Algoritmi Euristici

Corso di Laurea in Informatica e Corso di Laurea in Matematica

Roberto Cordone

DI - Università degli Studi di Milano

Lezioni:	Lunedì 13.30 - 15.30 in Aula G30
	Giovedì 13.30 - 15.30 in Aula G30
Ricevimento:	su appuntamento
Tel.:	02 503 16235
E-mail:	roberto.cordone@unimi.it
Web page:	http://homes.di.unimi.it/~cordone/courses/2018-ae/2018-ae.html

Ottimizzazione Combinatoria

Un problema di Ottimizzazione Combinatoria si può formulare come

$$\begin{array}{l} \text{opt } f(x) \\ x \in X \end{array}$$

con $X \subseteq 2^B$ e B finito

Consideriamo una serie di possibili classi di esempi:

- problemi su insiemi (pesati, dotati di metrica, da ripartire...)
- problemi su funzioni logiche
- problemi su matrici numeriche
- problemi su grafo

Perché una galleria di problemi?

Abbiamo bisogno di conoscere molti problemi perché

- il corso presenta idee astratte, ma occorre imparare a concretizzare ciascuna idea in algoritmi diversi per problemi diversi
- la stessa idea può funzionare bene o male secondo il problema cui viene applicata
- alcune idee richiedono problemi con una struttura specifica
- problemi diversi possono avere relazioni non evidenti, che si possono sfruttare per progettare algoritmi

Quindi, se vi troverete ad affrontare problemi nuovi dopo il corso, potrete

- applicare le idee astratte anche ai problemi nuovi
- intuire e sfruttare relazioni fra problemi nuovi e problemi noti

Certo, esiste il rischio "Magical Number Seven"...

Per controllarlo faremo degli intermezzi dedicati a osservazioni generali

Problemi su insiemi pesati: il problema dello zaino (KP)

Si vuole scegliere da un insieme di oggetti voluminosi un sottoinsieme di valore massimo che si possa racchiudere in uno zaino di capacità limitata

- un insieme O di oggetti elementari
- una funzione $v : O \to \mathbb{N}$ che descrive il volume di ogni oggetto
- un numero $V \in \mathbb{N}$ che descrive la capacità di uno zaino
- una funzione $\phi : O \to \mathbb{N}$ che descrive il valore di ogni oggetto

L'insieme base è banalmente quello degli oggetti: $B = O$

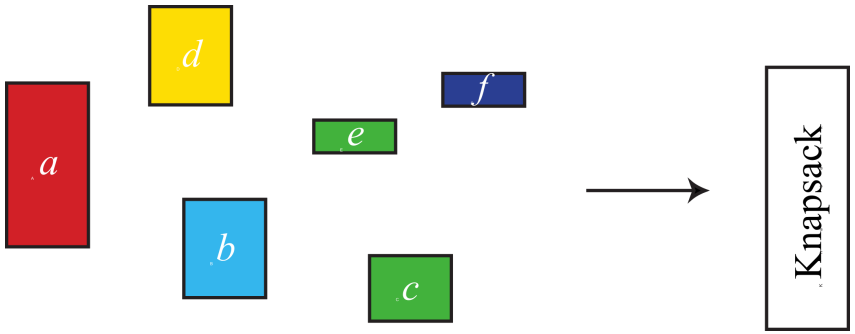
La regione ammissibile contiene i sottoinsiemi di oggetti di volume totale non superiore alla capacità dello zaino

$$X = \left\{ x \subseteq B : \sum_{j \in x} v_j \leq V \right\}$$

L'obiettivo è massimizzare il valore complessivo degli oggetti scelti

$$\max_{x \in X} f(x) = \sum_{j \in x} \phi_j$$

Esempio

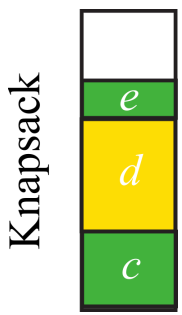


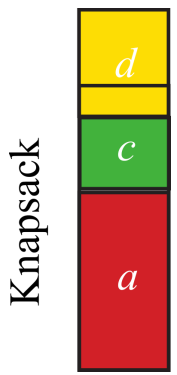
O	a	b	c	d	e	f
ϕ	7	2	4	5	4	1
v	5	3	2	3	1	1

$V = 8$

$x' = \{c, d, e\} \in X$
$f(x') = 13$

$x'' = \{a, c, d\} \notin X$
$f(x'') = 16$





Problemi su insiemi dotati di metrica: il *Maximum Diversity Problem* (*MDP*)

Si vuole scegliere da un insieme di punti un sottoinsieme di k punti con la massima somma delle distanze reciproche

- un insieme P di punti
- una funzione $d : P \times P \to \mathbb{N}$ che dà la distanza fra coppie di punti
- un numero $k \in \{1, \ldots, |P|\}$ che dà il numero di punti da scegliere

L'insieme base è l'insieme dei punti: $B = P$

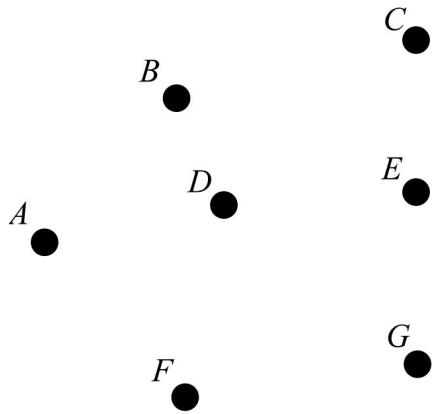
La regione ammissibile contiene i sottoinsiemi di k punti

$$X = \{x \subseteq B : |x| = k\}$$

L'obiettivo è massimizzare la distanza totale reciproca fra i punti scelti

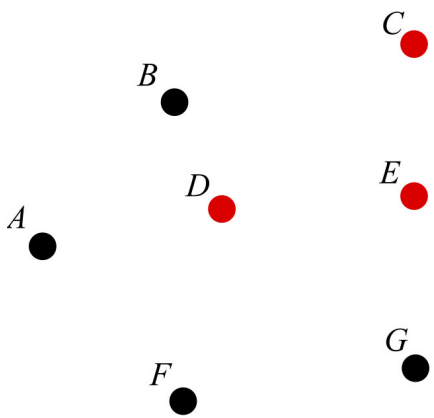
$$\max_{x \in X} f(x) = \sum_{i,j \in x} d_{ij}$$

