Claudio Arbib
Università di L'Aquila

Ricerca Operativa

Problemi combinatorici

(Gennaio 2006)

Sommario

Problemi combinatorici

Definizione. Un problema combinatorico è individuato da una coppia $(U, \Im)$ dove

$U = \{u_1, \ldots, u_n\}$ è un insieme finito (insieme universo)

$\Im = \{I \subseteq U: \wp(I)\} \subseteq 2^U$ è la famiglia dei sottoinsiemi di U (regione ammissibile) che verificano una certa proprietà $\wp$

La sua soluzione consiste nel rispondere alla domanda:

$$\Im = \emptyset?$$

esibendo, in caso contrario, un insieme $X \in \Im$.

Esempi

Zaino (*Knapsack* 0-1). Dati n reali positivi $a_1, \ldots, a_n$ trovare un loro sottoinsieme la cui somma non superi un dato reale $b > 0$

$$U = \{1, \ldots, n\}$$

$$\Im = \{X \subseteq U: \textstyle\sum_{i \in X} a_i \leq b\}$$

Edge-cover. Dato un grafo $G = (V, E)$, trovare un insieme C di archi tale che ogni vertice di G è estremo di almeno un arco di C (edge-cover).

$$U = E$$

$$\Im = \{X \subseteq U: \forall u \in V, \exists a \in X \text{ tale che } u \in a\}$$

Proposizione. Un edge-cover minimale è privo di cicli

Dimostrazione: sia Q un ciclo $\subseteq C$; allora $Q - e$ copre $V(Q)$.

Esempi

Insieme stabile. Dato un grafo $G = (V, E)$, trovare un insieme S di vertici a due a due non adiacenti (insieme stabile).

$$U = V$$

$$\Im = \{X \subseteq U : uv \notin E \ \forall u, v \in X\}$$

Matching. Dato un grafo $G = (V, E)$, trovare un insieme M di archi a due a due non adiacenti (matching).

$$U = E$$

$$\Im = \{X \subseteq U : a \cap b = \varnothing \ \forall a, b \in X\}$$

Definizione. Un matching che è anche un edge-cover si dice perfetto.

Esempi

Insieme dominante. Dato un grafo $G = (V, E)$, trovare un insieme D di vertici tale che ogni vertice di $V - D$ è adiacente ad almeno un vertice di D (insieme dominante).

$$U = V$$

$$\mathfrak{I} = \{X \subseteq U : X \text{ domina } V\}$$

Proposizione. Ogni insieme stabile massimale di G è anche dominante.

Dimostrazione: Se S non fosse dominante esisterebbe un $v \in V - S$ non adiacente ad alcun vertice di S.

Ma allora $S \cup \{v\}$ sarebbe stabile, e quindi S non massimale.

Esempi

Albero ricoprente. Dato un grafo $G = (V, E)$, trovare un insieme F di archi che individui un sottografo connesso e privo di cicli che tocca tutti i vertici di G (albero ricoprente).

$$U = E$$

$$\Im = \{X \subseteq U \colon X \text{ privo di cicli}\}$$

Osservazione. Il generico sottografo (V, X) di G con $X \in \Im$, in quanto non necessariamente connesso, non è in generale un albero bensì una foresta ricoprente G costituita da un certo numero di alberi (componenti connesse).

E' facile però verificare che ogni insieme massimale di $\Im$ individua un albero ricoprente G se e solo se G è connesso.

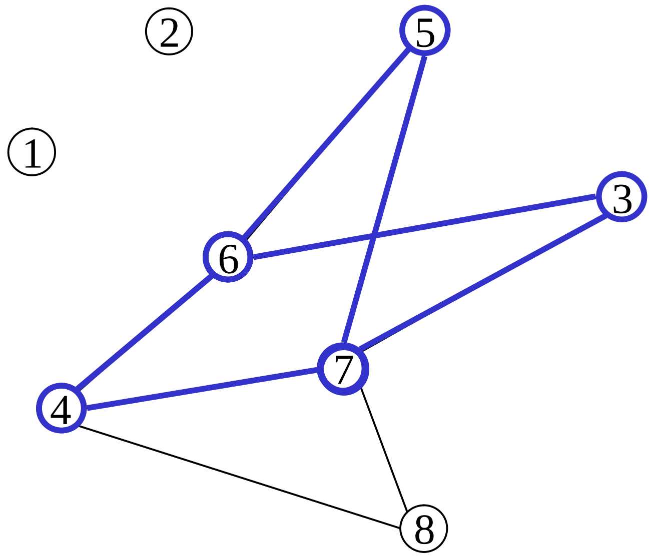
Esempi

Sottografo bipartito completo e bilanciato. Dato un grafo $G = (V, E)$, trovare un insieme F di archi che individui un sottografo isomorfo a $K_{m,m}$, per qualche $m > 0$.

$$U = E$$

$$\mathfrak{I} = \{X \subseteq U\colon \exists m > 0,\ (V(X), X) \cong K_{m,m}\}$$

$V(F)$ = insieme dei vertici di G che sono estremi di qualche arco di F.



