Appunti sui problemi di matching

A. Agnetis*

1 Formulazione

I problemi di *matching* (talvolta chiamati problemi di accoppiamento, o abbinamento) sono tra i più importanti e più studiati problemi di ottimizzazione su grafi, e hanno molteplici applicazioni, sia come modello di problemi reali che come strumento da utilizzare nella risoluzione di problemi di ottimizzazione più complessi.

Si consideri un grafo $G = (N, A)$, non orientato, con n nodi e m archi. Un *matching* M è un sottoinsieme di archi tale che ogni nodo di G incide al più un solo arco di M. Un arco $(i, j) \in M$ si dice *accoppiato*, mentre un arco $(i, j) \notin M$ si dice *libero*. Un nodo su cui incida un arco di M si dice *accoppiato* (e il nodo j tale che $(i, j) \in M$ è il suo *compagno*), mentre un nodo che non incide alcun arco di M si dice *esposto*.

Il più semplice e più studiato problema di questa classe consiste nel determinare un matching di cardinalità massima, ove per cardinalità di un matching si intende il numero di archi di cui è composto. Vediamo come è possibile formulare questo problema in termini di programmazione lineare binaria. Associamo a ogni arco (i, j) una variabile binaria x_{ij} tale che $x_{ij} = 1$ se $(i, j) \in M$ e $x_{ij} = 0$ se $(i, j) \notin M$. Ricordando che $\delta(i)$ indica l'insieme degli archi incidenti il nodo i, possiamo formulare il problema nel modo seguente.

$$\max \sum_{(i,j)\in A} x_{ij} \tag{1}$$

$$\sum_{(i,j)\in\delta(i)} x_{ij} \leq 1, \qquad i \in N \tag{2}$$

$$x_{ij} \in \{0, 1\}, (i, j) \in A \tag{3}$$

Ciascun vincolo (2) esprime il fatto che un nodo può essere incidente al più a un solo arco del matching. Abbiamo dunque un vincolo per ciascun nodo, e osserviamo che ciascuna variabile x_{ij} compare in esattamente due vincoli, ossia quelli corrispondenti ai due nodi

*Dipartimento di Ingegneria dell'Informazione - Università di Siena

i e j. Questo consente di riconoscere che la matrice dei coefficienti del problema (a meno di una matrice identità che compare allorché si vuole porre il problema in forma standard) coincide con la matrice di incidenza del grafo G. Di conseguenza, se G è bipartito, possiamo risolvere il problema come qualsiasi problema di programmazione lineare, sostituendo i vincoli (3) con i vincoli lineari:

$$x_{ij} \geq 0, \tag{4}$$

Si noti che i vincoli (2) implicano $x_{ij} \leq 1$ per ogni (i, j). Se invece il grafo non è bipartito, il vincolo di interezza non può essere trascurato. Infatti, si consideri ad esempio il grafo completo con 3 nodi. Chiaramente la soluzione ottima del problema di massimo matching ha valore 1, perché è possibile scegliere solo uno dei tre archi del grafo senza violare i vincoli (2). Ma se andiamo a risolvere il problema rilassando i vincoli di interezza, otteniamo una soluzione ottima in cui tutte e tre le variabili associate agli archi hanno valore $1/2$, e dunque la funzione obiettivo assume valore $3/2$, strettamente migliore di 1, ma chiaramente una soluzione di questo tipo non ha nessun senso in termini del problema che vogliamo risolvere.

Si noti che, ovviamente, se esiste un matching di cardinalità pari a $n/2$, questo è ottimo perché tutti i nodi risultano accoppiati. Tale matching si dice *perfetto*. Tuttavia, in generale non esiste un matching perfetto e dunque la soluzione ottima del problema potrà avere valore inferiore a $n/2$. (Si noti che su un grafo bipartito un matching perfetto può esistere solo se i due insiemi di nodi hanno ambedue cardinalità $n/2$.)

