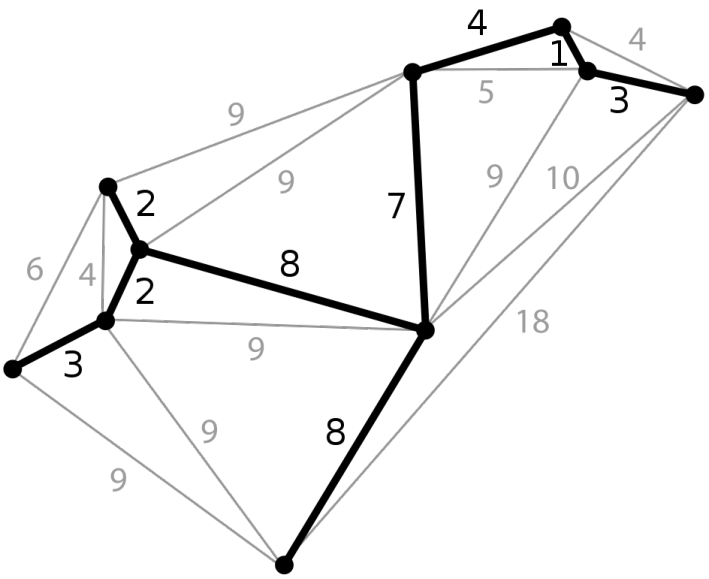


METODI ED ALGORITMI DI OTTIMIZZAZIONE PER IL PROBLEM SOLVING

Docente:
Aristide Mingozzi

Adattamento:
Edoardo Rosa

Anno Accademico 20014-2015

INDICE

ELENCO DELLE FIGURE

Copertina: `http://commons.wikimedia.org/wiki/File:Minimum_spanning_tree.svg`

ELENCO DELLE TABELLE

CAPITOLO 1

MODELLI E FORMULAZIONI MATEMATICHE

1.1 The Traveling Salesman Problem

Il Traveling Salesman Problem (TSP) è il problema più noto dell'ottimizzazione combinatoria. Siano date n città e i costi c_{ij} per andare dalla città i alla città j. Si vuole determinare un cammino che parte da una città (diciamo i_1), visitare una ed una sola volta tutte le rimanenti città e terminare nella città di partenza i_1. Inoltre si vuole che il costo di tale cammino sia minimo.
Ha molteplici applicazioni pratiche e teoriche perche è la struttura di molti problemi pratici.
Si è soliti modella il TSP come segue:

- è dato un grafo orientato (o non orientato) $G = (N, A)$ dove N è un insieme di n vertici e A è un insieme di m archi.

  Ad ogni arco $(i,j) \in A$ è associato un costo c_{ij}.

  Un circuito hamiltoniano di G è un circuito che passa per ogni vertice una ed una sola volta. Il costo di un circuito hamiltoniano di G è pari alla somma dei costi degli archi che compongono il circuito;

- il problema del TSP è di trovare un grafo G, con una data matrice dei costi $[c_{ij}]$, un circuito hamiltoniano di costo minimo.

1.1.1 Formulazioni Matematiche del TSP

In letteratura esistono molteplici (e a volte fantasiose) formulazioni del TSP.
Presentiamo le due formulazioni più note e su cui si basano i metodi esatti più efficienti.

1.1.1.1 TSP asimmetrico

I costi c_{ij} non verificano $c_{ij} = c_{ji}\ \forall\ i, j$ con $i < j$.
Sia x_{ij} una variabile $(0-1)$ associata ad ogni arco $(i,j) \in A$ dove $x_{ij} = 1$ se l'arco (i,j) è nella soluzione ottima e $x_{ij} = 0$ altrimenti.

$$Min \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{1.1}$$

$$s.t. \sum_{i \in N} x_{ij} = 1, \ \ \forall j \in N \tag{1.2}$$

$$\sum_{j \in N} x_{ij} = 1, \ \ \forall i \in N \tag{1.3}$$

$$\sum_{i \in S} \sum_{j \in N \setminus S} x_{ij} \geq 1, \ \ \forall S \subset N \tag{1.4}$$

$$x_{ij} \in \{0,1\}\ ,\ \forall (i,j) \in A \tag{1.5}$$

Il vincolo 1.4 impone che ogni soluzione ammissibile debba contenere almeno un arco (i,j) con $i \in S$ e $j \in N \setminus S$ per ogni sottoinsieme S di N. Un'alternativa al vincolo 1.4 è:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \ \ \forall S \subset N \tag{1.4'}$$

1.1.1.2 TSP simmetrico

Sia dato un grafo non-orientato $G = (N, A)$ con $c_{ij} = c_{ji}\ , \forall i, j \in N$.
Gli archi di A sono numerati da 1 a m. L'arco di indice l corrisponde a (α_l, β_l) con $\alpha_l < \beta_l$.
A_i è il sottoinsieme degli indici degli archi che incidono sul vertice i:

$$A_i = \{l :\ l = 1, m \ \ s.t.\ \alpha_l = i\ or\ \beta_l = i\}$$

Per una dato $S \in N$ e $\bar{S} = N \setminus S$ indichiamo con $(S, \bar{S})$ il sottoinsieme degli indici degli archi per cui $\alpha_l \in S$ e $\beta_l \in \bar{S}$ oppure $\alpha_l \in \bar{S}$ e $\beta_l \in S$.

Ad ogni arco di incide l è associato un costo $d_l = c_{\alpha_l \beta_l}$ e $x_l \in \{0,1\}$ è una variabile che vale 1 se e solo se l'arco di indice l è nella soluzione ottima.

$$Min \sum_{l=1} d_l \, x_l \tag{1.6}$$

$$s.t. \sum_{l \in A_i} x_l = 2, \ \forall i \in N \tag{1.7}$$

$$\sum_{l \in (S,\bar{S})} x_l \geq 1, \ \forall S \subset N \tag{1.8}$$

$$x_l \in \{0,1\}, \ \ l = 1, \ldots, m \tag{1.9}$$

1.1.2 Eliminazione subtours di Miller, Tucker, Zemlin (1960)

Sia u_i una variabile intera il cui valore sappresenta la posizione che il vertice i occupa nel tour.

Es. tour (1,4,5,3,2,1) per TSP con n=5 vertici, si ha $u_1 = 1, \ u_2 = 5, \ u_3 = 4, \ u_4 = 2, \ u_5 = 3$

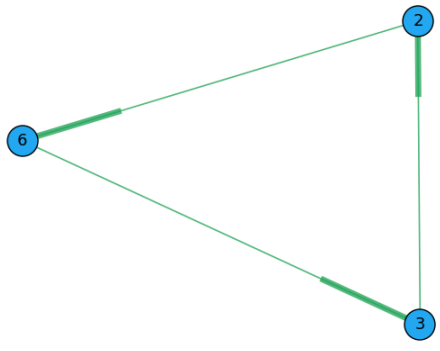
Miller, Tucker e Zemlin propongono in alternativa a:

$$\sum_{i \in S} \sum_{j \in N \setminus S} x_{ij} \geq 1, \ \ \forall S \subset N \tag{*}$$

hanno imposto i seguenti vincoli:

$$u_i - u_j + n x_{ij} \leq n - 1, \ \ i = 1, \ldots, n \, , \ j = 2, \ldots, n \tag{1.10}$$

Ogni tour hamiltoniano soddisfa questi vincoli e ogni subtour li viola.



$$u_2 - u_6 + n \ \cdot x_{2,6} \leq n - 1$$
$$u_6 - u_3 + n \ \cdot x_{6,3} \leq n - 1$$
$$u_3 - u_2 + n \ \cdot x_{3,2} \leq n - 1$$
$$\downarrow$$
$$3n \leq 3(n-1)$$

Figura 1.1: Grafo orientato

1.1.3 Il Traveling salesman problem con time windows (TSPTW)

È una variante del TSP che ha molte applicazioni.

Sia dato un grafo orientato $G = (V, A)$ di $n + 1$ vertici ($V = \{0, 1, \ldots, n\}$).

Ad ogni arco $(i, j) \in A$ sono associati

- un costo $c_{ij} \geq 0$
- un tempo di percorrenza $\theta_{ij} \geq 0$

Ad ogni vertice è associato un intervallo $[r_i, d_i]$ chiamato "time window" che rappresenta l'orario in cui il vertice i può essere vistato dal "salesman".

Ovvero il salesman può visitare i ad ogni tempo $t \in \mathbb{Z}^+$ con $r_i \leq t \leq d_i$.
Il problema consiste nel trovare una sequenza dei vertici di G che parte dal vertice 0 al tempo 0 e finisce al nodo 0 tale che sia il minimo il costo del circuito e il tempo di arrivo al nodo i sia nell'intervallo $[r_i, d_j]$, $\forall i \in V$.
Si consideri la sequenza $(0, i, .., i_{k-1}, i_k, \ldots, i_n, 0)$ e sia t_{i_k} il tempo di arrivo al vertice i_k, $k = 0, 1, \ldots, n + 1$.

I tempi di arrivo sono calcolati come:

$$t_0 = 0 \tag{1.11}$$

$$t_{i_k} = \max\{t_{i_{k-1}} + \theta_{i_{k-1}} \cdot i_k,\ r_{i_k}\} \tag{1.12}$$

1.1.3.1 Formulazione del TSPTW

Sia x_{ij} una variabile binaria intera che assume il valore 1 se il vertice i è visitato immediatamente prima di i e $x_{ij} = 0$ altrimenti.

$$Min \quad \sum_{(i,j)\in A} c_{ik} x_{ij} \tag{1.13}$$

$$s.t. \quad \sum_{i\in A_j^-} x_{ij} = 1, \ \ \forall j \in V \tag{1.14}$$

$$\sum_{j\in A_i^+} x_{ij} = 1, \ \ \forall i \in V \tag{1.15}$$

$$t_i + \theta_{ij} - t_j \leq M(1 - x_{ij}, \ \ \forall (i, j) \in A, \ j \neq 0) \tag{1.16}$$

$$t_i \leq d_i, \ \ \forall i \in V \tag{1.17}$$

$$t_i \geq r_i, \ \ \forall i \in V \tag{1.18}$$

$$x_{ij} \in \{0, 1\}, \ \ \forall \in A \tag{1.19}$$

$$t_i \in \mathbb{N}^+, \ \ \forall i \in V \tag{1.20}$$

dove

$$A_i^+ = \{j \in V : (i,j) \in A\}$$
$$A_i^- = \{j \in V : (i,j) \in A\}$$
$$M \ \textit{un intero grande a piacere}$$
$$r_0 = d_0 = 0$$

1.2 Project scheduling with resource constraints (PSR)

È dato un insieme $\mathbb{X} = \{1, \ldots, n\}$ di n jobs.

Sono disponibili m risorse dove ogni risorsa k ha una disponibilità b_k ad ogni istante del periodo di scheduling.

Ogni job i ha un tempo di processo d_i e la sua esecuzione, una volta iniziata, non può essere interrotta.

Il job i per essere eseguito richiede b_{ik} unità della risorsa k per ciascun intervallo di tempo in cui rimane in esecuzione.
È dato un grafo $G = (X, H)$ di precedenze, dove ogni arco $(i, j) \in H$ impone che il job j può iniziare solo dopo che il job i è stato completato.

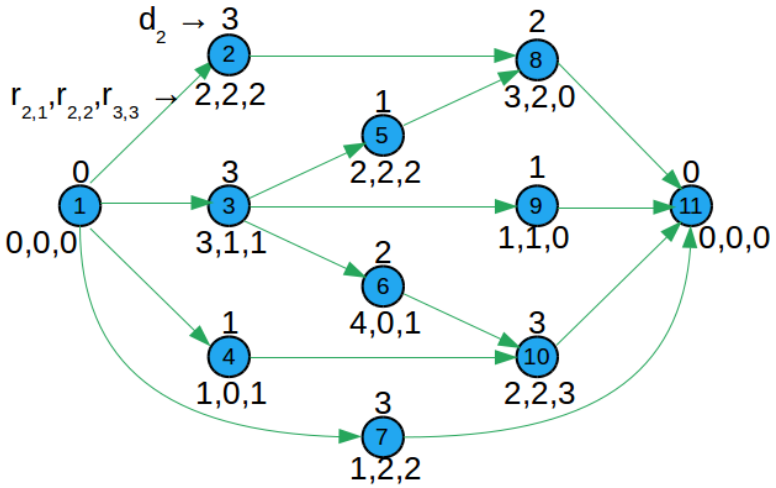
- Si vuole determinare il tempo di inizio di processo di ogni job in modo che siano soddisfatti i vincoli di precedenza, i vincoli sulle risorse e sia minima la durata complessiva del progetto

1.2.1 Esempio di PSR

Siano dati $n = 11$ jobs e $m = 3$ risorse con $b_1 = b_2 = b_3 = 4$ e un grafo H delle precedenze corrispondenti agli archi della figura 1.2.

Si osservi che i jobs 2 e 3 non possono essere eseguiti in parallelo poiché $r_{2,1} + r_{3,1} = 5 > b_1$!

Figura 1.2: Grafo H delle precedenze



1.2.2 Formulazione del PSR

Sia ξ_{it} una variabile binaria 0-1 che vale 1 se e solo se il job i viene messo in esecuzione al tempo t.

Sia T_{max} un upper bound sulla durata del progetto.

$$Min \sum_{t=1}^{T_{max}} t\, \xi_{nt} \tag{1.21}$$

$$s.t. \sum_{t=1}^{T_{max}} t\, \xi_{it} = 1, \quad i = 1, \ldots, n \tag{1.22}$$

$$\sum_{t=1}^{T_{max}} t\, \xi_{jt} - \sum_{t=1}^{T_{max}} t\, \xi_{it} \geq d_i, \quad \forall (i,j) \in H \tag{1.23}$$

$$\sum_{i=1}^{n} r_{ik} \sum_{\tau=t-d_i+1}^{t} \xi_{i\tau} \leq b_k, \quad t = 1, \ldots, T_{max} \; e \; k = 1, \ldots, m \tag{1.24}$$

$$\xi_{it} \in \{0,1\}, \quad i = 1, \ldots, n \; e \; t = 1, \ldots, T_{max} \tag{1.25}$$

Si osservi che:

$$\sum_{\tau=t-d_i+1}^{t} \xi_{i\tau} = 1 \quad \textit{se il job i in esecuzione al tempo t}$$

1.2.2.1 Esempio

Sia $d_i = 4$.

Se $\xi_{i3} = 1$, allora i è in esecuzione nei tempi 3,4,5 e 6. Infatti avremo:
$\sum_{\tau=t-d_i+1}^{t} \xi_{i\tau} = 1$ per $t = 3,4,5,6$ e $\sum_{\tau=t-d_i+1}^{t} \xi_{i\tau} = 0$ per $t < 3$ e $t > 6$

1.3 Fixed Charge Transportation Problem (FCTP)

Il Problema del Trasporto di Carico Fisso è una generalizzazione del classico Problema del Trasporto.

Si differenzia nel definire che il costo per la spedizione di una quantità non-zero di beni, da ogni origine alla sua destinazione, è composto da un costo proporzionale all'ammontare dei beni inviati più un costo fisso.

1.3.1 Descrizione del FCTP

Il FCTP è definito su un grafo completo e bipartito $G = (S, T, A)$ dove $S = 1, 2, \ldots, m$ è un insieme di m sorgenti e $T = 1, 2, \ldots, n$ è un insieme di n destinazioni.

Per ogni sorgente $i \in S$ è disponibile è una quantità intera $a_i > 0$ di merce e per ogni destinazione $j \in T$ è necessaria una quantità intera $b_j > 0$ di merce dalle sorgenti.
L'insieme A degli archi è definito come: $A = \{(i, j) : \ i \in S, \ j \in T\}$; ogni arco $(i, j) \in A$ è associato ad un costo unitario c_{ij} per il trasporto di una unità della merce dalla sorgente i alla destinazione j più un costo fisso f_{ij} for usare l'arco (i, j).

Senza perdere di generalità si assume che:

$$\sum_{i \in S} a_i = \sum_{j \in T} b_j$$

1.3.2 Formulazione del FCTP

Sia x_{ij} una variabile rappresentante la quantità di merce trasportata dalla sorgente i alla destinazione j e y_{ij} una variabile (0-1) che vale 1 se e solo se $x_{ij} > 0$.

Sia $m_{ij} = min a_i, b_j, \ (i, j) \in A$.
Una semplice formulazione matematiche del FCTP è:

$$z(F0) = min \sum_{i \in S} \sum_{j \in T} (c_{ij} x_{ij} + f_{ij} y_{ij}) \tag{1.26}$$

$$s.t. \ \sum_{j \in T} x_{ij} = a_i, \ \ i \in S \tag{1.27}$$

$$\sum_{i \in S} x_{ij} = b_j, \ \ j \in T \tag{1.28}$$

$$x_{ij} \leq m_{ij} y_{ij}, \quad (i, j) \in A \tag{1.29}$$

$$x_{ij} \geq 0, \qquad (i, j) \in A \tag{1.30}$$

$$y_{ij} \in \{0, 1\} \tag{1.31}$$

Si denota con $LF0$ il rilassamento lineare del problema $F0$ e con $z(LF0)$ il costo della soluzione ottima. Notare che, per ogni soluzione ottima di $LF0$, le variabili $x_{ij} > 0$ corrispondono ad una soluzione base fattibile dei vincoli 1.27 e 1.28, e $y_{ij} = x_{ij}/m_{ij}$ *con* $(i, j) \in A$.

1.4 Assegnamento dei veicoli alle baie di carico

Sia dato un insieme N di veicoli che devono scaricare presso un deposito che ha un insieme L di linee di scarico.

Per ogni linea di scarico $j \in L$ è definito l'insieme degli istanti di tempo T_j in cui è operativa.
Per ogni veicolo $i \in N$ sono noti:

- il sottoinsieme di linee $L_i \subseteq L$ compatibili con le operazioni di scarico richieste dal veicolo;
- iltempo di arrivo a_i del veicolo al deposito;
- la durata dello scarico d_{ij} sulla linea $j \in L_i$.

Si assume che lo scarico di un veicolo non possa essere interrotto, ovvero, se lo scarico del veicolo i sulla linea $j \in L_i$ inizia al tempo t, allora la linea j deve essere disponibile per tutti gli istanti di tempo $\tau = t, \ldots, t + d_{ij} - 1$ (ovvero $\tau \in T_j$ per ogni $\tau = t, \ldots, t + d_{ij} - 1$). Indichiamo con I_{ij} l'insieme degli istanti di tempo in cui può iniziare lo scarico del veicolo i sulla linea $j \in L_i$, ovvero per ogni $t \in I_{ij}$ si assume che la linea j disponibile per ogni istante $\tau = i, \ldots, d_{ij} - 1$.

Sia c_{ijt} è il costo per iniziare lo scarico del veicolo $i \in N$ sulla linea $j \in Li$ al tempo $t \in I_{ij}$.
Il problema richiede che ogni veicolo sia assegnato ad una linea di scarico compatibile in modo che ogni scarico sia fatto senza interruzioni e sia minimo il costo dell'assegnamento.

1.4.1 Formulazione matematica F

Per ogni $i \in N$, $j \in L_i$ e $t \in I_{ij}$ poniamo $\delta_{ijt\tau} = 1$ per $\tau = t, \ldots, t + d_{ij} - 1$ e $\delta_{ijt\tau} = 0$ per ogni $\tau \in T_j$ tale che $\tau < t$ oppure $\tau > t + d_{ij} - 1$.

Indichiamo con $N_j \subseteq N$ il sottoinsieme di veicoli che possono essere scaricati sulla linea j, ovvero $N_j = \{i \in N : \ j \in L_i\}$.

1.4.1.1 Variabili

x_{ijt} è una variabile (0-1) che vale 1 se e solo se il veicolo $i \in N$ inizia lo scarico sulla linea $j \in L_i$ al tempo $t \in I_{ij}$.

$s_{j\tau}$ è una variabile (0-1) che vale 1 se e solo se la linea j non viene utilizzata nell'istante di tempo τ.

La formulazione matematica F del problema è la seguente.

$$z(F) = min \sum_{j \in L} \sum_{i \in N_j} \sum_{t \in I_{ij}} c_{ijt} + x_{ijt} + \sum_{j \in L} \sum_{\tau \in T_j} g_{j\tau} s_{j\tau} \tag{1.32}$$

$$s.t. \quad \sum_{j \in L_i} \sum_{t \in I_{ij}} x_{ijt} = 1, \quad i \in N \tag{1.33}$$

$$\sum_{i \in N_j} \sum_{t \in I_{ij}} \delta_{ijt\tau} x_{ijt} + s_{j\tau} = 1, \quad j \in L, \ \tau \in T_j \tag{1.34}$$

$$x_{ijt} \in 0, 1, \qquad i \in N, \ j \in L_i, \ t \in I_{ij} \tag{1.35}$$

$$s_{j\tau} \in 0, 1, \qquad j \in L, \ \tau \in T_j \tag{1.36}$$

Il vincolo 1.33 impone che ad ogni veicolo venga assegnato una linea compatibile ed un tempo di scarico a sua volta compatibile sia con il veicolo stesso che con la linea a lui assegnata.

Il vincolo 1.34 impone che per ogni linea ed ogni istante di tempo compatibile con la linea vi sia in scarico al più un solo veicolo.
La formulazione F richiede $\hat{n} = |N| \times |L| \times \hat{I}$ variabili, dove $\hat{I} = max|I_{ij}| : i \in N, \ j \in L_i$ e al più $\hat{m} = |N| + |L| \times \hat{T}$ vincoli, dove $\hat{T} = max|T_j| : \ j \in L$.

Supponiamo di discretizzare il tempo a 5 minuti, che ogni linea sia disponibile al più 10 ore (i.e. $\hat{T} = 120$) e che un veicolo quando arriva non possa aspettare più di 5 ore (i.e. $\hat{I} = 60$). Avremo $\hat{n} = 200 \cdot 20 \cdot 60 = 240.000$ e $\hat{m} = 200 + 20 \cdot 120 = 2600$.

1.5 Lot Sizing Problem

Il termine ***Lot Sizing*** indica il processo decisionale mediante il quale un'azienda definisce la politica ottima di investimenti, produzione e stoccaggio dei prodotti per soddisfare le richieste dei clienti nel rispetto dei vincoli di produzione e di magazzino.

Non esiste un unico modello di lot sizing che rappresenti in modo generale le varie realtà operative. Sistemi di produzione anche marginalmente diversi possono richiedere modelli aventi complessità computazionale molto diverse.

Non esiste in letteratura un modello generale che contenga come sottocasi tutti i problemi reali noti di lot sizing.

Per questi motivi non esistono software commerciali general pourpose.
Diverse aziende di consulenze nel settore della supply chain vendono software basati su modelli semplificati che non necessariamente producono soluzioni operative ma lasciano all'utente il compito di modificare manualmente la soluzione prodotta per tener conto delle specifiche complessità del problema reale.
I problemi reali sono varianti complesse delle seguenti tre classi di lot sizing problem di un singolo prodotto che sono risolvibili in tempo polinomiale:

- lot sizing senza vincoli di capacità produttiva;
- lot sizing con back logging senza vincoli di capacità;
- lot sizing con vincoli di capacità.

Molti problemi reali possono essere risolti rilassando in modo lagrangiano i vincoli reali per cui il problema lagrangiano risultante corrisponde ad uno dei tre problemi suddetti.

1.5.1 Lot sizing senza vincoli di capacità

Si consideri un'azienda che deve pianificare la propria produzione per un orizzonte temporale di T periodi (ad esempio, T mesi).

Per ciascun periodo $t = 1, \ldots, T$ sono noti:

d_t domanda complessiva dei clienti;

A_t costo fisso di set up per attivare la produzione;

p_t costo per produrre un'unità di prodotto;

h_t costo per unità di prodotto presente nel magazzino alla fine del periodo t.

Per ciascun periodo t, deve essere deciso il numero di unità che devono essere prodotto al fine di soddisfare la domanda in ciascun periodo.
Si suppone che la quantità prodotto nel periodo t sia subito disponibile e che la quantità non venduta alla fine di ogni mese viene depositata in magazzino. L'obiettivo è di minimizzare i costi complessivi di set up, produzione e stoccaggio.

1.5.1.1 Formulazione Matematica (modello di Wagner-Whitin)

Variabili decisonali associate a ciascun periodo t=1,. . . ,T

x_t quantità prodotta all'inizio del periodo t;

I_t livello del magazzino alla fine del periodo t;

$y_i \in (0,1) : y_t = 1$ se nel periodo t vi è produzione, $y_t = 0$ altrimenti.

$$Min\ z = \sum_{t=1}^{T}(p_t x_t + h_t I_t + A_t y_t) \tag{1.37}$$

$$x_t + I_{t-1} = I_t + d_t,\ t = 1, \ldots, T \tag{1.38}$$

$$x_t \leq M y_t,\ t = 1, \ldots, \tag{1.39}$$

$$x_t,\ I_t \geq 0,\ t = 1, \ldots, T \tag{1.40}$$

$$y_t \in \{0,1\},\ t = 1, \ldots, T \tag{1.41}$$

$$dove\ M = \sum_{t=1}^{T} d_t \text{ e, per semplicità, si suppone che } I_0 = 0. \tag{1.42}$$

1.5.1.2 Metodo di soluzione

Al modello si associa il grafo $R = (N, A)$ senza vincoli di capacità sugli archi tale che ogni soluzione del problema corrisponde ad un flusso in R.
Il grafo R si compone di $2T + 1$ nodi:

- nodo sorgente S da cui parte un flusso pari a $\sum_{t=1}^{T} d_t$;
- per ciascun periodo t una coppia di nodi U_t, V_t dove:

  U_t rappresenta il magazzino,

  V_t corrisponde alla domanda.

