertex Cover Dato un grafo non orientato $G = (V, E)$ si cerca il sottoinsieme di vertici di cardinalità minima tale che ogni lato del grafo vi incida.

Maximum Clique Problem Dato un grafo non orientato e una funzione peso definita sui vertici, si cerca il sottoinsieme di vertici fra loro adiacenti di peso massimo.

Maximum Independent Set Problem Dato un grafo non orientato e una funzione di peso definita sui vertici, si cerca il sottoinsieme di vertici fra loro non adiacenti di peso massimo.

Minimum Steiner Tree Dato un grafo non orientato e una funzione costo definita sui lati, si cerca un albero ricoprente di costo minimo.

Boolean satisfiability problem or SAT Data una forma normale congiunta (CNF), si cerca un assegnamento di verità alle variabili logiche che la soddisfi.

Versione pesata (MAX-SAT) Viene considerata anche una funzione peso associata alle formule che compongono la CNF. L'obbiettivo è massimizzare il peso totale delle formule soddisfatte.

2
Matching Covers

2.1 Matching

Definizione 2.1.1 (Matching o Accoppiamento). Dato un grafo $G = (V, E)$, un **matching** è un sottoinsieme $M \subseteq E$ di archi *a due a due non adiacenti.*

Definizione 2.1.2 (Matching massimo). Matching M^* di cardinalità massima.

Definizione 2.1.3 (Matching bipartito). Se il grafo G è **bipartito**, allora anche M si dice **bipartito**.

Definizione 2.1.4 (Matching perfetto). Se la cardinalità del matching è pari a metà del numero di vertici, allora si dice **perfetto**:

$$|M| = \frac{|V|}{2}$$

Definizione 2.1.5 (Matching massimale). Un matching M si dice **massimale** se ogni elemento di $E \setminus M$ è adiacente ad almeno un elemento di M.

Un matching massimale **non** necessariamente è massimo, mentre un matching massimo è sempre massimale.

2.2 Insieme stabile

Definizione 2.2.1 (Insieme stabile o indipendente). Dato un grafo simmetrico $G = (V, E)$, un qualunque sottoinsieme S di vertici si dice **indipendente** o **stabile** se esso è costituito da elementi a due a due non adiacenti.

Definizione 2.2.2 (Insieme stabile massimo). Un insieme stabile S^* si dice **massimo** se $|S^*| \geq |S|$, per ogni insieme stabile S di G.

Definizione 2.2.3 (Insieme stabile massimale). Un insieme stabile S si dice **massimale** se ogni elemento di $V \setminus S$ è adiacente ad almeno un elemento di S.

2.3 Copertura

Definizione 2.3.1 (Copertura). Dato un grafo simmetrico $G = (V, E)$, un qualunque sottoinsieme T di vertici (F di archi) tale che ogni arco di E (vertice di V) incide su almeno un elemento di T (di F) si dice **copertura**. In particolare, l'insieme T è detto **trasversale** o **vertex cover** mentre l'insieme F è detto **edge cover**.

Definizione 2.3.2 (Copertura minima). Una copertura X^* si dice **minima** se $|X^*| \leq |X|$, per ogni insieme copertura X di G.

Definizione 2.3.3 (Copertura minimale). Una copertura X si dice **minimale** se $X \setminus \{x\}$ non è una copertura per ogni $x \in X$.

2.4 Disuguaglianze duali deboli

Teorema 2.4.1 (Disuguaglianze duali deboli). Indichiamo con $\alpha(G)$ l'**insieme stabile massimo** di G, con $\mu(G)$ il **matching massimo** di G, con $\rho(G)$ l'**edge cover minimo** di G e $\tau(G)$ **trasversale minimo** di G. Per un grafo G valgono le seguenti due disuguaglianze:

$$\alpha(G) \leqslant \rho(G)$$
$$\mu(G) \leqslant \tau(G)$$

L'**insieme stabile minimo di un grafo** è sempre di cardinalità minore dell'**edge cover minimo**. Inoltre, l'**insieme stabile massimo** è sempre minore o uguale dell'**insieme trasversale minimo**.

Dimostrazione 2.4.2 (Disuguaglianze duali deboli). Siano X l'**insieme stabile** di G e Y l'**edge cover** di G.

Poiché Y copre V, ogni elemento di X incide su almeno un elemento di Y.

D'altra parte, nessun elemento di Y copre contemporaneamente due elementi di X altrimenti i due elementi sarebbero adiacenti e quindi non potrebbero appartenere all'insieme stabile X.

Pertanto, per ogni $x \in X$ esiste un distinto $y \in Y$ che lo copre, e quindi $|X| \leqslant |Y|$.

Riscrivendo la precedente relazione per gli insiemi massimi X^* e Y^* si ottiene:

$$\alpha(G) \leqslant \rho(G)$$

Scambiando il ruolo di V ed E, si ottiene $\mu(G) \leqslant \tau(G)$. $\square$

2.5 Teorema di Gallai

Teorema 2.5.1 (Teorema di Gallai). Per ogni grafo G con n nodi si ha:

$$\alpha(G)+\tau(G)=n$$

Se inoltre G non ha nodi isolati

$$\mu(G)+\rho(G)=n$$

Dimostrazione 2.5.2 (Teorema di Gallai). Iniziamo ottenendo la prima equazione: Sia S un insieme stabile di G. Allora $V\setminus S$ è un insieme trasversale. In particolare, $|V\setminus S|\geqslant\tau(G)$. Se consideriamo l'insieme stabile massimo S^*, otteniamo:

$$\tau(G)\geqslant\left|V\setminus S^*\right|=n-\alpha(G)$$

da cui ricaviamo:

$$\alpha(G)+\tau(G)\leqslant n$$

Viceversa, sia T un insieme trasversale di G. Allora $V\setminus T$ è un insieme stabile.

In particolare, $|V-T|\leqslant\alpha(G)$.

Se consideriamo l'insieme trasversale minimo T^*, otteniamo:

$$\alpha(G)\geqslant\left|V\setminus T^*\right|=n-\tau(G)$$

da cui ricaviamo

$$\alpha(G)+\tau(G)\geqslant n$$

Considerando la condizione ottenuta precedentemente possiamo concludere che:

$$\alpha(G)+\tau(G)=n$$

Procediamo a dimostrare la seconda equazione Sia G un grafo privo di nodi isolati e sia M^* il matching massimo di G. Indichiamo con V_{M^*} i nodi che sono estremi degli archi in M^*.

Sia H un insieme minimale di archi tale che ogni nodo in $V\setminus V_{M^*}$ è estremo di qualche arco in H.

Segue che:

$$|H|=|V\setminus V_{M^*}|=n-2\left|M^*\right|$$

Osserviamo che l'insieme $C=H\cup M^*$ è un edge-cover di G.