Il problema del massimo matching è un problema non pesato, ma esistono alcune versioni pesate la cui formulazione differisce in minima parte da quella già vista. Ad esempio, si può affrontare il problema in cui il grafo G è pesato sugli archi, e si vuole determinare il matching di peso massimo. L'unica differenza da apportare consiste nel sostituire la funzione obiettivo (1) con

$$\max \sum_{(i,j)\in A} w_{ij} x_{ij} \tag{5}$$

dove ovviamente w_{ij} è il peso dell'arco (i, j). Essendo invariata la struttura dei vincoli, anche questo problema, se il grafo è bipartito, è risolubile come PL. Un'altra variante significativa è quella in cui il grafo bipartito è completo, e si vuole trovare un *matching perfetto* di peso minimo (o massimo). Nella formulazione del problema, basterà sostituire i vincoli (2) con

$$\sum_{(i,j)\in\delta(i)} x_{ij} = 1, \qquad i \in N \tag{6}$$

e anche in questo caso, ovviamente, la matrice dei vincoli è totalmente unimodulare, essendo il grafo bipartito. Quest'ultimo problema è noto come *problema dell'assegnamento*, che analizzeremo successivamente (Capitolo 5).

2 Definizioni e risultati fondamentali

Le seguenti importanti definizioni sono tutte date rispetto a un grafo G (non necessariamente bipartito) *e a un matching* M. Un cammino sul grafo G si dice *alternante* se è costituito da archi accoppiati e archi liberi, alternati. Un cammino alternante si dice *aumentante* se il nodo iniziale e il nodo finale sono esposti, ovvero, il che è lo stesso, se il primo e l'ultimo arco del cammino sono archi liberi.

Dati due insiemi di archi X e Y, la loro *differenza simmetrica* $X \oplus Y$ è l'insieme ottenuto considerando gli archi che appartengono *solo* a X oppure *solo* a Y, ma non a tutti e due, ovvero è l'insieme $(X - Y) \cup (Y - X)$. (Dal punto di vista logico, è l'*or esclusivo* dei due insiemi X e Y.)

TEOREMA 1 *Sia M un matching su un grafo G, e sia P un cammino aumentante. La differenza simmetrica $M' = M \oplus P$ è un matching di cardinalità $|M| + 1$.*

Dim. Consideriamo un matching M su un grafo G, e sia P un cammino aumentante. Effettuiamo ora la differenza simmetrica tra gli insiemi M e P, e sia M' l'insieme cosìottenuto. Osserviamo che: (i) gli archi del matching M che non facevano parte di P rimangono inalterati, e dunque si ritrovano in M'; (ii) invece gli archi di M che fanno parte anche di P vengono esclusi da M', ma vengono però inseriti in M' gli archi di P che non facevano parte di M. L'insieme M' è ancora un matching, in quanto: per i nodi che non sono toccati da P, non è cambiato nulla, su quelli intermedi di P prima incideva un arco di M e adesso un arco di $P - M$, i nodi estremi di P erano esposti e ora sono invece accoppiati. Poiché P è un cammino aumentante rispetto a M, se esso è di lunghezza $2k + 1$, esso conterrà k archi del matching M e $k + 1$ archi liberi. Facendo la differenza simmetrica, gli archi liberi entrano a far parte del matching (M') mentre quelli accoppiati ne escono, e dunque M' ha un arco in più rispetto a M. □

Dunque, se esiste un cammino aumentante, possiamo ottenere un matching di cardinalità superiore facendo la differenza simmetrica del cammino e del matching attuale. Viene da chiedersi allora se sia possibile caratterizzare un matching massimo in modo analogo a quanto accade nel problema del massimo flusso.