$\{(4, 6)\} \in \mathfrak{I}$

$\{(5, 6), (5, 7), (3, 6), (3, 7)\} \in \mathfrak{I}$

$\{(4, 6), (4, 7)\} \notin \mathfrak{I}$

Ottimizzazione combinatoria

Definizione. Un problema di ottimizzazione combinatoria è una terna $P = (U, \Im, c)$, dove c: $U \rightarrow$ IR è una funzione che associa un numero reale a ogni elemento dell'insieme universale (funzione peso).

Definizione. Per ogni $X \subseteq U$ si definisce peso di X il numero reale

$$c(X) = \sum_{u \in X} c(u)$$

Un problema di ottimizzazione combinatoria consiste nel trovare una $X^* \in \Im$ tale che

$$c(X^*) \leq c(X) \qquad \text{(problema di minimo)}$$

ovvero

$$c(X^*) \geq c(X) \qquad \text{(problema di massimo)} \quad \forall X \in \Im$$

Definizione. L'insieme X^* si dice soluzione ottima di P.

Algoritmo universale

Un problema di ottimizzazione combinatorica è apparentemente semplice: a differenza di altri problemi, le sue soluzioni sono infatti in numero finito.

E' quindi possibile costruire un algoritmo in grado di risolvere qualunque problema di ottimizzazione combinatoria basandosi su tre funzionalità:

1) enumerare tutti i sottoinsiemi di U
2) verificare $\wp(X)$, cioè se il sottoinsieme X correntemente enumerato è ammissibile
3) In caso affermativo, valutare $c(X)$ e, se migliore del valore ottimo corrente, aggiornare la soluzione.

Algoritmo universale

Esempio. La portata di un ascensore è di 260 chilogrammi.
Al piano terra si trovano:

Andrea:	24 anni	80 kg
Bruno:	60 anni	110 kg
Claudio:	43 anni	75 kg
Daniele:	16 anni	70 kg

Sebbene il galateo indichi di far salire prima le persone più anziane, in tempi di democrazia si preferisce un compromesso: massimizzare l'età complessiva del gruppo che sale per primo.

Cominciamo allora a enumerare le possibili soluzioni insieme al peso e al valore corrispondente.

Algoritmo universale

Soluzione	*Peso*	*Valore*	*Ottimo corrente*
$S_0 = \emptyset$	0	0	S_0
$S_1 = \{A\}$	80	24	S_1
$S_2 = \{B\}$	110	60	S_2
$S_3 = \{C\}$	75	43	S_2
$S_4 = \{D\}$	70	16	S_2
$S_5 = \{A, B\}$	190	84	S_5
$S_6 = \{A, C\}$	155	67	S_5
$S_7 = \{A, D\}$	150	40	S_5
$S_8 = \{B, C\}$	185	103	S_8
$S_9 = \{B, D\}$	180	76	S_8

Algoritmo universale

Soluzione	*Peso*	*Valore*	*Ottimo corrente*
S_{10} = {C, D}	145	59	S_8
S_{11} = {A, B, C}	265	(inammissibile)	S_8
S_{12} = {A, B, D}	260	100	S_8
S_{13} = {A, C, D}	225	83	S_8
S_{14} = {B, C, D}	255	119	S_{14}
S_{15} = {A, B, C, D}	335	(inammissibile)	S_{14}

La soluzione ottima è stata individuata dopo 16 passi di verifica e valutazione

Algoritmo universale

La complessità di un algoritmo è sostanzialmente legata a due fattori

- la quantità di memoria e
- il numero di passi elementari

che l'algoritmo richiede per terminare correttamente.

Ora, l'algoritmo universale richiede di costruire, verificare e valutare 2^n insiemi.