Sicuramente, $|C|\geqslant\rho(G)$, quindi:

$$\rho(G)\leqslant|C|=\left|M^*\right|+|H|=\left|M^*\right|+n-2\left|M^*\right|=n-\left|M^*\right|=n-\mu(G)$$

da cui ricaviamo:

$$\rho(G)+\mu(G)\leqslant n$$

Sia C il minimo edge-cover su G, cioè tale che $|C|=\rho(G)$ e sia $H=(V,C)$ il sottografo indotto da C. Valgono quindi le seguenti proprietà:

1. H è un grafo aciclico.
2. Ogni cammino di H è composto al più da due archi.

Dalle proprietà precedenti concludiamo che il grafo $H=(V,C)$ ha $|V|=n$ vertici e $|C|=\rho(G)$ archi. Può infine essere decomposto in N componenti connesse aventi la forma di stella.

Consideriamo l'i-esima componente connessa di H. Indichiamo con s_i il numero di nodi della componente connessa e con s_i-1 il numero di archi della componente connessa. Pertanto:

$$n=\sum_{i=1}^{N}s_i \quad \text{e} \quad \rho(G)=\sum_{i=1}^{N}(s_i-1)=n-N\Rightarrow N=n-\rho(G)$$

Sia M un matching con un arco per ogni componente di H. Si ottiene:

$$\mu(G)\geqslant|M|=n-\rho(G)\Rightarrow\rho(G)+\mu(G)\geqslant n$$

Considerando la condizione ottenuta precedentemente, possiamo concludere che:

$$\rho(G)+\mu(G)=n$$

□

2.6 Cammino alternante e aumentante

Sia M un matching di $G = (V, E)$.

Definizione 2.6.1 (Arco accoppiato). Un arco $(i, j) \in E$ si dice **accoppiato** se:

$$(i, j) \in M$$

Altrimenti è detto **libero**.

Definizione 2.6.2 (Vertice accoppiato). Un vertice $i \in V$ si dice **accoppiato** se su di esso incide un arco di M. Altrimenti si dice che **non incide**.

Definizione 2.6.3 (Cammino alternante). Un cammino P sul grafo G si dice **alternante** rispetto a M se esso è costituito alternativamente da archi accoppiati e liberi.

Definizione 2.6.4 (Cammino aumentante). Un cammino P *alternante* rispetto ad M che abbia entrambi gli estremi esposti si dice **aumentante**.

Teorema 2.6.5. Sia M un matching di G e sia P un cammino aumentante rispetto a M. La differenza simmetrica:

$$M' = (M \setminus P) \cup (P \setminus M)$$

È un matching di cardinalità $|M| + 1$.

Dimostrazione 2.6.6. Sia M un matching di G e sia P un cammino aumentante rispetto a M. L'insieme $M' = (M \setminus P) \cup (P \setminus M)$ gode delle seguenti proprietà:

1. M' è un matching:
   (a) I nodi che non sono toccati da P non è cambiato nulla: su di essi incide un solo arco di M che ora appartiene anche ad M'.
   (b) Sui nodi intermedi di P incide soltanto un arco di $P \setminus M$, e quindi di M'.
   (c) I nodi estremi di P prima erano esposti e adesso sono accoppiati e su di essi incide soltanto un arco di $P \setminus M$.
2. M' ha un elemento in più di M:
   (a) Sia $|M| = m_1 + m_2$ con $m_1 = |M \setminus P|$ ed m_2 = numero di archi del matching appartenenti al cammino.
   (b) Poiché P è aumentante, $|P| = m_2 + (m_2 + 1)$ dove $m_2 + 1 = |P \setminus M|$.
   (c) $\left|M'\right| = |M \setminus P| + |P \setminus M| = m_1 + m_2 + 1 = |M| + 1$

□

Teorema 2.6.7 (Teorema di Berge). Un matching M di G è massimo **se e solo se** G non ammette cammini aumentanti rispetto a M.

Dimostrazione 2.6.8 (Teorema di Berge). La condizione sufficiente segue dal teorema precedente. Per la condizione necessaria, facciamo vedere che, se non esistono cammini aumentanti rispetto a un certo matching M, allora quel matching M è massimo:

Supponiamo che G ammetta un matching M' con un elemento in più di M. Vogliamo dimostrare che allora esiste un cammino aumentante per M.

Consideriamo l'insieme di archi:

$$F = \{M' \cup M\} \setminus \{M' \cap M\}$$

e sia G' il sottografo di G avente gli stessi nodi di G ma contenente solo l'insieme di archi di F. Analizziamo il grado di ciascun nodo di G', considerando tutti i casi possibili:

1. Un nodo su cui incide lo stesso arco appartenente sia ad M che ad M' è un nodo isolato su G' e quindi ha grado 0.
2. Un nodo su cui incide sia un arco di M sia un arco di M' è un nodo che ha grado 2 su G'.
3. Un nodo su cui incide un arco di M e nessun arco di M' o viceversa è un nodo che ha grado 1 su G'.
4. Un nodo esposto sia rispetto ad M che rispetto ad M' è un nodo isolato su G' e quindi ha grado 0.

Pertanto in G' nessun nodo ha un grado superiore a 2 e possiamo concludere che le componenti connesse di G' sono o nodi isolati o percorsi o cicli.

Nessun ciclo può essere dispari altrimenti ci sarebbero due archi dello stesso matching incidenti sullo stesso nodo e questo è impossibile.

Non possono essere tutti cicli pari altrimenti $|M| = \left|M'\right|$. Deve esistere una componente connessa che è un percorso.

Non tutti i percorsi possono essere pari altrimenti, nuovamente, $|M| = \left|M'\right|$.

Quindi, senza perdita di generalità, possiamo assumere che esista un percorso dispari che inizia e termina con un arco di M'.

Questo percorso è aumentante per M. ☐

2.7 Teorema del cammino aumentante

Teorema 2.7.1 (Teorema del cammino aumentante). Sia v un vertice esposto in un matching M. Se non esiste un cammino aumentante per M che parte da v, allora esiste un matching massimo avente v esposto.

Dimostrazione 2.7.2 (Teorema del cammino aumentante). Sia M^* un matching massimo in cui v è accoppiato. Consideriamo $\{M^* \cup M\} \setminus \{M^* \cap M\}$: questo insieme non può contenere un cammino alternante con i vertici degli archi di M esposti, altrimenti sarebbe aumentante per esso.

Però deve contenere un cammino composto dallo stesso numero di archi dei due insiemi, M e M^*: un cammino con un solo arco di un insieme, infatti, sarebbe aumentante per l'altro e viceversa.