TEOREMA 2 *Dato un grafo $G = (N, A)$, un matching M è massimo se e solo se non esistono cammini aumentanti.*

Dim. Il "solo se" è ovvio, per il Teorema 1. Vediamo allora di dimostrare che se non esistono cammini aumentanti rispetto a un certo matching M, allora quel matching M è ottimo. Supponiamo per assurdo che dunque, pur non esistendo cammini aumentanti, M non sia ottimo, e che esista perciò un matching $\hat{M}$ tale che $|\hat{M}| > |M|$. (Si noti che la cardinalità di $\hat{M}$ può essere anche maggiore di $|M| + 1$.) Consideriamo ora l'insieme di archi $Y = \hat{M} \oplus M$, e sia $G' = (N, Y)$ il grafo che ha lo stesso insieme di nodi di G ma che contiene solo gli archi dell'insieme Y. Andiamo ad analizzare i gradi dei nodi di G', considerando tutti i casi possibili.

(i) Un nodo che in ambedue i matching M e $\hat{M}$ è accoppiato con lo stesso compagno, è un nodo isolato in G'.

(ii) Un nodo che in ambedue i matching M e $\hat{M}$ è accoppiato, ma con compagni diversi, ha grado 2 in G'.

(iii) Un nodo accoppiato in M ed esposto in $\hat{M}$ o viceversa, in G' ha grado 1.

(iv) Un nodo esposto in ambedue i matching, in G' è un nodo isolato.

Dunque, in G' nessun nodo ha grado superiore a 2. Ecco che allora le componenti connesse di G' possono essere solo di due tipi: (i) cicli *pari* (in quanto se un ciclo avesse un numero dispari di archi ci sarebbero due archi dello stesso matching incidenti uno stesso nodo, il che è impossibile), e (ii) cammini (eventualmente con 0 archi), costituiti alternatamente da archi di M e archi di $\hat{M}$, . Ora, ricordando che stiamo supponendo $|\hat{M}| > |M|$, deve risultare che in Y vi sono più archi di $\hat{M}$ che non di M. Ma dove sono questi archi in più? Non, ovviamente, nei cicli pari, dal momento che questi sono costituiti dallo stesso numero di archi di $\hat{M}$ e di M. Quindi, deve esistere almeno un cammino costituito da un numero di archi di $\hat{M}$ superiore al numero di archi di M. Ma perché ciò accada questo cammino deve necessariamente iniziare e terminare con archi di $\hat{M}$, e dunque questo cammino, *rispetto al matching* M, è un cammino aumentante. Avendo così contraddetto l'ipotesi secondo la quale non esistono cammini aumentanti, segue la tesi. □

Questo teorema suggerisce un'immediata strategia per la ricerca del matching di massima cardinalità: partendo da un matching iniziale (eventualmente, vuoto), cercare un cammino aumentante. Una volta trovato, possiamo aumentare il matching di 1. Se ne cerca quindi un altro, e così via fino al punto in cui non esistono più cammini aumentanti: il Teorema 2 ci garantisce che quando ciò si verifica, siamo arrivati alla soluzione ottima.

Il problema viene ricondotto dunque a quello, dato un matching M, di ricercare un cammino aumentante, o certificarne l'inesistenza. Benché il Teorema 2 valga per grafi

qualsiasi, la ricerca di un cammino aumentante presenta diversa complessità a seconda che il grafo sia bipartito o no. Noi ci limitiamo qui a considerare il caso bipartito.

3 Algoritmo risolutivo per il Matching bipartito

Gli algoritmi più semplici per la ricerca di un matching di cardinalità massima costruiscono una successione di cammini aumentanti, sfruttando il risultato del Teorema 2. La ricerca dei cammini aumentanti può essere effettuata in vari modi.

Nel seguito utilizziamo una metodologia particolarmente semplice e suggestiva.

Consideriamo un grafo bipartito $G = (U, V, A)$ e un matching $M \subseteq A$. Costruiamo allora un grafo *orientato* ausiliario $D = (U, V, A')$, ottenuto da G orientando gli archi come segue:

- Se $(u, v) \in M$, l'arco è orientato da v a u;
- Se $(u, v) \notin M$, l'arco è orientato da u a v.