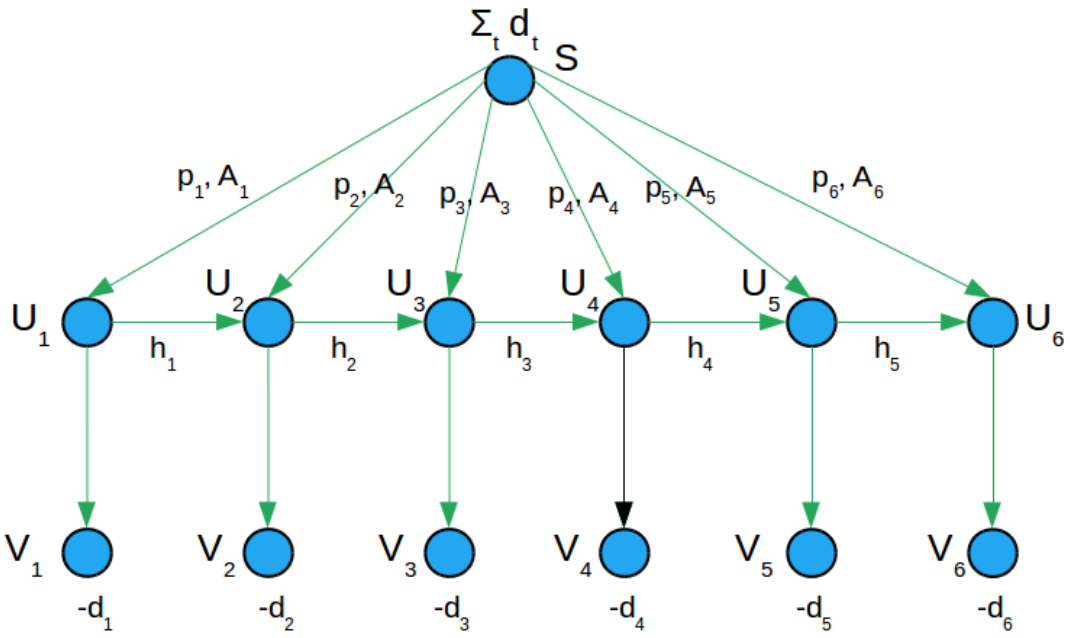
Per ciascun periodo $t = 1, .., T$ vi sono gli archi:

(S, U_t) il cui flusso corrisponde alla produzione x_t;

(U_t, U_{t+1}) il cui flusso è pari al livello I_t del magazzino alla fine del periodo t;

(U_t, V_t) il cui flusso deve essere pari alla domanda d_t.

Figura 1.3: Esempio della rete di flusso (modello di Wagner-Whitin)

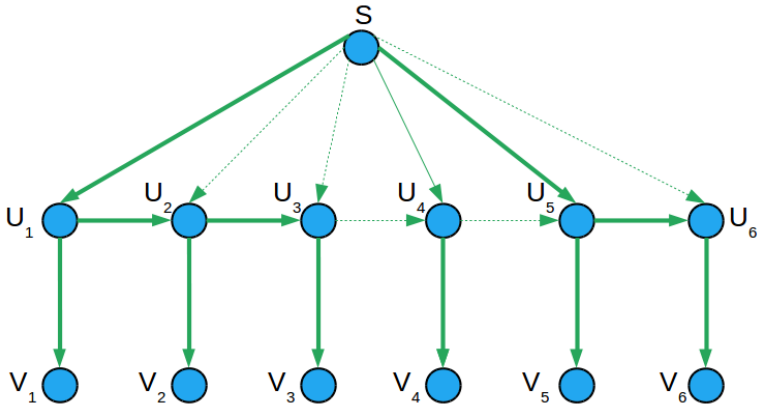


1.5.1.3 Proprietà della soluzione ottima

Teorema. In una soluzione ottima non può mai avvenire che la domanda del periodo t venga soddisfatta sia dalla produzione che dal magazzino, ovvero:

$$I_{t-1} \cdot x_t = 0;\ t = 1, \ldots, T$$

Figura 1.4



1.5.1.4 Algoritmo di soluzione (di complessità $O(T^2)$)

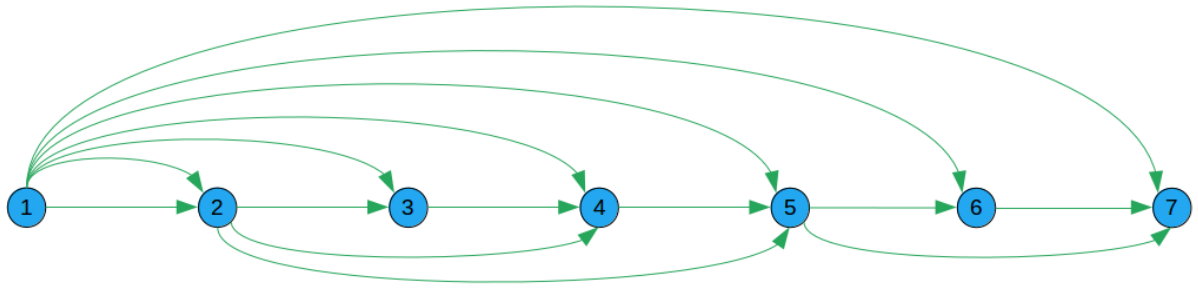
Si costruisca un grafo aciclico di $T+1$ vertici.
Si definiscano gli archi $j,k)$ per $j=0,\ldots,T-1$ e $k=j+1,\ldots,T$.

L'arco (j,k) rappresenta la decisione di produrre all'inizio del periodo $j+1$ quanto serve per soddisfare le domanda complessiva dei periodo $j+1,\ j+2,\ldots,k$.

Il costo M_{jk} dell'arco (j,k) è pari al costo per produrre nel periodo j+1 la quantità $\sum_{r=j+1}^{k} d_r$ più i costi di stoccaggio:

$$M_{jk} = A_{j+1} + p_{j+1} \sum_{r=j+1}^{k} d_r + \sum_{t=j+1}^{k-1} h_t(\sum_{r=t+1}^{k} d_r)$$

Figura 1.5



Ogni soluzione del modello di Wagner-Whitin corrisponde ad un cammino da 0 a t in questo grafo aciclico.
Il cammino di costo minimo fornisce la soluzione ottima.

CAPITOLO 2

INTRODUZIONE ALLA PROGRAMMAZIONE LINEARE A NUMERI INTERI

Si consideri il seguente problema.

$$Min\ cx \tag{2.1}$$
$$Ax = b \tag{2.2}$$
$$x \geq 0 \tag{2.3}$$
$$x\ intero \tag{2.4}$$

Le variabili devono assumere valori interi:

$$Es: \quad x_i = Numero\ di\ uomini\ che\ devono\ essere\ assegnati\ al\ lavoro\ i. \tag{2.5}$$
$$= Numero\ di\ automezzi\ che\ devono\ operare\ il\ trasporto\ lungo\ la\ "tratta\ i". \tag{2.6}$$
$$\tag{2.7}$$

2.1 Arrotondamento ad una soluzione non-intera

Si risolva il problema ignorando i vincoli $[x : intero]$. Le variabili che risultano non intere, nella soluzione ottima del problema continuo, vengano arrotondate al valore intero più vicino.

$$Es: \quad Min\ z = -2x_1 + 3x_2 \tag{2.8}$$
$$x_1 + x_2 \geq 3 \tag{2.9}$$
$$3x_1 + x_2 \leq 6 \tag{2.10}$$
$$x_2 \leq 5 \tag{2.11}$$
$$x_1,\ x_2 \geq 0\ ed\ intere \tag{2.12}$$

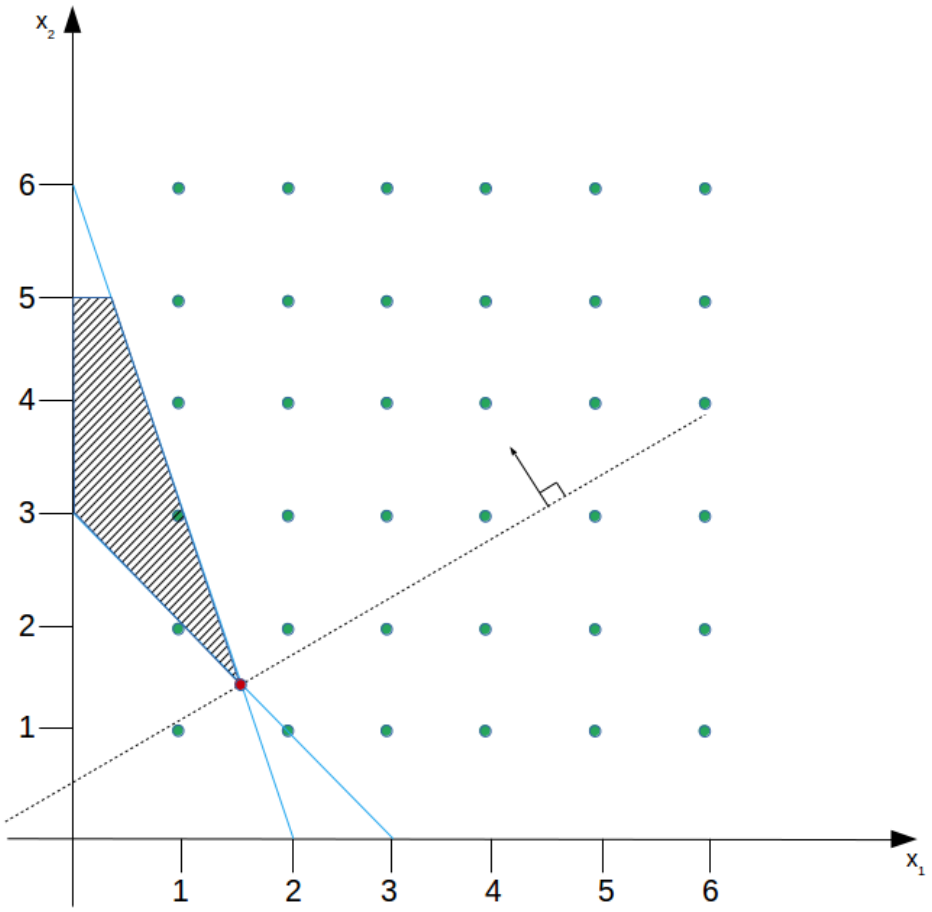


Figura 2.1:
Soluzione continua: $z = \frac{3}{4}$; $x_1 = \frac{3}{2}$, $x_2 = \frac{3}{2}$
Soluzione intera: $z = 4$; $x_1 = 1$, $x_2 = 2$

In questo esempio la soluzione arrotondata coincide con la soluzione ottima.

$$Es: \quad Min\ z = 8x_1 + 6x_2 \tag{2.13}$$
$$4x_1 + 3x_2 \geq 6 \tag{2.14}$$
$$x_1,\ x_2 \geq 0\ ed\ intere \tag{2.15}$$

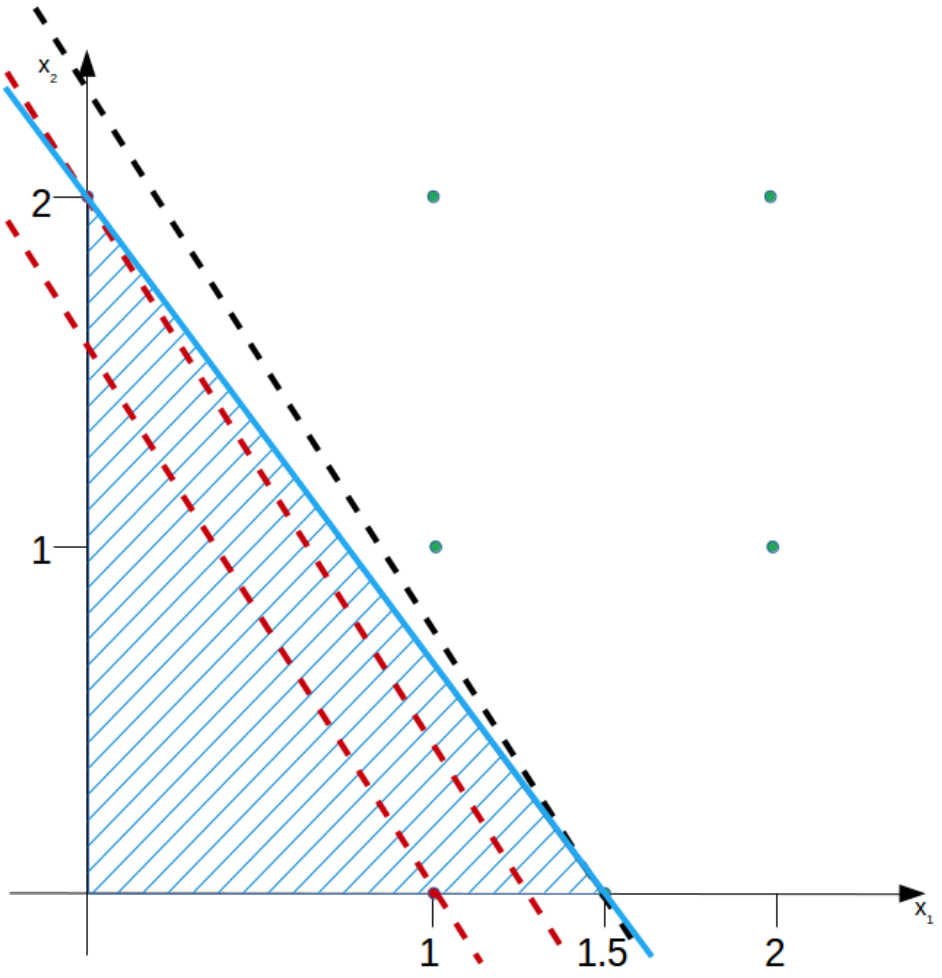
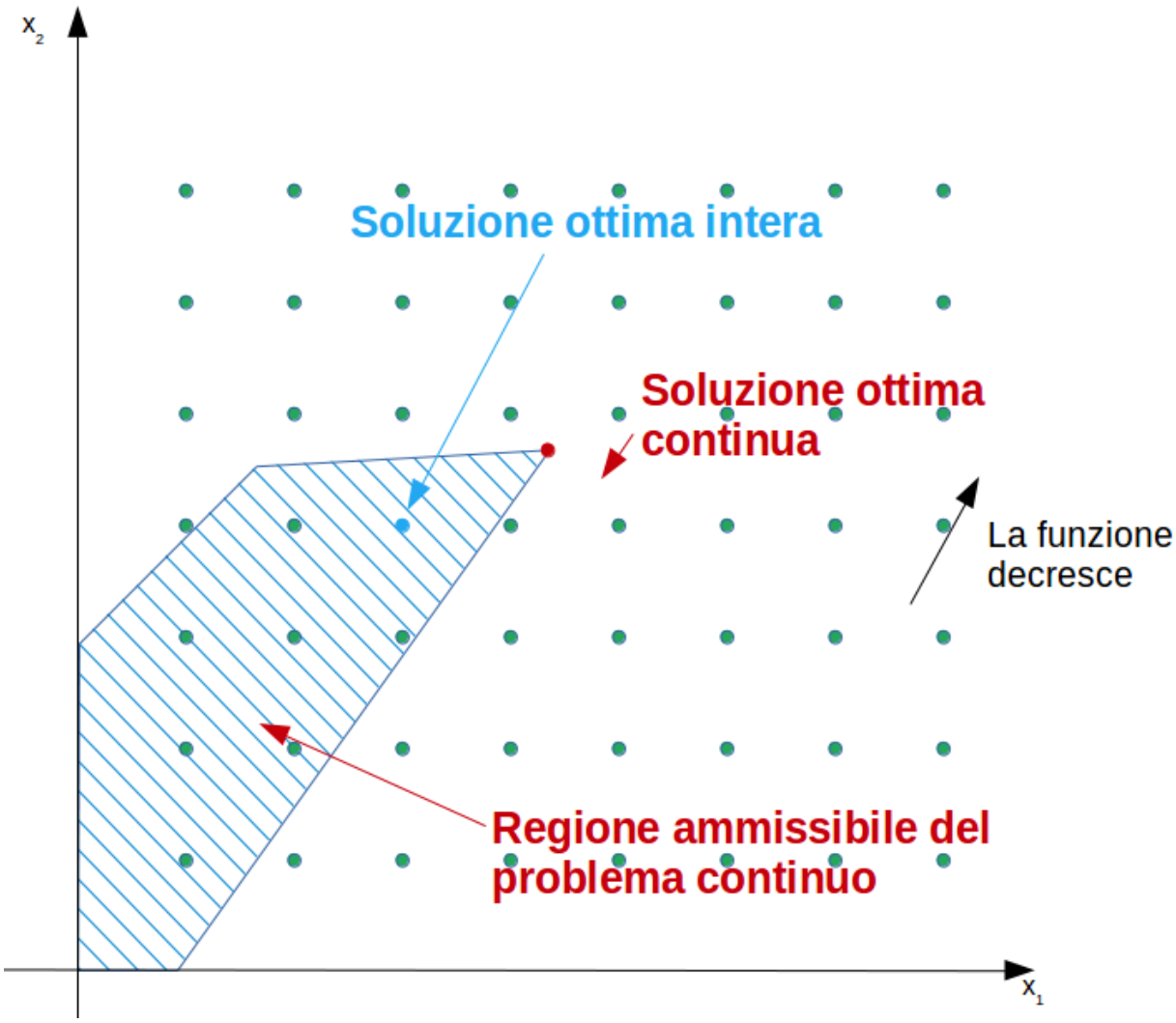


Figura 2.2:
Soluzione continua: $z = 12$; $x_1 = 1,5$, $x_2 = 0$
Soluzione arrotondata $z = 8$; $x_1 = 1$, $x_2 = 0$
Soluzione intera: $z = 10$; $x_1 = 0$, $x_2 = 2$

La soluzione arrotondata si discosta notevolmente dalla soluzione ottima.

$$Es: \quad Min\ z = 8x_1 + 6x_2 \tag{2.16}$$
$$4x_1 + 3x_2 \geq 6 \tag{2.17}$$
$$x_1,\ x_2 \geq 0\ ed\ intere \tag{2.18}$$



I quattro punti interi più vicini alla soluzione continua non sono ammissibili.

2.2 Unimodularità

La matrice intera A di m righe ed n colonne è totalmente unimodulare se ogni sua sottomatrice quadrata B non singolare è unimodulare, ovvero $det(B) = \pm 1$.

Teorema. Se la matrice intera A è totalmente unimodulare allora tutti i punti estremi dell'insieme pd. convesso $X = x : \ Ax = b, \ x \geq 0$ sono interi per ogni vettore intero b.

Dimostrazione. Sia B una base ammissibile e x_b le variabili base: $Bx_B = b$.
Per la regola di Cramer:

$$x_{b_i} = \frac{\det(B_i)}{\det(B)} \tag{2.19}$$

Dove B_i si ottiene da B sostituendo la i-esima colonna di B con b. È ovvio che $\det(B_i)$ è un numero intero e quindi anche ciascun x_{B_i} è intero.

Teorema. Una matrice intera A i cui elemento sono $0, +1, -1$ è totalmente unimodulare se:

1. In ogni colonna A compaiono al più due elementi non-nulli (cioè $1, -1$);
2. L'insieme delle righe R può essere suddiviso in due insieme disgiunti R_1 e R_2 ($R_1 \cup R_2 = R$) per cui:
   (a) Se una colonna contiene due elementi non-nulli dello stesso segno allora la riga corrispondente ad uno dei due elementi appartiene a R_1 mentre la riga relativa all'altro elemento è in R_2;
   (b) Se una colonna contiene due elementi di segno opposto entrambe le righe appartengono allo stesso insieme.

Esempi.

$$A = \begin{bmatrix} -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 1 & -1 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 1 & -1 & 0 \end{bmatrix}$$

$$\begin{aligned} R &= 1,2,3,4 \\ R_1 &= 1,2,3,4 \\ R_2 &= \emptyset \end{aligned} \qquad \begin{aligned} R &= 1,2,3,4,5 \\ R_1 &= 1,2,3 \\ R_2 &= 4,5 \end{aligned}$$

La totale unimodularità della matrice A è **condizione sufficiente** affinchè la soluzione ottima

x^* sia intera per

$$\begin{aligned} &Min\ cx \\ &\quad Ax = b\ (bintero) \\ &\quad x \geq 0 \end{aligned}$$

La condizione non è **necessaria**.

Esempio:

dato il sistema di vincoli

$$\begin{aligned} 6x_1 + x_2 &= 7 \\ 2x_1 + x_2 &= 3 \end{aligned}$$

L'unica soluzione è $(x_1 = 1,\ x_2 = 1)$ mentre la matrice

$$A = \begin{bmatrix} 6 & 1 \\ 2 & 1 \end{bmatrix}$$

non risulta essere totalmente unimodulare.

2.3 Metodo dei piani di taglio

Sia dato il problema

$$ILP \begin{cases} Min\ z_{ILP} = cx \\ Ax = b \\ x \geq 0\ e\ intero \end{cases}$$

Supponiamo A,c,b interi.

Si consideri il problema rilassato che si ottiene da 'ILP' ignorando i vincoli di interezza. Indichiamo tale problema con ***LP***.

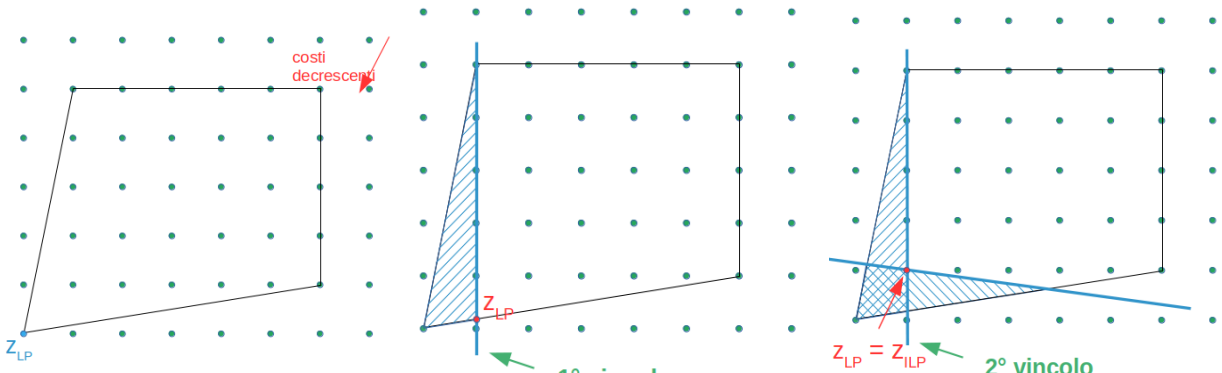
$$LP \begin{cases} z_{LP} = Min\ cx \\ ax = b \\ x \geq 0 \end{cases}$$

È noto che $z_{LP} \leq z_{ILP}$.

2.3.1 Piani di taglio

Si risolva LP; se la soluzione è intera tale soluzione è anche l'ottimo di ILP.

Altrimenti vengono aggiunti a LP vincoli, *che non escludono soluzioni intere*, fino a che la soluzione del problema LP risultante non risulti intera.



In (A) viene mostrata la regione ammissibile di LP ed il punto di ottimo.

In (B) viene mostrata la regione ammissibile di LP più un vincolo che rende non-ammissibile l'ottimo ottenuto in (A) ma che non esclude nessuno dei punti interi.

In (C) viene mostrato come l'aggiunta di un secondo vincolo rende la soluzione intera.

Nell'esempio sono sufficienti 2 vincoli per rendere la soluzione intera.

In generale bisogna aggiungere vincoli fino a che la soluzione non risulti intera o si scopra che il problema non ha soluzioni intere.

2.3.2 Gomory cuts

Si consideri il tableau ottimo relativo a LP:

	z	x_1	...	x_m	x_{m+1}	...	x_j	...	x_n	
	1	0	...	0						
x_1		1								$\bar{b}_1$
			1							
			...							
x_r				1	y_r^{m+1}	...	y_r^j	...	y_r^n	$\bar{b}_r$
				...						
x_m				1						$\bar{b}_m$

Supponiamo la soluzione ottima non intera.

Sia $\bar{b}_r$ non intero.

L'equazione associata a x_r è:

$$x_r + \sum_{j=m+1}^{n} y_r^j x_j = \bar{b}_r \tag{2.20}$$

Poniamo:

$$y_r^j = I_r^j + F_r^j, \text{ dove } I_r^j = \lfloor y_r^j \rfloor, \ (0 \leq F_r^j < 1) \tag{2.21}$$

Inoltre:

$$\bar{b}_r = I_r + F_r \text{ essendo } 0 \leq F_r < 1 \tag{2.22}$$

Sostituendo, la 2.20 diviene:

$$x_r + \sum_{j=m+1}^{n} (I_r^j + F_r^j) x_j = (I_r + F_r) \tag{2.23}$$

o anche:

$$\underbrace{x_r + \sum_{j=m+1}^{n} I_r^j x_j - I_r}_{\text{intero per ogni x intero}} = \underbrace{F_r - \sum_{j=m+1}^{n} F_r^j x_j}_{< 1 \text{ per } x \geq 0} \tag{2.24}$$

Ne segue:

$$F_r - \sum_{j=m+1}^{n} F_r^j x_j \leq 0 \tag{2.25}$$

La soluzione corrente non soddisfa il vincolo 2.20 in quanto $x_j = 0$ $j = m+1, \ldots, n$ mentre $F_r > 0$ poiché $\bar{b}_r$ si è supposto non intero.

Se il vincolo 2.20 viene aggiunto al problema LP allora la soluzione corrente risulterà non ammissibile.
Per determinare una nuova soluzione che soddisfi il vincolo 2.20 può essere impiegato il *Simplesso Duale* partendo dal tableau ottimo relativo alla soluzione corrente.