Consideriamo quindi un cammino P composto da un ugual numero di archi dai due insiemi e consideriamo un nuovo matching $M' = \{M^* \cup P\} \setminus \{M^* \cap P\}$. Vanno osservate due proprietà:

1. La cardinalità del nuovo insieme e del matching massimo sono uguali:

$$\left|M'\right| = \left|M^*\right|$$

2. Il nodo v è esposto rispetto ad M'.

Pertanto abbiamo individuato un nuovo matching massimo con v esposto. □

2.8 Teorema di König

Teorema 2.8.1 (Teorema di König). Se $G = (X, Y, E)$ è un grafo bipartito, allora $\mu(G) = \tau(G)$.

Dimostrazione 2.8.2 (Teorema di König). Sia M^* un matching massimo, e siano:

1. X_1 un insieme dei nodi x di X **accoppiati** rispetto ad M^*
2. X_2 un insieme dei nodi x di X **esposti** rispetto ad M^*
3. Y_1 insieme dei nodi y di Y raggiungibili da x in X_2. Questi nodi, per definizione, sono **accoppiati** altrimenti M^* non sarebbe massimo.
4. $Y_2 = Y \setminus Y_1$

Definizione 2.8.3 (Nodo raggiungibile). Un nodo $y \in Y$ è raggiungibile se esiste P alternante rispetto ad M^* da x in X_2 tale che l'ultimo arco non appartiene ad M^*.

Consideriamo un set di nodi Z definito come:

$$Z = \{z_1, z_2, \dots, z_{\mu(G)}\} \quad \text{con} \quad \begin{cases} z_i = y_i & \text{se } y_i \text{ è raggiungibile} \\ z_i = x_i & \text{altrimenti} \end{cases}$$

Procediamo ora a dimostrare che il set Z è *trasversale*.

Iniziamo dimostrando che non esistono archi da nodi in X_2 verso nodi in Y non coperti da Z:

1. Non può esistere un arco non coperto da Z tra un nodo in X_2 e un nodo in Y_2, altrimenti il matching non sarebbe massimo.
2. Non può esistere un arco non coperto da Z tra un nodo in X_2 e un nodo in Y_1 perché i nodi in Y_1 sono raggiungibili e quindi l'arco necessariamente deve essere coperto.

Dimostriamo ora che non esistono archi da nodi in X_1 verso nodi in Y non coperti da Z:

Consideriamo un arco da X_1 a Y_2: se non fosse coperto, allora esisterebbe un nodo, estremo dell'arco del matching, raggiungibile da X_2 in Y_2. Ciò implicherebbe l'esistenza di un cammino aumentante ed il matching sarebbe pertanto non massimo.

Consideriamo ora un arco da X_1 a Y_1: se il nodo terminale non fosse coperto non sarebbe raggiungibile (per la definizione di Y_1 e di Z) e non apparterrebbe in primo luogo a Y_1, quindi l'arco non esisterebbe.

Pertanto, Z è un insieme trasversale di cardinalità pari a $\mu(G)$. □

Dimostrazione 2.8.4 (Teorema di König (Rizzi, 1999)). Chiamiamo $|G|_M$ la cardinalità massima di un matching di G e $|G|_C$ la cardinalità minima di una cover di G. Vale che $|G|_M \leq |G|_C$.

Sia G = (V, E) un contro esempio minimo: il più piccolo grafo **connesso** che non sia un **circuito** ne un **cammino**.

Ne segue che G ha un nodo u di grado almeno 3.

Sia v uno dei nodi vicini di u: se $|G \setminus v|_M < |G|_M$, allora per minimalità $G \setminus v$ ha una cover W' tale che $\left|W'\right| < |G|_M$.

Ne segue che $W' \cup \{v\}$ è una cover di G con cardinalità al più $|G|_M$.

Possiamo assumere di conseguenza che deve esistere un matching massimo M di G privo di lati incidenti a v.

Sia f un lato di $G \setminus M$ incidente a u ma non a v. Sia W' una cover di $G \setminus f$ con $\left|W'\right| = |G|_M$. Siccome nessun lato di M è incidente a v, ne segue che W' non contiene v.

Di conseguenza, W' contiene u ed è una cover di G. □

2.9 Teorema di Hall o dei matrimoni

Teorema 2.9.1 (Teorema di Hall). Sia $G = (V_1 \cup V_2, E)$ un grafo bipartito con $|V_1| \geqslant |V_2|$. Dato un insieme di vertici $J \subset V_2$, sia $\Gamma(J)$ il set dei vertici in V_1 che sono adiacenti a qualche vertice in J. Allora il grafo G ammette un **matching massimo** se e solo se:

$$|\Gamma(J)| \geqslant |J| \quad \forall J \subset V_2$$

Dimostrazione 2.9.2 (Teorema di Hall). Sia M un **matching massimo** di G e J un sottoinsieme qualsiasi di V_2. Sia $E(J)$ l'insieme dei lati contenuti nel matching M che sono incidenti con un vertice in J. Allora, i vertici terminali dei lati in $E(J)$ che sono contenuti in V_1 formano un sottoinsieme di cardinalità $|J|$ di $\Gamma(J)$.

Per assurdo, supponiamo che la condizione $|\Gamma(J)| \geqslant |J|$ è valida e che la cardinalità massima di un matching in G è minore di $|T|$. Allora per il **Teorema di König** è possibile costruire una **vertex cover** $X = V_1' \cup V_2'$ dove $V_1' \subset V_1$ e $V_2' \subset V_2'$ e $\left|S'\right| + \left|T'\right| < |T|$ ma quindi i vertici terminali u dei lati $\overline{uv}$, dove v è uno dei $|V_2| - \left|V_2'\right|$ vertici in $V_2 \setminus V_2'$ sono tutti contenuti in V_1, quindi:

$$\left|\Gamma(V_2 \setminus V_2')\right| \leqslant \left|V_1'\right| < |V_2| - \left|V_2'\right| = \left|V_2 \setminus V_2'\right|$$

da cui la contraddizione. □

2.10 Formula di Tutte-Berge e Teorema di Tutte

2.10.1 Formula di Tutte-Berge

Teorema 2.10.1 (Formula di Tutte-Berge). La dimensione di un matching massimo in un grafo G = (V, E) è pari a:

$$\frac{1}{2} \min_{U \subseteq V} \{|U| - \operatorname{odd}(G - U) + |V|\}$$

dove $\operatorname{odd}(H)$ è il numero di componenti connesse del grafo H con un numero dispari di vertici.

Questa formula implica immediatamente una condizione per l'esistenza di un matching perfetto: il Teorema di Tutte.

2.10.2 Teorema di Tutte

Teorema 2.10.2 (Teorema di Tutte). Un grafo G = (V, E) contiene un matching perfetto se e solo se per ogni sottoinsieme di nodi A vale che:

$$\operatorname{odd}(G \setminus A) \leqslant |A|$$

2.11 Proposizioni sugli alberi alternanti

Definizione 2.11.1 (Albero alternante). Un albero alternante è costruito a partire da un matching M di un grafo G = (V, E) ed un nodo $r \in V$ di M esposto.