In altre parole, dato un matching M, in D gli archi liberi sono orientati da sinistra verso destra, mentre quelli del matching da destra verso sinistra. Siano inoltre U_M e V_M gli insiemi di nodi esposti appartenenti a U e V rispettivamente. A questo punto, è immediato riconoscere che un qualsiasi cammino su D che inizia in un nodo di U_M e termina in un nodo di V_M è un cammino aumentante. Dunque, la determinazione di un cammino aumentante consiste semplicemente nell'effettuare una visita dei nodi di D a partire dai nodi che fanno parte di U_M. Una volta trovato un cammino aumentante, il matching M viene aggiornato, e di conseguenza anche il grafo D. L'algoritmo prosegue fino a che nessun nodo di V_M risulta raggiungibile dai nodi di U_M, il che certifica che il matching corrente è massimo.

Osserviamo che il calcolo di ciascun cammino aumentante, richiedendo semplicemente una visita del grafo D, ha complessità $O(m)$, e tale è anche la complessità richiesta dall'aggiornamento del grafo D (in sostanza, a ogni iterazione, si tratta di invertire l'orientamento degli archi che fanno parte del cammino aumentante trovato). Dunque, in questo modo, un matching di massima cardinalità su un grafo bipartito può essere calcolato in $O(mn)$.

E' appena il caso di notare che l'idea alla base della ricerca dei cammini aumentanti in questo algoritmo risolutivo non può essere banalmente estesa al caso generale di grafi non bipartiti. Tuttavia, se pure con complessità maggiore, anche nel caso generale un cammino aumentante può essere calcolato, se esiste, in tempo polinomiale[1].

[1] Il più semplice algoritmo per il matching su grafi generali è l'*algoritmo di Edmonds*, avente complessità $O(mn^2)$, si veda ad esempio ancora il testo di Papadimitriou e Steiglitz.

4 Il problema di Node Cover

Si consideri il seguente problema. Dato un grafo non orientato $G(N, A)$, una *copertura* o *cover* è un sottoinsieme di nodi $S \subset N$ tale che, per ogni arco $(i, j) \in A$, $i \in S$ oppure $j \in S$ (o entrambi), vale a dire che ogni arco è "coperto" da (almeno) un nodo di S. Il problema noto come Node Cover consiste nel determinare la copertura con il minimo numero di nodi.

Associando a ogni nodo $j \in N$ una variabile binaria x_j tale che $x_j = 1$ se $j \in S$ e $x_j = 0$ se $j \notin S$, possiamo formulare il problema nel modo seguente.

$$\min \sum_{j \in N} x_j \tag{7}$$

$$x_i + x_j \geq 1, \qquad (i, j) \in A \tag{8}$$

$$x_j \in \{0, 1\}, j \in N \tag{9}$$

Ciascun vincolo (8) esprime il fatto che un arco deve essere coperto da almeno uno dei nodi a cui l'arco è incidente. Si noti che la matrice dei coefficienti del problema (al solito, a meno di una matrice identità cambiata di segno se si vuole portare il problema in forma standard) è la trasposta della matrice di incidenza del grafo G, e dunque se G è bipartito si può rilassare il vincolo di interezza (9) sostituendolo con il vincolo di nonnegatività, e risolvere il problema come un qualsiasi problema di programmazione lineare. Nel seguito *supporremo dunque che il grafo G sia bipartito.*

Confrontando i problemi (1)–(3) e (7)–(9) si può osservare che la formulazione del problema di Node Cover e il problema di matching di massima cardinalità costituiscono una coppia primale/duale. Questo fatto ha interessanti implicazioni dal punto di vista algoritmico. Anzitutto, dalla dualità debole e dalla dualità forte discendono immediatamente le seguenti proprietà:

Teorema 3 *Dato un grafo G bipartito, la cardinalità di un matching è sempre minore o uguale a quella di un node cover.*

Teorema 4 *Dato un grafo G bipartito, la massima cardinalità di un matching e la minima cardinalità di un node cover coincidono.*

Osserviamo dalla complementarietà che se, all'ottimo, un arco $(i, j) \in M$, e dunque $x_{ij} = 1$, il corrispondente vincolo nel problema di Node Cover deve essere attivo, ossia *esattamente uno* dei due nodi estremi degli archi che appartengono al matching massimo

fa parte della copertura minima. Questo fatto può essere sfruttato per ricostruire la copertura minima dalla soluzione ottima del problema di matching massimo.