Al tableau va aggiunto il vincolo:

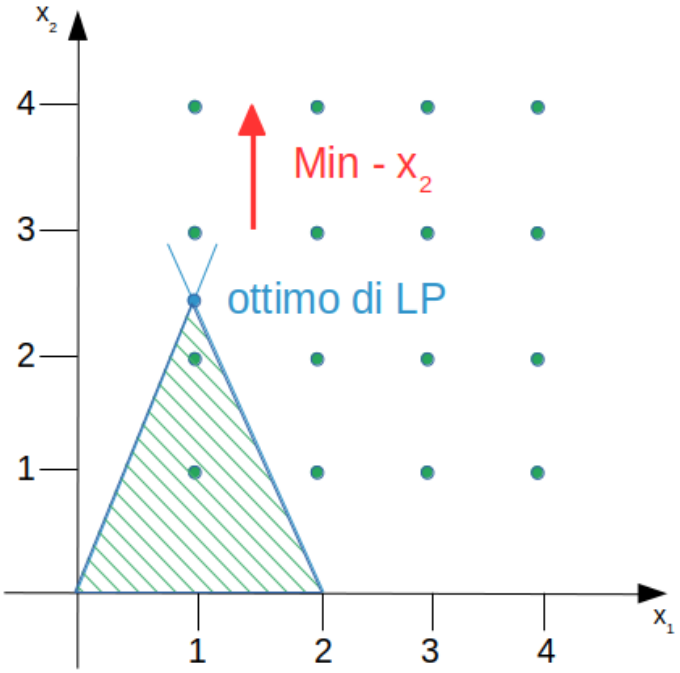
$$-\sum_{j=m+1}^{j} F_r^j x_j + s = -F_r \tag{2.26}$$

La nuova variabile slack s è una nuova variabile base.
Il nuovo tableau è non ammissibile per il Primale ma duale ammissibile.

Esempio.

$$\begin{aligned} Min \ & -x_2 \\ & 3x_1 + 2x_2 \leq 6 \\ & -3x_1 + 2x_2 \leq 0 \\ & x_1,\ x_2 \geq 0 \\ & x_1,\ x_2 \ intere \ \geq 0 \end{aligned}$$

Si noti che l'ottimo cade nel punto $x = (1; 1)$.



	z	x_1	x_2	x_3	x_4	RHS
z	1	0	1	0	0	0
x_3	0	3	2	1	0	6
x_4	0	-3	②	0	1	0

	z	x_1	x_2	x_3	x_4	RHS
z	1	3/2	0	0	-1/2	0
x_3	0	⑥	0	1	-1	6
x_2	0	-3/2	1	0	1/2	0

		x_1	x_2	x_3	x_4	RHS
z	1	0	0	-1/4	-1/4	-3/2
x_1	0	1	0	1/6	-1/6	1
x_2	0	0	1	1/4	1/4	3/2

Tabella 2.1: Tableau ottimo. Soluzione continua!

Ricordando che il cut da aggiungere è:

$$-\sum_{j=m+1}^{n} F_r^j x_j + s = -F_r \tag{2.27}$$

Dalla riga di x_2 si ha la sequente equzione:

$$x_2 + \frac{1}{4}x_3 + \frac{1}{4}x_4 = \frac{3}{2} \tag{2.28}$$

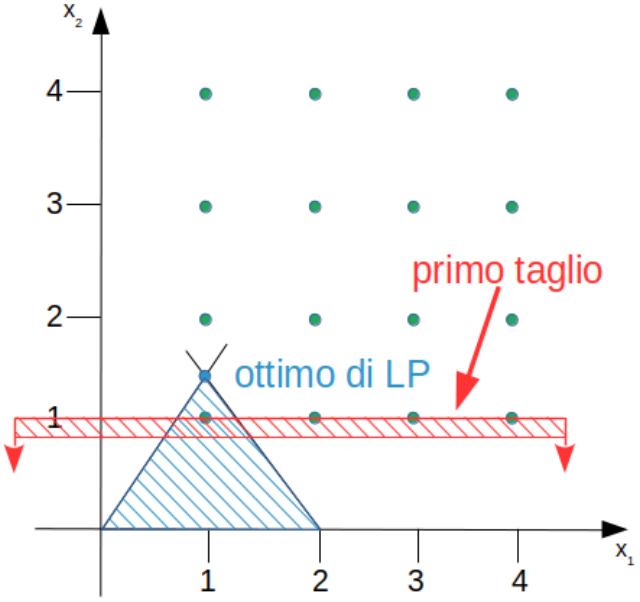
da cui:

$$-\frac{1}{4}x_3 - \frac{1}{4}x_4 + s_1 = -\frac{1}{2} \tag{2.29}$$

Si osservi che per definizione di x_3 e x_4 si ha

$$x_3 = 6 - 3x_1 - 2x_2 \text{ ed } x_4 = 3x_1 - 2x_2 \tag{2.30}$$

Sostituendo in 2.29 si ottiene $x_2 \leq 1$:



Aggiungendo il cut al tableau ottimo precedente:

		x_1	x_2	x_3	x_4	s_1	RHS
z	1	0	0	-1/4	-1/4	0	-3/2
x_1	0	1	0	1/6	-1/6	0	1
x_2	0	0	1	1/4	1/4	0	3/2
s_1	0	0	0	**-1/4**	-1/4	1	1

Continuando con il simplesso duale:

		x_1	x_2	x_3	x_4	s_1	RHS
z	1	0	0	0	0	-1	-1
x_1	0	1	0	0	-1/3	2/3	2/3
x_2	0	0	1	0	0	1	1
s_1	0	0	0	1	1	-4	2

Dalla riga di x_1 si ha i cut:

$$-\frac{2}{3}x_4 - \frac{2}{3}s_1 + s_2 = -\frac{2}{3} \tag{2.31}$$

Il nuovo tableau, quindi, diviene:

e ottimizzando con li simplesso duale: L'algoritmo converge in un numero finito di passi purché venga impiegata una appropiata regola lessicografica per la scelta del pivot.

		x_1	x_2	x_3	x_4	s_1	s_2	RHS
z	1	0	0	0	0	-1	0	-1
x_1	0	1	0	0	-1/3	2/3	0	2/3
x_2	0	0	1	0	0	1	0	1
s_1	0	0	0	1	1	-4	0	2
s_2	0	0	0	0	-2/3	-2/3	1	2/3

		x_1	x_2	x_3	x_4	s_1	s_2	RHS
z	1	0	0	0	0	-1	-1/2	-1
x_1	0	1	0	0	0	-1	0	1
x_2	0	0	1	0	0	1	0	1
s_1	0	0	0	1	0	-5	3/2	2
s_2	0	0	0	0	1	1	-3/2	1

Tabella 2.2: Tableau ottimo.

2.3.2.1 Come evitare un numero indefinito di righe e colonne

Qualora una variabile di slack s_i, associata all'i-esimo cut, entra in base, si elimina sia il cut sia la variabile s_i.

In questo modo il numero delle righe aggiunte (relative ai cuts) non supera n-m.

2.4 Metodi Branch and Bound

Sia P_0 un problema a cui corrisponde l'insieme S_0 di soluzioni ammissibili.

Ad esempio

$$P_0 \begin{cases} Min \;\; cx \\ \qquad Ax = b \\ \qquad x \geq 0 \; e \; intero \end{cases}$$

$$S_0 = \{x : Ax = b, \; x \geq 0 \; intero\} \tag{2.32}$$

Principio base dei metodi Branch and Bound
Suddividre il problema P_0 nei sottoproblemi $P_1, P_2, \ldots, P_k$ a cui corrispondono gli insiemi di soluzioni ammissibili $S_1, S_2, \ldots, S_k$. La suddivisione è tale per cui

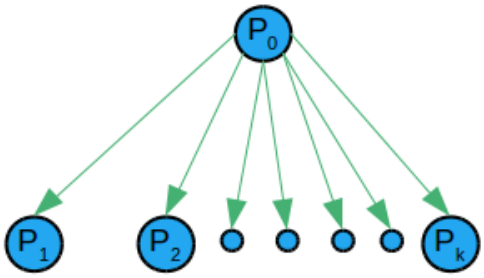
$$S_1 \cup S_2 \cup \cdots \cup S_k = S_0 \tag{2.33}$$

È evidente che:

$$\min_{x \in S_0} cx = MIN\{\min_{x \in S_1} cx, \min_{x \in S_1} cx, \ldots, \min_{x \in S_k} cx\} \tag{2.34}$$

La risoluzione di ogni sottoproblema $P_1, P_2, \ldots, P_k$ può risultare molto più semplice della risoluzione di P_0.

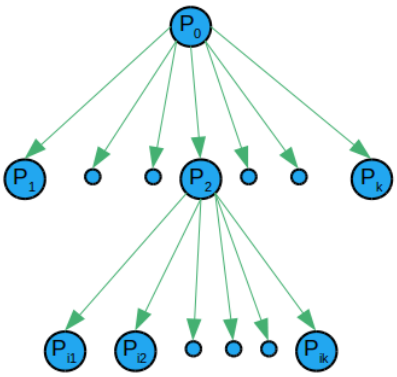
Possiamo rappresentare la suddivisione di P_0 in $P_1, P_2, \ldots, P_k$ mediante un albero



Nel caso in cui la riduzione di uno o più sottoproblemi risulti *difficile* questi possono essere ulteriormente suddivisi.

Supponiamo che P_i risulti difficile, allora può essere suddiviso nei sottoproblemi $P_{i1}, P_{i2}, \ldots, P_{ik}$ a cui corrispondono gli insiemi di soluzioni ammissibili $S_{i1}, S_{i2}, \ldots, S_{ik}$ tali che $S_{i1} \cup S_{i2} \cup \ldots \cup S_{ik} = S_i$

Si ha il seguente albero



Risolvere P_0 equivale a risolvere $P_1, \ldots, P_{i-1}, (P_{i1}, P_{i2}, \ldots, P_{ir}), P_{i+1} \ldots, P_k$

Il processo di suddivisione di un problema in un numero finito di sottoproblemi viene chiamato **BRANCHING**.

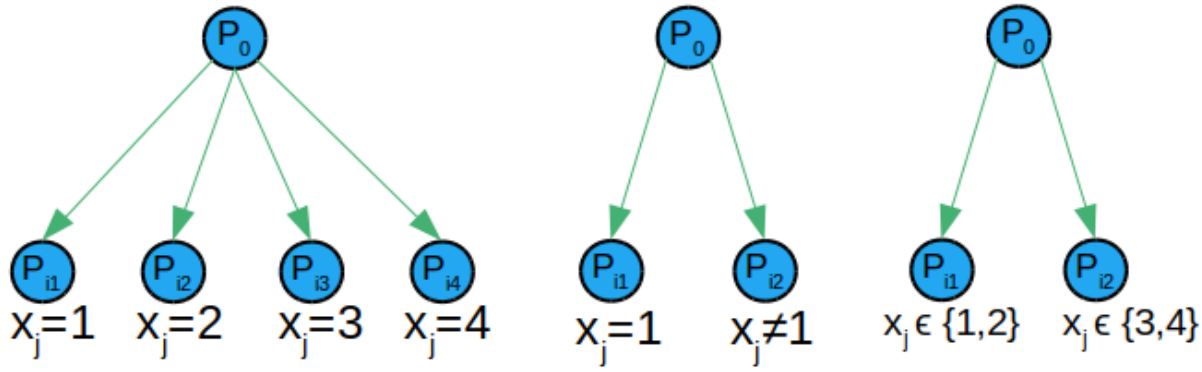
Una buona strategia di branching consiste nel suddividere P_i nei sottoproblemi $P_{i1}, P_{i2}, \ldots, P_{ir}$ in modo che, per ogni coppia $P_{i\alpha}, P_{i\beta}$ con $(\alpha \neq \beta)$, gli insiemi $S_{i\alpha}$ e $S_{i\beta}$ siano disgiunti.

$$S_{i\alpha} \cap S_{i\beta} = \emptyset$$

Se la condizione sopra è soddisfatta allora $\{S_i, \ldots, S_{ir}\}$ è una **PARTIZIONE** di S_i.
Si noti che la condizione non è *necessaria* ma rende computazionalmente efficiente il processo di branching.

2.4.0.1 Esempi di Branching

Si consideri il problema P_i in n variabili dove la variabile x_j può assumere i valori $1, 2, 3, 4$.

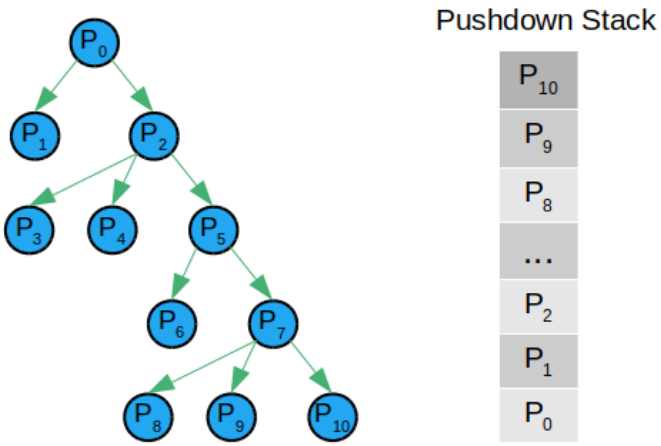


2.4.1 Tipi di Branching

Ogni sottoproblema che non può essere risolto può essere suddiviso in sottoprblemi più piccoli. Dato un insieme di sottoproblemi da suddividere quale sottoproblema suddividere per primo?

2.4.1.1 Dept-first search

In questo tipo di branching il sottoproblema che viene suddiviso per primo è l'ultimo generato. Ciò si ripete fino ad ottenere un sottoproblema che può essere risolto.



BACKTRACKING: quando un sottoproblema è risolto viene scelto il penultmo sottoproblema generato e su questo viene effettuato il branching.

2.4.1.2 Breadth-first search

Il branching procede da livello a livello, ovvero il problema P_0 è suddiviso in $P_1, P_2, \ldots, P_k$ che sono i sottoproblemi a livello 1.

Ogni sottoproblema a livello 1 viene suddiviso in un numero di sottoproblemi che costituiscono il livello 2.

In generale quando viene esaminato un sottoproblema a livello K sono stati già esaminati tutti i sottoproblemi a livello $K-1$.

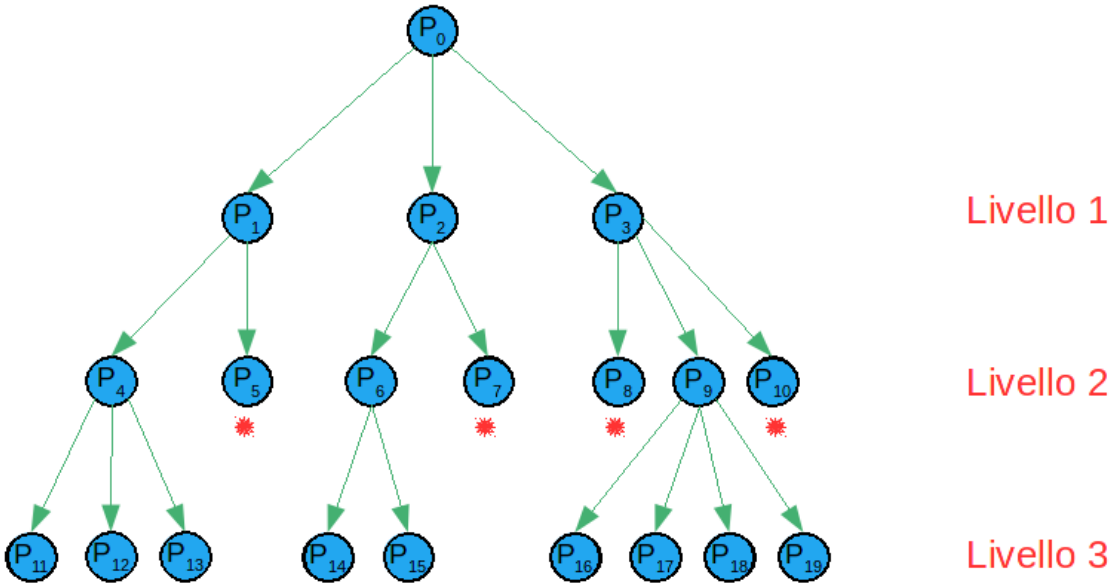


Figura 2.3: * Problemi risolti

2.4.2 Bounds

La ricerca della solutione ottima di P_0 è completa quando sono stati risolti tutti i sottoproblemi generati.
Questo processo può essere migliorato calcolando, per ogni sottoproblemi P_j un *bound* (*Lower Bound* se il problema è di minimizzazione).

Lower Bound: diremo che LB_i è un lower bound al sottoproblema P_i se

$$LB_i \leq \underset{x \in S_i}{Min}\{cx\} \tag{2.35}$$

Upper Bound: diremo che UB_i è un upper bound al sottoproblema P_i se

$$UB_i \leq \underset{x \in S_0}{Min}\{cx\} \tag{2.36}$$

È possible trascurare il sottoproblema P_i se $LB_i \geq UB$, infatti, poiché $LB_i \leq \underset{x \in S_i}{Min}\{cx\}$ si avrebbe $\underset{x \in S_i}{Min}\{cx\} \geq UB$ e quindi il sottproblema P_i non contiene la soluzione ottima.

2.4.2.1 Calcolo del Lower Bound

Sia dato il problema

$$P_0 \begin{cases} Min \;\; z = cx \\ \qquad\;\; Ax = b \\ \qquad\;\; x \geq 0 \; e \; intero \end{cases}$$

Sia z^* il costo della soluzione ottima di P_0.

I seguenti medoti producono validi Lower Bounds a P_0.

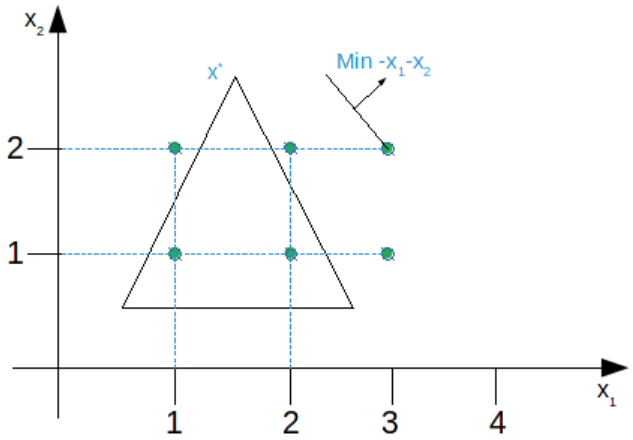
Rilassamento continuo Si ignori il vincolo x intero; il problema risultante è risolvibile con la programmazione lineare.
Sia z^*_{LP} il costo di tale soluzione; si ha

$$z^*_{LP} \leq z^* \tag{2.37}$$

- Se la soluzione del problema continuo è intera allora è anche la soluzione ottima intera e quindi $z^*_{LP} = z^*$;

- Se la soluzione è frazionaria allora può essere usato un metodo Branch & Bound per trovare la soluzione ottima intera.

Esempio.

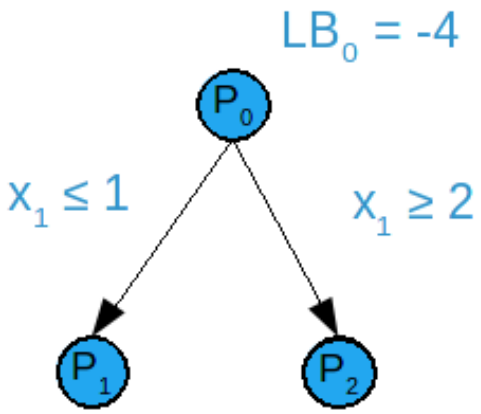


Rilassamento continuo $x^* = (\frac{3}{2}, \frac{5}{2}),\; z^*_{LP} = cx^* = -4$

P_0 è suddiviso in due sottoproblemi P_1 *e* P_2

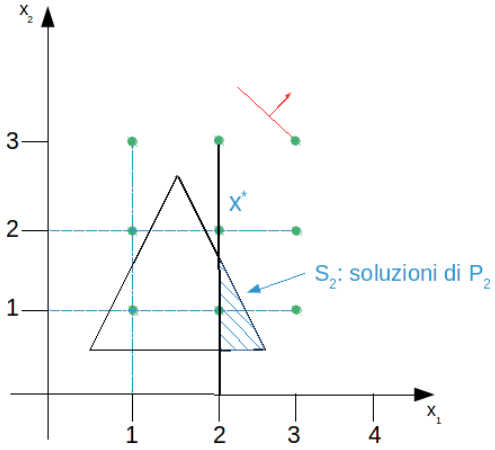
- P_1 imponiamo che $x_1 \leq 1$

- P_2 imponiamo che $x_1 \geq 2$



Si usi una strategia Depth-First e quindi si esamini il problema P_2.

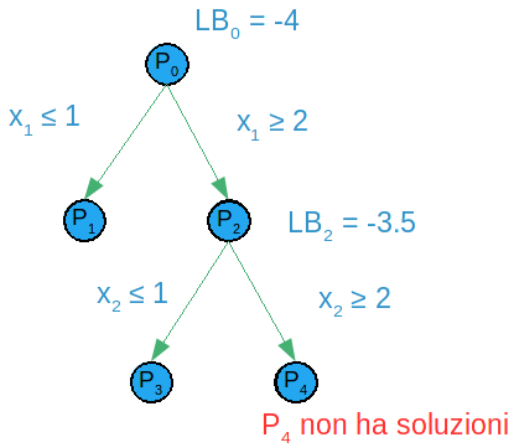
Esame del sottoproblema P_2
Il lower bound si ottiene imponendo il vincolo $x_1 \geq 2$.



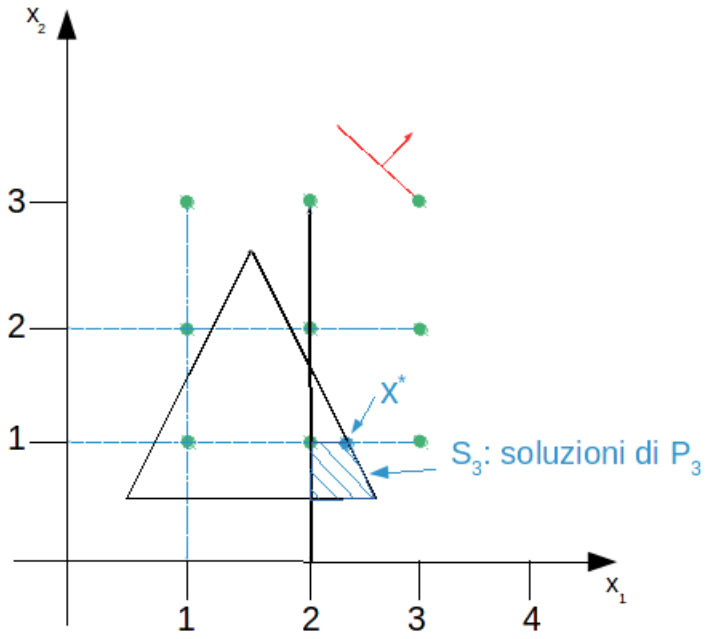
L'ottimo per P_2 è $z^*_{LP} = -3.5$ con componenti $x^*_1 = 2$ e $1 < x^*_2 < 2$

Il problema P_2 vien suddiviso in P_3 e P_4 dove

- P_3 imponiamo $x_1 \geq 2$ e $\underline{\boldsymbol{x_2 \leq 1}}$
- P_4 imponiamo $x_1 \geq 2$ e $\underline{\boldsymbol{x_2 \geq 2}}$



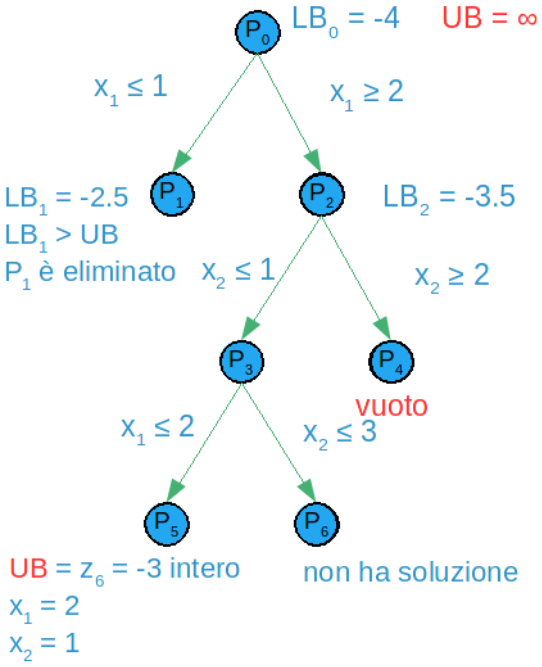
Esame del sottoproblema P_3



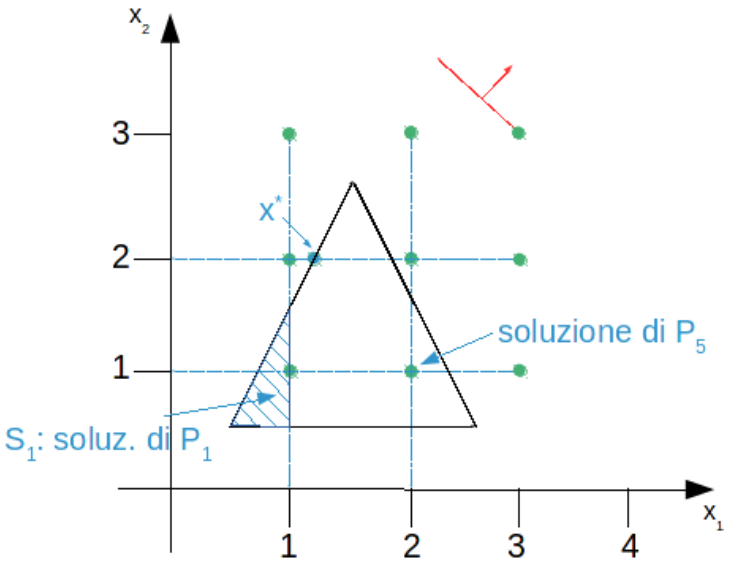
L'ottimo di P_3 è $z^*_{LP} = -3.25$ con componenti $x^*_2 = 1$ e $2 < x^*_1 < 3$.