Costruiamo iterativamente set di nodi A e B, in modo tale che ogni nodo A sia al termine di un cammino alternante di M di lunghezza dispari che inizia con r, mentre ogni nodo B al termine di quelli pari.

Tali insiemi possono essere costruiti iniziando da $A = \emptyset, B = \{r\}$ e seguendo la regola seguente:

$$vw \in E, v \in B, w \notin A \cup B, wz \in M \Rightarrow A' = A \cup \{w\}, B' = B \cup \{z\}$$

L'albero così costruito possiede due proprietà:

1. Ogni nodo dell'albero oltre alla radice è coperto da un lato di $M \cap E(T)$.
2. Per ogni nodo v dell'albero, il cammino nell'albero da v alla radice è alternante nel matching M associato all'albero.

L'albero alternante così ottenuto è utile per trovare un **cammino aumentante**

Definizione 2.11.2 (Albero alternante frustrato). Un albero alternante in un grafo G = (V, E) si dice frustrato se ogni lato di G avente un termine in $B(T)$ possiede l'altro termine in $A(T)$.

Proposizione 2.11.3 (Prima proposizione su AAF (5.6)). Sia G = (V, E) un grafo con un matching M ed un albero alternante frustrato T associato ad esso. Allora G non possiede un matching perfetto.

Dimostrazione 2.11.4 (Prima proposizione su AAF (5.6)). Ogni elemento di $B(T)$ è una componente dispari a nodo singolo in $G \setminus A(T)$.

Siccome $|A(T)| < |B(T)|$, per il teorema di Tutte G non possiede nessun matching perfetto. □

Proposizione 2.11.5 (Seconda proposizione su AAF (5.7)). Sia G = (V, E) un grafo bipartito, M un matching definito su G e T un albero M-alternante tale che nessun lato di G sia posto tra un nodo in $B(T)$ ed un nodo non in $V(T)$. Allora risulta che T è frustrato e di conseguenza G non ha un matching perfetto.

Dimostrazione 2.11.6 (Seconda proposizione su AAF (5.7)). Procediamo a mostrare che ogni lato avente un termine in $B(T)$ possiede l'altro termine in $A(T)$. Dalle ipotesi del teorema, l'unica possibile eccezione sarebbe un lato tra due nodi in $B(T)$. Ma questo lato, insieme con i lati che lo uniscono alla radice r, formerebbero un ciclo di lunghezza dispari, una cosa impossibile in un grafo bipartito.

Di conseguenza T è frustrato e quindi per la prima proposizione su AAF (5.6), il grafo G non possiede un matching perfetto. □

Definizione 2.11.7 (Pseudonodo). In riferimento ad un grafo G' ottenuto da G, vengono detti **pseudonodi** qui nodi che sono parte di G' ma non di G.

Proposizione 2.11.8 (Terza proposizione su AAF (5.8)). Sia G' un grafo derivato da G, sia M' un matching di G' e sia T un albero M'-alternante frustrato di G' tale che nessun elemento di $A(T)$ è uno pseudonodo.

Se T è frustrato, allora G non possiede un matching perfetto.

Dimostrazione 2.11.9 (Terza proposizione su AAF (5.8)). Se eliminiamo $A(T)$ da G otteniamo una componente con un insieme di nodi $S(v)$ per ogni nodo $v \in B(T)$. Utilizzando quindi la regola di Tutte-Berge si ha che:

$$\operatorname{odd}(G \setminus A(T)) > |A(T)|$$

Ne Segue che G non possiede un matching perfetto. □

2.12 Algoritmo Blossom

L'algoritmo procede derivando grafi da un grafo bipartito G iniziale: se riesce ad identificare un matching perfetto in un grafo derivato allora è noto che esiste un matching perfetto in G e se riusciamo ad identificare un certo tipo di AAF all'interno di un grafo derivato possiamo concludere che G non possiede un matching perfetto. I grafi derivati G' vengono ottenuti identificando circuiti dispari all'interno dell'albero alternante T ottenuto da G e rimuovendoli attraverso un procedimento detto di "shrinking" (questo tipo di circuito veniva chiamato blossom, da cui il nome dell'algoritmo.)

Teorema 2.12.1 (Algoritmo Blossom). L'algoritmo Blossom termina dopo $O(n)$ passi di "augmentations", $O(n^2)$ passi di "shrinking" e $O(n^2)$ passi di estensione dell'albero. Inoltre, determina correttamente se il grafo G possiede un matching perfetto.

Dimostrazione 2.12.2 (Algoritmo Blossom). L'algoritmo inizia con un matching qualsiasi $M = M'$ di G, che non cessa di essere tale per tutto il procedimento. Siccome ad ogni passo di "augmentation" il numero di nodi esposti decresce, vi saranno $O(n)$ passi di "augmentation". Tra questi, ogni passo di "shrinking" riduce il numero dei nodi in G' senza tuttavia cambiare il numero dei nodi in T ed ogni passo di estensione dell'albero riduce il numero di nodi non in T non modificando il numero di nodi in G'.

Ne segue che il numero di passi di "shrinking" ed estensione dell'albero tra passi di "augmentation" è $O(n)$ e quindi in totale sono $O(n^2)$.

Infine, siccome ogni G' è un grafo derivato da G, se l'algoritmo termina con la conclusione che G' non possiede un matching perfetto, allora significa che ha trovato un albero con le proprietà descritte nella terza proposizione su AAF (5.8) e quindi G non possiede un matching perfetto. □

2.13 Il postino cinese (Chinese postman problem, CPP)

Sia G = (V, E) un grafo connesso e sia $w : E \to \mathbb{R}_0^+$ una lunghezza definita su G. Vogliamo trovare un percorso chiuso C di lunghezza minima $w(C)$ che contiene ogni lato di G almeno una volta.

Definizione 2.13.1 (Grafo e cammino euleriano). Un grafo è detto **euleriano** se è possibile tracciare un cammino che passa una sola volta per tutti gli archi del grafo. Un tale cammino è detto a sua volta euleriano.

Equivalentemente, un grafo G è euleriano se e solo se ogni vertice ha grado pari e un cammino euleriano può essere costruiti con complessità $O(|E|)$.

Se il grafo G risulta euleriano allora la soluzione del CPP è banale: qualsiasi cammino euleriano andrebbe bene.

Altrimenti si procede come segue: sia X l'insieme di tutti i vertici di G con grado dispari. Aggiungiamo un insieme di lati E' a G tale che le seguenti tre condizioni siano soddisfatte:

1. Ogni lato $e' \in E'$ è parallelo a qualche lato $e \in E$. Estendiamo la distanza w a E' definendo $w(e') = w(e)$.
2. In V, E', solo i vertici di X hanno grado dispari.
3. La distanza $w(E')$ è minima: $w(E') \leqslant w(E'')$ per ogni insieme E'' che soddisfa le due condizioni precedenti.