Esempio

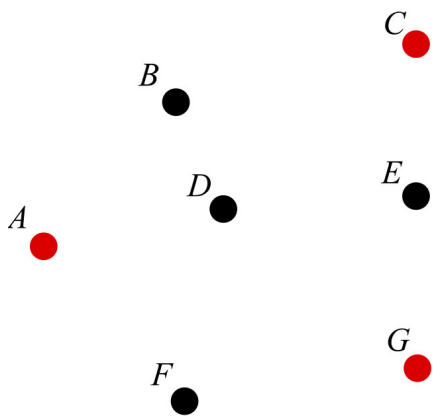


$$k = 3$$

$$x' = \{C, D, E\} \in X$$
$$f(x') = 24$$



$$x'' = \{A, C, G\} \in X$$
$$f(x'') = 46$$



Intermezzo 1: la funzione obiettivo

La funzione obiettivo associa valori interi a sottoinsiemi ammissibili

$$f : X \to \mathbb{N}$$

Calcolare la funzione obiettivo può essere complicato (persino esaustivo)

Abbiamo visto due casi semplici

- il *KP* ha una funzione obiettivo additiva che somma valori di una funzione ausiliaria definita sull'insieme base

$$\phi : B \to \mathbb{N} \text{ induce } f(x) = \sum_{j \in x} \phi_j : X \to \mathbb{N}$$

- il *MDP* ha una funzione obiettivo quadratica

Entrambe sono definite non solo su X, ma su tutto 2^B (*può essere utile?*)

Entrambe sono facili da calcolare, ma le funzioni additive $f(x)$ sono anche rapide da ricalcolare se il sottoinsieme x cambia di poco: basta

- sommare ϕ_j per ogni elemento j aggiunto a x
- sottrarre ϕ_j per ogni elemento j tolto da x

Per la funzione quadratica, la cosa pare più complicata (*ne riparleremo*)

Problemi su insiemi da ripartire: il *Bin Packing Problem* (*BPP*)

Si vuole dividere un insieme di oggetti voluminosi nel minimo numero di contenitori di capacità data

- un insieme O di oggetti elementari
- una funzione $v : O \to \mathbb{N}$ che descrive il volume di ogni oggetto
- un insieme C di contenitori
- un numero $V \in \mathbb{N}$ che dà il volume dei contenitori

L'insieme base $B = O \times C$ contiene le coppie (oggetto,contenitore)

La regione ammissibile contiene le partizioni degli oggetti tra i contenitori tali da non eccedere la capacità di alcun contenitore

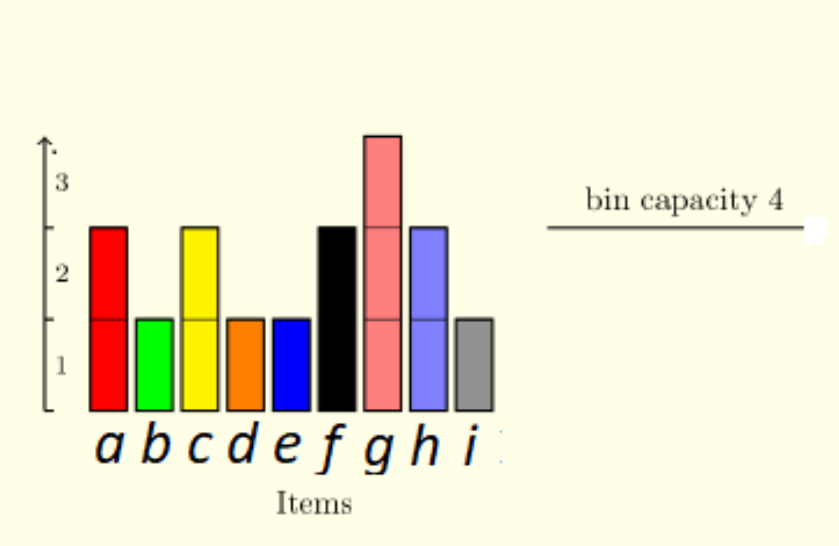
$$X = \left\{ x \subseteq B : |x \cap B_o| = 1 \; \forall o \in O, \sum_{(o,c) \in B^c} v_o \leq V \; \forall c \in C \right\}$$

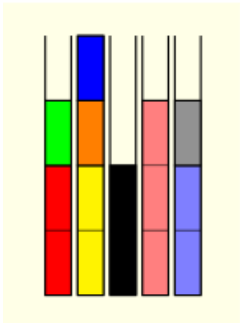
con $B_o = \{(i,j) \in B : i = o\}$, $B^c = \{(i,j) \in B : j = c\}$

L'obiettivo è minimizzare il numero di contenitori usati

$$\min_{x \in X} f(x) = |\{c \in C : x \cap B^c \neq \emptyset\}|$$

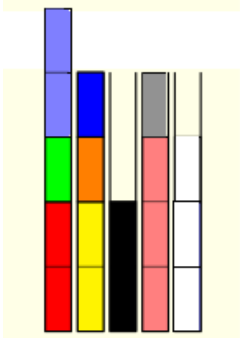
Esempio





$$x' = \{(a,1),(b,1),(c,2),(d,2),(e,2),(f,3),(g,4),(h,5),(i,5)\} \in X$$

$$f(x') = 5$$



$$x'' = \{(a,1),(b,1),(c,2),(d,2),(e,2),(f,3),(g,4),(h,1),(i,4)\} \notin X$$

$$f(x'') = 4$$

Problemi su insiemi da ripartire: il *Parallel Machine Scheduling Problem* (*PMSP*)

Si vuole dividere un insieme di lavorazioni fra un dato insieme di macchine minimizzando il tempo di completamento

- un insieme L di lavorazioni
- una funzione $d : L \to \mathbb{N}$ che descrive la durata di ogni lavorazione
- un insieme M di macchine

L'insieme base $B = L \times M$ contiene le coppie (lavorazione,macchina)

La regione ammissibile contiene le partizioni delle lavorazioni tra le macchine (*l'ordine delle lavorazioni è irrilevante!*)

$$X = \left\{ x \subseteq B : |x \cap B_\ell| = 1 \; \forall \ell \in L \right\}$$

L'obiettivo è minimizzare la massima durata totale per ogni macchina

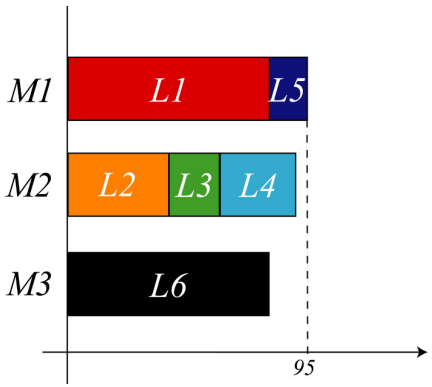
$$\min_{x \in X} f(x) = \max_{m \in M} \sum_{\ell:(\ell,m) \in x} d_\ell$$