Infatti, indicando ora con M un matching massimo, si consideri l'insieme R_M di nodi raggiungibili da U_M (Figura 1). Si noti che tutti i nodi di $R_M \cap V$ sono nodi accoppiati: se uno di essi non lo fosse, esisterebbe un cammino aumentante che termina proprio su quel nodo. Dunque, ogni nodo di $R_M \cap V$ è incidente a un arco di M.

Consideriamo ora i nodi dell'insieme $U \setminus R_M$: anche ognuno di questi nodi è accoppiato, altrimenti farebbe parte di U_M, che ovviamente è incluso in R_M. Ogni nodo di $U \setminus R_M$ è accoppiato con un nodo che *non* fa parte di $R_M \cap V$, in quanto altrimenti farebbe parte anch'esso di R_M.

A questo punto, è facile vedere che $C \equiv (U \setminus R_M) \cup (R_M \cap V)$ costituisce un node cover. Infatti, se vi fossero archi non coperti dall'insieme C, questi dovrebbero avere un estremo in $R_M \cap U$ e l'altro in $V \setminus R_M$, ma se un arco (u, v) di questo tipo esistesse, il nodo v sarebbe stato raggiungibile da u e avrebbe quindi fatto parte di R_M. Dunque, C è un node cover. Ma a questo punto, osservando che ciascun nodo di C è incidente un arco diverso di M, si ha che $|C| \leq |M|$. Poiché dalla dualità forte deve essere $|C| = |M|$, si ha che C è il node cover di minima cardinalità.

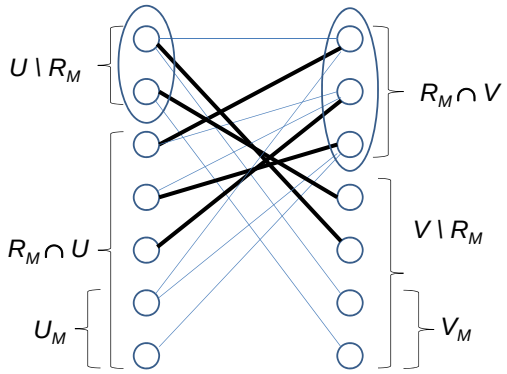

Figure 1: Un matching massimo (in grassetto) e una copertura minima $(U \setminus R_M) \cup (R_M \cap V)$.

5 Il problema dell'assegnamento

Dato un grafo bipartito completo $G = (U, V, A)$, con $|U| = n$ e $|V| = n$, e indicando con c_{ij} il costo di assegnare il nodo $u_i \in U$ al nodo $v_j \in V$, il *problema dell'assegnamento* è il

seguente:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{10}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad u_i \in U \tag{11}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad v_j \in V \tag{12}$$

$$x_{ij} \in \{0,1\}, (u_i, v_j) \in A \tag{13}$$

(Si noti che a differenza della (6), abbiamo indicato separatamente i vincoli relativi ai nodi di U e quelli relativi ai nodi di V.) Essendo la matrice dei coefficienti totalmente unimodulare, possiamo rilassare i vincoli di interezza e riformulare il problema come:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{14}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad u_i \in U \tag{15}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad v_j \in V \tag{16}$$

$$x_{ij} \geq 0, \quad (u_i, v_j) \in A \tag{17}$$

Il problema può dunque essere risolto con il metodo del simplesso. Tuttavia, in questo caso il metodo del simplesso potrebbe non risultare molto efficiente, a causa del fatto che, come si può facilmente osservare, tutti i vertici del politopo (15–17) sono fortemente degeneri: infatti, in ogni soluzione di base ammissibile, soltanto n variabili di base (su $2n-1$) sono diverse da zero. Nel seguito vedremo allora un algoritmo specifico per risolvere in modo efficiente il problema dell'assegnamento, tralasciando alcuni dettagli dimostrativi.