Ne segue che $\left(V, E \cup E'\right)$ è un multigrafo euleriano, e ogni percorso euleriano introduce un cammino chiuso di lunghezza minima $w(E) + w(E')$ in G.

2.13.1 Applicazione su grafo orientato

Definizione 2.13.2 (Grafo orientato euleriano). Un grafo orientato è detto **euleriano** se tutti i vertici hanno grado entrante uguale al grado uscente.

Consideriamo un grafo orientato non euleriano G e sia $S(T)$ l'insieme di tutti i vertici che hanno un eccesso di archi entranti (o uscenti) rispetto a quelli uscenti (o entranti).

Indichiamo con $d^{+(i)}$ (o $d^{-(i)}$) l'eccesso di grado entrante (o uscente) nei nodi di $S(T)$. Per pareggiare il grado dei nodi $i \in S(T)$ devo aggiungere $d^{+(i)}$ (o $d^{-(i)}$) archi uscenti (o entranti). Poiché ogni arco contribuisce con un grado in uscita ed uno in ingresso, la somma totale dei gradi in uscita coincide con quella dei gradi in ingresso.

Come conseguenza la somma delle etichette $d^{(i)}$ dei vertici in S coincide con quella dei vertici in T. Calcolo il costo C_{ij} di un cammino di costo minimo fra ogni vertice $i \in S$ ed ogni vertice $j \in T$.

Costruisco una istanza di un problema di trasporto con gli insiemi S e T, dove le quantità prodotte dai vertici in S e da trasportare a quelli in T sono le etichette $d^{+(i)}$ mentre la domanda dei vertici in T sono i valori $d^{-(i)}$, ed il costo per unità di merce trasportata lungo l'arco (i, j) è C_{ij}. La soluzione del problema di trasporto indica gli archi da aggiungere.

Se la soluzione dice di trasportare k unità di merda da i a j allora aggiungiamo k archi paralleli ad ogni arco nel cammino di costo minimo da i a j.

3

Minimum spanning trees (MST)

3.1 Minimum spanning trees

Problema 3.1.1 (Connector Problem). Dato un grafo connesso G = (V, E) ed un costo positivo $c_e \in \mathbb{R}^+ \forall e \in E$, trovare il sottografo connesso di G a costo minimo.

Lemma 3.1.2. Un lato $e = uv$ di G è un lato di un circuito in G se e solo se esiste un cammino in $G \setminus \{e\}$ da u a v.

Di conseguenza se eliminiamo un lato in un circuito di un grafo connesso, il grafo risultante rimane connesso ed ogni soluzione ottima al *connector problem* non avrà circuiti.

Definizione 3.1.3 (Albero e Foresta). Un grafo privo di circuiti è detto **foresta**, mentre una foresta connessa è detta **albero**.

È possibile risolvere il Connector Problem risolvendo il problema del Minimum Spanning Tree (MST):

Problema 3.1.4 (Minimum Spanning Tree Problem). Dato un grafo connesso G = (V, E) ed un costo $c_e \in \mathbb{R} \forall e \in E$, trovare l'albero ricoprente a costo minimo in G.

Lemma 3.1.5. Un sottografo ricoprente connesso di G è un albero ricoprente se e solo se ha esattamente $n-1$ lati.

3.2 Algoritmi che risolvono MST

3.2.1 Algoritmo di Kruskal

Sia $H = (V, F)$ una foresta ricoprente di G, con $F = \emptyset$ inizialmente.

A ogni passo viene aggiunto ad F il lato a costo minimo $e \not\in F$ tale che H rimane una foresta.

Il procedimento si interrompe quando H diviene un albero ricoprente.

3.2.2 Algoritmo di Prim

Sia $H = (V(H), T)$ un albero di G = (V, E) con inizialmente $V(H) = \{r\}$ per qualche $r \in V$ e $T = \emptyset$.

A ogni passo viene aggiunto a T il lato a costo minimo $e \not\in T$ tale che H rimanga un albero.

Il procedimento si interrompe quando H diviene un albero ricoprente.

3.2.3 Validità degli algoritmi su MST

Definizione 3.2.1 (Taglio). Dato un grafo G = (V, E), chiamiamo $\delta(A)$ con $A \subseteq V$ l'insieme:

$$\delta(A) = \{e \in E : \text{entrambi i vertici di } e \text{ sono in } A\}$$

Questo insieme per qualche A è detto **taglio** di G.

Teorema 3.2.2 (Tagli di grafi connessi). Un grafo G = (V, E) è connesso se e solo se non esiste un insieme non vuoto $A \subset V$ tale che:

$$\delta(A) = \emptyset$$

Dimostrazione 3.2.3 (Tagli di grafi connessi). È chiaro che se $\delta(A) = \emptyset$, $u \in A$ e $v \not\in A$, allora non ci può essere un cammino da u a v e quindi se $A \neq \emptyset$ e $A \subset V$ allora G non è connesso.

Dobbiamo dimostrare che, se G non è connesso, allora non esiste un tale set A: scegliamo quindi due vertici $u, v \in V$ tali che non esiste un cammino tra u a v.

Definiamo A come:

$$A = \{w \in V : \text{esiste un cammino tra } u \text{ a } w\}$$

Quindi $u \in A$ e $v \not\in A$, pertanto $A \neq \emptyset$ e $A \subset V$.

Per dimostrare che $\delta(A) = \emptyset$ procediamo per assurdo: supponiamo che $p \in A$, $q \not\in A$ e che $e = pq \in E$. Aggiungendo e a qualsiasi cammino da u a p si ottiene un cammino da u a q, contraddicendo il fatto che $q \not\in A$. □

Definizione 3.2.4 (Sottografo estensibile a MST). Un sottoinsieme di lati di G = (V, E) $A \subseteq E$ è estensibile a un MST se A è contenuto in un insieme di lati di qualche MST di G.

Teorema 3.2.5 (Estensione a MST). Dato un grafo G = (V, E), sia $B \subseteq E$ un sottoinsieme estendibile a MST e e sia un lato a costo minimo di qualche taglio D tale che $D \cap B = \emptyset$. Allora $B \cup \{e\}$ è estendibile ad un MST.

Lemma 3.2.6 (Alberi ricoprenti). Sia $H = (V, T)$ un albero ricoprente di G, $e = vw$ un lato di G ma non di H e sia f un lato di un cammino semplice in T da v a w. Allora il sottografo:

$$H' = \left(V, (T \cup \{e\}) \setminus \{f\}\right)$$

è un albero ricoprente di G.