Esempio

$$L = \{L1, L2, L3, L4, L5, L6\}$$

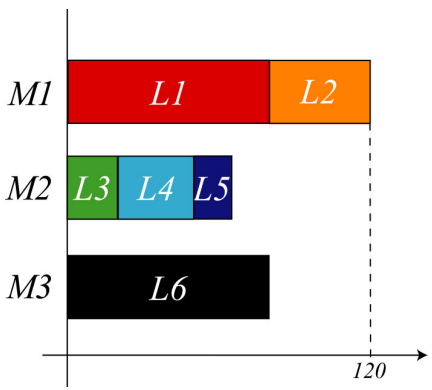
$$M = \{M1, M2, M3\}$$

Lavorazione	$L1$	$L2$	$L3$	$L4$	$L5$	$L6$
d	80	40	20	30	15	80



$$x' = \{(L1, M1), (L2, M2), (L3, M2), (L4, M2), (L5, M1), (L6, M3)\} \in X$$

$$f(x') = 95$$



$$x'' = \{(L1, M1), (L2, M1), (L3, M2), (L4, M2), (L5, M2), (L6, M3)\} \in X$$

$$f(x'') = 120$$

Intermezzo 2: ancora la funzione obiettivo

La funzione obiettivo di *BPP* e *PMSP*

- non è additiva
- non è banale da calcolare

Piccole modifiche nella soluzione hanno impatto variabile sull'obiettivo

- impatto uguale alla durata delle lavorazioni spostate (es., $L5$ su $M1$ in x'')
- impatto nullo (es., $L5$ su $M3$ in x'')
- impatto intermedio (es., $L2$ su $M2$ in x'')

Infatti, l'impatto di una modifica alla soluzione dipende

- sia dagli elementi modificati
- sia da quelli non modificati

La funzione obiettivo è "piatta": molte soluzioni hanno lo stesso valore

Problemi su funzioni logiche: il *Max-SAT*

Data una CNF, si cerca l'assegnamento di verità alle variabili logiche che soddisfi l'insieme di formule di peso/cardinalità massima

- un insieme V di variabili logiche x_j
- una forma congiuntiva normale (*CNF*) definita su tali variabili
- una funzione peso w associata alle formule che compongono la *CNF*

dove

- variabile logica x_j è una variabile che assume valore in $\mathbb{B} = \{0, 1\}$
- letterale ℓ_j è una funzione ridotta a una variabile affermata o negata

$$\ell_j(x) \in \{x_j, \bar{x}_j\}$$

- formula logica è una disgiunzione o somma logica (*OR*) di letterali

$$C_i(x) = \ell_{i,1} \vee \ldots \vee \ell_{i,n_i}$$

- forma congiuntiva normale (*CNF*) è una congiunzione o prodotto logico (*AND*) di formule logiche

$$CNF(x) = C_1 \wedge \ldots \wedge C_n$$

- soddisfare una funzione logica significa farle assumere valore 1

Problemi su funzioni logiche: il *Max-SAT*

L'insieme base è l'insieme degli assegnamenti di verità

$$B = V \times \mathbb{B} = \{(x_1, 0), (x_1, 1), \ldots, (x_n, 0), (x_n, 1)\}$$

La regione ammissibile contiene i sottoinsiemi di assegnamenti

- completi: per ogni variabile compare almeno un letterale
- coerenti: per ogni variabile compare un solo letterale

$$X = \{x \subseteq B : |x \cap B_v| = 1 \; \forall v \in V\}$$

con $B_{x_j} = \{(x_j, 0), (x_j, 1)\}$

L'obiettivo è massimizzare il peso totale delle formule soddisfatte

$$\max_{x \in X} f(x) = \sum_{i : C_i(x) = 1} w_i$$

Esempio

- Variabili
$$V = \{x_1, x_2, x_3, x_4\}$$
- Letterali
$$L = \{x_1, \bar{x}_1, x_2, \bar{x}_2, x_3, \bar{x}_3, x_4, \bar{x}_4\}$$
- Formule logiche
$$C_1 = \bar{x}_1 \vee x_2 \qquad \ldots \qquad C_7 = x_2$$
- Forma congiuntiva normale
$$CNF = (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_4) \wedge x_1 \wedge x_2$$
- Peso uniforme:
$$w_i = 1 \qquad i = 1, \ldots, 7$$

$x = \{(x_1, 0), (x_2, 0), (x_3, 1), (x_4, 1)\}$ soddisfa 5 formule su 7: $f(x) = 5$

Cambiando valore a una variabile, $f(x)$ può cambiare o no

(es., complementare x_1 aumenta $f(x)$ di 1, x_4 la lascia invariata)

Problemi su matrici numeriche: Set Covering (*SCP*)

Data una matrice binaria e un vettore di costi associati alle colonne, si cerca il sottoinsieme di colonne di costo minimo che copra tutte le righe

- una matrice binaria $A \in \mathbb{B}^{m,n}$ costituita da un insieme di righe R e un insieme di colonne C
- la colonna $j \in C$ copre la riga $i \in R$ significa che $a_{ij} = 1$
- una funzione $c : C \to \mathbb{N}$ che indica il costo di ogni colonna

L'insieme base è l'insieme delle colonne: $B = C$

La regione ammissibile contiene i sottoinsiemi di colonne che coprono tutte le righe

$$X = \left\{ x \subseteq B : \sum_{j \in x} a_{ij} \geq 1 \; \forall i \in R \right\}$$

L'obiettivo è minimizzare il costo totale delle colonne scelte

$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

Esempio

c	4	6	10	14	5	6

A						
	0	1	1	1	1	0
	0	0	1	1	0	0
	1	1	0	0	0	1
	0	0	0	1	1	1
	1	1	1	0	1	0

A							
	0	1	1	1	1	0	2
	0	0	1	1	0	0	1
	1	1	0	0	0	1	1
	0	0	0	1	1	1	1
	1	1	1	0	1	0	3

$$x' = \{c_1, c_3, c_5\} \in X$$

$$f(x') = 19$$

A							
	0	1	1	1	1	0	1
	0	0	1	1	0	0	0
	1	1	0	0	0	1	2
	0	0	0	1	1	1	2
	1	1	1	0	1	0	2

$$x'' = \{c_1, c_5, c_6\} \notin X$$

$$f(x'') = 15$$

"Set Covering": coprire un insieme (righe) con sottoinsiemi (colonne)

Intermezzo 3: il test di ammissibilità

In un algoritmo euristico, può capitare di dover risolvere il problema:

Dato un sottoinsieme x, si indichi se è ammissibile o no: $x \in X$?