Il problema P_3 viene suddiviso in P_5 e P_6 dove

- P_5 imponiamo $x_1 \geq 2$, $x_2 \leq 1$ e $\underline{\boldsymbol{x_1 \leq 2}}$
- P_6 imponiamo $x_1 \geq 2$, $x_2 \leq 2$ e $\underline{\boldsymbol{x_1 \geq 3}}$



Al nodo dell'albero, corrispondente al problema P_5 si è ottenuta la prima soluzione ammissibile di P_0 di costo -3. Quindi poniamo $UB = 3$.
Il backtracking conduce ad esaminare il problema P_1



La soluzione ottima di P_1 è $z^*_{LP} = -2.5$, quindi, $LB_1 = -2.5$ e poiché $LB_1 > UB$ il sottoproblema P_1 non può condurre ad alcuna soluzione migliore di quella trovata per P_5.

Essendo stati esaminati tutti i nodi dell'albero l'algoritmo termina e $z^* = -3$ è la soluzione ottima.

2.4.3 Eliminazione di alcuni vincoli

Si consideri il problema

$$P_0 \begin{cases} Min \;\; z = cx \\ \qquad Ax = b \;\; m_1 \; vincoli \\ \qquad Dx = h \;\; m_2 \; vincoli \\ \qquad x \geq 0 \;\; intero \end{cases}$$

Si consideri il problema RP che deriva da P_0 eliminando i vincoli $Ax = b$

$$RP \begin{cases} Min \;\; z_{RP} = cx \\ \qquad Dx = b \\ \qquad x \geq 0 \; e \; intero \end{cases}$$

Sia z^*_{RP} il valore ottimo di RP; si ha

$$z^*_{RP} \leq z^* \quad (\; z^* \text{ ottimo di } P_0 \;) \tag{2.38}$$

L'estensione di questo metodo è il Rilassamento Lagrangiano mediante il quale è possibile tener conto dei vincoli rilassati nella funzione obiettivo.

2.4.4 Rilassamento Surrogato

Sia dato il problema

$$P_0 \begin{cases} Min \;\; z = cx \\ \qquad \displaystyle\sum_{j=1}^{n} a_{ij} x_j \geq b_i \;\; i = 1, \ldots, m_1 \\ \qquad Dx = h \;\; m_2 \; vincoli \\ \qquad x \geq 0 \;\; intero \end{cases}$$

Si consideri il probelma che si ottiene da P_0 sostituendo i primi m_1 vincoli $a^i x \geq b_i$ con una loro combinazione lineare

$$SP \begin{cases} Min \;\; z_{SP} = cx \\ \qquad \displaystyle\sum_{j=1}^{m_1} \pi_i \sum_{j=1}^{n} a^j_{ij} \geq \sum_{i=1}^{m_1} \pi_i b_i \;\; \pi_i \geq 0 \\ \qquad Dx = h \;\; m_2 \; vincoli \\ \qquad x \geq 0 \;\; intero \end{cases}$$

z^*_{SP}, ottimo di SP, è un valido lower bound a P_0.

$$P \begin{cases} z^* = \;\; Min \; cx \\ \qquad s.t. \; Ax \geq b \\ \qquad x \geq 0 \;\; intero \end{cases}$$

Rilassamento lineare

$$LP \begin{cases} z^*_{LP} = \ Min\ cx & = \ Max\ \omega b \\ \quad s.t.\ Ax \geq b & \omega A \leq c \\ \quad x \geq 0 \ \ intero & \omega \geq 0 \end{cases}$$

Sia ω^* la soluzione duale ottima.

$$SP \begin{cases} z^*_{SP} = \ Min\ cx \\ \quad (\omega^* A)x \geq \omega^* b \\ \quad x \geq 0 \ \ intero \end{cases}$$

Ottenuto tramite **rilassamento surrogato**.

Teorema. $z^*_{SP} \geq z^*_{LP}$ Poiché w^* è la soluzione ottima del duale di LP si ha

$$z^*_{LP} = \omega^* b \quad \text{e} \quad c - \omega^* A x^* \tag{2.39}$$

Sia x^* la soluzione ottima intera di SP; si ha:

$$(c - \omega^* A)x^* \geq 0 \quad \text{ovvero} \quad cx^* \geq \omega^* A x^* \tag{2.40}$$

ma anche (da SP):

$$\omega^* A x^* \geq \omega^* b \tag{2.41}$$

Da 2.40 e 2.41:

$$cx^* \geq \omega^* A x^* \geq \omega^* b = z^*_{LP} \tag{2.42}$$

ovvero

$$z^*_{SP} = cx^* \geq z^*_{LP} \tag{2.43}$$

□

2.5 Assegnamento Generalizzato

Allocazione ottimale di n oggetti in m contenitori in modo che ogni oggetto sia assegnato ad un solo contenitore e non sia superata la portata di ogni contenitore.

Indichiamo con:

b_i : portata del contenitore $i, i = 1, \ldots, m$

a_{ij} : spazio del contenitore i occupato dall'oggetto j (se j viene assegnato a i)

c_{ij} : costo per assegnare al contenitore i l'oggetto j

x_{ij}

$= 1$ se l'oggetto viene assegnato al contenitore i

$= 0$ altrimenti

2.5.1 Formulazione matematica

$$Min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.44}$$

$$\sum_{i=1}^{m} x_{ij},\ j = 1, \ldots, n \tag{2.45}$$

$$\sum_{j=1}^{n} a_{ij}x_{ij},\ i = 1, \ldots, m \tag{2.46}$$

$$x_{ij} \in \{0, 1\},\ \forall i, j \tag{2.47}$$

Dove:

2.45 ogni oggetto j deve essere assegnato ad un solo contenitore

2.46 il "peso" complessivo assegnato ad ogni contenitore i non deve superare la portata b_i

2.5.2 Rilassamento lagrangiano

2.5.2.1 (a) Rispetto ai vincoli 2.45

$$L(u) = Min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} - \overbrace{\sum_{j=1}^{n} u_j(\sum_{i=1}^{m} x_{ij} - 1)}^{-u(Ax-b)} = \tag{2.48}$$

$$L(u) = Min \sum_{i=1}^{m}(\sum_{j=1}^{n}(c_{ij} - u_j)x_{ij}) + \sum_{j=1}^{n} u_j \tag{2.49}$$

$$s.t. \sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i,\ i = 1, \ldots, m \tag{2.50}$$

$$x_{ij} \in \{0, 1\} \tag{2.51}$$

L'ottimo $L(u)$ si ottiene risolvendo m problemi di *Knapsack* per il contenitore i del tipo:

$$z_i = Min \sum_{j=1}^{n}(c_{ij} - u_j)x_{ij}$$
$$\sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i$$
$$x_{ij} \in \{0, 1\}$$

Per cui $L(u) = \sum_{i=1}^{m} z_i \sum_{j=1}^{n} u_j$

Esempio.

$m = 2$ contenitori; $n = 4$ oggetti.

$$a_{ij} = \begin{bmatrix} 5 & 7 & 4 & 2 \\ 3 & 1 & 6 & 4 \end{bmatrix} \quad c_{ij} = \begin{bmatrix} 3 & 3 & 4 & 9 \\ 2 & 6 & -9 & 3 \end{bmatrix}$$

$b = (30, 12)$

Poniamo $u^0 = 0$ e $z^* = 3$ (soluzione euristica iniziale)

$$L(u^0) = \min_x \sum_{i=1}^{m}(\sum_{j=1}^{n}(c_{ij} - u_j^0)x_{ij} + \sum_{j=1}^{n} u_j^0) \tag{2.52}$$

$$L(u^0) =; \min_x 3x_{11} + 3x_{12} + 4x_{13} + 9x_{14} + 2x_{21} + 6x_{22} - 9x_{23} + 3x_{24} + 0 \tag{2.53}$$

$$5x_{11} + 7x_{12} + 4x_{13} + 2x_{14} \leq 30 \tag{2.54}$$

$$3x_{21} + x_{22} + 6x_{23} + 4x_{24} \leq 12 \tag{2.55}$$

$$x_{11}, \ldots, x_{24} \in \{0, 1\} \tag{2.56}$$

Si decompone in due problemi: $L(u^0) = z_1 + z_2 + 0$

$$1^0 problema \begin{cases} z_1 = \ Min\ 3x_{11} + 3x_{12} + 4x_{13} + 9x14 \\ \qquad 5x_{11} + 7x_{12} + 4x_{13} + 2x_{14} \leq 30 \end{cases}$$

Soluzione ottima: $z_1 = 0;\ x^0_{11} = x^0_{12} = x^0_{13} = x^0_{14} = 0$

$$2^0 problema \begin{cases} z_1 = \ Min\ 2x_{21} + 6x_{22} - 9x_{23} + 3x24 \\ \qquad 3x_{21} + x_{22} + 6x_{23} + 4x_{24} \leq 12 \end{cases}$$

Soluzione ottima: $z_2 = -9;\ x^0_{21} = x^0_{22} = 0,\ x^0_{23} = 1,\ x^0_{24} = 0$

Quindi $L(u^0) = z_1 + z_2 + 0 = 0 - 9 + 0 = -9$

La soluzione di $L(u * 0)$ non soddisfa i vincoli 2.45; infatti:
$x = (x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{24})$
$x^0 = (\ 0,\ \ 0,\ \ \ 0,\ \ \ 0,\ \ \ 0,\ \ 0,\ \ 1,\ \ 0)$

$$Vincoli\ 2.45 \begin{cases} x_{11} + x_{21} = 1 \quad (j = 1) \\ x_{12} + x_{22} = 1 \quad (j = 2) \\ x_{13} + x_{23} = 1 \quad (j = 3) \\ x_{14} + x_{24} = 1 \quad (j = 4) \end{cases}$$

Si noti che la soluzione di $L(u^0)$ viola i vincoli 2.45 per $j = 1, 2, 4$ mentre soddisfa quello per $j = 3$.

Aggiornamento delle penalità {u}

$$u^k = u^{k-1}\alpha_k \frac{(z^* - L(u^{k-1}))}{||Ax^{k-1} - b||^2} \cdot (Ax^{k-1} - b) \tag{2.57}$$

Poniamo $k = 1$ ed $\alpha_1 = 2$; si è già assunto $z^* = 3$

$$u^k_j = u^{k-1}_j - 2 \cdot \frac{(+3 - (-9))}{\sum_{j=1}^{n}(\sum_{i=1}^{m} x_{ij} - 1)^2} \cdot (\sum_{i=1}^{m} x_{ij} - 1) \tag{2.58}$$

$$u^1_1 = u^0_1 - 2 \cdot \frac{12}{3} \cdot (-1) = 0 + 8 = 8 \tag{2.59}$$

$$u^1_2 = u^0_2 - 2 \cdot 4 \cdot (-1) = 0 + 8 = 8 \tag{2.60}$$

$$u^1_3 = u^0_3 - 2 \cdot 4 \cdot (0) = 0 + 0 = 0 \tag{2.61}$$

$$u^1_4 = u^0_4 - 2 \cdot 4 \cdot (-1) = 0 + 8 = 8 \tag{2.62}$$

Calcolo di $L(u1)$

$$L(u^1) = \ Min \sum_{i=1}^{m}(\sum_{j=1}^{n}(c_{ij} - u^1_j)x_{ij}) + \sum_{j=1}^{n} u^1_j \tag{2.63}$$

$c_{ij} = \begin{bmatrix} 3 & 3 & 4 & 9 \\ 2 & 6 & -9 & 3 \end{bmatrix}$
$u^1 = (8, 8, 0, 8)$

Come fatto in precedenza $L(u^1) = z_1 + z_2 + 24$ dove:

$$1^0 problema \begin{cases} z_1 = \ Min\ 5x_{11} - 5x_{12} + 4x_{13} + x14 \\ \quad 5x_{11} + 7x_{12} + 4x_{13} + 2x_{14} \leq 30 \\ \quad x_{11}, \ldots, x_{14} \in \{0, 1\} \end{cases}$$

Soluzione ottima: $z_1 = -10;\ x^1_{11} = x^1_{12} = 1;\ x^1_{13} = x^1_{14} = 0$

$$2^0 problema \begin{cases} z_2 = \ Min\ -6x_{21} - 2x_{22} - 9x_{23} - 5x24 \\ \quad 3x_{21} + x_{22} + 6x_{23} + 4x_{24} \leq 12 \\ \quad x_{21}, \ldots, x_{24} \in \{0, 1\} \end{cases}$$

Soluzione ottima: $z_2 = -17;\ x^1_{21} = x^1_{22} = x^1_{23} = 1;\ x^1_{24} = 0$

Quindi $L(u^1) = z_1 + z_2 + 24 = -10 - 17 + 24 = -3$

I vincoli 2.45 sono violati dalla soluzione $x1$ di $L(u^1)$.

$x = (x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{24})$
$x^1 = (\ 1,\ \ 1,\ \ 0,\ \ \ 0,\ \ \ 1,\ \ 1,\ \ 1,\ \ 0)$

$$\begin{cases} x_{11} + x_{21} = 1 \\ x_{12} + x_{22} = 1 \\ x_{13} + x_{23} = 1 \\ x_{14} + x_{24} = 1 \end{cases}$$

Tutti i vincoli, eccetto il terzo, sono violati!

Aggiornamento delle penalità

Poniamo $k = 2$ e manteniamo $\alpha_2 = 2$ in quanto $L(u^1) > L(u^0)$

$$u^k_j = u^{k-1}_j - \alpha_2 \cdot \frac{(z^* - L(u^{k-1}))}{\sum_{j=1}^{n}(\sum_{i=1}^{m} x_{ij} - 1)^2} \cdot (\sum_{i=1}^{m} x_{ij} - 1) \tag{2.64}$$

$$u^2_1 = u^1_1 - 2 \cdot \frac{6}{3} \cdot (1) = 8 - 4 = 4 \tag{2.65}$$

$$u^2_2 = u^1_2 - 2 \cdot 2 \cdot (1) = 8 - 4 = 4 \tag{2.66}$$

$$u^2_3 = u^1_3 - 2 \cdot 2 \cdot (0) = 0 - 0 = 0 \tag{2.67}$$

$$u^2_4 = u^1_4 - 2 \cdot 2 \cdot (-1) = 8 + 4 = 12 \tag{2.68}$$

Calcolo di $L(u^2)$

$$L(u^2) = \ Min \ \sum_{i=1}^{m}(\sum_{j=1}^{n}(c_{ij} - u_j^2)x_{ij}) + \sum_{j=1}^{n} u_2^1 \tag{2.69}$$

$$L(u^2) = z_1 + z_2 + 20 \tag{2.70}$$

dove z_1 e z_2 sono i valori ottimi dei problemi sequenti

$c_{ij} = \begin{bmatrix} 3 & 3 & 4 & 9 \\ 2 & 6 & -9 & 3 \end{bmatrix}$
$u^2 = (4, 4, 0, 12)$

$$1^0 problema \begin{cases} z_1 = \ Min \ -x_{11} - x_{12} + 4x_{13} - 3x14 \\ \qquad 5x_{11} + 7x_{12} + 4x_{13} + 2x_{14} \leq 30 \\ \qquad x_{11}, \ldots, x_{14} \in \{0, 1\} \end{cases}$$

Soluzione ottima: $z_1 = -5;\ x_{11}^2 = x_{12}^2 = 1,\ x_{13}^2 = 0,\ x_{14}^2 = 1$

$$2^0 problema \begin{cases} z_2 = \ Min \ -2x_{21} + 2x_{22} - 9x_{23} - 9x24 \\ \qquad 3x_{21} + x_{22} + 6x_{23} + 4x_{24} \leq 12 \\ \qquad x_{21}, \ldots, x_{24} \in \{0, 1\} \end{cases}$$

Soluzione ottima: $z_2 = -18;\ x_{21}^2 = x_{22}^2 = 0,\ x_{23}^2 = x_{24}^2 = 1$

Quindi $L(u^2) = z_1 + z_2 + 24 = -5 - 18 + 20 = -3$

L'unico vincolo violato è $x_{14} + x_{24} = 1$; per cui

$$u_j^3 = u_j^2 - \alpha_3 \cdot \frac{(z^* - L(u^2))}{\sum_{j=1}^{n}(\sum_{i=1}^{m} x_{ij} - 1)^2} \cdot (\sum_{i=1}^{m} x_{ij} - 1) \tag{2.71}$$

poniamo $\alpha_3 = \alpha_2/2 = 1$. Le nuove penalità sono:

$$u_1^3 = u_1^2 - 6 \cdot (0) = 4 \tag{2.72}$$
$$u_2^3 = u_2^2 - 6 \cdot (0) = 4 \tag{2.73}$$
$$u_3^3 = u_3^2 - 6 \cdot (0) = 0 \tag{2.74}$$
$$u_4^3 = u_4^2 - 6 \cdot (1) = 6 \tag{2.75}$$

Calcolo di $L(u^3) = z_1 + z_2 + 14$ dove

$$1^0 problema \begin{cases} z_1 = \ Min \ -x_{11} - x_{12} + 4x_{13} + 3x14 \\ \qquad 5x_{11} + 7x_{12} + 4x_{13} + 2x_{14} \leq 30 \\ \qquad x_{11}, \ldots, x_{14} \in \{0, 1\} \end{cases}$$

Soluzione ottima: $z_1 = -2;\ x_{11}^3 = x_{12}^3 = 1,\ x_{13}^3 = x_{14}^3 = 0$

$$2^0 problema \begin{cases} z_2 = \ Min \ -2x_{21} + 2x_{22} - 9x_{23} - 3x24 \\ \qquad\qquad\quad 3x_{21} + x_{22} + 6x_{23} + 4x_{24} \leq 12 \\ \qquad\qquad\quad x_{21}, \ldots, x_{24} \in \{0,1\} \end{cases}$$

Soluzione ottima: $z_2 = -12;\ x^3_{21} = x^3_{22} = 0,\ x^3_{23} = x^3_{24} = 1$

Quindi $L(u^2) = -2 - 12 + 14 = 0$.

□ Si noti che x^3 è ammissibile e quindi la soluzione è ottima.

2.5.2.2 (b) Rispetto ai vincoli 2.46

$$L(u) = Min \overbrace{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{m}\lambda_i(\sum_{j=1}^{n} a_{ij}x_{ij} - b_i)}^{cx-\lambda(Ax-b)} = \tag{2.76}$$

$$L(u) = Min \sum_{j=1}^{n}(\sum_{i=1}^{m}(c_{ij} - \lambda_i a_{ij})x_{ij}) + \sum_{i=1}^{m}\lambda_i b_i \tag{2.77}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \tag{2.78}$$

$$x_{ij} \in \{0,1\} \tag{2.79}$$

L'ottimo si ottiene ponendo, per ogni j

$$x_{i^*j} = 1 \quad \text{per} \quad c_{i^*j} - \lambda_{i^*j}a_{i^*j} = \min_i\{c_{ij} - \lambda_i a_{ij}\} \tag{2.80}$$

Ovvero, ogni oggetto j viene assegnato al contenitore i^* rispetto al quale j ha costo minimo.

Esempio *(Problema precedente con $m = 2$, $n = 4$)*

$$a_{ij} = \begin{bmatrix} 5 & 7 & 4 & 2 \\ 3 & 1 & 6 & 4 \end{bmatrix} \ c_{ij} = \begin{bmatrix} 3 & 3 & 4 & 9 \\ 2 & 6 & -9 & 3 \end{bmatrix}$$

$b = (30, 12)$

Poniamo $\lambda = (0, 0)$ e assumiamo $z^* = 3$ (come in precedenza).

$$L(\lambda^0) = Min \sum_{j=1}^{n}(\sum_{i}^{m}(c_{ij} - \lambda_i a_{ij})x_{ij} + \sum_{i=1}^{m}\lambda_i b_i) \tag{2.81}$$

$$L(\lambda^0) = 3x_{11} + 3x_{12} + 4x_{13} + 9x_{14} + 2x_{21} + 6x_{22} - 9x_{23} + 3x_{24} + 0 \tag{2.82}$$

$$\begin{cases} x_{11} + x_{21} = 1 \\ x_{12} + x_{22} = 1 \\ x_{13} + x_{23} = 1 \\ x_{14} + x_{24} = 1 \end{cases} \tag{2.83}$$

Poniamo $x_{i^*j} = 1$ dove $c_{i^*j} = \underset{i}{Min}\{c_{ij}\};\ \ \forall j$

Dal 1^o vincolo $x_{11} = 0,\ x_{21} = 1$
Dal 2^o vincolo $x_{12} = 1,\ x_{22} = 0$
Dal 3^o vincolo $x_{13} = 0,\ x_{23} = 1$
Dal 4^o vincolo $x_{14} = 0,\ x_{24} = 1$

Quindi $L(\lambda^0) = -1$

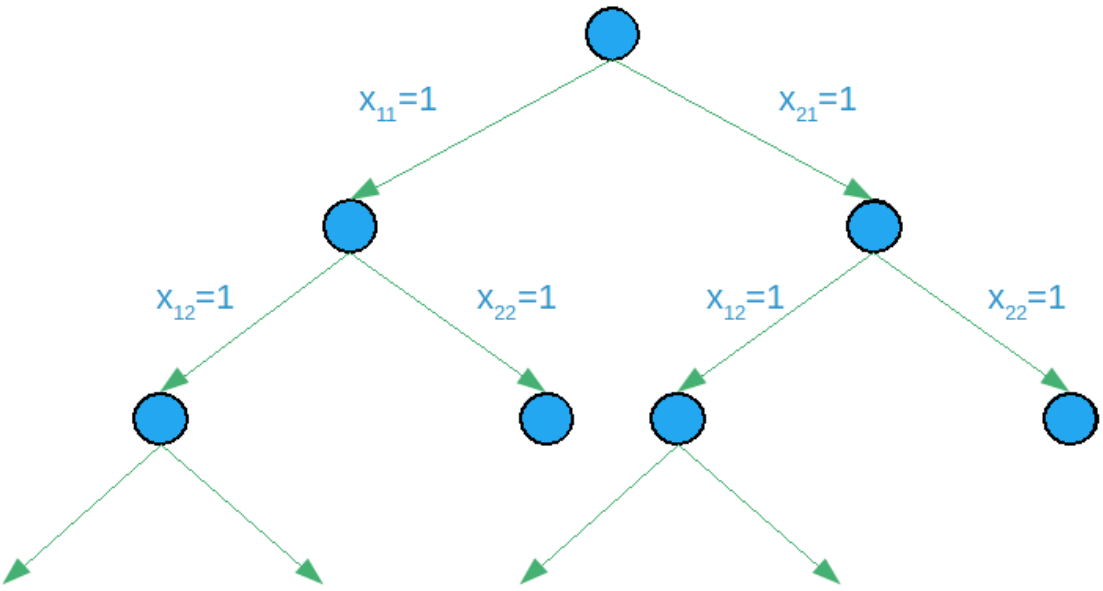
Soluzione ottenuta per $L(u^0)$
$x = (x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{24})$
$x^1 = (\ 0,\ \ 1,\ \ \ 0,\ \ \ 0,\ \ \ 1,\ \ 0,\ \ 1,\ \ 1)$

$5x_{11} + 7x_{12} + 4x_{13} + 2x_{14} \leq 30$ soddisfatto
$3x_{21} + x_{22} + 6x_{23} + 4x_{24} \leq 12$ violato!

Ulteriori iterazioni possono migliorare il lower bound...

2.5.3 Algoritmo Branch & Bound

(Ad esempio)



Ad ogni livello j viene deciso in quale contenitore inserire l'oggetto j.

Da ogni nodo vengono generati m nodi.

CAPITOLO 3

RILASSAMENTO LAGRANGIANO PER IL CALCOLO DI LOWER BOUNDS

Si consideri il seguente problema P di programmazione a numeri interi:

$$P : \begin{cases} z(P) = Min\ cx \\ \quad s.t.\ Ax \geq b \\ \quad\quad Bx \geq d \\ \quad\quad x \in \{0,1\}^n \end{cases}$$

Il valore ottimo $z(LP)$ del rilassamento lineare LP del problema P fornisce un valido lower bound, ovvero

$$z(LP) \leq z(P) \tag{3.1}$$

LP si ottiene da P sostituendo $x \in \{0,1\}^n$ con $0 \leq x \leq 1$:

$$LP : \begin{cases} z(LP) = Min\ cx \\ \quad s.t.\ Ax \geq b \\ \quad\quad Bx \geq d \\ \quad\quad 0 \leq x \leq 1 \end{cases}$$

In molti casi:

- è pribitivo risolvere LP: troppe variabili e/o vincoli;
- $z(LP)$ è troppo distante da $z(P)$ e quindi non utilizzabile in un algoritmo Branch and Bound.

3.1 Rilassamento Lagrangiano di P rispetto ai vincoli $Ax \geq b$

Viene così definito il problema RL_u che si ottiene da P rimuovendo i vincoli $Ax \geq b$ e sottraendo dalla funzione obiettivo il termine $u(Ax - b)$ dove $u \geq 0$ è il vettore dei **Moltiplicatori Lagrangiani**.

$$RL_u : \begin{cases} L(u) = Min\ cx - u(Ax - b) \\ \quad s.t.\ Bx \geq d \\ \quad x \in \{0,1\} \end{cases}$$

$L(u)$ viene detta "*Funzione Lagrangiana*"

3.1.1 Esempio

$$P : \begin{cases} z(P) = Min\ 3x_1 + 7x_2 + 10x_3 \\ \quad s.t.\ x_1 + 3x_2 + 5x_3 \geq 7 \\ \quad x_1, x_2, x_3 \in \{0,1\} \end{cases}$$

$$RL_u : \begin{cases} L(u) = Min\ x_1 + 7x_2 + 10x3 - u(x_1 + 3x_2 + 5x_3 - 7) \\ \quad s.t.\ x_1, x_2, x_3 \in \{0,1\} \end{cases}$$

3.2 Validità e importanza di RL_u

Possiamo dimostrare che $L(u) \leq z(P),\ \forall u \geq 0$ e quindi $\underset{u\geq 0}{Max}[L(u)] \leq z(P)$.
In certe condizioni la soluzione ottima di RL_u è anche la soluzione ottima di P.