Dimostrazione 3.2.7 (Estensione a MST). Sia $H = (V, T)$ un MST tale che $B \subseteq T$. Se $e \in T$, allora abbiamo terminato il procedimento.

Altrimenti, sia P un cammino semplice appartenente a H da v a w, dove $vw = e$. Siccome non esiste nessun cammino in $G \setminus D$ da v a w, esiste un lato $f \in P$ tale che $f \in D$.

Quindi $c_f \geqslant c_e$, e per il lemma sugli alberi ricoprenti anche $\left(V, (T \cup \{e\}) \setminus \{f\}\right)$ è un MST.

Siccome $D \cap B = \emptyset$, ne segue che $f \not\in B$, pertanto $B \cup \{e\}$ è estendibile a un MST. □

3.3 MST e programmazione lineare

Teorema 3.3.1 (Legame tra MST e PL). Sia x^o il vettore caratteristico di un MST in rispetto ai costi $\underline{\boldsymbol{c}}$. ALlora x^o è una soluzione ottima del problema di programmazione lineare:

$$\begin{aligned} &\min \underline{\boldsymbol{c}}^T \underline{\boldsymbol{x}} \\ x\big(\gamma(S)\big) &\leqslant |S| - 1 \quad \forall S \neq \emptyset, S \subset V \\ x(E) &= |V| - 1 \\ x_e &\geqslant 0 \quad \forall e \in E \end{aligned}$$

Dimostrazione 3.3.2 (Legame tra MST e PL). Per un sottoinsieme di lati A, sia $k(A)$ il numero di componenti del sottografo (V, A) di G. Prendiamo in considerazione una versione equivalente del problema proposto nel teorema:

$$\begin{aligned} &\min \underline{\boldsymbol{c}}^T \underline{\boldsymbol{x}} \\ &x(A) \leqslant |V| - k(A) \quad \forall A \subset E \\ &x(E) = |V| - 1 \\ &x_e \geqslant 0 \qquad \forall e \in E \end{aligned}$$

Sia $A \subseteq E$ e siano $S_1, \dots, S_k$ gli insiemi di vertici delle componenti del sottografo (V, A). Allora:

$$x(A) \leqslant \sum_{i=1}^{k} x\big(\gamma(S_i)\big) \leqslant \sum_{i=1}^{k} (|S_i| - 1) = |V| - k$$

Procediamo ora a mostrare che x^o è la soluzione ottima del problema semplificato proposto e che è sufficiente perché ciò sia vero che essa è il vettore caratteristico di un albero ricoprente T generato dall'algoritmo di Kruskal.

Mostriamo che x^o è ottimale mostrando che l'algoritmo di Kruskal può essere utilizzato per calcolare una soluzione ammissibile al problema PL duale che soddisfa gli scarti complementari con x^o. Riscrivendo la funzione obbiettivo del problema primale come $\max -\underline{\boldsymbol{c}}^T \underline{\boldsymbol{x}}$, il duale risulta:

$$\begin{aligned} \min \sum_{A \subseteq W} (|V| - k(A))\, \underline{\boldsymbol{y}}_A& \\ \sum \Big(\underline{\boldsymbol{y}}_a : e \in A\Big) &\geqslant -c_e, \quad \forall e \in E \\ \underline{\boldsymbol{y}}_A &\geqslant 0, \quad \forall A \subset E \end{aligned}$$

Sia $e_1, \dots, e_m$ l'ordine con cui l'algoritmo di Kruskal considera i lati. Sia $R_i = \{e_1, \dots, e_i\}, \forall i \in \mathbb{N} \cap [1, m]$.

Sia $\underline{\boldsymbol{y}}^o_A = 0$ a meno che $\exists i : A = R_i$, e assegniamo $\underline{\boldsymbol{y}}^o_{R_i} = c_{e_{i+1}} - c_{e_i} \forall i \in \mathbb{N} \cap [1, m-1]$. Infine assegniamo $\underline{\boldsymbol{y}}^o_{R_m} = -c_{e_m}$.

Segue dall'ordine dei lati considerati dall'algoritmo che $\underline{\boldsymbol{y}}^o_A \geqslant 0$ se $A \neq E$. Considerando ora il vincolo a sommatoria del problema, dove $e = e_i$ abbiamo:

$$\sum_{\underline{\boldsymbol{y}}^o_A : e \in A} = \sum_{j=i}^{m} \underline{\boldsymbol{y}}^o_{R_j} = \sum_{j=i}^{m-1} \big(c_{e_{i+1}} - c_{e_i}\big) - c_{e_m} = -c_{e_i} = -c_e$$

In altre parole, tutte le disuguaglianze considerate valgono come uguaglianze, pertanto la soluzione $\underline{\boldsymbol{y}}^o$ è ammissibile e rispetta le condizioni degli scarti complementari: rimane solo una condizione da verificare, cioè se:

$$\underline{\boldsymbol{y}}^o_A > 0 \Rightarrow \underline{\boldsymbol{x}}^o \text{ soddisfa le condizioni degli scarti complementari.}$$

Per verificare questo, sappiamo che $\exists i : A = R_i$. Se i vincoli del problema primale non soddisfano le condizioni degli scarti complementari per R_i, allora esiste qualche lato di R_i la cui aggiunta a $T \cap R_i$ porterebbe a ridurre il numero di componenti di $(V, T \cap R_i)$. Ma un tale lato avrebbe termini in due componenti diverse di $(V, R_i \cup T)$, e quindi sarebbe stato aggiunto a T dall'algoritmo di Kruskal.

Per cui, $\underline{\boldsymbol{x}}^o$ e $\underline{\boldsymbol{y}}^o$ soddisfano le condizioni degli scarti complementari. Ne segue che $\underline{\boldsymbol{x}}^o$ è la soluzione ottima del problema di PL. $\square$

4

Matroidi e Algoritmo Greedy

4.1 Sistema di Indipendenza

Definizione 4.1.1 (Sistema di Indipendenza (SI)). Sia $E = \{e_1, \dots, e_n\}$ e $F \subseteq 2^E$ tale che per tutti $I, J \subseteq E$ valga la proprietà:

$$J \subset I \in F \to J \in F$$

Allora la coppia (E, F) è detta **Sistema di Indipendenza** ed i membri di F vengono detti **insiemi indipendenti**.

4.2 GREEDSUM

L'algoritmo riceve un insieme di elementi E ed una funzione di costo c ed inizia con l'insieme soluzione $I = \emptyset$.

Successivamente itera $|E|$ volte, scegliendo ogni volta l'elemento di E che massimizza la funzione c.

Questo elemento $e^* = \max_e c(E)$ viene quindi rimosso dal set degli oggetti: $E = E - \left\{e^*\right\}$.