È un problema di decisione

Il test di ammissibilità può richiedere di

- calcolare e verificare un singolo valore
  (il volume totale nel *KP*, la cardinalità totale nel *MDP*)
- scorrere un insieme di attributi (uno e un solo valore per variabile nel *Max-SAT*, una e una sola macchina per lavorazione nel *PMSP*)
- calcolare e verificare un insieme di valori
  (il volume di ciascun contenitore nel *BPP*)

Il tempo richiesto può cambiare secondo che il test sia eseguito

- su un sottoinsieme generico x
- su un sottoinsieme x' ottenuto applicando una modifica δ a una soluzione ammissibile x

Talvolta si può prefiltrare le modifiche lecite (*MDP*, *PMSP*, *Max-SAT*), in altri casi si può solo valutarle a posteriori (*KP*, *BPP*, *SCP*)

Problemi su matrici numeriche: Set Packing

Data una matrice binaria e un vettore di valori associati alle colonne, si cerca il sottoinsieme di colonne di valore massimo che non confliggano

- una matrice binaria $A \in \mathbb{B}^{m,n}$ costituita da un insieme di righe R e un insieme di colonne C
- le colonne j' e $j'' \in C$ confliggono significa che $a_{ij'} = a_{ij''} = 1$
- una funzione $\phi : C \to \mathbb{N}$ che indica il valore di ogni colonna

L'insieme base è l'insieme delle colonne: $B = C$

La regione ammissibile contiene i sottoinsiemi di colonne che non confliggono

$$X = \left\{ x \subseteq B : \sum_{j \in x} a_{ij} \leq 1 \; \forall i \in R \right\}$$

L'obiettivo è massimizzare il valore totale delle colonne scelte

$$\max_{x \in X} f(x) = \sum_{j \in x} \phi_j$$

Esempio

ϕ	4	6	10	14	5	6

A						
	0	1	0	0	1	0
	0	0	1	1	0	0
	1	0	0	0	0	1
	0	0	0	1	1	1
	1	1	1	0	0	0

A							
	0	1	0	0	1	0	1
	0	0	1	1	0	0	1
	1	0	0	0	0	1	0
	0	0	0	1	1	1	1
	1	1	1	0	0	0	1

$$x' = \{c_2, c_4\} \in X$$

$$f(x') = 20$$

A							
	0	1	0	0	1	0	2
	0	0	1	1	0	0	0
	1	0	0	0	0	1	1
	0	0	0	1	1	1	2
	1	1	1	0	0	0	1

$$x'' = \{c_1, c_5, c_6\} \notin X$$

$$f(x'') = 15$$

"Set Packing": impaccare sottoinsiemi (colonne) in un insieme (righe)

Problemi su matrici numeriche: Set Partitioning (*SPP*)

Data una matrice binaria e un vettore di costi associati alle colonne, si cerca il sottoinsieme di colonne di costo minimo che copra tutte le righe senza conflitti

- una matrice binaria $A \in \mathbb{B}^{m,n}$ costituita da un insieme di righe R e un insieme di colonne C
- una funzione $c : C \to \mathbb{N}$ che indica il costo di ogni colonna

L'insieme base è l'insieme delle colonne: $B = C$

La regione ammissibile contiene i sottoinsiemi di colonne che coprono tutte le righe e non confliggono

$$X = \left\{ x \subseteq C : \sum_{j \in x} a_{ij} = 1 \; \forall i \in R \right\}$$

L'obiettivo è minimizzare il costo totale delle colonne scelte

$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

Esempio

c	4	6	10	14	5	6

A						
	0	1	0	0	1	0
	0	0	1	1	0	0
	1	0	0	0	0	1
	0	0	0	1	1	0
	1	1	1	0	0	0

A							
	0	1	0	0	1	0	1
	0	0	1	1	0	0	1
	1	0	0	0	0	1	1
	0	0	0	1	1	0	1
	1	1	1	0	0	0	1

$$x' = \{c_2, c_4, c_6\} \in X$$

$$f(x') = 20$$

A							
	0	1	0	0	1	0	2
	0	0	1	1	0	0	0
	1	0	0	0	0	1	1
	0	0	0	1	1	0	1
	1	1	1	0	0	0	1

$$x'' = \{c_1, c_5, c_6\} \notin X$$

$$f(x'') = 15$$

"Set Partitioning": ripartire un insieme (righe) in sottoinsiemi (colonne)

Intermezzo 4: la ricerca di soluzioni ammissibili

In un algoritmo euristico, può capitare di dover risolvere il problema

Trovare una soluzione ammissibile $x \in X$

È un problema di ricerca

Spesso la soluzione è banale

- nel *KP*, $x = \emptyset$ è ammissibile
- nel *MDP*, qualsiasi sottoinsieme di k elementi è ammissibile
- nel *SCP*, $x = B$ è ammissibile (oppure non ci sono soluzioni)
- in altri problemi, basta rispettare semplici vincoli di coerenza

Per alcuni problemi la ricerca d'una soluzione ammissibile è difficile

- nel *BPP* occorre che ci siano abbastanza contenitori (spesso si ipotizza che ce ne sia uno per ogni oggetto)
- nel *SPP* non ci sono algoritmi polinomiali per rispondere

Si può allargare la regione ammissibile da X a $X' \supseteq X$ (rilassamento)

- occorre estendere l'obiettivo f da X a X' (*l'avevamo detto*)
- però spesso le soluzioni di $X' \setminus X$ sono migliori (*. . . e allora?*)

Problemi su grafo: Vertex Cover (VCP)

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

- un grafo non orientato $G = (V, E)$

L'insieme base è l'insieme dei vertici: $B = V$

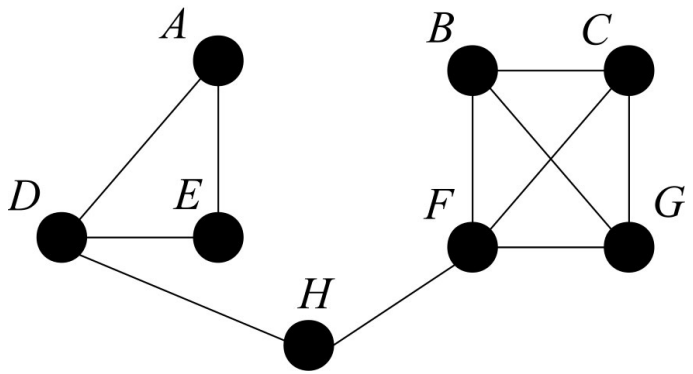
La regione ammissibile contiene i sottoinsiemi di vertici tali che tutti i lati del grafo vi incidono

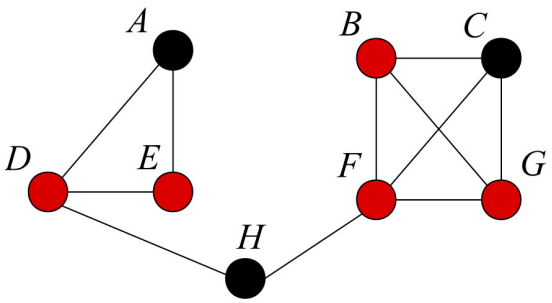
$$X = \left\{ x \subseteq B : x \cap (i,j) \neq \emptyset \; \forall \, (i,j) \in E \right\}$$