Se ad esempio $n = 100$, le operazioni richieste sono

$$2^{100} = (2^{10})^{10} = 1024^{10} > 1000^{10} = 10^{30} =$$

1.000.000.000.000.000.000.000.000.000.000

Algoritmo universale

Un processore operante a 500MHz (in grado quindi di eseguire, in linea di principio, 500 milioni di operazioni al secondo) richiederebbe più di 2.000.000.000.000.000.000.000 secondi = circa 63.419.583.967.529 anni.

(l'età dell'universo è stimata in 13 miliardi di anni)

L'algoritmo universale può dunque largamente eccedere le possibilità di qualsiasi computer e di qualsiasi tecnologia.

Ù

Un algoritmo meno complesso

Un'idea molto intuitiva per risolvere un problema di ottimizzazione combinatorica può essere quella di costruire la soluzione aggiungendo a ogni passo un elemento di U scelto tra i più convenienti, fino ad esaurire gli elementi disponibili.

Questo metodo viene detto "greedy" (ingordo), e si può descrivere così:

Algoritmo Greedy (per un problema di massimizzazione)

Inizializzazione: $S := \emptyset$

1) Scegli un $u \in U$ tale che $c(u) \geq c(x)$ per ogni $x \in U$
2) Se $S \cup \{u\} \in \mathfrak{I}$, allora $S := S \cup \{u\}$
3) Poni $U := U - \{u\}$; se $U \neq \emptyset$ torna a (1), altrimenti restituisci S e stop.

Un algoritmo meno complesso

L'algoritmo greedy richiede chiaramente n passi di verifica, contro i 2^n dell'algoritmo universale.

Applicato al problema dell'ascensore si comporta così:

Soluzione	*Peso*	*Valore*	*Elementi restanti*
$S_0 = \emptyset$	0	0	A, B, C, D
$S_1 = \{B\}$	110	60	A, C, D
$S_3 = \{B, C\}$	185	103	A, D
$S_4 = \{B, C, A\}$	265	(inammissibile)	D
$S_5 = \{B, C, D\}$	255	119	

Limiti dell’algoritmo greedy

Sorgono ora spontanee 2 domande:

1) Nel caso dell’ascensore l’algoritmo greedy ha fornito una soluzione ammissibile. Ciò è sempre vero?
2) Inoltre, com’è facile verificare confrontando con l’algoritmo universale, la soluzione proposta dall’algoritmo greedy nell’esempio dell’ascensore è anche ottima. Ciò è sempre vero?

Ammissibilità della soluzione

La risposta alla domanda (1) è no, a meno di non modificare la codifica sopra descritta.

Infatti l'algoritmo greedy costruisce la soluzione in maniera incrementale, presupponendo implicitamente che se X è una soluzione lo è anche qualunque sottoinsieme Y di X.

(Il problema del Grande Fratello, ad esempio, non può essere risolto con la codifica descritta).

Diamo allora la seguente:

Definizione. Un problema di ottimizzazione combinatoria si dice subclusivo se $\emptyset \in \Im$ e

$$\forall X \in \Im, Y \subseteq X \quad \Rightarrow \quad Y \in \Im$$

(proprietà di subclusione)

Ammissibilità della soluzione

I problemi dell'insieme stabile, dello zaino, del matching sono subclusivi.

I problemi dell'edge-cover, dell'insieme dominante, del matching perfetto, dell'albero ricoprente non sono subclusivi.

Possiamo allora dire che se il problema è subclusivo, la nostra codifica dell'algoritmo greedy è in grado di fornire una soluzione ammissibile.

Problemi non subclusivi

Il caso di alcuni problemi non subclusivi può essere trattato modificando opportunamente la codifica. Questi problemi rispondono alla seguente:

Definizione. Chiamiamo superclusivo un problema di ottimizzazione combinatoria nel quale $U \in \Im$ e

$$\forall X \in \Im, Y \supseteq X \quad \Rightarrow \quad Y \in \Im.$$

I problemi dell'edge-cover e dell'insieme dominante sono superclusivi.

Quelli del matching perfetto e dell'albero ricoprente non lo sono.

Problemi non subclusivi

Per risolvere un problema superclusivo si può provare a costruire la soluzione per sottrazione a partire dall'insieme universale.