Se l'insieme $I \cup \left\{e^*\right\}$ risulta essere indipendente, allora il nuovo elemento viene aggiunto all'insieme di soluzione: $I = I \cup \left\{e^*\right\}$.

Definizione 4.2.1 (Matroide). Un SI che gode della proprietà del primo teorema di Rado 4.3.1 si dice **matroide**.

4.3 I teoremi di Rado

I teoremi di Rado consentono di fare un'importante affermazione su GREEDSUM: "GREEDSUM fornisce l'ottimo se e solo se il SI è un matroide".

Teorema 4.3.1 (Primo teorema di Rado). Sia (E,F) un Sistema di Indipendenza (SI). Se per due qualunque indipendenti I e J tali che $|I| = |J| + 1$ vale la proprietà:

$$\exists e \in I : J \cup \{e\} \in F$$

allora **GREEDSUM** fornisce un membro di F di valore massimo per qualunque funzione di valutazione: $e : E \to \mathbb{R}^+$.

Dimostrazione 4.3.2 (Primo teorema di Rado). La proprietà del teorema implica che se $A \subseteq E$ e $I, I' \subseteq A$ sono insiemi indipendenti massimali, allora $|I| = \left|I'\right|$.

Infatti se **per assurdo** $|I| < \left|I'\right|$, si può trovare $I'' \subseteq I'$ tale che $\left|I''\right| = |I| + 1$ eliminando da I' un numero di elementi pari a $\left|I'\right| - |I| - 1$.

Per la proprietà del teorema esiste $e \in I'' \setminus I$ tale che $I \cup \{e\} \in F$ e I non sarebbe massimale.

Supponiamo adesso ancora per assurdo che **GREEDSUM** non dia un risultato ottimo, cioè che per una certa funzione di valutazione $c : E \to \mathbb{R}^+$ l'algoritmo fornisce un insieme indipendente $I = \{e_1, \dots, e_i\}$ mentre ne esiste un altro $J = \left\{e'_1, \dots, e'_j\right\}$ tale che $c(j) > c(i)$.

Supponiamo senza perdita di generalità di aver ordinato gli elementi di I e J in ordine di valore non crescente.

Per costruzione I è massimale e quindi anche J per quanto visto sopra, per cui $i = j$. Vedremo che per $m = 1, 2, \dots, i$ vale che $c(e_m) \geqslant c(e'_m)$, contraddicendo l'ipotesi per cui $c(J) > c(I)$.

Procediamo quindi per induzione: quando $m = 1$, la proprietà è vera per come **GREEDSUM** sceglie il primo elemento.

Supponiamo allora che per qualche $m > 1$ si abbia $c(e_m) < c(e'_m)$ mentre $c(e_s) \geqslant c(e'_s)$ per $s = 1, \dots, m-1$. Si $A = \left\{e \in E : c(e) \geqslant c(e'_m)\right\}$.

Allora l'insieme $\{e_1, \dots, e_{m-1}\}$ è un indipendente massimale di A dato che se $\{e_1, \dots, e_{m-1}, e\} \in I$ e $c(e) \geqslant c(e'_m) > c(e_m)$, **GREEDSUM** avrebbe scelto e invece di e_m come successivo elemento da aggiungere.

Ciò contraddice quanto dimostrato all'inizio dato che $\left\{e'_1, \dots, e'_m\right\}$ è un altro insieme indipendente di A ed avrebbe cardinalità maggiore. □

Teorema 4.3.3 (Secondo teorema di Rado). Sia (E,F) un Sistema di Indipendenza (SI). Se per qualunque funzione di valutazione non negativa $c : E \to \mathbb{R}^+$ degli elementi di E, **GREEDSUM** fornisce sempre un membro di F di valore massimo, allora il SI gode della proprietà del primo teorema di Rado 4.3.1.

Dimostrazione 4.3.4 (Secondo teorema di Rado). Procediamo **per assurdo** e supponiamo che la proprietà del primo teorema di Rado 4.3.1 non valga.

Ciò significa che esistono due insiemi indipendente I e J con $|I| = p$ e $|J| = p+1$ tali che per nessun elemento $e \in J \setminus I$ accade che $I \cup \{e\} \in I$.

Consideriamo allora la seguente maniera di valutare gli elementi di E:

$$c(e) = \begin{cases} p+2 & e \in I \\ p+1 & e \in J \setminus I \\ 0 & e \not\subseteq I \cup J \end{cases}$$

Se ne deduce che $c(J) \geqslant \left(p+1\right)^2 > p(p+2) - c(I)$ e quindi I non è ottimo.

GREEDSUM d'altra parte per sua natura inizierà a inserire nella soluzione per primi tutti gli elementi di I e non potrà poi aumentare il valore della soluzione, dato che sarà costretto a selezione solo elementi di valore nullo.

Ma allora abbiamo trovato una funzione di valutazione per cui l'algoritmo non fornisce l'ottimo, contrariamente a quanto assunto. □

5
Algoritmo di Dijkstra

5.1 Utilità

L'algoritmo di Dijkstra viene utilizzato per trovare il cammino più breve tra due nodi di un grafo.

5.2 Complessità

Complessità computazionale 5.2.1 (Algoritmo di Dijkstra). L'implementazione dell'algoritmo di Dijkstra iniziale ha complessità $O\left(|2|^2\right)$.

5.3 Il funzionamento di Dijkstra

1. Inizialmente tutti i nodi sono marcati come **non visitati** e viene assegnata ad ognuno di essi un valore di distanza: 0 per il nodo iniziale ed ∞ per tutti gli altri.
2. Per il nodo corrente, prendiamo in considerazione tutti i nodi vicini non visitati: per ognuno calcoliamo la distanza dalla radice passando attraverso il nodo corrente e confrontiamo il valore di distanza preesistente con quello nuovo e manteniamo il valore minore.
3. Una volta determinata la distanza per tutti i vicini del nodo corrente marchiamo il nodo corrente come **visitato**.
4. Se il **nodo destinazione** è stato marcato visitato o se la lunghezza minore tra i nodi non visitati è infinita (come quando si vuole raggiungere un nodo non connesso) l'algoritmo si interrompe.
5. Altrimenti si sceglie un nodo non visitato a distanza minima: esso diviene il nuovo nodo corrente e l'operazione si ripete dal punto 2.

5.4 Correttezza di Dijkstra

Dimostrazione 5.4.1 (Correttezza di Dijkstra). Procediamo a dimostrare la correttezza dell'algoritmo di Dijkstra **per induzione** sul numero di nodi visitati.

Per ogni nodo visitato v, la distanza $d(v)$ è la distanza minore dal nodo sorgente al nodo v. Per ogni non non visitato u, $d(u)$ è assunta essere la distanza più breve attraverso solo nodi visitati, dalla radice a u.