L'obiettivo è minimizzare il numero di vertici scelti

$$\min_{x \in X} f(x) = |x|$$

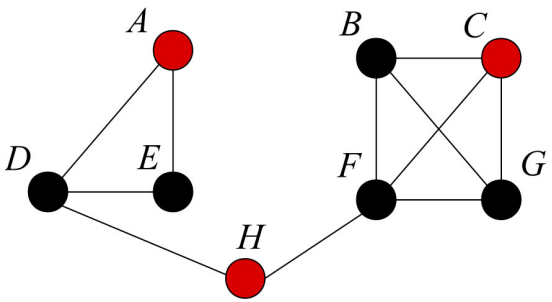
Esempio





$$x' = \{B, D, E, F, G\} \in X$$

$$f(x') = 5$$



$$x'' = \{A, C, H\} \notin X$$

$$f(x'') = 3$$

Problemi su grafo: 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

- un grafo non orientato $G = (V, E)$
- una funzione $w : V \to \mathbb{N}$ che indica il peso di ogni vertice

L'insieme base è l'insieme dei vertici: $B = V$

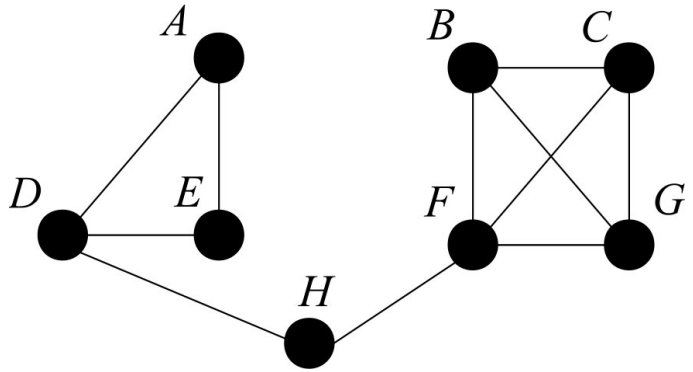
La regione ammissibile contiene i sottoinsiemi di vertici reciprocamente adiacenti

$$X = \left\{ x \subseteq B : (i,j) \in E \; \forall i \in x, \forall j \in x \setminus \{i\} \right\}$$

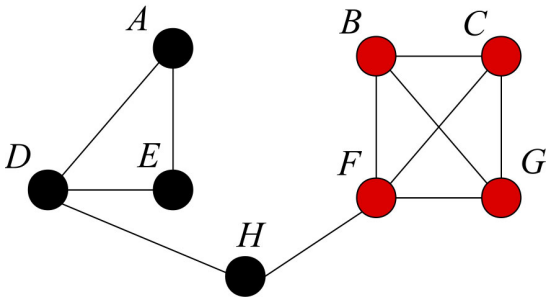
L'obiettivo è massimizzare il peso dei vertici scelti

$$\max_{x \in X} f(x) = \sum_{j \in x} w_j$$

Esempio

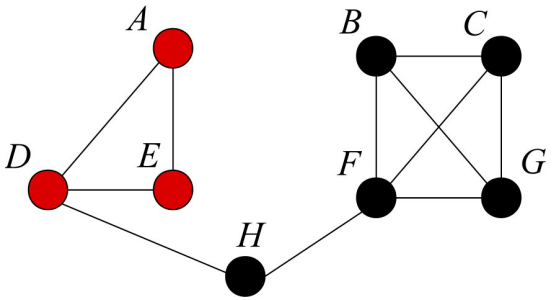


Peso uniforme: $w_i = 1$ per ogni $i \in V$



$$x' = \{B, C, F, G\} \in X$$

$$f(x') = 4$$



$$x'' = \{A, D, E\} \in X$$

$$f(x'') = 3$$

Problemi su grafo: Maximum Independent Set Problem

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

- un grafo non orientato $G = (V, E)$
- una funzione $w : V \to \mathbb{N}$ che indica il peso di ogni vertice

L'insieme base è l'insieme dei vertici: $B = V$

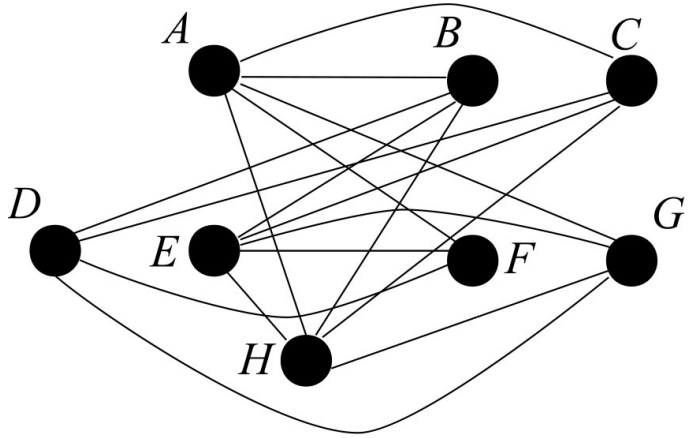
La regione ammissibile contiene i sottoinsiemi di vertici reciprocamente non adiacenti

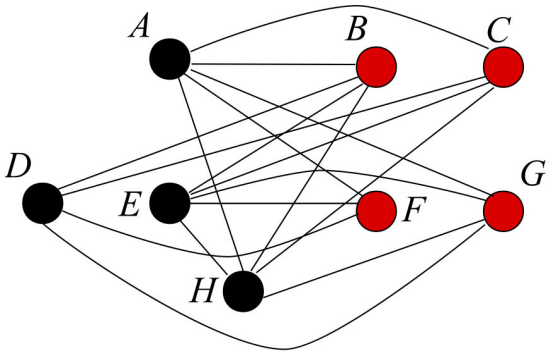
$$X = \left\{ x \subseteq B : (i,j) \notin E \; \forall i \in x, \forall j \in x \setminus \{i\} \right\}$$