Nel seguito, indicheremo con C la matrice quadrata $n \times n$ contenente tutti i costi. Supporremo inoltre che $c_{ij} \geq 0$ per tutti gli i e j, ma come vedremo questa assunzione non è restrittiva. L'algoritmo *ungherese*[2] si basa su due considerazioni estremamente semplici: la prima è che se, in un problema di assegnamento, aggiungiamo o sottraiamo una quantità θ a tutti gli elementi di una stessa riga o di una stessa colonna di C, otteniamo un problema equivalente: infatti, il valore della funzione obiettivo di qualunque soluzione ammissibile varierà della stessa quantità θ, ma un assegnamento ottimo rimane tale.

La seconda considerazione è la seguente. Supponiamo che tutti i coefficienti di costo c_{ij} siano non negativi e che alcuni di essi siano nulli. Se allora consideriamo *solo* gli

[2]Gli autori sono infatti i matematici ungheresi König e Egervary, e l'algoritmo fu chiamato "ungherese" da Kuhn, che lo fece conoscere in occidente negli anni '50.

archi relativi a queste coppie (u_i, v_j), e con questi riusciamo a formare un matching di cardinalità n, di sicuro tale matching costituisce una soluzione ottima per il problema di assegnamento.

L'idea di fondo dell'algoritmo ungherese è quella di manipolare i coefficienti di costo, sempre attraverso opportune operazioni di somma o sottrazione di costanti a righe e/o colonne di C, allo scopo di far apparire sempre più zeri, fino a riuscire a trovare un assegnamento che faccia uso solo di tali costi. Al termine, il valore della soluzione ottima sarà dato dalla somma dei coefficienti di costo originari degli archi utilizzati nel matching perfetto. Nel seguito vediamo una descrizione di questo metodo, tralasciando alcuni dettagli dimostrativi[3]. Per semplicità notazionale, continueremo a usare C per indicare la matrice dei costi *correnti*, inizialmente pari alla matrice dei costi originari. Si comincia sottraendo da ciascuna riga di C il proprio elemento minimo, ossia, per ogni riga i:

$$c_{ij} := c_{ij} - \min_j\{c_{ij}\}$$

si noti che, dopo l'aggiornamento, almeno un c_{ij} risulterà nullo. Fatto questo, si effettua un'analoga operazione per le colonne, ossia si pone, per ogni colonna j:

$$c_{ij} := c_{ij} - \min_i\{c_{ij}\}$$

A questo punto, sia E l'insieme degli archi (u_i, v_j) tali che $c_{ij} = 0$, e consideriamo il problema di trovare il matching di cardinalità massima sul grafo $G = (U, V, E)$, che supponiamo di affrontare utilizzando l'algoritmo risolutivo visto nel capitolo 3. Se tale matching M è perfetto, allora abbiamo trovato la soluzione ottima del problema di assegnamento. Altrimenti, procediamo come segue.

Nel corso dell'ultima iterazione dell'algoritmo risolutivo , alcuni nodi di U e di V sono stati etichettati, siano essi U^* e V^* rispettivamente (tali nodi definiscono la sezione S^* di capacità minima). Si noti che almeno un nodo di U^* è sicuramente esposto, altrimenti il matching corrente sarebbe perfetto. A questo punto, andiamo a marcare le righe e le colonne della matrice C corrispondenti ai nodi etichettati, e individuiamo il valore θ definito come:

$$\theta = \min\{c_{ij}|\ i \in U^*, j \notin V^*\}$$

A questo punto, alteriamo i valori della C andando a *sottrarre* θ da tutte le righe marcate, e a *sommare* θ a tutte le colonne marcate. La situazione è cioè quella raffigurata nella Tabella 1, in cui sono indicate righe e colonne marcate: di certo, uno dei valori che fanno