3.2.1 Esempio

$$P: \begin{cases} z(P) = Min\ 2x_1 + 3x_2 + 4x_3 + 5x_4 \\ \quad s.t.\ x_1 + x_3 \geq 1 \\ \quad\quad x_1 + x_4 \geq 1 \\ \quad\quad x_2 + x_3 + x_4 \geq 1 \\ \quad\quad \forall i \in \{0,1\},\ i = 1,\ldots,4 \end{cases}$$

La soluzione ottima è $x_1 = x_2 = 1,\ x_3 = x_4 = 0$ e $z(P) = S$.
Il rilassamento lagrangiano dei tre vincoli richiede tre moltiplicatori u_1, u_2, u_3; quindi

$$\begin{aligned} (RL_u) \quad L(u)) = Min\ & 2x_1 + 3x_2 + 4x_3 + 5x_4 - u_1(x_1 + x_3 - 1) && (3.2)\\ & - u_2(x_1 + x_4 - 1) && (3.3)\\ & - u_3(x_2 + x_3 + x_4 - 1) && (3.4)\\ s.t.\ & x_i \in \{0,1\},\ i = 1,\ldots,4 && (3.5) \end{aligned}$$

ma anche

$$(RL_u)\ L(u) = Min\ (2 - u_1 - u_2)x_1 + (3 - u_3)x_2 + (4 - u_1 - u_3)x_3 + (5 - u_2 - u_3)x_4 + u_1 + u_2 + u_3 \quad (3.6)$$

$$s.t.\ x_i \in \{0,1\},\ i = 1,\ldots,4 \quad (3.7)$$

Dato u, la soluzione ottima di RL_u e, conseguentemente, il valore di $L(u)$ si ottiene ponendo

$$x_i = 0 \text{ se il coefficiente di } x_i \text{ è } \geq 0$$
$$x_i = 1 \text{ se il coefficiente di } x_i \text{ è } < 0$$

Poniamo $u_1 = 1.5$, $u_2 = 1.6$ e $u_3) = 2.2$

$$L(u) = Min\ -1.1x_1 + 0.8x_2 + 0.3x_3 + 1.2x_4 + 1.5 + 1.6 + 2.2$$

La soluzione ottima è

$$x_1 = 1,\ x_2 = x_3 = x_4 = 0$$

quindi

$$L(u) = -1.1 + 1.5 + 1.6 + 2.2 = 5.3 - 1.1 = 4.2$$

Ponendo $u_1 = 1$, $u_2 = 1$ e $u_3 = 3$

$$L(u) = Min\ 0x_1 + 0x_2 + 0x_3 + x_4 + 1 + 1 + 3$$

Una soluzione ottima è

$$x_1 = 1 = x_2 = x_3 = x_4 = 0$$

di costo $L(u) = 0 + 0 + 0 + 0 + 1 + 1 + 3 = 5 \equiv z(P)$.
Si noti che esistono soluzioni ottime alternative tutte di costo $L(u)$=5 che si ottengono ponendo $x_1 = 1$ e/o $x_2 = 1$ e/o $x_3 = 1$ e $x_4 = 0$. Fra tali soluzioni esiste quella ottima! $x_1 = x_2 = 1$ e $x_3 = x_4 = 0$

3.3 TEOREMA: Dualità Lagrangiana debole

Il valore ottimo $z(P)$ del problema

$$P: \begin{cases} z(P) = Min \;\; cx \\ \qquad s.t. \; Ax \geq b \\ \qquad\quad Bx \geq d \\ \qquad\quad x \in \{0,1\} \end{cases}$$

è maggiore o uguale al valore ottimo $L(u)$ del problema

$$RL_u: \begin{cases} L(u) = Min \; cx - u(Ax - b) \\ \qquad s.t. \; Bx \geq d \\ \qquad\quad x \in \{0,1\} \\ \qquad\quad \forall u \geq 0 \end{cases}$$

3.3.1 Dimostrazione

Sia x^* la soluzione ottima di P. Si noti che x^* è anche una soluzione ammissibile per RL_u per ogni $u \geq 0$, ma non necessariamente l'ottimo di RL_u per un dato u.
Si ha, quindi, che

$$cx^* - u(Ax^* - b) \geq L(u) \tag{3.8}$$

ma $u(Ax^* - b) \geq 0$ (poichè $u \geq 0$ e $Ax^* \geq b$ essendo per ipotesi x^* l'ottimo di P); quindi

$$cx^* \geq L(u) \text{ ovvero } z(P) \geq L(u) \qquad \square \tag{3.9}$$

3.4 Lagrangiano Duale

Dal teorema della dualità debole per cui $L(u) \leq z(P)$, $\forall u \geq 0$, si ha che l'ottimo $z(D_L)$ del seguente problema:

$$D_L \qquad z(D_L) = \underset{u \geq 0}{Max} \left[L(u) \right] \tag{3.10}$$

è un valido lower bound a $z(P)$; ovvero $z(D_L) \leq z(P)$.
Il problema D_L è detto *Lagrangiano Duale* di P.

3.5 Duality Gap

Nel caso in cui $z(D_L) < z(P)$ allora si dice che esiste un **duality gap** fra il problema P e il problema D_L.
Supponiamo che l'ottimo di D_L si ottenga risolvendo $L(\bar{u})$ per un dato $\bar{u} \geq 0$, ovvero, $z(D_L) = L(\bar{u})$.
Indichiamo con $\bar{x}$ la soluzione ottima di $RL_{\bar{u}}$ ovvero:

$$z(D_L) = L(\bar{u}) = c\bar{x} - \bar{u}(A\bar{x} - b) \tag{3.11}$$

Si consideri il caso in cui $\bar{x}$ è anche l'ottimo di P, ovvero, $z(P) = c\bar{x}$.

È evidente che $z(D_L) < z(P)$ se $\bar{u}(A\bar{x} - b) > 0$.

3.5.1 Esempio

$$P : \begin{cases} z(P) = Min\ 3_x 1 + 7x_2 + 10x_3 \\ \qquad s.t.\ x_1 + 3x_2 + 5x_3 \geq 7 \end{cases}$$

$$L(u) = Min\ 3x_1 + 7x_2 + 10x_3 - u(x_1 + 3x_2 + 5x_3 - 7)$$

$$x_1, x_2, x_3 \in \{0, 1\}$$

Per calcolare $z(D_L) \underset{u \geq 0}{Max}[L(u)]$ calcoliamo $L(u)$, $u \geq 0$

$u = 0$	$L(0) = 0$	$x = (0, 0, 0)$
$u = 1$	$L(1) = 7$	$x = (0, 0, 0)$
$u = 2$	$L(2) = 14$	$x = (0, 0, 0)$ oppure $x = (0, 0, 1)$
$u = \frac{7}{3}$	$L(\frac{7}{3}) = \frac{44}{3}$	$x = (0, 0, 1)$ oppure $x = (0, 1, 1)$
$u = 3$	$L(3) = 14$	$x = (0, 1, 1)$ oppure $x = (1, 1, 1)$
$u > 3$	$L(u) = -2u + 20$	$x = (1, 1, 1)$

Quindi $z(D_L) = \frac{44}{3}$ mentre $z(P) = 17$ e $x^* = (0, 1, 1)$ che corrisponde ad una delle soluzioni di $L(\frac{7}{3}) = \frac{44}{3}$ ma esiste un gap di dualità.

3.6 TEOREMA: Dualità Lagrangiana Forte

Sia $\bar{x}$ la soluzione ottima di $L(u)$, per un dato $\bar{x} \geq 0$.
Se $\bar{x}$, $\bar{u}$ soddisfano le seguenti condizioni:

$$A\bar{x} \geq b \tag{3.12}$$
$$\bar{u}(A\bar{x} - b) = 0 \tag{3.13}$$

allora $\bar{x}$ è la soluzione ottima di P ed inoltre $z(D_L) = L(\bar{u}) = z(P)$.

3.6.1 Dimostrazione

Dimostraimo che se $\bar{x}$, $\bar{u}$ soddisfano le 3.12 e 3.13 allora $\bar{x}$ è una soluzione ottima di P.
Poichè $\bar{x}$ soddisfa la 3.12 allora è soluzione ammissibile di P e quindi

$$c\bar{x} \geq z(P) \tag{3.14}$$

Per il teorema della dualità Lagrangiana debole si ha:

$$z(P) \geq L(u) = c\bar{x} - \underbrace{\bar{u}(A\bar{x} - b)}_{=0 \text{ per la } 3.13} \tag{3.15}$$

Quindi da 3.14 e 3.15 si ottiene

$$c\bar{x} \geq z(P) \geq c\bar{x} \text{ ovvero } z(P) = c\bar{x}. \tag{3.16}$$

Dimostriamo che se $\bar{x}$ e $\bar{u}$ soddisfano le 3.12 e 3.13 allora $z(D_L) = L(\bar{u}) = z(P)$.
Per come è definito il problema D_L si ha che:

$$\begin{aligned} z(D_L) &\geq L(\bar{u}) \\ z(P) &\geq z(D_L) \end{aligned} \tag{3.17}$$

Abbiamo dimostrato che se valgono 3.12 e 3.13 allora

$$z(P) = L(\bar{u}) = c\bar{x} \tag{3.18}$$

Quindi da 3.17 e 3.18 si ottiene

$$z(D_L) = z(P) \tag{3.19}$$

□

3.6.2 Osservazioni

- Qual è il migliore sottoinsieme di vincoli da rilassare in modo Lagrangiano?
- Come risolvere D_L: ovvero come scegliere i valori numerici di u in modo da ottenere il miglior possibile lower bound.

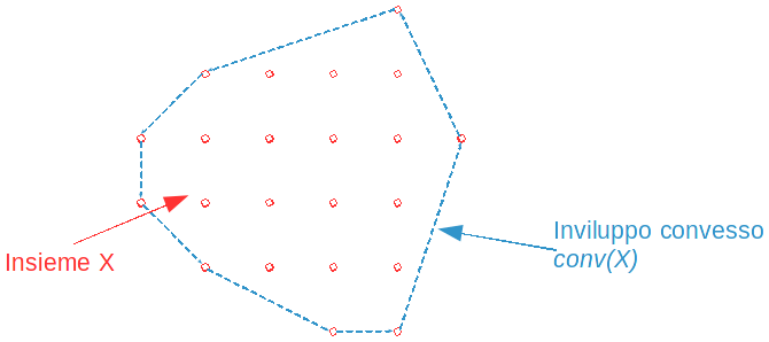
- Che relazione esiste tra $z(D_L)$ e $z(LP)$ il valore del Rilassamento Lineare di P?

3.7 Caratterizzazione del Lagrangiano Duale

Al fine di stabilire una relazione tra D_L ed il rilassamento lineare LP di P è utile riformulare D_L come un problema di programmazione lineare.

3.7.1 Definizione

Indichiamo $X = \{x : Bx \geq d,\ x \in (0,1)\}$ e con $conv(X)$ l'invilupopo convesso di tutti i punti di X (ovvero, $conv(X)$ è l'intersezione di tutti gli insieme convessi che contengono X).



Si osservi che l'ottimo del problema Lagrangiano:

$$RL_u \begin{cases} L(u) = Min\ cx - u(Ax - b) \\ \qquad\qquad s.t.\ Bx \geq d \\ \qquad\qquad\quad x \in (0,1) \end{cases}$$

corrisponde ad un punto estremo di $conv(X)$

3.7.2 TEOREMA

Il Lagrangiano Duale D_L corrisponde al seguente problema di programmazione lineare.

$$D_L \begin{cases} z(D_L) = Min\ cx \\ \qquad\qquad s.t.\ Ax \geq d \\ \qquad\qquad\quad x \in conv(X) \end{cases}$$

dove $X = \{x : Bx \geq d,\ x \in (0,1)\}$ e $conv(X)$ è l'inviluppo convesso di X.

3.7.2.1 Dimostrazione

Si ricordi che:

$$D_L \qquad z(D_L) = \underset{u \geq 0}{Max}\ [L(u)]$$

e, per come è stato definite X, il problema RL_u diviene

$$RL_u \quad L(u) = \underset{x \in X}{Min}\ (cx - u(Ax - b))$$

Quindi, il problema D_L può essere scritto come:

$$D_L \quad z(D_L) = \underset{u \geq 0}{Max}\ \underbrace{[\underset{x \in X}{Min})\ (cx - u(Ax - b))]}_{L(u)}$$

o anche

$$D_L \quad z(D_L) = \underset{u \geq 0}{Max}\ [\underset{x \in conv(X)}{Min})\ (cx - u(Ax - b))]$$

poichè $L(u)$ raggiunge l'ottimo in un punto estremo di $conv(X)$.
Indichiamo con $x^i,\ i = 1, \ldots, t$ i punti estremi di $conv(X)$. Il problema D_L può essere scritto come:

$$D_L \quad z(D_L) = \underset{u \geq 0}{Max}\ [\underset{1 \leq i \geq t}{Min}\ (cx^i - u(Ax^i - b))]$$

quest'ultimo problema può essere riformulato mediante la programmazione lineare come segue:

$$D_L \begin{cases} z(D_L) = Max\ v \\ \qquad s.t.\ v \leq cx^i - u(Ax^i - b),\ i = 1, \ldots, t \\ \qquad\quad \text{v qualsiasi} \\ \qquad\quad u \geq 0 \end{cases}$$

Il duale di questo problema è il seguente

$$DD_L \begin{cases} z(D_L) = Min\ \sum_{i=1}^{t} \lambda_i (cx^i) \\ \qquad s.t.\ \sum_{i=1}^{t} \lambda_i = 1 \\ \qquad\quad \sum_{i=1}^{t} \lambda_i (Ax^i - b) \geq 0 \\ \qquad\quad \lambda_i \geq 0,\ i = 1, \ldots, t \end{cases}$$

Si noti che $\sum_{i=1}^{t} \lambda_i (cx^i) = c(\sum_{i=1}^{t} \lambda_i x^i)$ ed inoltre $\sum_{i=1}^{t} = \lambda_i (Ax^i - b) = A(\sum_{i=1}^{t} \lambda_i x^i) - b(\sum_{i=1}^{t} \lambda_i)$.
Quindi DD_L può essere riscritto come

$$DD_L \begin{cases} z(D_L) = Min\ c(\sum_{i=1}^{t} \lambda_i x^i) & (3.20) \\ s.t.\ \sum_{i=1}^{t} \lambda_i = 1 & (3.21) \\ A(\sum_{i=1}^{t} \lambda_i x^i) \geq b(\sum_{i=1}^{t} \lambda_i) & (3.22) \\ \lambda_i \geq 0,\ i = 1, \ldots, t & (3.23) \end{cases}$$

Si osservi che, per ognu t-pla $\lambda_i, \ldots, \lambda_t$ che soddisfa i vincoli 3.21 e 3.23, il punto $x = \sum_{i=1}^{t} \lambda_i x^i$ appartiene a $conv(X)$. Quindi il problema DD_L può essere riscritto come

$$DD_L \begin{cases} z(D_L) = Min\ cx \\ s.t.\ Ax \geq b \\ x \in conv(X) \end{cases} \quad (3.24)$$

□

3.8 Lagrangiano Duale e Rilassamento Lineare

3.8.1 TEOREMA

$$z(D_L) \geq z(LP) \tag{3.25}$$

3.8.2 Dimostrazione

Il rilassamento lineare LP è definito come

$$LP \begin{cases} z(LP) = Min\ cx \\ \qquad s.t.\ Ax \geq b \\ \qquad\quad Bx \geq d \\ \qquad\quad 0 \leq x \leq 1 \end{cases} \tag{3.26}$$

Definiamo $\bar{X} = \{x : Bx \geq d,\ 0 \leq x \leq 1\}$, quindi

$$LP \begin{cases} z(LP) = Min\ cx \\ \qquad s.t.\ Ax \geq b \\ \qquad\quad x \in \bar{X} \end{cases} \tag{3.27}$$

Per come abbiamo definito $\bar{X}$ è facile osservare che:

$$conv(X) \subseteq \bar{X} \tag{3.28}$$

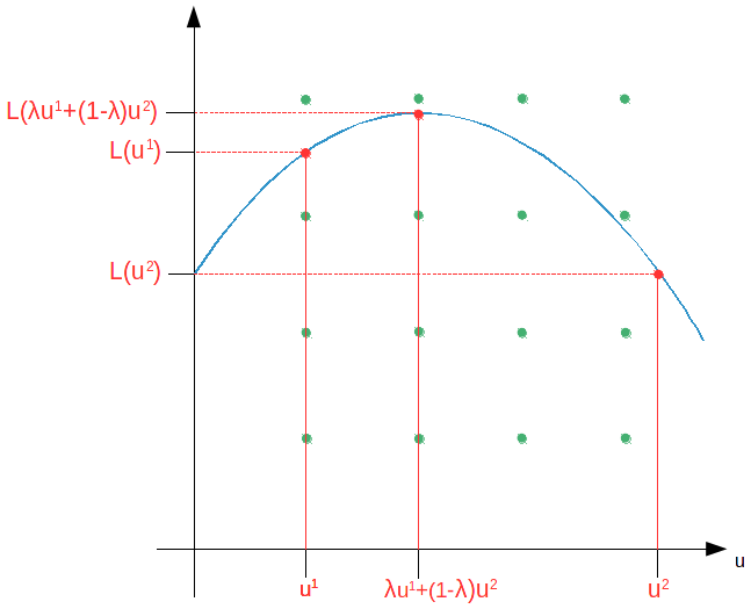
e poichè abbiamo dimostrato che

$$D_L \begin{cases} z(D_L) = Min\ cx \\ \qquad s.t.\ Ax \geq b \\ \qquad\quad x \in conv(X) \end{cases} \tag{3.29}$$

si ha che $z(D_L) \geq z(LP)$.
□

3.8.3 TEOREMA: $L(u)$ è concava

La funzione lagrangiana $L(u)$ è concava, ovvero $L(\lambda u^1 + (1-\lambda)u^2) \geq \lambda L(u^1) + (1-\lambda)L(u^2)$, $\lambda \in [0,1]$



3.8.3.1 Dimostrazione

Siano $u^1, u^2 \geq 0$ e $u^0 = \lambda u^1 + (1-\lambda)u^2$ con $\lambda in [0,1]$.
Indichiamo con x^0 la soluzione ottima di RL_{u^0}:

$$L(u^0) = cx^0 - u^0(Ax^0 - b) \tag{3.30}$$

x^0 è soluzione ammissibile di RL_{u^1} e RL_{u^2} quindi

$$L(u^1) \leq cx^0 - u^1(Ax^0 - b) \tag{3.31}$$
$$L(u^2) \geq cx^0 - u^2(Ax^0 - b) \tag{3.32}$$

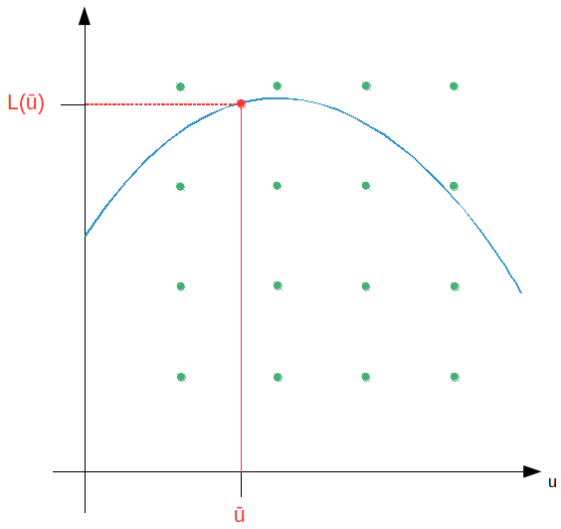
Moltiplicando la 3.31 per λ, la 3.32 e sommando:

$$\lambda L(u^1) + (1-\lambda)L(u^2) \leq cx^0 - \underbrace{(\lambda u^1 + (1-\lambda)u^2)}_{u^0}(Ax^0 - b) = L(u^0) \tag{3.33}$$

3.9 Subgradiente di $L(u)$

Un vettore è detto subgradiente di $L(u)$ in $\bar{u}$ se soddisfa

$$L(u) \geq L(\bar{u}) + y(u - \bar{u}) \tag{3.34}$$



Come calcolare y?
Sia $\bar{x}$ tale che

$$L(\bar{u}) = c\bar{x} - \bar{u}(A\bar{x} - b) \tag{3.35}$$

Per ogni $u \geq 0$ si ha

$$L(u) \leq c\bar{x} - u(A\bar{x} - b) \tag{3.36}$$

Sottraendo dalla 3.36 la 3.35 si ottiene

$$L(u) = L(\bar{u}) \geq -(A\bar{x} - b)(u - \bar{u}) \tag{3.37}$$

ma anche

$$L(u) \leq L(\bar{u}) - (A\bar{x} - b)(u - \bar{u}) \tag{3.38}$$

ne segue che $y = -(A\bar{x} - b)$ è un subgradiente di $L(u)$ in $\bar{u}$

3.9.1 Metodo del subgradiente

Metodo iterativo per risolvere il lagrangiano duale

$$D_L \left\{ z(D_L) = \underset{u \geq 0}{Max}[L(u)] \right.$$

Il metodo genera una sequenza finita di punti $(u^1, u^2, \ldots, u^k)$ e, quindi, calcola

$$z(D_L) = \underset{u \in \{u^1, u^2, \ldots, u^k\}}{Max}[L(u)]$$

3.9.1.1 Generazione di u^r in funzione di u^{r-1}

Sia x^{r-1} tale che $L(u^{r-1}) = cx^{r-1} - u^{r-1}(Ax^{r-1} - b)$.
Abbiamo dimostrato che

$$L(u^r) \leq L(u^{r-1}) - (Ax^{r-1} - b)(u^r - u^{r-1}) \tag{3.39}$$

se vogliamo che $L(u^r)$ possa essere maggiore di $L(u^{r-1})$ è necessario che

$$-(Ax^{r-1} - b)(u^r - u^{r-1}) > 0 \tag{3.40}$$

Si noti che una scelta di u^r che verifichi la suddetta condizione non è sufficiente per garantire che $L(u^r) > L(u^{r-1})$.
Come definire u^r affinchè 3.40 sia verificata?
Supponiamo che A abbia m righe e quindi $u = (u_1, \ldots, u_m)$ indicando con $a^i \geq b_i$ la i-esima disequazione di $Ax \geq b$; la condizione 3.40 può essere scritta come

$$-\sum_{i=1}^{m}(a_i^{r-1} - b_i)(u_i^r - u_i^{r-1}) > 0$$

Per soddisfare 3.9.1.1 è sufficiente determinare ogni u_i^r in modo che

$$-(a^i x^{r-1} - b_i)(u_i^r - u_i^{r-1}) > 0, \ \forall i = 1, \ldots, m \tag{3.41}$$

Da cui seguono i sequenti casi:

- $a^i x^{r-1}$: x^{r-1} viola il vincolo i-esimo
  definisci $u_i^r > u_i^{r-1}$
- $a^i x^{r-1} > b_i$: x^{r-1} soddisfa il vincolo i-esimo
  definisci $u_i^r < u_r^{r-1}$ ma imponi $u_i^r \geq 0$
- $a^i x^{r-1} = b_i$: x^{r-1} satura il vincolo i-esimo
  u_i^r qualsiasi (è buona norma $u_i^r = u_i^{r-1}$!)

1. Inizializza $u^1 = 0$ e poni $r = 1$ e $LB = -\infty$
2. Risolvi:
$$L(u^r) \begin{cases} Min\ cx - u^r(Ax - b) \\ s.t.; Bx \geq d \\ x \in (0, 1) \end{cases}$$
Sia x^r la soluzione ottima
Se $L(u^r) > LB$ allora poni $LB = L(u^r)$ e $u^* = u^r$
Se $Ax^r \geq b$ e $u^r(Ax^r - b) = 0$ allora x^r è soluzione ottima di P: STOP
3. Definisci i moltiplicatori di u^{r+1}
$$u_i^{r+1} = Max[0,\ u_i^r - \alpha \cdot \frac{z_{UB} - L(u^r)}{\sum_{i=1}^m \widetilde{y_i^2}} \cdot \widetilde{y_i}],\ \forall i$$

dove $\widetilde{y_i} = a^i x^r - b_i$ e α è una costante $(0 < \alpha \leq 2)$.
Poni $r \leftarrow r + 1$ e ritorna allo step *2.*

4. Il metodo potrebbe non arrestarsi: è quindi necessario imporre un numero massimo di iterazioni.
5. È opportuno diminuire il valore di α ($\alpha \leftarrow \alpha/2$) se per δ iterazioni consecutive $L(u) \leq LB$
6. I valori di α e δ vanno determinati sperimentalmente: tipicamente $\alpha = 2$ e $\delta = 30$.