Algoritmo Greedy (modificato per un problema di minimizzazione superclusivo)

Inizializzazione: $S := U$

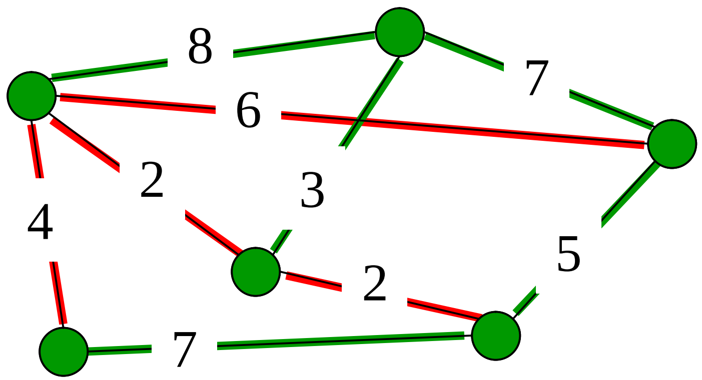
1) Scegli un $u \in S$ tale che $c(u) \geq c(x)$ per ogni $x \in S$
2) Se $S - \{u\} \in \Im$, allora $S := S - \{u\}$
3) Poni $U := U - \{u\}$; se $U \neq \emptyset$ torna a (1), altrimenti restituisci S e stop.

Problemi non subclusivi

- Infine osserviamo che in alcuni casi non subclusivi né superclusivi la soluzione cercata risulta massimale ovvero minimale in una determinata famiglia $\mathfrak{I}$.
- Ad esempio un matching perfetto (se esiste) è senza dubbio massimale nella famiglia dei matching di G e minimale nella famiglia degli edge-cover di G.
- Allo stesso modo un albero ricoprente è massimale nella famiglia degli insiemi di archi di G che non formano cicli.
- Ora la soluzione fornita al termine della prima (della seconda) codifica dell'algoritmo è senza dubbio massimale (minimale). Tuttavia non è detto che questa soluzione sia ammissibile.

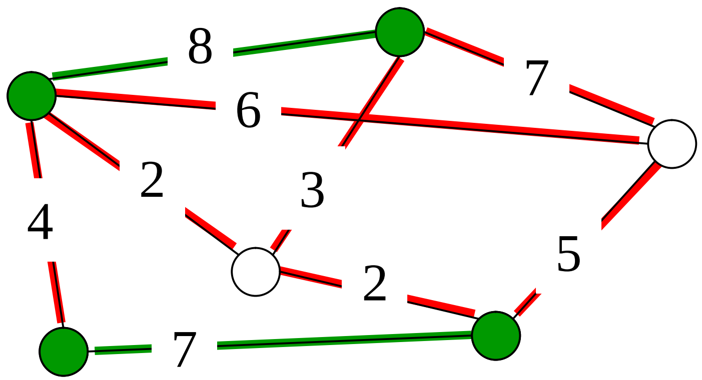
Problemi non subclusivi

Nel caso dell'albero ricoprente la cosa funziona perché un grafo massimale privo di cicli è sempre connesso (e quindi è un albero).



Problemi non subclusivi

Nel caso del matching perfetto invece in generale la cosa non funziona, perché un matching massimale non è necessariamente perfetto.



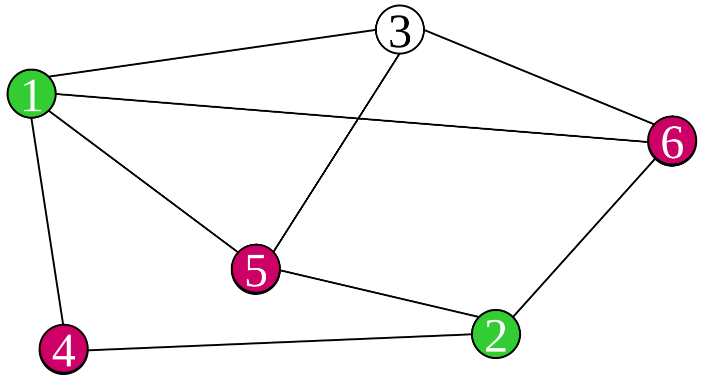
Ottimalità della soluzione

Ma, domanda (2), possiamo sempre dire che la soluzione fornita dall’algoritmo è ottima?

La risposta, in generale, è ancora una volta no. Vediamo qualche esempio.