Questa assunzione vale solo se un cammino esiste, altrimenti la distanza è inizializzata a ∞.

Il caso base è quando esiste solo un nodo visitato, cioè il nodo sorgente, in tal caso l'ipotesi è banale.

Altrimenti, assumiamo l'ipotesi per $n-1$ nodi visitati. In tal caso, scegliamo un lato vu dove u possiede la distanza minima di ogni nodo non visitato ed il lato vu è tale che:

$$d(u) = d(v) + \text{lunghezza}(v, u)$$

La distanza $d(u)$ è considerata la distanza minore dalla **radice** a u perché se esistesse un cammino più breve e se w fosse il primo nodo non visitato su quel cammino allora l'ipotesi originale che $d(w) > d(u)$ **non sarebbe più valida**.

Similarmente se vi fosse un cammino più breve per giungere ad u senza utilizzare nodi non visitati e se il penultimo nodo su questo cammino fosse w, allora si avrebbe che $d(u) = d(w) + \text{lunghezza}(w, u)$ **che è nuovamente una contraddizione**.

Dopo aver processato il nodo u, le ipotesi iniziali si manterranno vere per ogni non visitato w: la distanza $d(w)$ sarà la distanza minore dalla radice a w utilizzando solo i nodi visitati perché se esistesse un cammino più breve non passante per u lo avremmo identificato precedentemente, e se esistesse un cammino più breve passante passante per u lo avremmo aggiornato mentre il nodo u veniva processato. $\square$

6 Algoritmo di Ford-Fulkerson

6.1 Utilità

Si tratta di un **algoritmo greedy** che calcola il **flusso massimo** un un grafo. Spesso è chiamato *metodo* invece di *algoritmo* perché l'approccio per trovare i cammini aumentanti nel grafo residuo non è completamente specificato. Spesso viene implementato completamente nell'algoritmo di **Edmonds-Karp**, in cui si procede a scegliere ad ogni iterazione il **cammino aumentante più corto**.

Definizione 6.1.1 (Cammino aumentante). Un cammino P da s a t tale che ogni arco $\left(i, j\right) \in P$ orientato verso t (in avanti o forward) ha capacità $x_{ij} < u_{ij}$ ed ogni arco orientato verso s (all'indietro o reverse) ha capacità $x_{ji} > 0$ si dice **cammino aumentante**.

Definizione 6.1.2 (Cammino incrementante). Un cammino P da s a v tale che $v \neq t$ e per ogni arco $\left(i, j\right) \in P$ forward ha capacità $x_{ij} < u_{ij}$ e per ogni arco $\left(i, j\right) \in P$ reverse ha capacità $x_{ji} > 0$ si dice **cammino incrementante**.

6.2 Algoritmo di Edmonds-Karp

Lemma 6.2.1 (Primo lemma di Edmonds-Karp).

$$\forall v \in \mathbb{N}, \quad d_{x'}(s,v) \geqslant d_x(s,v) \quad \wedge \quad d_{x'}(s,v) \geqslant d_x(s,v)$$

Dimostrazione 6.2.2 (Primo lemma di Edmonds-Karp). Iniziamo dividendo il lemma in due parti:

$$\forall v \in \mathbb{N}, \quad \underbrace{d_{x'}(s,v) \geqslant d_x(s,v)}_{\text{Prima parte}} \quad \wedge \quad \underbrace{d_{x'}(s,v) \geqslant d_x(s,v)}_{\text{Seconda parte}}$$

Procediamo ora a dimostrare la prima parte del lemma.

Supponiamo che esista un nodo v tale che $d_{x'}(s,v) < d_x(s,v)$ e scegliamo v in modo che $d_{x'}(s,v)$ sia la distanza più piccola il possibile.

Essendo $v \neq s$, $d_{x'}(s,v) > 0$.

Sia P' un cammino (s,v) in $G_{x'}$ e w il penultimo nodo di P'. Si ha che:

$$d_x(s,v) > d_{x'}(s,v) = d_{x'}(s,w) + 1 \geqslant d_x(s,w) + 1$$

Se $d_x(s,v) > d_x(s,w) + 1$, allora l'arco (w,v) non appartiene a G_x, perché altrimenti $d_x(s,v) = d_x(s,w) + 1$.

Se l'arco (w,v), che appartiene a $G_{x'}$ non appartiene a G_x allora esiste un indice $i : w = v_i e v = v_{i-1}$, pertanto:

$$d_x(s,v) = i - 1 > d_x(s,w) + 1 = i + 1$$

e si ha una **contraddizione**.

Analogamente si procede a dimostrare la seconda parte del lemma. □

Lemma 6.2.3 (Secondo lemma di Edmonds-Karp). Durante l'esecuzione dell'algoritmo di Edmonds e Karp, un arco $\left(i,j\right)$ scompare (e compare) in G_x al più $\frac{n}{2}$ volte.

Dimostrazione 6.2.4 (Secondo lemma di Edmonds-Karp). Se un arco $\left(i,j\right)$ "scompare" dalla rete ausiliaria, significa che esso è su un cammino aumentate e che il corrispondente arco in G si satura oppure si svuota. Pertanto, nella rete ausiliaria successiva compare l'arco $\left(j,i\right)$. Sia x_f il flusso al momento della "scomparsa dell'arco".

Supponiamo che ad una successiva iterazione l'arco $\left(i,j\right)$ ricompaia in G_{x_h}. Ciò significa che il cammino aumentante che ha generato x_h contiene l'arco $\left(j,i\right)$.

Allora se x_g è il flusso a partire dal quale si è generato x_h, si ha per il primo lemma 6.2.1 che:

$$d_g\left(s,i\right) = d_g\left(s,j\right)+1 \geqslant d_f\left(s,j\right)+1 = d_f\left(s,i\right)+2$$

Pertanto, nel passare del flusso x_f al flusso x_h, $d\left(s,u\right)$ è aumentata almeno di 2. Poiché il massimo valore che può assumere $d\left(s,u\right)$ è n, un arco può scomparire e riapparire al più $\frac{n}{2}$ volte. □

Lemma 6.2.5 (Terzo lemma di Edmonds-Karp). L'algoritmo di Edmonds-Karp ha complessità $O\left(nm^2\right)$.

Dimostrazione 6.2.6 (Terzo lemma di Edmonds-Karp). Ogni arco può scomparire al più $\frac{n}{2}$ volte durante l'esecuzione dell'algoritmo per il secondo lemma 6.2.3. Ogni volta che effettuiamo un aumento di flusso, scompare almeno un arco. Pertanto, almeno un arco. Pertanto, durante l'esecuzione, si hanno al più $\frac{mn}{2}$ "sparizioni". Ogni operazione di aumento richiede $O\left(m\right)$ e, quindi, la complessità dell'algoritmo è $O\left(nm^2\right)$. □