[3]Per una trattazione completa, si rinvia ad altri testi, ad esempio Papadimitriou, C., Steiglitz, K., Combinatorial Optimization: Algorithms and Complexity, 1982, Prentice-Hall.

parte delle aree indicate con "$-\theta$" diventerà pari a zero. Per costruzione, tale operazione lascia invece inalterati tutti gli zeri *corrispondenti ad archi del matching corrente*. Infatti, si noti che gli unici elementi di C che subiscono un aumento netto (di θ) sono gli elementi c_{ij} con $i \notin U^*$ e $j \in V^*$. Ma se l'arco (i, j) appartenesse a M, allora, quando l'algoritmo di visita del grafo ausiliario D arriva ad etichettare j, etichetterebbe anche i (come arco inverso non vuoto), che farebbe quindi parte di U^*. Di conseguenza, chiamando E' il nuovo insieme di archi, si ha che il matching M di cardinalità massima per $G = (U, V, E)$ è comunque un matching ammissibile sul grafo $G' = (U, V, E')$, e la ricerca di un matching di cardinalità massima su G' può quindi riprendere da M.

		$+\theta$	
•	-θ		-θ
•			
		$+\theta$	
•	-θ		-θ
		$+\theta$	

Tabella 1: Modifica della matrice dei costi.

È importante osservare che l'insieme dei nodi etichettati (durante la ricerca del nuovo cammino aumentante) sul nuovo grafo G' di certo *contiene strettamente* l'insieme dei nodi etichettati su G. Infatti, gli unici archi che possono sparire passando da E a E' sono archi (u_i, v_j) con $u_i \notin U^*$ e $v_j \in V^*$. Ma tali archi sono ininfluenti rispetto all'insieme dei nodi etichettati, nel senso che quest'ultimo rimane lo stesso anche se si rimuovessero tutti questi archi. D'altro canto, invece, l'aggiornamento della matrice C produce, in E', almeno un nuovo arco (u_i, v_j) con $u_i \in U^*$ e $v_j \notin V^*$, che ha quindi l'effetto di etichettare v_j, che non era etichettato in G. Quindi, dopo $O(n)$ passaggi, certamente si arriverà ad etichettare un nodo v_j esposto, e si sarà individuato un cammino aumentante. A quel punto si aggiorna il matching e si ricomincia, andando avanti fino a che il matching di cardinalità massima conterrà n archi.

Vediamo un esempio. Si consideri la matrice

$$C = \begin{pmatrix} 3 & 3 & 3 & 7 & 8 \\ 5 & 7 & 8 & 5 & 11 \\ 7 & 8 & 11 & 3 & 10 \\ 7 & 10 & 11 & 4 & 4 \\ 6 & 6 & 10 & 8 & 1 \end{pmatrix}$$

sottraendo a ogni riga l'elemento minimo, si ottiene

$$C = \begin{pmatrix} 0 & 0 & 0 & 4 & 5 \\ 0 & 2 & 3 & 0 & 6 \\ 4 & 5 & 8 & 0 & 7 \\ 3 & 6 & 7 & 0 & 0 \\ 5 & 5 & 9 & 7 & 0 \end{pmatrix}$$

siccome ogni colonna contiene almeno uno 0, sottraendo l'elemento minimo da ogni colonna la matrice non cambia. A questo punto possiamo calcolare un matching di massima cardinalità con gli archi corrispondenti agli elementi $c_{ij} = 0$. Tale matching è $M = \{(u_1, v_3); (u_2, v_1); (u_3, v_4); (u_4, v_5)\}$, di cardinalità 4. Nel corso dell'ultima iterazione dell'algoritmo di visita visto nel capitolo 3, ossia l'iterazione nella quale si certifica l'ottimalità del matching, vengono etichettati (nell'ordine) i nodi u_5, v_5, u_4, v_4, u_3.