L'obiettivo è massimizzare il peso dei vertici scelti

$$\max_{x \in X} f(x) = \sum_{j \in x} w_j$$

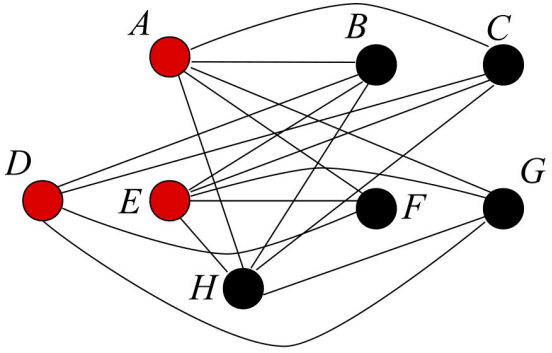
Esempio





$$x' = \{B, C, F, G\} \in X$$

$$f(x') = 4$$



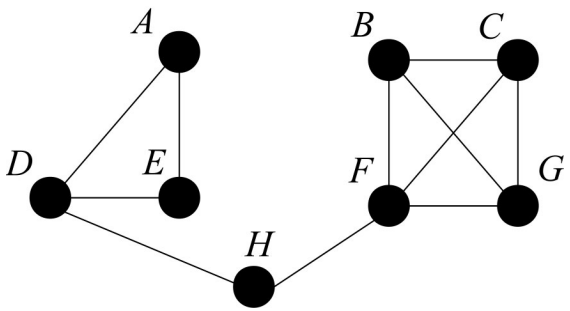
$$x'' = \{A, D, E\} \in X$$

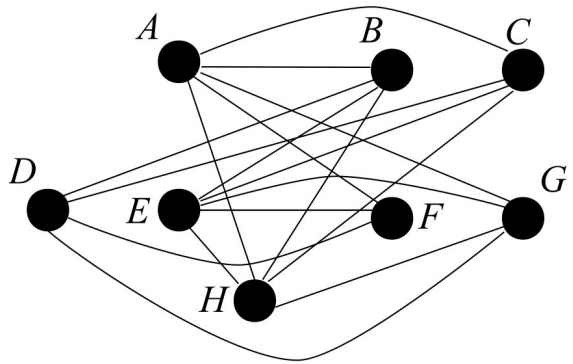
$$f(x'') = 3$$

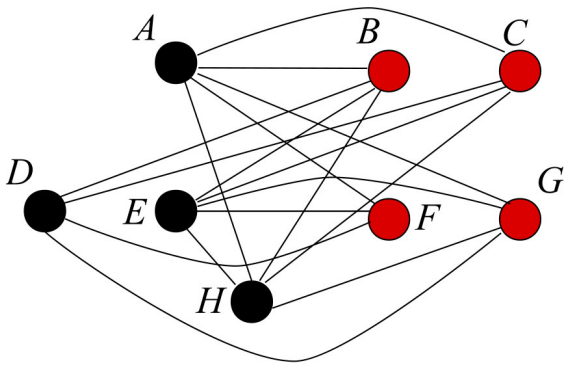
Intermezzo 5: le relazioni fra problemi (1)

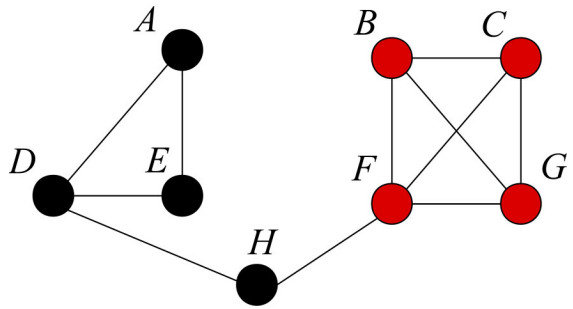
Ogni istanza del *MCP* è equivalente a un'istanza del *MISP*

1. dato il grafo $G = (V, E)$
2. si costruisce il grafo complementare $\bar{G} = (V, (V \times V) \setminus E)$
3. si trova una soluzione ottima del *MISP* su $\bar{G}$
4. i vertici corrispondenti danno una soluzione ottima del *MCP* su G







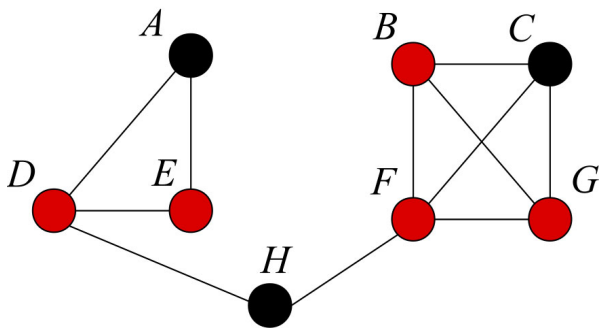


Si può fare anche il percorso esattamente inverso

Intermezzo 5: le relazioni fra problemi (2)

Il *VCP* e il *SCP* sono anche loro legati, ma in modo diverso:
ogni istanza del *VCP* si può tradurre in un'istanza di *SCP*:

- ogni lato i corrisponde a una riga della matrice di copertura A
- ogni vertice j corrisponde a una colonna di A
- se il lato i incide nel vertice j, si pone $a_{ij} = 1$; altrimenti $a_{ij} = 0$
- la soluzione ottima del *SCP* dà la soluzione ottima del *VCP*



	A	B	C	D	E	F	G	H
(A, D)	1	0	0	1	0	0	0	0
(A, E)	1	0	0	0	1	0	0	0
(B, C)	0	1	1	0	0	0	0	0
(B, F)	0	1	0	0	0	1	0	0
(B, G)	0	1	0	0	0	0	1	0
(C, F)	0	0	1	0	0	1	0	0
(C, G)	0	0	1	0	0	0	1	0
(D, E)	0	0	0	1	1	0	0	0
(D, H)	0	0	0	1	0	0	0	1
(F, G)	0	0	0	0	0	1	1	0
(F, H)	0	0	0	0	0	1	0	1

Però non è possibile fare l'inverso

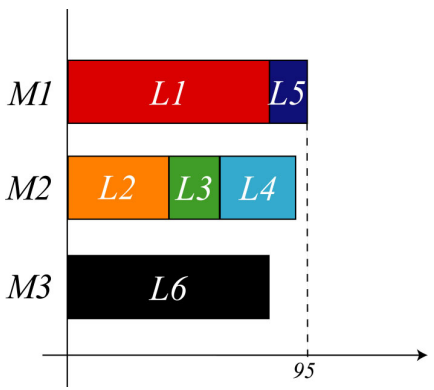
Intermezzo 5: le relazioni fra problemi (3)

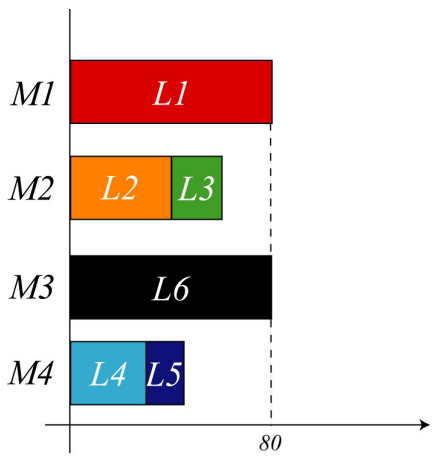
Il *BPP* e il *PMSP* sono equivalenti, ma in modo più sofisticato:

- le lavorazioni corrispondono agli oggetti
- le macchine corrispondono ai contenitori, ma
  - nel *BPP* è data la capacità, e si minimizza il numero di contenitori
  - nel *PMSP* è dato il numero di macchine, e si minimizza il tempo di completamento

Per minimizzare il numero di contenitori del *BPP* con una data capacità

1. si fa un'ipotesi sul valore ottimo (ad es., 3)
2. si costruisce il *PMSP* corrispondente
3. si valuta il tempo di completamento ottimo (ad es., 95)
   - se eccede la capacità (ad es., 80), si aumenta l'ipotesi (4 o 5)
   - se non la eccede, si prova a ridurre l'ipotesi (2 o 1)





Si può fare anche l'inverso

I due problemi sono equivalenti, ma ciascuno va risolto più volte

Problemi su grafo: Travelling Salesman Problem

Dato un grafo orientato e una funzione costo definita sugli archi, si cerca il ciclo di costo minimo che ricopra tutti i nodi del grafo

- un grafo orientato $G = (N, A)$
- una funzione $c : A \to \mathbb{N}$ che indica il costo di ogni arco

L'insieme base è l'insieme degli archi: $B = A$

La regione ammissibile contiene i cicli che coprono tutti i nodi del grafo (cicli hamiltoniani)

Come si fa a sapere che un sottoinsieme è soluzione ammissibile?