Ottimalità della soluzione

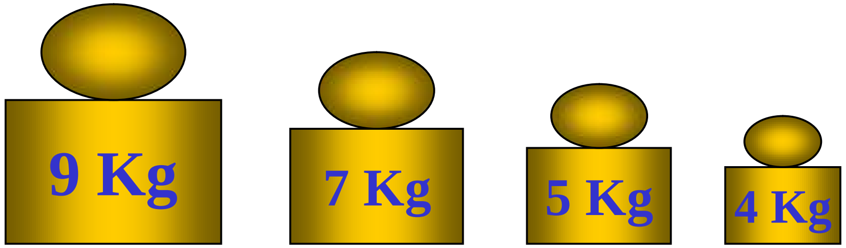
Insieme stabile. Applichiamo l’algoritmo greedy per individuare il più grande insieme stabile:



Siccome i pesi sono tutti pari a 1, l’algoritmo greedy può selezionare inizialmente il nodo 1, dopodiché qualunque nodo venga scelto si termina in un insieme massimale.
Tuttavia esiste un insieme stabile di 3 elementi.

Ottimalità della soluzione

Zaino. Applichiamo l'algoritmo greedy alla ricerca della composizione più conveniente di uno zaino in grado di portare fino a b = 12 chilogrammi:

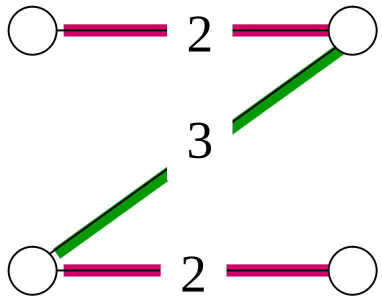


10€ 8€ 7€ 5€

L'algoritmo greedy selezionerà inizialmente l'oggetto da 10€, ma dopo questo nessun altro oggetto può entrare nello zaino, per cui l'oggetto scelto forma un insieme massimale.
Tuttavia esiste un insieme ammissibile del valore di 15€.

Ottimalità della soluzione

Matching massimo. Utilizziamo l'algoritmo greedy per cercare un matching di peso massimo in un grafo bipartito:



L'algoritmo greedy selezionerà inizialmente l'arco di peso 3, ma dopo questo nessun altro arco potrà essere aggiunto, per cui l'arco scelto forma un insieme massimale. Tuttavia esiste un matching di peso 4.

Ottimalità della soluzione

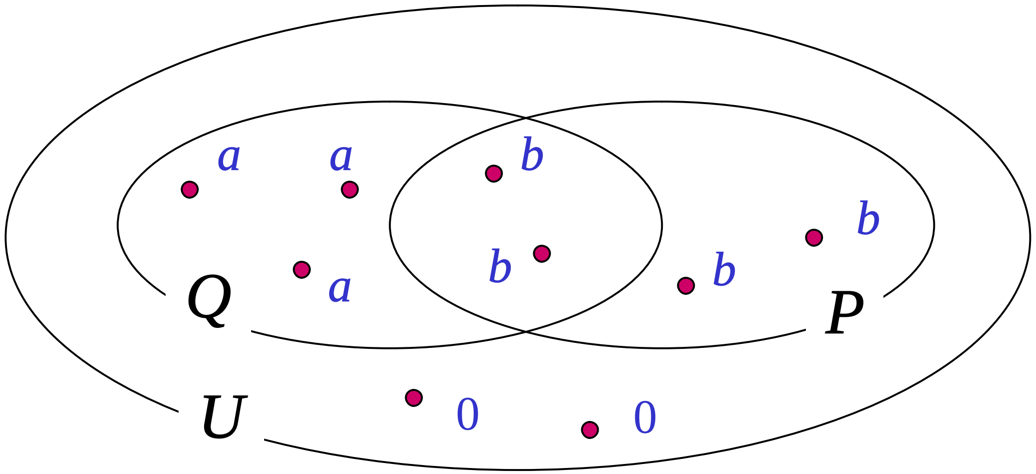
- Esiste allora qualche problema risolto sempre all'ottimo dall'algoritmo greedy?
- Se esaminiamo i fallimenti finora osservati e ci chiediamo cosa abbiano in comune, possiamo osservare che in tutti i problemi esaminati l'algoritmo greedy fallisce perché $\Im$ contiene un insieme massimale che non è anche massimo.
- Questo comportamento è del tutto generale e si può sintetizzare nel seguente teorema:

Teorema 1. Se $\Im$ contiene massimali di cardinalità diversa, allora è sempre possibile costruire una funzione peso c: $U \rightarrow$ IR in grado di far fallire l'algoritmo greedy nella ricerca di una soluzione ottima del problema di massimizzazione (U, $\Im$, c).