3.9.2 Vincoli Misti

$$z(P) \begin{cases} Min\ cx \\ \quad A_1 x \geq b_1 \quad m_1 \text{ righe e } u^1 \geq 0 \\ \quad A_2 x = b_2 \quad m_2 \text{ righe e } u^2 \in \mathbb{R}^{m_2} \\ \quad A_3 x \leq b_3 \quad m_3 \text{ righe e } u^3 \leq 0 \\ \quad Bx \geq d \\ \quad\ x \in \{0,1\}^m \qquad\qquad u = (u^1, u^2, u^3) \end{cases}$$

$$L(u) \begin{cases} Min\ cx - u^1(A_1 x - b_1) - u^2(A_2 x - b_2) - u^3(A_3 x - b_3) \\ \quad s.t.\ Bx \geq d \\ \quad\ x \in \{0,1\}^m \end{cases}$$

ma anche:

$$L(u) \begin{cases} Min\ (c - u^1 A_1 - u^2 A_2 - u^3 A_3)x + u^1 b_1 + u^2 b_2 + u^3 b_3 \\ \quad s.t.\ Bx \geq d \\ \quad\ x \in \{0,1\} \end{cases}$$

3.9.3 Subgradiente per vincoli mist

Ad una generica iterazione.
Sia $\bar{x}$ la soluzione ottima di $L(u)$.
Calcola $y^1 = A_1\bar{x} - b_1$, $y^2 = A_2\bar{x} - b_2$, $y^3 = A_3\bar{x} - b_3$.
Poni $y = (y^1, y^2, y^3)$ e $t = \alpha \dfrac{z_{UB} - L(u)}{\sum_{i=1}^{m} y_1^2}$

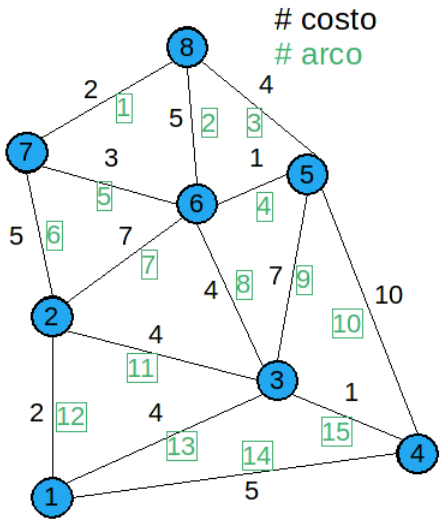
$$u_i^1 \leftarrow max[0,\ u_i^1 - ty_i^1],\ i = 1, \ldots, m_1$$
$$u_i^2 \leftarrow u_i^2 - ty_i^2,\ i = 1, \ldots, m_2$$
$$u_i^3 \leftarrow min[0,\ u_i^3 - ty_i^3],\ i = 1, \ldots, m_3$$

3.10 Traveling Salesman Problem

3.10.1 Costi Simmetrici

n vertici, m archi.
$G = (N, A)$: grafo non-orientato.
$N = \{1, \ldots, n\}$ insieme dei vertici.
$A = \{1, \ldots, m\}$ insieme degli archi.
Costi simmetrici: $c_{ij} = c_{ji}\ \forall\ ij$ Indichiamo con c_l il costo dell'arco $l \in A$.
Per ogni arco $l \in A$ siano (α_l, β_l) i due vertici terminali.
Inoltre sia $B_i \subset A$ l'insieme degli archi incidenti nel vertice $i \in N$.

3.10.1.1 Esempio



$n = 8$ vertici
$m = 15$ archi
04 arco 14 $\alpha_{14} = 1$, $\beta_{14} = 4$
$B_4 = \{10, 14, 15\}$
$B_3 = \{8, 9, 15, 13, 11\}$

3.10.2 Fomulazione Matematica (TSP Simmetrico)

$x_l = 1$, se l'arco l è nella soluzione ottima
$x_l = 0$, altrimenti.

$$\begin{cases} Min\ z = \sum_{l=1}^{m} c_l x_l & (3.42) \\ \sum_{l \in B_i} x_l = 2;\ i = 1, \ldots, n & (3.43) \\ \sum_{l \in K_t} x_l \geq 1;\ \forall K_t = (S_t,\ N \setminus S_t) \quad S_t \subset N,\ S_t \neq \emptyset,\ |S_t| \geq 2 & (3.44) \\ x_l \in \{0, 1\};\ l = 1, \ldots, m & (3.45) \end{cases}$$

3.10.2.1 1^o Rilassamento Lagrangiano (SST)

I vincoli 3.43 vengono portati nella fuzione obiettivo e sostituiti nella formulazione con il "surrogato" $\sum_{i=1}^{n} = (\sum_{l \in B_i} x_l) = 2n$.

$$L(\lambda) = min \sum_{l=1}^{m} c_l x_l - \sum_{i=1}^{n} \lambda_i (\sum_{l \in B_i} x_l - 2) \tag{3.46}$$

$$\sum_{l=1}^{m} x_l = n \tag{3.47}$$

$$\sum_{l \in K_t} x_l \geq 1; \;\; \forall K_t = (S_t, \; N \setminus S_t) \quad S_t \subset N, \; S_t \neq \emptyset \tag{3.48}$$

$$x_l \in \{0,1\} \;\; l = 1, \ldots, m \tag{3.49}$$

$$L(\lambda) = min \sum_{l=1}^{m} c_l x_l - \sum_{i=1}^{n} \lambda_i (\sum_{l \in B_i} x_l - 2)$$

$$L(\lambda) = min \sum_{l=1}^{m} c_l x_l + 2 \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n} \sum_{l \in B_i} \lambda_i x_l$$

Si noti che l'arco l ha come vertici terminali α_l, β_l e quindi compare per $i = \alpha_l$ ed $i = \beta_l$.

$$\sum_{i=1}^{n} \sum_{l \in B_i} \lambda_i x_l = \sum_{l=1}^{m} (\lambda + \lambda) x_l$$

$$L(\lambda) = min \sum_{l=1}^{m} (\underbrace{c_l - \lambda_{\alpha_l} - \lambda_{\beta_l}}_{c'_l}) x_l + 2 \sum_{i=1}^{n} \lambda_i$$

$$\sum_{l=1}^{m} x_l = n$$

$$\sum_{l \in K_t} x_l \geq 1; \quad \forall K_t \equiv (S_t, \; N \setminus S_t) \quad S_t \subset N, \; S_t \neq \emptyset$$

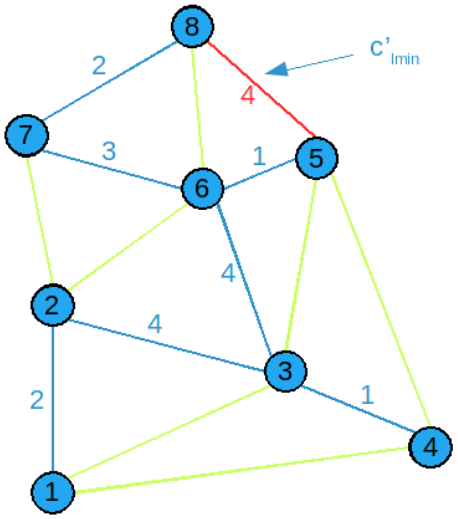
$$x_l \in \{0,1\}$$

La soluzione ottima si ottiene calcolando l'albero di costo minimo, usando i costi $(c_l - \lambda_{\alpha_l} - \lambda_{\beta_l})$, detto $v(SST)$ tale costo si ha che

$$L(\lambda) = v(SST) + c'_{l_{min}} + 2 \sum_{i=1}^{n} \lambda_i \tag{3.50}$$

dove $c'_{l_{min}} = min\{(c_l - \lambda_{\alpha_l} - \lambda_{\beta_l}) : \; l \notin SST\}$

3.10.3 Calcolo di $L(\lambda^0)$ per $\lambda^0 = 0$



L'albero di costo minimo è $SST = \{(3,4),(6,5),(1,2),(7,8),(7,6),(2,3),(3,6)\}$ mentre l'arco minimo è $(5,8)$ e $c'_{l_{min}} = 4$

$$L(\lambda^0) = v(SST) + c'_{l_{min}} = 17 + 4 = 21 \tag{3.51}$$

3.10.4 Calcolo Penalità Lagrangiane

Poniamo $d_i = \sum_{l \in B_i} x_l$ per cui il vincolo

$$\sum_{l \in B_i} x_l = 2; \quad i = 1, \ldots, n \tag{3.52}$$

diviene

$$d_i = 2; \quad i = 1, \ldots, n \tag{3.53}$$

Nella soluzione prodotta per $\lambda^0 = 0$

$$d^0 = (1,2,3,1,2,3,2,2) \tag{3.54}$$

$$\boxed{\begin{array}{l} \lambda^k = \lambda^{k-1} - t_k(Ax^{k-1} - b) \\ \text{dove} \\ t_k = \lambda_k \cdot \dfrac{(z^* - L(\lambda^{k-1}))}{||Ax^{k-1} - b||^2} \end{array}}$$

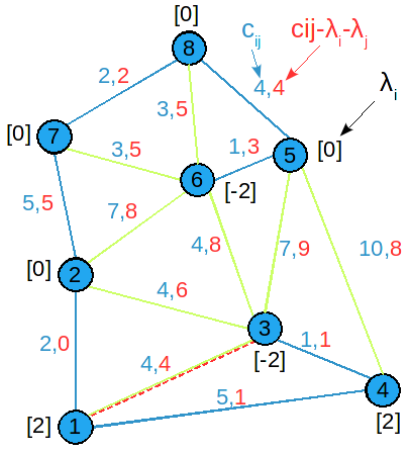
3.10.4.1 Prima iterazione: $k = 1$ e $\lambda^0 = 0$, $\alpha_1 = 2$

$$t_1 = 2 \cdot \frac{(25 - 21)}{\sum_i (d_i^0 - 2)^2} = 2 \cdot \frac{4}{4} = 2$$
$$\lambda_i^1 = 0 - 2(d_i^0 - 2) \quad i = 1, \ldots, n$$

Quindi

$$\lambda_i^1 > 0 \qquad \text{se } d_i^0 < 2$$
$$\lambda_i^1 = 0 \qquad \text{se } d_i^0 = 2$$
$$\lambda_i^1 < 0 \qquad \text{se } d_i^0 > 2$$
$$\lambda^1 = (+2, 0, -2, +2, 0, -2, 0, 0)$$

3.10.4.2 Calcolo di $L(\lambda^1)$



Albero di costo minimo SST usando i costi $\{c_{ij}, \lambda_i - \lambda_j\}$

$$SST = (1,2), (1,4), (3,4), (7,8), (5,6), (5,8), (2,7)$$
$$v(SST) = 16$$

Nella soluzione prodotta per $\lambda^1 = (2, 0, -2, 2, 0, -2, 0, 0)$ si ha

$$d^1 = (3, 2, 2, 2, 2, 1, 2, 2)$$

3.10.4.3 Nuova iterazione per $k \leftarrow k + 1$ ossia $k = 2$

$$\lambda_i^2 = \lambda_i^1 - \alpha_2 \cdot \frac{(z^* - L(\lambda^1))}{\sum_i (d_i^1 - 2)^2} \cdot; \quad i = 1, \ldots, n$$

dove $\alpha_2 = \alpha_1/2 = 1$

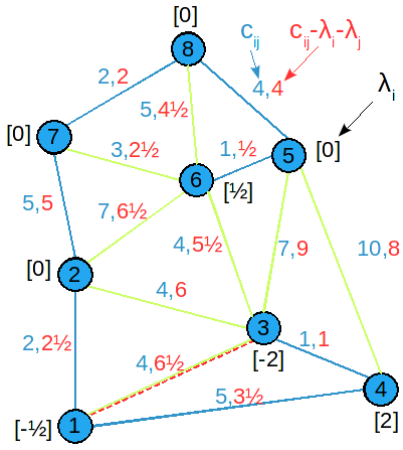
$$\lambda_i^2 = \lambda_i^1 - 1 \cdot \frac{5}{2} \cdot (d_i^1 - 2)$$

Quindi

$$\lambda_1^2 = 2 - \frac{5}{2} - 1 = -\frac{1}{2}$$
$$\lambda_i^2 = \lambda_i^1, \quad i = 2,3,4,5$$
$$\lambda_6^2 = -2 - \frac{5}{2} \cdot (-1) = -2 + \frac{5}{2} = \frac{1}{2}$$
$$\lambda_7^2 = \lambda_7^1,\ \lambda_8^2 = \lambda_8^1$$

$$\lambda_2 = (-\frac{1}{2}, 0, -2, +2, 0, \frac{1}{2}, 0, 0)$$

3.10.4.4 Calcolo di $L(\lambda^2)$



Albero a costo minimo SST usando i costi $\{c_{ij} - \lambda_i - \lambda_j\}$

$$SST = \{(5,6),(3,4),(7,8),(1,2),(6,7),(1,4),(2,7)\}$$
$$v(SST) = 17$$

Arco a costo minimo $\notin SST$ è $(5,8)$ e $c'_{l_{min}} = 4$

$$L(\lambda^2) = v(SST) + c_{l_{min}} + 2\sum_i \lambda_i = 17 + 4 + 0 = 21$$

Nella soluzione per $\lambda^2 = (-\frac{1}{2}, 0, -1, +2, 0, \frac{1}{2}, 0, 0)$ si ha che

$$d^2 = (2,2,1,2,2,2,3,2)$$

3.10.4.5 Nuova iterazione per $k = 3$, $\alpha_3 = \frac{1}{2}$ $(\alpha_{3=\alpha_2/2})$

$$\lambda_i^3 = \lambda_i^2 - \alpha_3 \cdot \frac{(z^* - L(\lambda^2))}{\sum_i (d_i^2 - 2)^2} \cdot (d_i^2 - 2)$$

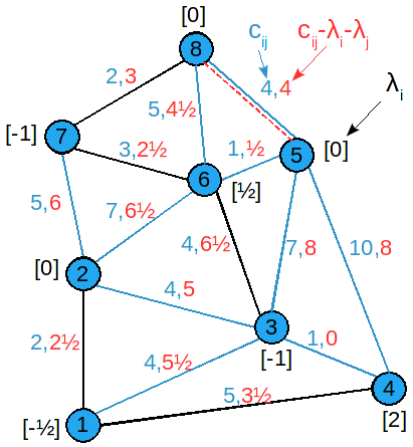
ovvero

$$\lambda_i^3 = \lambda_i^2 - \frac{1}{2} \cdot \frac{4}{2} \cdot (d_i^2 - 2)$$

Quindi

$$\lambda_3^3 = -2 - 1 \cdot (-1) = -1$$
$$\lambda_7^3 = 0 - 1 \cdot 1 = -1$$
$$\text{altrimenti } \lambda_i^3 = \lambda_i^2, \quad \forall i \neq= 3, 7$$
$$\lambda^3 = (-\frac{1}{2}, 0, -1, +2, 0, \frac{1}{2}, -1, 0)$$

3.10.4.6 Calcolo di $L(\lambda^3)$



$$SST = \{(3,4), (5,6), (1,2), (7,8), (1,4), (7,6), (3,6)\}$$
$$v(SST) = 17.5$$

Arco a costo minimo $\notin SST$ è $(5,8)$ e $c'_{l_{min}} = 4$

$$L(\lambda^2) = v(SST) + c_{l_{min}} + 2\sum_i \lambda_i = 17.5 + 4 = 21.5$$

$$d^3 = (2, 1, 2, 2, 2, 3, 2, 2)$$

3.10.4.7 Nuova iterazione per $k = 4$, $\alpha_4 = \frac{1}{4}$

$$\lambda_i^4 = \lambda_i^3 - \frac{1}{4} \cdot \frac{3.5}{2}(d_i^3 - 2)$$

per semplificare si usi:

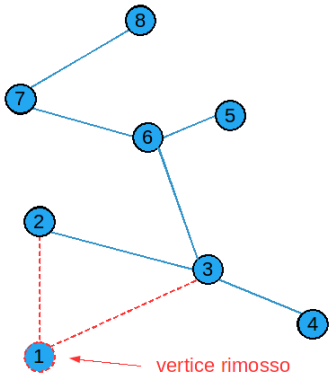
$$\lambda_i^4 = \lambda_i^3 - \frac{1}{2} \cdot (d_i^3 - 2)$$

continuare per esercizio...

3.10.5 Rilassamento 1-TREE

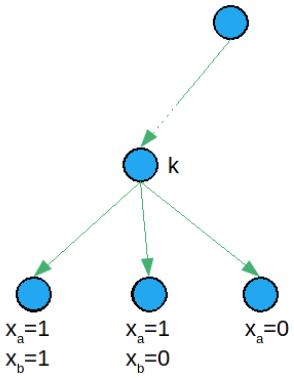
Held and Karp

- Si rimuova dal grafo un vertice;
- Si calcolo lo shortest spanning tree (SST) sul grafo rimanente;
- Si aggiungano i due links di costo minimo che incidono sul vertice rimosso;
- Il lower bound è dato dalla somma del costo dello SST e dei costi dei due link aggiunti



3.10.6 Regola di branching TSP simmetrico

Al nodo K dell'albero decisionale: scegli un vertice i il cui grado sia maggiore a 2 è due links a e b che nello SST incidono su i e liberi.
Genera 3 nodi come segue

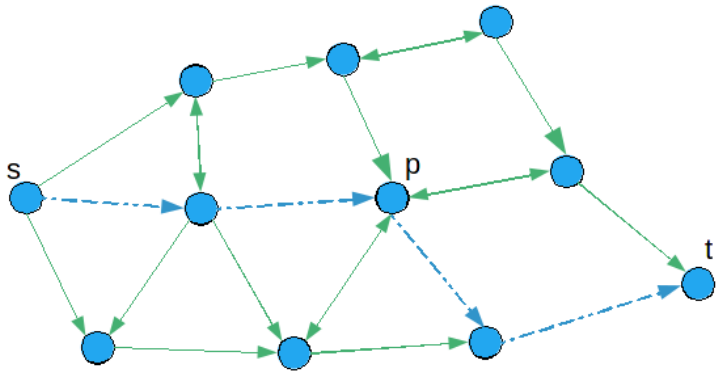

k
x_a=1
x_b=1
x_a=1
x_b=0
x_a=0

CAPITOLO 4

PROGRAMMAZIONE DINAMICA

4.1 Motivazioni

Si consideri il problema del cammino minimo di un grafo da s a t.



4.1.1 Osservazione 1

Se il cammino minimo (s,t) passa per il vertice p, allora, i sottocammini (s,p) e (p,t) sono i cammini minimi di s a p e da p a t, rispettivamente.

4.1.1.1 Dimostrazione

Se per assurdo uno dei due cammini (s,p) e (p,t) non fosse un cammino minimo, allora (s,t) non potrebbe essere il cammino minimo da s a t.

4.1.2 Osservazione 2

Indichiamo con $d(v)$ il costo del cammino minimo da s a v e dall'oservazione 1 si ottiene che se conosciamo il costo di $d(i)$ del cammino minimo da s ad ogni predecessore $i \in \Gamma^{-1}(v)$ del vertice v allora:

$$d(v) = \min_{i \in \Gamma^1(v)} [d(i) + c_{iv}] \tag{4.1}$$

Tale osservazione non è sufficiente per stabilire un algoritmo per calcolare il cammino minimo in un qualsiasi tipo di grafo.
È sufficiente per grafi aciclici.

4.1.3 Osservazione 3

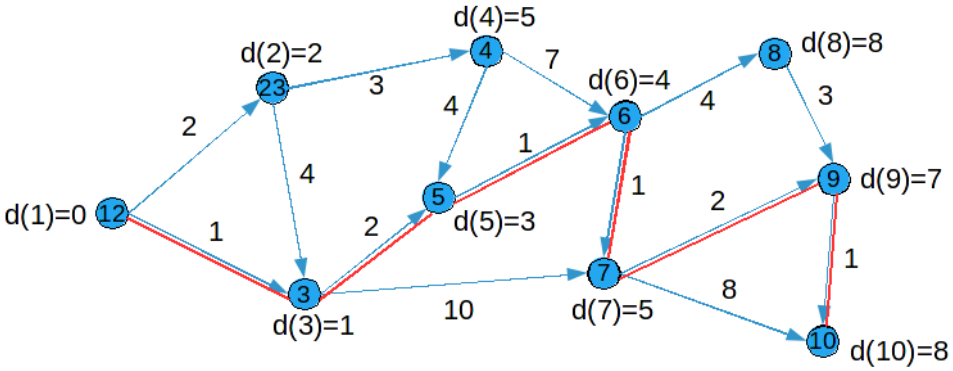
Sia $G = (V, A)$ un grafo orientato aciclico di $n = |V|$ vertici e $m = |A|$ archi i cui vertici sono ordinati così che $i < j \ \forall (i,j) \in A$ (ovvero $i < v$, $\forall i \in \Gamma^{-1}(v)$).
Supponiamo siano noti i costi $d(1), d(2), \ldots, d(k)$ dei cammini minimi dal vertice 1 ai vertici $(1, 2, \ldots, k) \subset V$.

Allora, per come è stato definite il grafo aciclico G, sono noti i costi $d(i)$, $\forall i \in \Gamma_{k+1}^{-1}$ e quindi

$$d_{k+1} = \min_{i \in \Gamma_{k+1}^{-1}} [d(i) + c_{i,k+1}] \tag{4.2}$$

4.2 Algoritmo

1. Poni $d(1) = 0$, $d(2) = \cdots = d(n) = \infty$;
2. Per $v = 2, \ldots, n$ calcola $d(v) = \min_{i \in \Gamma^{-1}(v)} [d(i) + c_{iv}]$.



4.3 Algoritmo Forward (grafi aciclici)

1. Poni $d(1) = 0$ e $d(j) = \infty$, $\forall j \in V \setminus \{1\}$; sia $v = 1$;
2. Per ogni $j \in \Gamma(v)$ aggiorna:

$$d(j) = min[d(j), d(v) + c_{vj}] \tag{4.3}$$

3. Se $v = n - 1$ STOP, altrimenti poni $v = v + 1$ e ritorna allo step 2.

Caso generale: grafi orientati con cicli e costi degli archi c_{ij} non-negativi.
Non si può applicare direttamente la ricorsione poichè è necessario imporre un ordine con cui calcolare la 4.1.

4.4 Algoritmo di Bellman

Sia $D(k, j)$ il costo del cammino minimo da s a j contenente al più k archi.
Si hanno due casi:

1. Il cammino minimo di costo $D(k, j)$ contiene al più $k - 1$ archi, quindi

$$D(k, j) = D(k - 1, j) \tag{4.4}$$

2. Il cammino minimo di costo $D(k, j)$ contiene k archi, quindi

$$D(k, j) = \min_{i \in \Gamma^{-1}(j)} [D(k - 1, i), +c_{ij}] \tag{4.5}$$

Partendo dalla 4.5 si a che la ricorsione è:

$$D(k, j) = min[D(k - 1, j), \min_{i \in \Gamma^{-1}(j)} [D(k - 1, i) + c_{ij}]] \tag{4.6}$$

La ricorsione 4.6 impone un ordine implicito di calcolo:
prima $D(1, j),\ \forall j \in V$, poi $D(2, j)$, $D(3, j)$, $\ldots$, $D(n - 1, j)$.

4.4.1 Schema dell'algoritmo cammini minimi da 1 ad ogni $j \in V$

1. Definisci

$$D(1, 1) = 0$$
$$D(1, j) \begin{cases} c_{1j}, & \forall j \in \Gamma(1) \\ \infty, & \forall j \in V \setminus \Gamma(1) \end{cases}$$

2. Per $k = 2, \ldots, n - 1$ calcola

$$D(k, j) = min[D(k - 1, j), \min_{i \in \Gamma^{-1}(j)} [D(k - 1, i) + c_{ij}]], \quad \forall j \in V$$

3. Poni $d(j) = D(n - 1, j), \quad \forall j \in V$

4.5 Knapsack 0-1

Sono dati n items ed un *knapsack* di capacità b.
L'item j ha peso a_j e profitto c_j.
Si vuole riempire il knapsack massimizzando il profitto complessivo degli items caricati.
Assumiamo che i coefficienti $\{a_j\}$ e b siano interi positivi

$$KP \begin{cases} z = max \sum_{j=1}^{n} c_j x_j & (4.7) \\ \sum_{j=1}^{n} a_j b_j \leq b & (4.8) \\ x_j \in \{0,1\}, \quad j = 1, \ldots, n & (4.9) \end{cases}$$

4.5.1 Esempio

$$\begin{cases} z = max 10x_1 + 7x_2 + 25x_3 + 24x_4 \\ \qquad\qquad 2x_1 + 1x_2 + 6x_3 + 5x_4 \leq 7 \\ \qquad\qquad x_j \in \{0,1\}, \quad j = 1, \ldots, 4 \end{cases}$$

La soluzione ottima è $x_i^* = 1,\ x_2^* = 0, x_3^* = 0, x_4^* = 1$.

4.5.2 Osservazione 1

Se $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ è una soluzione ottima di KP allora $(x_1^*, x_2^*, \ldots, x_k^*)$ con $k \leq n$ è una soluzione ottima del seguente sottoproblema $KP_k(q)$.

$$KP_k(q) \begin{cases} f_k(q) = max \sum_{j=1}^{k} c_j x_j & (4.10) \\ \sum_{j=1}^{k} a_j x_j \leq q & (4.11) \\ x_j \in \{0,1\}, \quad j = 1, \ldots, k & (4.12) \\ \text{dove } q = \sum_{j=1}^{k} a_j x_j^* & (4.13) \end{cases}$$

4.5.2.1 Esempio

$(x_1^* = 1, x_2^* = 0, x_3^* = 0)$ deve essere la soluzione ottima del seguente problema

$$KP_3(q) \begin{cases} f_3(q) = max 10x_1 + 7x_2 + 25x_3 \\ 2x_1 + 1x_2 + 6x_3 \leq q = 2 \\ x_1, x_2, x_3 \in \{0,1\} \text{dove } q = 2x_1^* + 1x_2^* + 6x_3^* = 2 \end{cases} \tag{4.14}$$

La dimostrazione dell'osservazionee 1 è ovvia!
Si consideri la famiglia degli $n(b+1)$ sottoproblemi

$$\{KP_k(q) : 1 \leq k \leq n,\ 0 \leq q \leq b\} \quad q \text{ intero} \tag{4.15}$$

dove q è detto stato e k è detto stadio.
Il problema originario KP è un membro di tale famiglia: $KP = KP_n(b)$ e $z = f_n(b)$ (z: costo ottimo di KP).