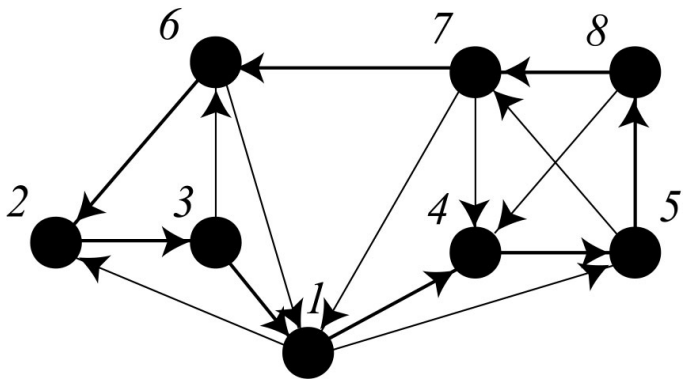
E nel caso di modifiche di una soluzione ammissibile?

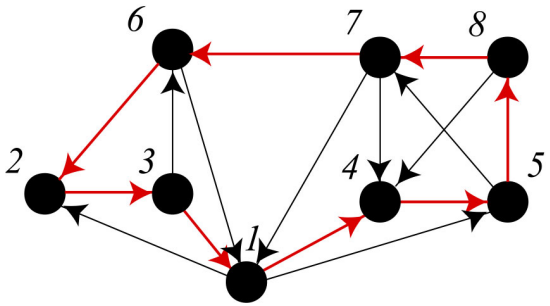
È complicato trovare una soluzione ammissibile?
(difficile in genere, banale su grafo completo)

L'obiettivo è minimizzare il costo degli archi scelti

$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

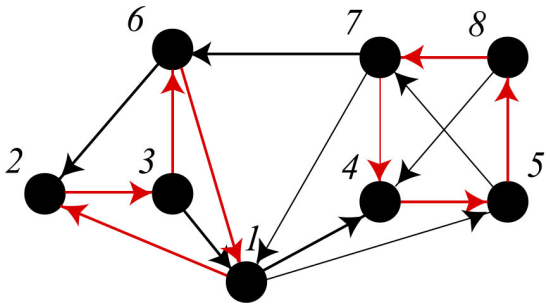
Esempio





$$x' = \{(1,4),(4,5),(5,8),(8,7),(7,6),(6,2),(2,3),(3,1)\} \in X$$

$$f(x') = 102$$



$$x'' = \{(4,5),(5,8),(8,7),(7,4),(1,2),(2,3),(3,6),(6,1)\} \notin X$$

$$f(x'') = 106$$

Problemi su grafo: Capacitated Spanning Tree Problem

Dato un grafo non orientato, un vertice radice, una funzione costo definita sui lati, una funzione peso definita sui vertici e una capacità massima, si cerca l'albero ricoprente di costo minimo tale che ogni sottoalbero appeso alla radice abbia peso non superiore alla capacità

- un grafo non orientato $G = (V, E)$ con un vertice radice $r \in V$
- una funzione $c : E \to \mathbb{N}$ che indica il costo di ogni lato
- una funzione $w : V \to \mathbb{N}$ che indica il peso di ogni vertice
- un numero $W \in \mathbb{N}$ che indica la capacità di ogni sottoalbero

L'insieme base è l'insieme dei lati: $B = E$

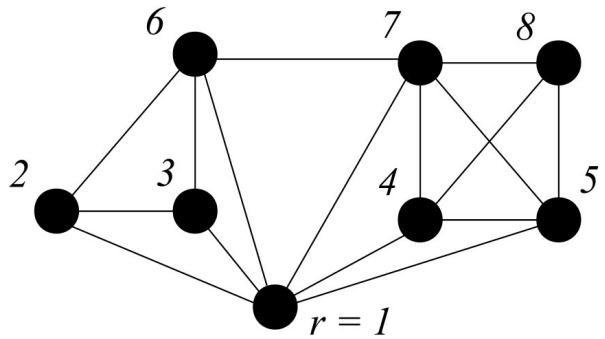
La regione ammissibile contiene gli alberi ricoprenti tali che il peso dei vertici ricoperti da ogni sottoalbero appeso alla radice non superi W

Il test di ammissibilità richiede la visita di un grafo

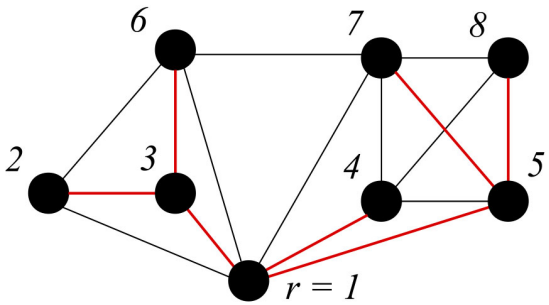
L'obiettivo è minimizzare il costo dei lati scelti

$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

Esempio

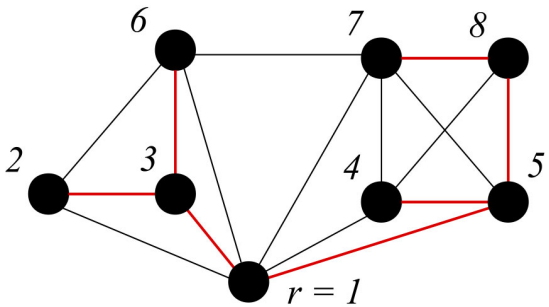


Peso uniforme ($w_i = 1$ per ogni $i \in V$) e capacità: $W = 3$



$$x' = \{(r,3),(3,2),(3,6),(r,4),(r,5),(5,7),(5,8)\} \in X$$

$$f(x') = 95$$



$$x'' = \{(r,3),(3,2),(3,6),(r,5),(5,4),(5,8),(8,7)\} \notin X$$

$$f(x'') = 87$$

Costo delle operazioni principali

Funzione obiettivo

- valutazione semplice: sommare i costi dei lati
- aggiornamento veloce: sommare i costi nuovi e sottrarre i vecchi

Può generare sottoalberi non ottimi per i sottoinsiemi di vertici

Ammissibilità

- valutazione meno semplice:
  - cercare eventuali cicli
  - visitare ogni sottoalbero per calcolarne il peso
- aggiornamento lento:
  - vedere se ogni arco tolto rompe un ciclo creato da un arco aggiunto
  - rivalutare i pesi dei sottoalberi

E se si ragionasse in termini di sottoinsiemi di vertici?