$$C = \begin{matrix} \\ \bullet \\ \bullet \\ \bullet \end{matrix} \begin{matrix} \begin{matrix} & & & \bullet & \bullet \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 4 & 5 \\ 0 & 2 & 3 & 0 & 6 \\ 4 & 5 & 8 & 0 & 7 \\ 3 & 6 & 7 & 0 & 0 \\ 5 & 5 & 9 & 7 & 0 \end{pmatrix} \end{matrix}$$

A questo punto, calcoliamo il minimo degli elementi (i, j) con $u_i \in U^*$ e $v_j \notin V^*$. Tale valore è $\theta = 3$, che si ha per l'elemento $(4, 1)$. Effettuando l'aggiornamento della matrice dei costi, si tratta quindi di sottrarre 3 alle righe 3, 4 e 5, e sommare 3 alle colonne 4 e 5:

$$C = \begin{pmatrix} 0 & 0 & 0 & 7 & 8 \\ 0 & 2 & 3 & 3 & 9 \\ 1 & 2 & 5 & 0 & 7 \\ 0 & 3 & 4 & 0 & 0 \\ 2 & 2 & 6 & 7 & 0 \end{pmatrix}$$

L'arco (u_4, v_1) viene quindi aggiunto a E (si noti che scompare invece (u_2, v_4)). Sul nuovo grafo non vi sono cammini aumentanti, per cui il matching precedente è ancora di cardinalità massima. Ora però sono cambiati gli insiemi U^* e V^*:

$$C = \begin{matrix} \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{matrix} \begin{matrix} \begin{matrix} & \bullet & & \bullet & \bullet \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 7 & 8 \\ 0 & 2 & 3 & 3 & 9 \\ 1 & 2 & 5 & 0 & 7 \\ 0 & 3 & 4 & 0 & 0 \\ 2 & 2 & 6 & 7 & 0 \end{pmatrix} \end{matrix}$$

evidentemente ora $\theta = 2$, che si ha per gli archi $(u_2, v_2), (u_3, v_2)$ e (u_5, v_2), i quali vengono quindi tutti aggiunti a E (da cui esce (u_1, v_1)). La nuova C è

$$C = \begin{pmatrix} 2 & 0 & 0 & 9 & 10 \\ 0 & 0 & 1 & 3 & 9 \\ 1 & 0 & 3 & 0 & 7 \\ 0 & 1 & 2 & 0 & 0 \\ 2 & 0 & 4 & 7 & 0 \end{pmatrix}$$

dove a questo punto, partendo dallo stesso matching, è possibile individuare il cammino aumentante costituito dal singolo arco (u_5, v_2), che consente di aumentare la cardinalità del matching al valore 5. Dunque in definitiva la soluzione ottima è data da $x_{13} = 1, x_{21} = 1, x_{34} = 1, x_{45} = 1, x_{52} = 1$ e tutte le altre $x_{ij} = 0$. Il valore della soluzione ottima è 21.

Accenniamo infine alla complessità dell'algoritmo ungherese. Ogni qual volta l'algoritmo di visita si "ferma", ossia non trova più cammini aumentanti, è necessario aggiornare la matrice C dei costi. Il risultato di tale aggiornamento, come si è visto, è un allargamento degli insiemi U^* e V^*. Poiché, come pure si è visto, l'etichettatura può proseguire dalla precedente, in $O(n)$ aggiornamenti si arriva comunque a determinare un nuovo cammino aumentante, e dunque a incrementare di 1 il valore della cardinalità del matching. Se l'aggiornamento di C è effettuato in $O(n^2)$, il nuovo cammino è determinato in $O(n^3)$ e dunque la complessità dell'algoritmo è di $O(n^4)$. In realtà, utilizzando alcune strutture dati per rendere più efficiente l'aggiornamento della matrice C, si riesce ad abbassare la complessità a $O(n^3)$.