Ottimalità della soluzione

Dimostrazione: supponiamo che $\Im$ contenga due insiemi massimali P, Q con $|P| < |Q|$. Costruiamo la funzione c: $U \rightarrow \mathrm{IR}$ come illustrato nello schema seguente

Supponiamo $0 < a < b$
$|P - Q| = p < q = |Q - P|$
$|P \cap Q| = k$.



L'algoritmo greedy sceglierà la soluzione P, di valore $c(P) = (k + p)b$.

Scegliendo $a \in (pb/q, b)$ (cosa lecita perché $p/q < 1$) si ha però $c(Q) > c(P)$, quindi l'algoritmo greedy fallisce.

La proprietà di scambio

È naturale a questo punto concentrare l'attenzione su quei problemi nei quali gli insiemi massimali hanno tutti la medesima cardinalità.

Questo accade senza dubbio se la regione ammissibile $\Im$ soddisfa la seguente

Proprietà (scambio): Qualunque siano $X, Y \in \Im$ con $|X| < |Y|$, esiste sempre un $y \in Y - X$ tale che $X \cup \{y\} \in \Im$.

In altre parole, comunque prendiamo due insiemi in $\Im$ è sempre possibile trovare nel più grande almeno un elemento che non è nel più piccolo il quale, aggiunto a esso, forma un insieme ammissibile.

La proprietà di scambio

È facile ora verificare la seguente

Proposizione: Se $\mathfrak{I}$ gode della proprietà di scambio, allora tutti i suoi massimali hanno lo stesso numero di elementi.

Dimostrazione: Supponiamo che $\mathfrak{I}$ contenga due massimali X e Y con $|Y| > |X|$. Se applichiamo la proprietà di scambio possiamo allora trovare un $y \in Y - X$ che, aggiunto a X, forma un insieme appartenente a $\mathfrak{I}$.

Ma allora X non sarebbe massimale.

Domanda: È vero il viceversa? Cioè, se tutti i massimali di $\mathfrak{I}$ hanno lo stesso numero di elementi vale la proprietà di scambio?

La risposta, in generale, è no. Provatelo con un esempio.

Caratterizzazione del greedy

A cosa serve la proprietà di scambio? La sua utilità è legata al seguente

Teorema 2 (Rado): Dato un problema di ottimizzazione combinatoria subclusivo $P = (U, \Im, c)$, l'algoritmo greedy determina una sua soluzione ottima qualunque sia la funzione peso c se e solo se $\Im$ gode della proprietà di scambio.

Dimostrazione ($\Leftarrow$). Per dimostrare che in assenza della proprietà di scambio il metodo greedy può fallire, basta costruire un'opportuna funzione peso, in modo analogo a quanto fatto nel Teorema 1.

($\Rightarrow$). Viceversa, dimostriamo ora che se vale la proprietà di scambio il metodo greedy non può fallire. Consideriamo la prima codifica del metodo e, per assurdo, supponiamo che il peso del massimale X fornito dal metodo greedy sia inferiore a quello di un'altro Y ammissibile:

$$c(X) < c(Y) \qquad (1)$$

Caratterizzazione del greedy

Segue dimostrazione ($\Rightarrow$).

Siano $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$.

Senza perdere in generalità supponiamo

$$c(x_1) \geq c(x_2) \geq \ldots \geq c(x_n)$$

$$c(y_1) \geq c(y_2) \geq \ldots \geq c(y_n)$$

Sia k il primo indice per cui $c(x_k) \neq c(y_k)$. Dalla (1) si ricava

$$\sum_{i \geq k} c(x_i) < \sum_{i \geq k} c(y_i)$$

e quindi esiste senz'altro un indice $j \geq k$ per il quale $c(x_j) < c(y_j)$:

$$\begin{array}{c} c(y_1) \geq \ldots \geq c(y_j) \geq \ldots \geq c(y_n) \\ \vee \\ c(x_1) \geq \ldots \geq c(x_j) \geq \ldots \geq c(x_n) \end{array} \qquad (2)$$

Caratterizzazione del greedy

Segue dimostrazione ($\Rightarrow$).

Siano allora $X_{j-1} = \{x_1, \ldots, x_{j-1}\}$ e $Y_j = \{y_1, \ldots, y_{j-1}, y_j\}$

Siccome P è subclusivo, $X_{j-1}, Y_j \in \Im$.

Inoltre $|X_{j-1}| < |Y_j|$.