Una descrizione alternativa

Occorre aggiungere un insieme di sottoalberi T

Quanti? Ne bastano $|V| - 1$, *alcuni anche vuoti*

L'insieme base è l'insieme delle coppie (vertice,sottoalbero): $B = V \times T$

La regione ammissibile contiene le partizioni dei vertici in sottoinsiemi di peso $\leq W$ (*come nel BPP*) connessi (*visita, ovvia su grafo completo*)

$$X = \left\{ x \subseteq B : |x \cap B_v| = 1 \; \forall v \in V \setminus \{r\}, \sum_{(i,j) \in B^t} w_i \leq W \; \forall t \in T, \ldots \right\}$$

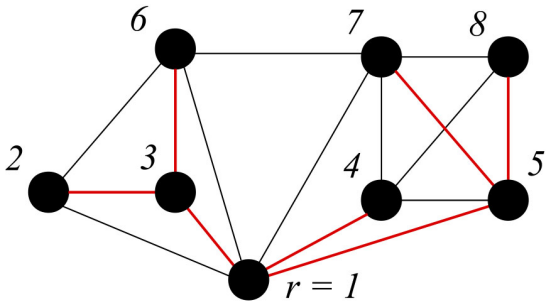
con $B_v = \{(i,j) \in B : i = v\}$, $B^t = \{(i,j) \in B : j = t\}$

L'obiettivo è minimizzare la somma dei costi degli alberi che ricoprono ciascun sottoinsieme di vertici più i lati che li collegano alla radice

Sono tanti problemi di albero ricoprente minimo indipendenti

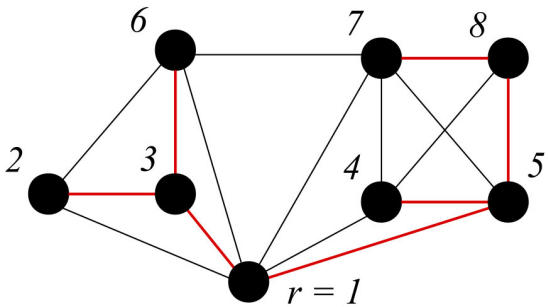
Esempio

Le soluzioni viste in precedenza hanno ora una rappresentazione diversa



$$x' = \{(2,1),(3,1),(6,1),(4,2),(5,3),(7,3),(8,3)\} \in X$$

$$f(x') = 95$$



$$x'' = \{(2,1),(3,1),(6,1),(4,2),(5,2),(7,2),(8,2)\} \notin X$$

$$f(x'') = 87$$

Costo delle operazioni principali

Funzione obiettivo

- valutazione lenta: calcolare diversi alberi ricoprenti minimi
- aggiornamento lento: ricalcolare gli alberi ricoprenti

I sottoalberi generati sono certamente ottimi

Ammissibilità

- valutazione semplice: sommare i pesi dei vertici
- aggiornamento veloce: sommare i pesi nuovi e sottrarre i vecchi

Obiettivo e ammissibilità si scambiano vantaggi e svantaggi

Problemi su grafo: Vehicle Routing Problem

Dato un grafo orientato, un nodo deposito, una funzione costo definita sugli archi una funzione peso definita sui nodi e una capacità, si cerca l'insieme di cicli di costo minimo che passano per il deposito e hanno ciascuno peso totale non superiore alla capacità

- un grafo orientato $G = (N, A)$ con un nodo deposito $d \in N$
- una funzione $c : A \to \mathbb{N}$ che indica il costo di ogni arco
- una funzione $w : N \to \mathbb{N}$ che indica il peso di ogni nodo
- un numero $W \in \mathbb{N}$ che indica la capacità di ogni ciclo

L'insieme base potrebbe essere

- l'insieme degli archi: $B = A$
- l'insieme delle coppie (nodo,ciclo): $B = N \times C$

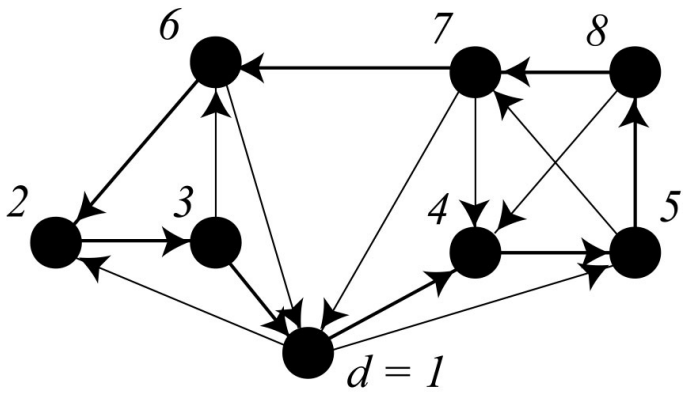
La regione ammissibile potrebbe contenere

- gli insiemi di archi che coprono tutti i nodi con cicli passanti per il deposito e di peso non superiore a W (*ancora la visita di un grafo*)
- le partizioni dei nodi in sottoinsiemi di peso ciascuno non superiore a W e copribile con un ciclo (*problema difficile!*)

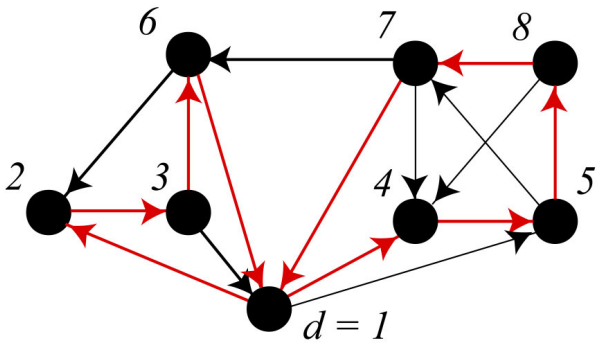
L'obiettivo è minimizzare il costo degli archi scelti

$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

Esempio



Le soluzioni potrebbero essere descritte come



- insiemi di archi
  $x = \{(d,2),(2,3),(3,6),(6,d),(d,4),$
  $(4,5),(5,8),(8,7),(7,d)\} \in X$
- partizioni di nodi
  $x = \{(2,1),(3,1),(6,1),$
  $(4,2),(5,2),(7,2),(8,2)\} \in X$

Ad ogni modo, $f(x) = 133$

Intermezzo 6: combinare descrizioni alternative

Il *CMSTP* e il *VRP* condividono un'interessante complicazione:
diverse definizioni dell'insieme base sono possibili e naturali

- la descrizione dell'insieme base B come insieme di lati/archi
- la descrizione dell'insieme base B come insieme di coppie (vertice,albero)/(nodo/ciclo)

La prima definizione si presta meglio a manipolare l'obiettivo,
la seconda a trattare l'ammissibilità e a ottenere soluzioni migliori
(ne parleremo)

Quale definizione adottare?

- quella che rende facili le operazioni più frequenti nell'algoritmo
- entrambe, caricandosi dell'obbligo di tenerle aggiornate e coerenti, se l'aggiornamento è molto meno frequente della consultazione