Come risolvere $KP_k(q)$ per un dato $k \leq n$ e $q \leq b$?
Sia $(x_1^*, \ldots, x_{k-1}^*, x_k^k)$ la soluzione ottima di $KP_k(q)$ di costo $f_k(q)$. Si hanno due casi:

1. $x_k^* = 0$, allora $q = \sum_{j=1}^{k} a_j x_j^* \equiv \sum_{j=1}^{k-1} a_j x_j^*$ e quindi $((x_1^*, \ldots, x_{k-1}^*)$ è la soluzione ottima di $KP_{k-1}(q)$ ovvero $f_k(q) = f_{k-1}(q)$;
2. $x_k^* = 1$, allora $q = \sum_{j=1}^{k-1} a_j x_j^* + a_k$ e quindi $(x_1^*, \ldots, x_{k-1}^*)$ è la soluzione ottima di $KP_{k-1}(q - ax)$ ovvero $f_k(q) = f_{k-1}(q - ax) + c_k$

Se conosciamo $f_{k-1}(q)$ e $f_{k-1}(q - ax)$ si ha la seguente ricorsione:

$$f_k(q) = max[f_{k-1}(q),\ f_{k-1}(q - ax) + c_k] \tag{4.16}$$

Per calcolare $f_k(q)$, $\forall q$ per un dato k dobbiamo conoscere i valori $f_{k-1}(q)$, $\forall q$.
Partiamo ponendo $f_0(q) = 0$, per ogni q tale che $0 \leq q \leq b$. Ciò consente di calcolare $f_1(q)$, $\forall q$, mediante la ricorsione.
Quindi, $f_1(\cdot)$ consente di calcolare $f_2(\cdot), \ldots,$ infine, $f_{n-1}(\cdot)$ consente di calcolare $f_n(\cdot)$.

4.5.2.2 Esempio

$$\begin{cases} z = max 10_1 + 7x_2 + 25x_3 + 24x_4 \\ 2x_1 + 1x_2 + 6x_3 + 5x_4 \leq 7 \\ x_j \in \{0,1\}, \quad j = 1, \ldots, 4 \end{cases}$$

$$f_k(q) = max[f_{k-1}(q), f_{k-1}(q - a_k) + c_k], \qquad \forall q \leq b, \quad k = b, \ldots, n$$

	f_1	f_2	f_3	f_4
$q=0$	0	0	0	0
$q=1$	0	7	7	7
$q=2$	10	10	10	10
$q=3$	10	17	17	17
$q=4$	10	17	17	17
$q=5$	10	17	17	24
$q=6$	10	17	25	31
$q=7$	10	17	32	34

Soluzione ottima:

$$
\begin{aligned}
&fl_4(7) > f_3(7) \implies x_4^* = 1\\
&fl_3(7-5) = f_2(2) \implies x_3^* = 0\\
&fl_2(2) = f_1(2) \implies x_2^* = 0\\
&fl_1(2) > f_0(2) \implies x_1^* = 1
\end{aligned}
$$

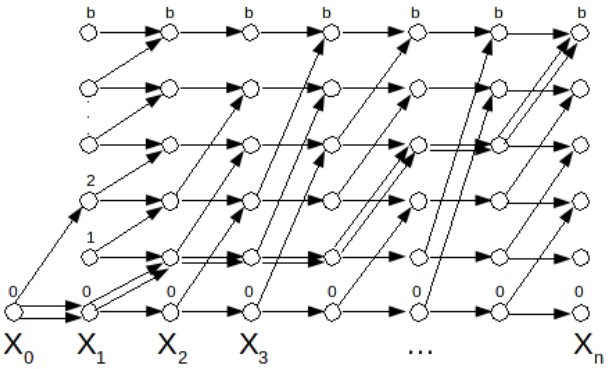
4.5.3 Grafo dello spazio degli stati

Alla ricorsione

$$f_k(q) = max[f_{k-1}(q), f_{k-1}(q-a_k) + c_k], \qquad q \leq b, \quad k = 1, \ldots, n \tag{4.17}$$

si può associare un grafo aciclico $H = (X, A)$ così fatto:

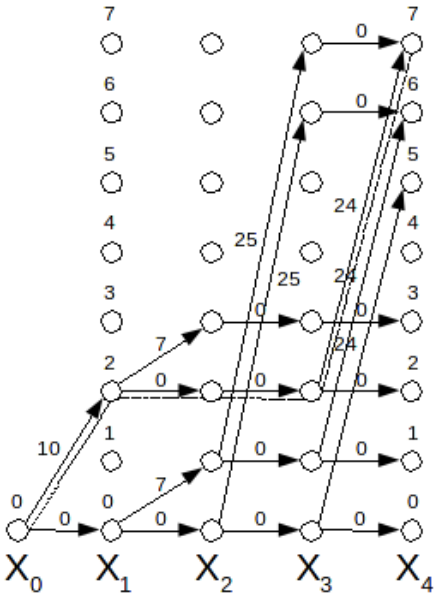
- X si compone di $n+1$ partizioni $X_0, X_1, X_2, \ldots, X_n$.
  $X_0 = (0)$ e ogni altra partizione X_k, $k = 1, \ldots, n$ contiene $b+1$ vertici corrispondenti agli stati $(0, 1, 2, \ldots, b)$;
- Su ogni vertice $q \in X_k$ terminano al più due archi: l'arco avente vertice iniziale in $q - a_k \in X_{k-1}$ di costo c_k (l'arco non esiste se $q \in X_{k-1}$ o se $q - a_k < 0$)



Il cammino di profitto massimo da $0 \in X_0$ a $b \in X_n$ corrisponde a $f_n(b)$.

4.5.4 Esempio

$$\begin{cases} z = max\ 10x_1 + 7x_2 + 25x_3 + 24x_4 \\ 2x_1 + 1x_2 + 6x_3 + 5x_4 \leq 7 \\ x_j \in \{0,1\}, \quad j = 1, \ldots, 4 \end{cases}$$



Non sono stati disegnati gli archi inutili, ovvero, non attraversabili da alcun cammino che parte da $0 \in X_0$.
Il grafo mostra che è inutile calcolare $f_1(q)$, $q \in \{1,3,4,5,6,7\}$ ma anche $f_2(q)$, $q = 4, \ldots, 7$, etc.

Qual è un algoritmo migliore per implementare la ricorsione per il knapsack 0-1?

4.5.5 Ricorsione Froward - Knapsack 0-1

1. Poni $f_0(0) = 0$ e $f_0(q) = -\infty$, $q = 1, \ldots, b$. Sia $k = 0$;
2. inizializza $f_{k+1}(q) = -\infty$, $q = 0, 1, \ldots, b$;
3. Per ogni $q = 0, 1, \ldots, b$ tale che $f_k(q) \geq 0$ aggiorna

$$f_{k+1}(q) = max\ [f_{k+1}(q), f_k(q)] \tag{4.18}$$

   se $q + a_{k+1} \leq b$ allora

$$f_{k+1}(q + a_{k+1}) = max\ [f_{k+1}(q + a_{k+1}), f_k(q) + c_{k+1}] \tag{4.19}$$

4. Se $k = n - 1$ *STOP*, altrimenti poni $k = k + 1$ e ritorna allo step 2.

4.6 Programmazione a numeri interi

$$F^* = Min \sum_{j=1}^{n} C_j x_j \tag{4.20}$$

$$\sum_{j=1}^{n} a_{1j} x_j = b_1 \tag{4.21}$$

$$\sum_{j=1}^{n} a_{2j} x_j = b_2 \tag{4.22}$$

$$\vdots \qquad \vdots \tag{4.23}$$

$$\sum_{j=1}^{n} a_{mj} x_j = b_m \tag{4.24}$$

$$x_j \geq 0 \text{ e intero} \tag{4.25}$$

Per semplicità assumiamo che

$$a_{ij} \geq 0 \qquad \forall i, j$$
$$b_i \geq 0 \qquad \forall i$$

4.7 Programmazione Dinamica

$$F_k(\underline{v}) = Min \sum_{j=1}^{k} C_j x_j \tag{4.26}$$

$$\sum_{j=1}^{k} a_{1j} x_j = v_1 \tag{4.27}$$

$$\sum_{j=1}^{k} a_{2j} x_j = v_2 \tag{4.28}$$

$$\vdots \qquad \vdots \tag{4.29}$$

$$\sum_{j=1}^{k} a_{mj} x_j = v_m \tag{4.30}$$

$$x_1, x_2, \ldots, x_k \geq 0 \text{ e intero} \tag{4.31}$$

$F_k(V)$ venga calcolato per $k = 1, \ldots, n$; $\forall \underline{v} \leq \underline{b}$ e $\underline{v} \geq 0$.
La soluzione ottima è data da

$$F^* = F_n(\underline{b}) \tag{4.32}$$

4.8 Ricorsione di Programmazione Dinamica

Le funzioni $F_k(\underline{v})$ si possono calcolare come segue:

$$F(\underline{v} = Min\ \{F_{k-1}(\underline{v}),\quad \underset{\substack{x_k>0\\ x_k\ intero}}{Min}\ [F_{k-1}(\underline{v} - a^k x_k) + c_k x_k]\}) \tag{4.33}$$

per $k = 1, \ldots, n$"; $\forall \underline{v} \leq \underline{b}$.
La ricorsione richide di conoscere $F_0(\underline{v})$

$$F_0(\underline{v}) \begin{cases} 0 \text{ per } v = 0 \\ \infty \text{ per ogni } 0 < \underline{v} \leq b \end{cases} \tag{4.34}$$

k rappresenta lo **stadio** della ricorsione mentre v è la **variabile di stato**.

4.9 Programmazione Dinamica: il TSP

È dato un grafo $G = (X, A)$ di $n = |X|$ vertici ed $m = |A|$ archi. Ad ogni archo $(x_i, x_j) \in A$ è associato un costo c_{ij}.

Determinare il cammino di costo minimo che parte da x_1, attraversa tutti i vertici di G una ed una sola volta e termina in x_1.

$f(S, x_i)$: costo minimo di un cammino che parte da x_i, attraversa tutti i vertici di $S \in X \setminus x_i$ una ed una sola volta e termina in $x_i \in S$.

4.9.1 Ricorsione per il calcolo di $f(S, x_i)$

$$f(S, x_i) = \underset{x_j \in S \setminus x_i}{Min} \{f(S \setminus x_i, x_j) + c_{ji}\} \tag{4.35}$$

Le $f(S, x_i)$ vanno calcolate:

$$\forall S \subseteq X \text{ tale che } x_1 \in S,\ |S| \geq 2,\ \forall x_i \in S \setminus x_1 \tag{4.36}$$

4.9.1.1 Calcolo del valore z^* della soluzione ottima

$$z^* = \underset{x_i \in X \setminus X_1}{Min} \{f(X, x_i) + c_{i1}\} \tag{4.37}$$

È richiesta la seguente inizializzazione:

$$f(\{x_1\}, x_1) = 0 \tag{4.38}$$

4.9.2 Considerazioni computazionali

Il calcolo di $f(S, x_i)$ richiede di conoscere il valore di $f(S \setminus \{x_i\}, x_j)\ \forall x_j \in S \setminus \{x_i\}$

Esempio: calcolo di $f(\{x_2, x_3, x_4x_5\}, x_2)$

$$f(\{x_2, x_3, x_4x_5\}, x_2) = \underset{x_j \in \{x_3, x_4, x_5\}}{Min} [f(\{x_3, x_4, x_5\}, x_j) + c_{j2}] \tag{4.39}$$

e quindi bisogna conoscere:

$$f(x_3, x_4x_5\}, x_3);\quad f(\{x_3, x_4x_5\}, x_4);\quad f(\{x_3, x_4x_5\}, x_5) \tag{4.40}$$

Le **variabili di stato** sono rappresentate da

$$(S, x_i);\qquad \forall\ S \subseteq X \setminus \{x_i\},\ \ \forall\ X_i \in S \tag{4.41}$$

Lo stadio può essere definito in modi diversi; deve corrispondere al sequente principio:

"se lo stato (S, x_i) è associato allo stadio K allora ognuno degli stati $(S \setminus \{x_i\}, x_j)$ deve essere associato ad uno stadio $K^{'} < K^{''}$."

Ad esempio: stadio: $|S|$ cardinalità di S.
È ovvio che allo stadio 1 vengono calcolate le $f(S, x_i)\ \forall\ S$ per cui $|S| = 1$.
Allo stadio 2 si calcola $f(S, x_i)$, $\forall\ S$ per cui $|S| = 2$.
⋮
Allo stadio k si calcola $f(S, x_i)$, $\forall\ S$ per cui $|S| = k$ e quindi sono note le $f(S \setminus \{x_i\}, x_j)$ in quanto sono state calcolate allo stadio $k - 1$.
Lo stadio può essere definito diversamente:

- Si associ un peso $q_i \geq 1$ ad ogni $x_i \in X$. Si assegni ad ogni stato (S, x_i) allo stadio $\sum_{i \in S}^{q_i}$. In tal modo gli stati vengono suddivisi in $\sum_{i=1}^{n} q_i$ stadi;
- si noti che questa definizione è corretta, infatti dato lo stato (S, x_i) corrispondente a $\sum_{i \in S} q_i = q(S)$ ogni stato $(S \setminus \{x_i, x_j\})$ corrisponde allo stadio $q(S) - q_i$.

È banale notare che ponendo $q_i = 1$, $\forall\ x_i \in X$ lo stadio corrisponde alla cardinalità di S (come visto in precedenza).

4.9.3 Esempio del TSP con 5 città

$$[c_{ij}] = \begin{bmatrix} - & 6 & 11 & 3 & 4 \\ 7 & - & 14 & 8 & 10 \\ 12 & 5 & - & 10 & 2 \\ 6 & 15 & 7 & - & 5 \\ 4 & 9 & 8 & 13 & 6- \end{bmatrix}$$

Applichiamo la ricorsione:

$$f(S, x_i) = \underset{x_j \in S \setminus \{x_i\}}{Min} \{f(S \setminus \{x_i, x_j\}) + c_{ji}\}$$

Inizializzazione:

$$f(\{x_i\}, x_i) = c_{1i}; \quad \forall\ x_i \in X \setminus \{x_1\} \tag{4.42}$$

$$f(\{2\}, 2) = 6$$
$$f(\{3\}, 3) = 11$$
$$f(\{4\}, 4) = 3$$
$$f(\{5\}, 5) = 4$$

4.9.3.1 Stadio 2: Calcolo $f(S, x_i),\ \forall\ S \subset X \setminus \{x_i,\ s.t.\ |S| = 2;\ \forall\ x_i \in S$

$$f(S, x_i) = \underset{x_j \in S \setminus \{x_j\}}{Min} \{f(S \setminus \{x_i\}, x_j) + c_{ji}\}$$

$$f(\{2,3\},2) = f(\{3\},3) + c_{3,2} = 11 + 5 = 16$$
$$f(\{2,3\},3) = f(\{2\},2) + c_{2,3} = 6 + 14 = 20$$

$$f(\{2,4\},2) = f(\{4\},4) + c_{4,2} = 3 + 15 = 18$$
$$f(\{2,4\},3) = f(\{2\},2) + c_{2,4} = 6 + 8 = 14$$

$$f(\{2,5\},2) = f(\{5\},5) + c_{5,2} = 4 + 9 = 13$$
$$f(\{2,5\},3) = f(\{2\},2) + c_{2,5} = 6 + 10 = 16$$

Similarmente:

$$f(\{3,4\},3) = 10$$
$$f(\{3,4\},4) = 21$$

$$f(\{3,5\},3) = 12$$
$$f(\{3,5\},5) = 13$$

$$f(\{4,5\},4) = 17$$
$$f(\{4,5\},5) = 8$$

4.9.3.2 Stadio 3: $f(S, x_i),\ \forall\ S \subset X \setminus \{x_i,\ s.t.\ |S| = 3;\ \forall\ x_i \in S$

$$f(\{2,3,4\},2) = \underset{x_j \in \{3,4\}}{Min} \{f(\{3,4\}, x_j) + c_{j2}\}$$

$$f(\{2,3,4\},2) = Min\ \{f(\{3,4\},3) + c_{3,2};\ f(\{3,4\},4) + c_{4,2}\} = Min\ \{10+5;\ 21+15\} = 15$$
$$f(\{2,3,4\},3) = Min\ \{f(\{2,4\},2) + c_{2,3};\ f(\{2,4\},4) + c_{4,3}\} = Min\ \{18+4;\ 14+7\} = 21$$

Similarmente:

$$f(\{2,3,4\},4) = 24$$

$$\begin{array}{lll} f(\{2,3,5\},2) = 17; & f(\{2,3,5\},3) = 24; & f(\{2,3,5\},5) = 22 \\ f(\{2,4,5\},2) = 17; & f(\{2,4,5\},4) = 21; & f(\{2,4,5\},5) = 19 \\ f(\{3,4,5\},3) = 16; & f(\{3,4,5\},4) = 22; & f(\{3,4,5\},5) = 12 \end{array}$$

4.9.3.3 Stadio 4

$$f(\{2,3,4,5\},2) = \underset{x_j \in \{3,4,5\}}{Min} \{f(\{3,4,5\},x_j) + c_{j2}\}$$

da cui

$$\begin{aligned} f(\{2,3,4,5\},2) = Min\ \{&f(\{3,4,5\},3) + c_{3,2},\ f(\{3,4,5\},4) + c_{4,2},\ f(\{3,4,5\},5) + c_{5,2}\} = \\ &Min\{16+5;\ 22+15;\ 12+9\} = 21 \end{aligned}$$

Similarmente

$$\begin{array}{c} f(\{2,3,4,5\},3) = 27 \\ f(\{2,3,4,5\},4) = 25 \\ f(\{2,3,4,5\},5) = 23 \end{array}$$

4.9.3.4 Soluzione ottima

$$\begin{aligned} z^* = &\underset{x_j \in \{2,3,4,5\}}{Min} \{f(\{2,3,4,5\},x_i) + c_{i1}\} = \\ &= \{21+7;\ 27+12;\ 25+6;\ 23+4\} = 27 \end{aligned}$$

4.9.3.5 Spazio degli stati del TSP

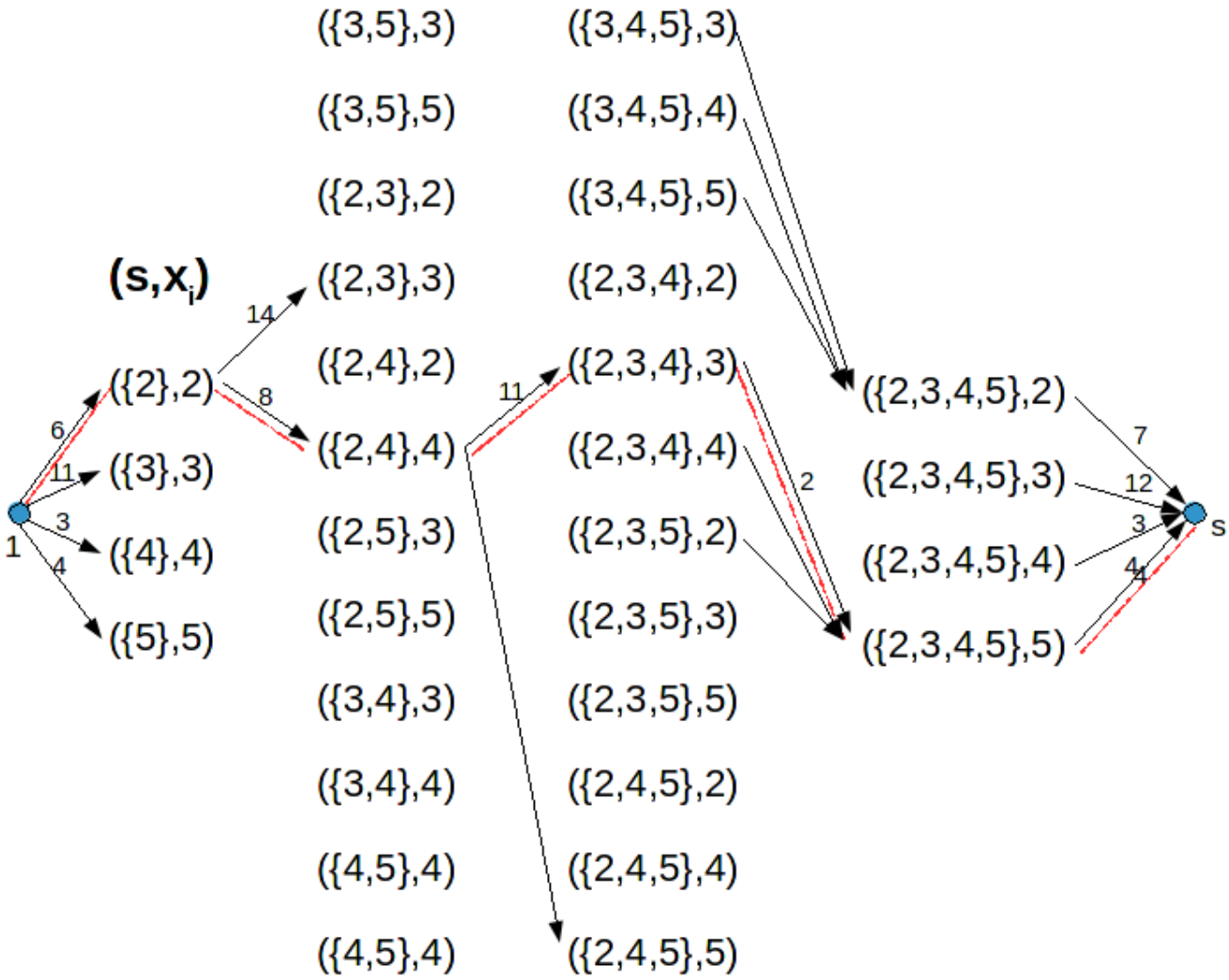


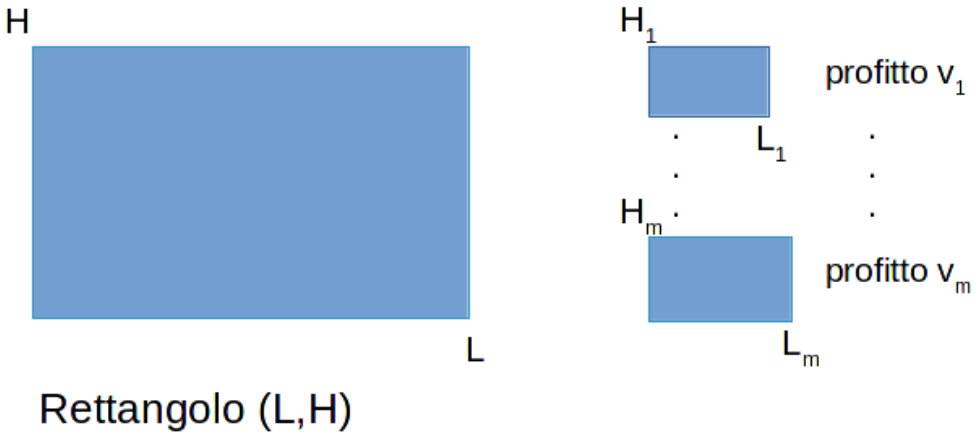
Figura 4.1: Solo alcuni degli archi sono raffigurati

4.10 Ricorsione Forward per il TSP

1. $\mathscr{L}_0 = \{(\{1\}, 1)\}$, $f(\{1\}, 1) = 0$, $p(\{1\}, 1) = 0$. Inizializza $\mathscr{L}_r = \emptyset$, $r = 1, \ldots, n$. Definisci $k = 1$.
2. Espansione degli stati del set $\mathscr{L}_{k-1}$: per ogni stato $(S, i) \in \mathscr{L}_{k-1}$ ripeti lo step 3.
3. Genera/Aggiorna gli stati di $\mathscr{L}_k$ raggiungibili da (S, i): per ogni vertice $j \in \Gamma_i \setminus S$ considera lo stato (S', i), dove $S' = S \cup \{j\}$, che si ottiene aggiungendo l'arco (i, j). Il costo di (S', j) è $h = f(S, i) + c_{ij}$.
   Si hanno i seguenti casi:
   - $(S', j) \in \mathscr{L}_k$, allora $\mathscr{L}_k = \mathscr{L}_k \cup \{f(S', j)\}$ e $f(S', j) = h$. Poni $p(S', j) = (S, i)$;
   - $(S', j) \in \mathscr{L}_k$ ma $f(S', j) > h$, allora $f(S', j) = h$ e $p(S', j) = (S, i)$.
4. Poni $k = k + 1$; se $k \leq n$ vai allora step 2.
5. Il costo ottimo z^* del TSP si ottiene come segue:

$$z^* = \underset{(X,j) \in \mathscr{L}_n}{Min} [f(X, j) + c_{j1}] \tag{4.43}$$

4.11 Taglio 2-dimesionale a ghigliottina



- Il rettangolo e i pezzi hanno dimensioni intere e non possono essere ruotati;
- Sono ammessi solo tagli a ghigliottina;
- Ogni tipo di pesso è disponibile in quantità illimitata

Obiettivo: massimizzare il profitto totale dei pezzi tagliati.

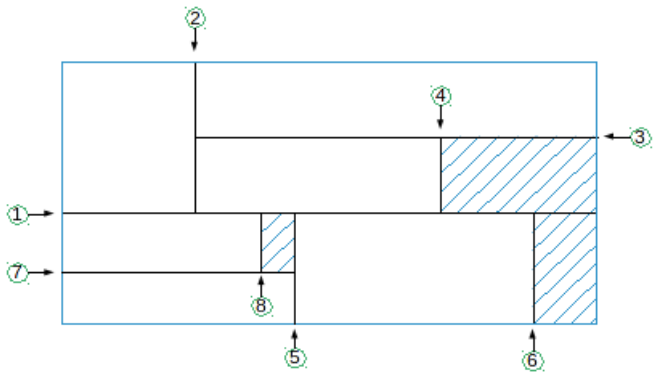


Figura 4.2: Esempio di taglio a ghigliottina

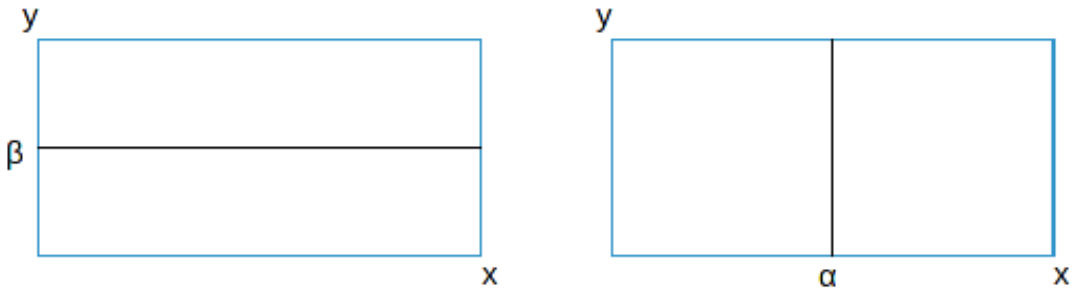
Indichiamo con:
$F(x, y)$ il profitto massimo per tagliare un rettangolo di dimensione (x, y) con $x \leq L$ e $y \leq H$.
$F(L, H)$ il valore della soluzione ottima.