Quindi, per la proprietà di scambio, esiste un $y_h \in Y_j - X_{j-1}$ che può essere aggiunto a X_{j-1}.
Ma per la (2) questo y_h ha un peso sicuramente superiore a quello di x_j.
Ciò contraddice la scelta operata dal metodo greedy.

Fine della dimostrazione

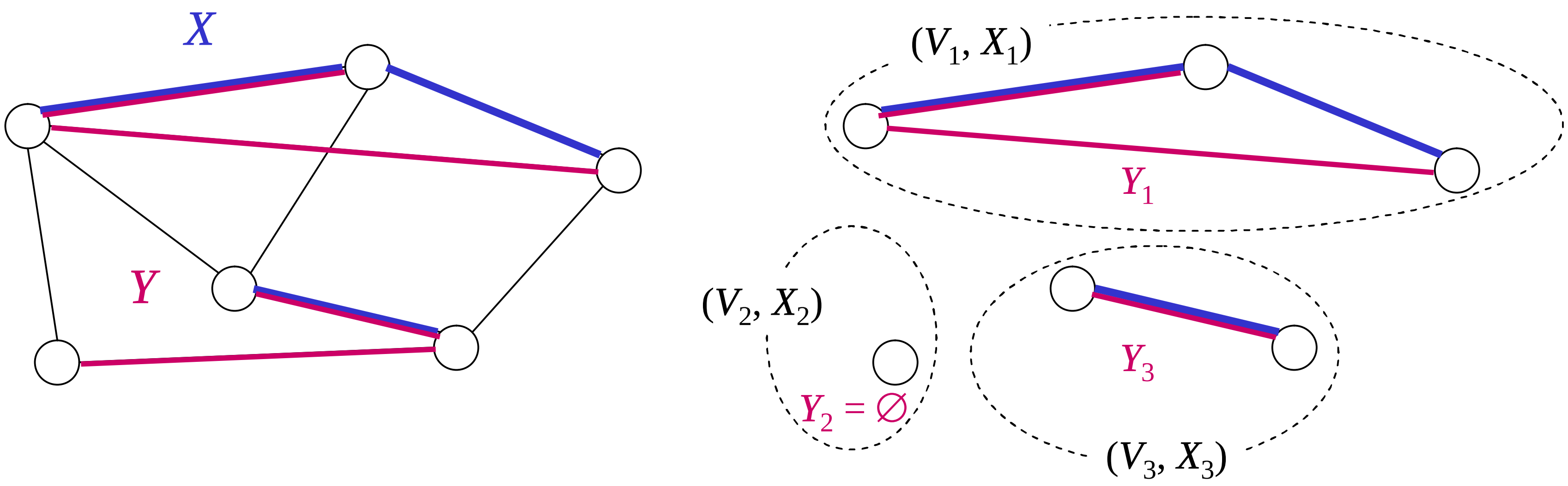
Matroidi

- Il Teorema di Rado pone l'accento su quei problemi subclusivi definiti da una coppia $(U, \Im)$ dotata della proprietà di scambio.
- Una tale coppia si dice matroide.
- Resta ora da far vedere che la caratterizzazione data ha un senso pratico. In altri termini, esistono esempi concreti di matroidi? In effetti, si può dimostrare quanto segue:

Teorema: La famiglia $\Im$ costituita dagli insiemi di archi privi di cicli gode della proprietà di scambio. Il metodo greedy risolve quindi all'ottimo il problema dell'albero ricoprente.

Dimostrazione. Sia $G = (V, E)$ un grafo simmetrico e $X, Y \subseteq E$ due insiemi di archi privi di cicli con $|X| < |Y|$. L'insieme X individua un sottografo di G con k componenti connesse $(V_1, X_1), \ldots, (V_k, X_k)$.

Matroidi



Segue dimostrazione. Indichiamo con Y_i l'insieme degli archi di Y che hanno entrambi gli estremi in V_i, $i = 1, \ldots, k$.

Siccome (V, Y) è privo di cicli, lo è anche (V_i, Y_i). Quindi

$$|Y_i| \leq |V_i| - 1$$

Matroidi

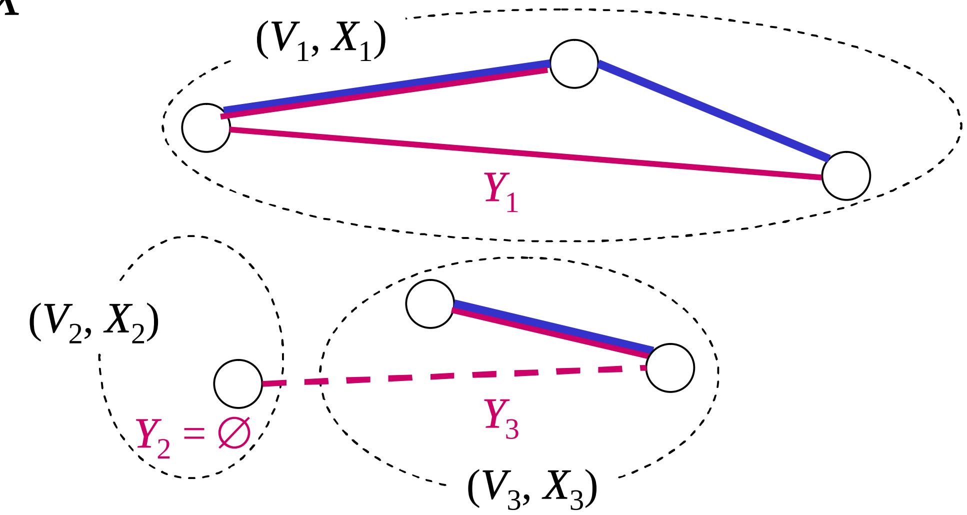
Segue dimostrazione. Possiamo allora scrivere

$$|Y| > |X| = \sum_{i=1..k} |X_i| = \sum_{i=1..k} (|V_i| - 1) \geq \sum_{i=1..k} |Y_i|$$

dal che si deduce che Y contiene almeno un elemento che non si trova in nessun Y_i.

In altre parole, esiste un arco di $Y - X$ che ha gli estremi in insiemi V_i differenti.

Tale arco non forma cicli con X e può quindi essere aggiunto a X senza problemi.



Matroidi

(Problema 5: produzione del vetro)

Obiettivo: produrre i pezzi nei quantitativi richiesti minimizzando l'area totale delle lastre utilizzate

Ipotesi: per semplicità, tutti i pezzi dello stesso tipo vengono tagliati da lastre di una medesima dimensione

domanda	713	611	… 248 …	897

	pz. 1	pz. 2	… pz. *i* …	pz. *m*
lastra 1	2685	2295	930	3375
lastra 2	2400	$\boxed{2040}$	840	3000
.				
lastra *k*	3012	2422	$\boxed{781}$	2317
.				
lastra *n*	$\boxed{2304}$	2520	918	$\boxed{2102}$

Il problema consiste nello scegliere un tipo di lastra per ogni tipo di pezzo da produrre.

Scegliere un certo tipo di lastra per un certo tipo di pezzo non influenza la scelta per gli altri tipi di pezzo

Quindi si può scegliere, per ogni tipo di pezzo, quel tipo di lastra che minimizzi l'area necessaria.

Proprietà 1: Una soluzione ottima può essere calcolata con l'algoritmo *greedy*.

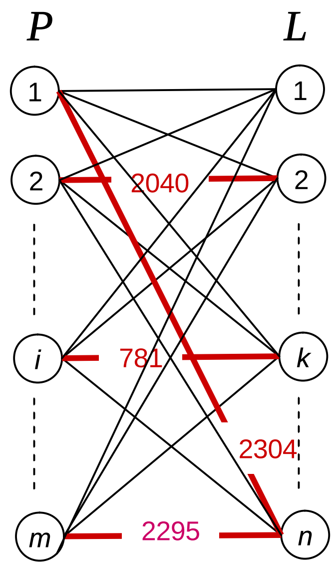
Matroidi

(Problema 5: produzione del vetro)

Problema: in un grafo bipartito completo $G = (P \cup L, P \times L)$, trovare un assegnamento di P a L avente peso minimo

domanda	713	611	… 248 …	897

	pz. 1	pz. 2	… pz. *i* …	pz. *m*
lastra 1	2685	2295	930	3375
lastra 2	2400	$\boxed{2040}$	840	3000
.				
lastra *k*	3012	2422	$\boxed{781}$	2317
.				
lastra *n*	$\boxed{2304}$	2520	918	$\boxed{2102}$



Teorema: La famiglia $\Im$ degli archi di un grafo bipartito che formano un assegnamento definisce il *matroide partizione*

Dimostrazione. Corollario della Proprietà 1 e del Teorema di Rado.