4.11.1 Calcolo di $F(x, y)$

Si hanno tre casi:

1. Il rettangolo (x, y) contiene al più un pezzo (o nessuno) $F(x, y) = max\ [0,\ vi :\ l_i \leq x,\ h_i \leq y,\ i = 1, \ldots, m]$

2. Il rettangolo (x, y) contiene due o più pezzi che per essere tagliati richiedono almeno un taglio a ghigliottina o parallelo alla lunghezza in posizione β parallelo all'altezza in posizione α.



$F(x, y) = F(x, \beta) + F(x, y - \beta)$ o $F(x, y) = F(\alpha, y) = F(x - \alpha, y)$

4.11.2 Inizializzazione

Per ogni $x = 1, ..., L$ e $y = 1, \ldots, H$ poni

$$F^0(x, y) = max\ [0,\ v_i :\ l_i \leq x,\ h_i \leq y,\ i = 1, \ldots, m] \tag{4.44}$$

- se $F^0(x, y) = v_{i^*}$ poni $X - cut(x, y) = l_{i^*}$ e $Y - cut(x, y) = h_{i^*}$
- se $F^0(x, y) = 0$ poni $X - cut(x, y) = Y - cut(x, y) = 0$

La ricorsione: per ogni $x = 1, \ldots, L$ e $y = 1, \ldots, H$ calcola

$$F(x, y) = max\ [F^0(x, y);\ F(x, \beta)+F(x, y-\beta),\ \beta = 1, \ldots, y-1,\ F(\alpha, y)+F(x-\alpha, y),\ \alpha = 1, \ldots, x-1] \tag{4.45}$$

- se $F(x, y) = F(x, \beta^*)+F(x, y-\beta^*)$ per qualche β^* poni $X-cut(x, y) = 0$ e $Y-cut(x, y) = \beta^*$
- se $F(x, y) = F(\alpha^*, x)+F(x-\alpha^*, y)$ per qualche α^* poni $X-cut(x, y) = \alpha^*$ e $Y-cut(x, y) = 0$

Si può migliorare la ricorsione sostituendo

$$\beta = 1, \ldots, y - 1 \text{ con } 1 \leq \beta \leq y/2$$
$$\alpha = 1, \ldots, x - 1 \text{ con } 1 \leq \alpha \leq x/2$$

4.11.3 Esempio

Sia $y = 5$

$$F(x, 5) = max\ [F^0(x, 5),\ \overbrace{F(x, 1) + F(x, 4)}^{T_1}, \overbrace{F(x, 2) + F(x, 3)}^{T_2}, \underbrace{F(x, 3) + F(x, 2)}_{T_3}, \underbrace{F(x, 4) + F(x, 1)}_{T_4}, \ldots]$$

Si noti che $T_1 \equiv T_4$ e $T_2 \equiv T_3$ è ciò giustifica la sostituzione di $\beta = 1, \ldots, y - 1$ con $1 \leq \beta \leq y/2$.
Complessità: $O(L \cdot H(L + H))$.

4.11.4 Normal Cuts

Riduce la complessità in molte situazioni reali I pezzi possono essere spostati in basso e a sinistra

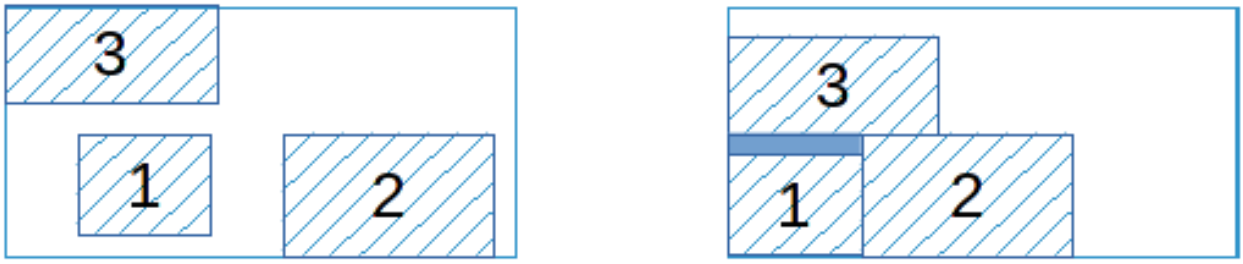


Figura 4.3: Sono equivalenti

fino a che il lato sinistro e il lato inferiore di ogni pezzo sono adiacenti ad un taglio o al lato sinistro e inferiore del rettangolo.
I tagli a ghigliottina possono avvenire solo nelle seguenti posizioni.

Posizioni possibili per tagli paralleli all'altezza

$$L_0 = (x: \ x = \sum_{i=1}^{m} l_i \xi_i, \ 1 \leq x \leq L, \ \xi \geq 0 intero) \quad U(L) \tag{4.46}$$

Posizioni possibili per tagli paralleli alla lunghezza

$$H_0 = (y: \ y = \sum_{i=1}^{m} h_i \xi_i, \ 1 \leq y \leq H, \ \xi \geq 0 intero) \quad U(H) \tag{4.47}$$

Definiamo

$$p(x) = max \ [0, \ \alpha : \alpha \leq x, \ \alpha \in L_0], \quad x = 1, \ldots, L-1$$
$$p(y) = max \ [0, \ \beta : \beta \leq y, \ \beta \in H_0], \quad y = 1, \ldots, H-1$$

La ricorsione diviene, per ogni $x = 1, \ldots, L$ e $y = 1, \ldots, H$:

$$F(x,y) = max \ [F^0(x,y); \ F(x,\beta) + F(x, q(y-\beta)), \ \beta \in H_0, \ \beta \leq y/2,$$
$$F(\alpha, y) + F(p(x-a), y), \ \alpha \in L_0, \ \alpha \leq x/2]$$

4.12 Rilassamento dello spazio degli stati

In molti casi lo spazio degli stati su cui è definita una ricorsione di programmazione dinamica (DP) ha dimensioni proibitive (si veda il caso del TSP).

4.12.1 Come ridurre lo spazio degli stati

1. Eliminare stati che non possono condurre ad alcuna soluzione ottima.
   Ad esempio usando un lower bound.
2. Riducendo in modo euristico gli stati fino a che lo spazio risultante non ha dimensioni "accettabili".
   Ad esempio scegliendo mediante qualche regola un sottoinsieme limitato di stato ad ogni stadio. Lo spazio risultante potrebbe non contenere la soluzione ottima.
3. Contraendo più stati in un unico stato in modo che la ricorsione di DP nello spazio rilassato produca un lower bound (questo metodo è noto: *state space relaxation*).

Vedremo come il metodo state space relaxation fornisca un lower bound da usare al punto 1.

4.12.2 Lower bound al cammino minimo

Il metodo della **State Space Relaxation** si basa sulla sequente semplice idea che consente di calcolare un lower bound al costo del cammino minimo in un grafo. I vertici sono clusterizzati in

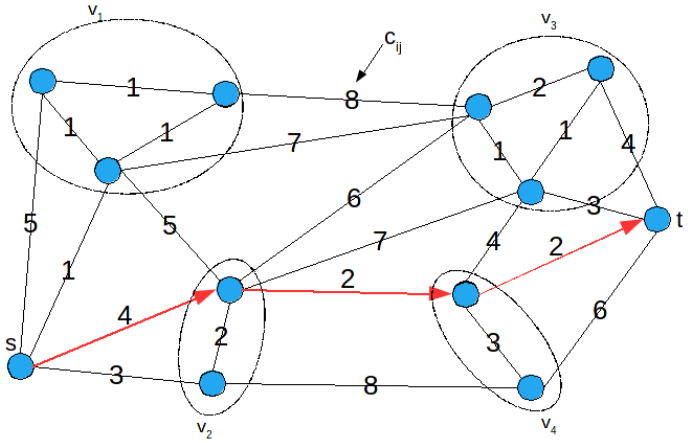
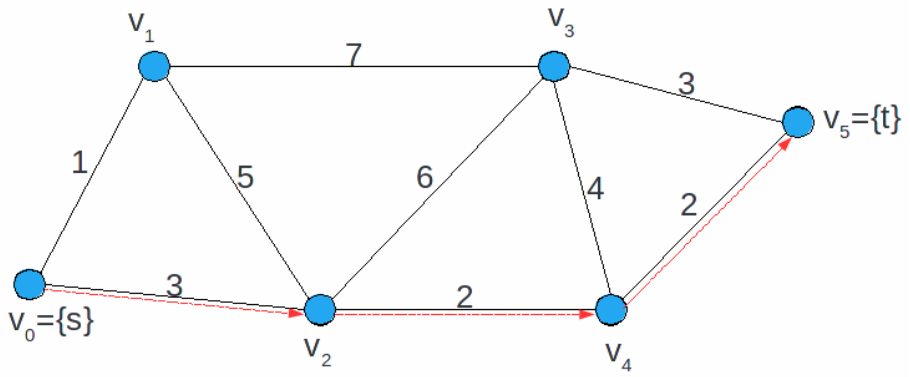


Figura 4.4: Il costo del cammino del grafo $G = (X, A)$ da s a t è 8

4 cluster come mostrato (non è importante il criterio di clustering per quanto segue).
Ogni $v_k \subset X$, $k = 1, \ldots, 4$ e $v_k \cap V_j = \emptyset$ $j \neq k = 1\, dots, 4$.

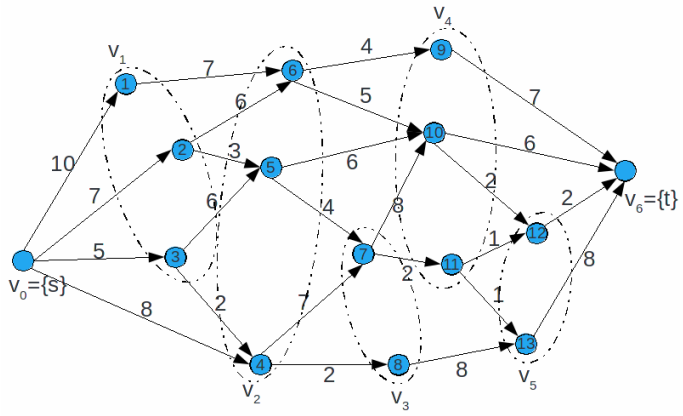
Grafo rilassato $\bar{G} = (\bar{X}, \bar{A})$: $\bar{X} = \{v_0, v_1, \ldots, v_4, v_5\}$



$$\bar{c_{ij}} = \min_{\substack{r \in V^i \\ s \in V_j}} [c_{rs}]$$

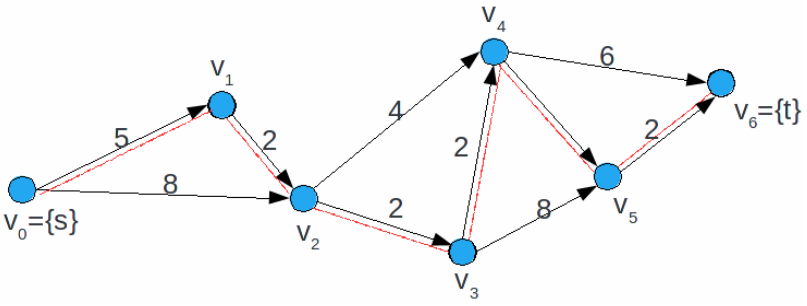
Il costo del cammino minimo da v_0 a $v_5 \in \bar{G}$ $(= 7)$ è un valido lower bound.

4.12.2.1 Esempio



Nel caso di un grafo aciclico, come nell'esempio, può essere conveniente che ogni cluster v_k sia tale che $\forall\ j, j \in v_k (i < j)$ non esista l'arco (i, j) in A.

Grafo rilassato



4.12.3 Sitema discreto multistadio

$s = (s_1, \ldots, s_m)$: variabile di stato
$\mathscr{L}_k$: insieme degli stati allo stadio k
$f_k(s)$ è il costo minimo per cambiare lo stato del sistema dallo stato iniziale (allo stadio 0) allora stato $s \in \mathscr{L}_k$ allora stadio k.

$$f_k(s) = \min_{s' \in \Delta^{-1}(s)} [f_{k-1}(s') + v(s', s)], \quad \forall\ s \in \mathscr{L}_k \tag{4.48}$$

dove $\Delta^{-1}(s)$ sono gli stati che raggiungono lo stato s e $v(s', s)$ è il costo per cambiare il sistema dallo stato s' allo stato s.

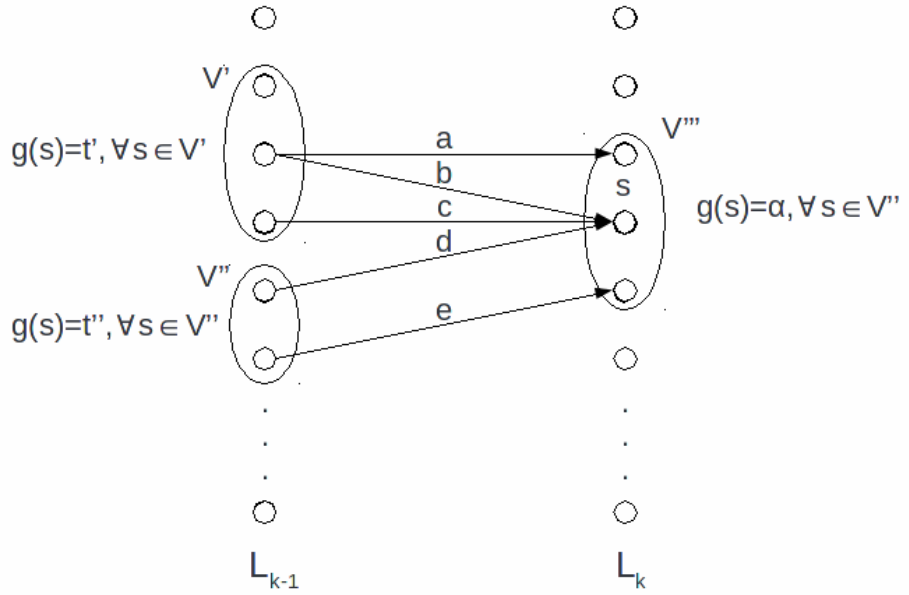
4.12.3.1 Esempio: il TSP

$$f(s, i) = \min_{j \in S \setminus \{i,1\}} [f(S \setminus \{i\}, j) + c_{ji}], \quad |S| \geq 2$$

Definiamo (ad esempio)

$$\begin{aligned} &s = (S, i) \\ &\mathscr{L}_k = \{(S, i) : S \subset X \ t.c.\ 1 \in S,\ |S| = k,\ i \in S\} \\ &\Delta^{-1}(S, i) = \{(S \setminus \{i\}, j) :\ j \in S\{i, 1\}\} \\ &v((S \setminus \{i\}, j), (S, i)) = c_{ji} \end{aligned}$$

Sia $g(\cdot)$ una funzione di "mapping" dallo spazio degli stati $\mathscr{L}$ allo spazio ridotto R.



Più stati di $\mathscr{L}$ vengono associati ad un unico stato di R.
Per ogni $s' \in \Delta^{-1}(s)$ esiste l'arco $(g(s'), g(s))$.
$F^{-1}(\alpha)$: predecessori di $\alpha \in R$ (ad esempio: $F^{-1}(\alpha) = \{t', t''\}$).

$\bar{v}(\alpha, \beta)$: $min[v(s', s) : \forall s', s \in \mathscr{L}$ *t.c.* $g(s') = \alpha$ *e* $g(s) = \beta]$.

Esempio
$\bar{v}(t', \alpha) = min[a, b, c]$, $\bar{v}(t'', \alpha) = min[d, e]$

Nello spazio R la ricorsione diviene

$$f_k(g(s)) = \min_{t \in F^{-1}(g(s))} [f_{k-1}(t) + \bar{v}(t, g(s))] \tag{4.49}$$

Si noti che $f_k(g(s)) \leq f_k(s)$, $\forall\ s \in \mathscr{L}$, ovvero, $f_k(g(s))$ è un lower bound a $f_k(s)$.
La funzione $g(\cdot)$ deve essere tale per cui

1. $p^{-1}(\alpha)$ può essere calcolato facilmente $\forall \alpha \in R$

2. $\bar{v}(\alpha, \beta)$ può essere calcolato facilmente o approssimato con un lower bound

4.12.4 Rilassamento dello spazio degli stati per il TSP

$$f(s, i) = \min_{j \in S \setminus i, 1} [f(S \setminus i, j) + c_{ji}], \quad |S| \geq 2 \tag{4.50}$$

(s, i): variabile di stato

Funzione di mapping: $(s, i) \rightarrow (g(s), i)$

$$\Delta^{-1}(s, i) = \{(S \setminus i, j) :\ j \in S \setminus \{i, 1\}\} \tag{4.51}$$
$$F^{-1}(g(s), i) = \{(g(S \setminus \{i\}), j) :\ j \in S \setminus \{i, 1\}\} \subseteq \{g(S \setminus \{i\}, j) :\ j \in \Gamma^{-1}\} \tag{4.52}$$
$$v((S \setminus \{i\}, j)), (s, i)) = c_{ji} \tag{4.53}$$

quindi $v()$ non dipende da S, per cui

$$\bar{v}((g(S \setminus \{i\}), j), (g(s), i)) = c_{ji} \tag{4.54}$$

La ricorsione nello spazio rilassato diviene

$$f(g(s), i) = \min_{j \in \Gamma^{-1} \setminus \{1\}} [f(g(S \setminus \{i\}), j) + c_{ji}], \quad |S| \geq 2 \tag{4.55}$$

Inizializzazione

$$f(g(\{1, i\}), i) = c_{1j}, \quad \forall\ i \tag{4.56}$$

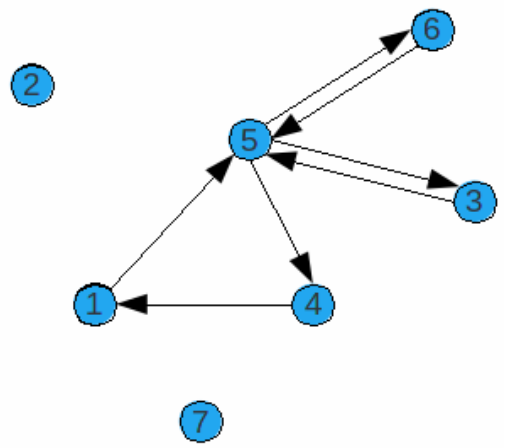
4.12.5 Funzioni di mapping $g(\cdot)$ per il TSP

4.12.5.1 Rilassamento *n-path*

$g(S) = |S| : \ (S, i) \rightarrow (k, i)$ dove $k = |S|$

$$f(k, i) = \min_{j \in \Gamma_i^{-1} \setminus \{1\}} [f(k-1, j) + c_{ji}], \quad k \geq 2 \tag{4.57}$$

Inizializza $f(1, i) = c_{1i}, \ \forall \ i$
$f(k, 1)$: cammino minimo di cardinalità k da 1 a i (tale cammino può essere non elementare).
$z^* = \min_i [f(n-1, i) + c_{i1}]$ è un lower bound al TSP.



Il vertice 5 è visitato 3 volte.
I vertici 2 e 7 non sono visitati.

Miglior lower bound
Si penalizzi in modo Lagrangiano i vertici non visitati esattamente una ed una sola volta.

4.12.6 Rilassamento q-path

Ad ogni vertice i si associ un peso $q_i \leq 1$ e $q_1 = 0$

$$g(S) = \sum_{i \in S} q_i : \ (S, i) \rightarrow (q, i) \text{ dove } q = \sum_{i \in S} q_i \tag{4.58}$$

$$f(q, i) = \min_{j \in \Gamma_i^{-1} \setminus \{1\}} [f(q - q_i, j) + c_{ji}], \ q > q_i \tag{4.59}$$

Inizializza:

$$f(q,i) = \begin{cases} c_{1i}, \text{ se } q = q_i \\ \infty, \text{ altrimenti} \end{cases} \quad , \quad \forall i \text{ e } \forall\, q = 1, \ldots, Q \tag{4.60}$$

$$\text{dove } Q = \sum_{i=1}^{k} q_i \tag{4.61}$$

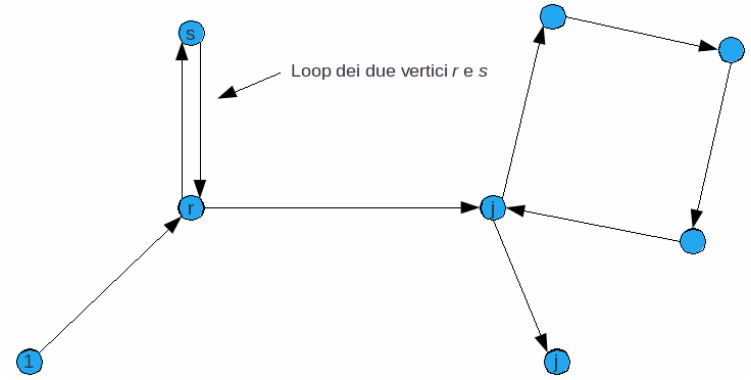
Il lower bound al TSP è dato da

$$z^* = \min_i [f(Q,i) + c_{j1}] \tag{4.62}$$

Il cammino di costo $f(q,i)$ può essere non elementare e quindi anche il circuito corrispondente a z^*.
Un miglior lower bound, anche in questo caso si ottiene mediante un ascent lagrangiano in cui vengono penalizzati i vertici non visitati esattamento una sola volta.

4.12.7 Eliminazione dei loops di 2 vertici

Sia $f(k,1)$ che $f(q,i)$ possono produrre cammini non elementari con loops di 2 vertici (vedi esempio)



I loops di 2 vertici possono essere eliminati senza aumentare la complessità delle ricorsioni $f(k,i)$ e $f(q,i)$ con il "trucco" qui descritto per $f(q,i)$.

4.12.7.1 Definizioni

- $f(q,i)$: costo del cammino di costo minimo e "peso" q da 1 a i
- $\pi(q,i)$: vertice che precede i nel cammino di costo $f(q,i)$
- $\phi(q,i)$: costo del cammino di costo minimo e peso q da 1 a i tale che il vertiche che precede i è $\neq \pi(q,i)$
- $\gamma(q,i)$: vertice che precede i nel cammino di costo $\phi(q,i)$

Per il calcolo di $f(q,i)$ e $\phi(q,i)$, $\forall\, i$ e un dato q si procede come segue.
Sia h_{ji} il costo del cammino minimo di peso q e senza loops di 2 vertici da 1 a i dove j precede i.

h_{ji}, $\forall i$ e j si calcola come segue

$$h_{ji} = \begin{cases} f(q - q_i, j) + c_{ji}, \text{ se } \pi(q - q_i, j) \neq= i \\ \phi(q - q_i, j) + c_{ji}, \text{ altrimenti} \end{cases} \quad , \forall\ i, j \tag{4.63}$$

Quindi calcola per ogni vertiche i

$$f(q, i) = \min_{j}[h_{ji}], \text{ sia } j^* \text{ il vertice che produce il } min \tag{4.64}$$

$$\pi(q, i) = j^* \tag{4.65}$$

$$\phi(q, i) = \min_{j \neq j^*}[h_{ji}], \text{ sia } \hat{j} \text{ il vertiche che produce il } min \tag{4.66}$$

$$\gamma(q, i) = \hat{j} \tag{4.67}$$

Inizializza

$$f(q, i) = \begin{cases} c_{1i}, \text{ se } q = q_i \\ \infty, \text{ altrimenti} \end{cases} \tag{4.68}$$

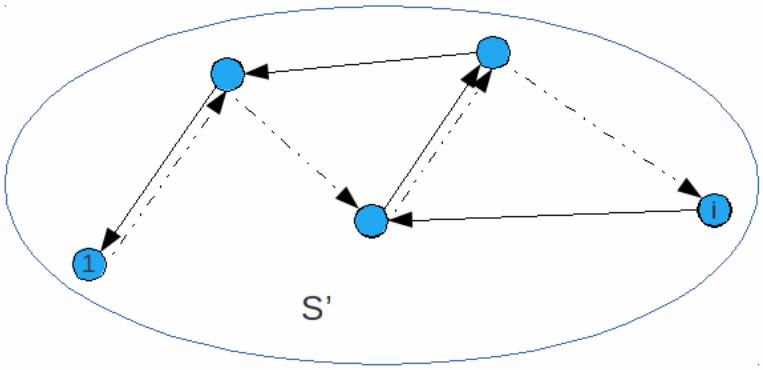
$$\pi(q_i, i) = 1 \tag{4.69}$$

$$\phi(q, i) = \infty, \quad \forall\ i \text{ e } \forall\ q \tag{4.70}$$

$$\gamma(q, i) = 0 \tag{4.71}$$

4.12.8 Revers function per il TSP

$f'(S, i)$: costo del cammino che parte dalla città $i \in S$ visita una e una sola volta tutti i vertici di S e termina nella città 1. Se $c[c_{ij}]$ è simmetrica allora $f'(S, i) = f(S, i)$.



Matrice $[c_{ij}]$ asimmetrica

$\rightarrow$ cammino $f'(S, i)$

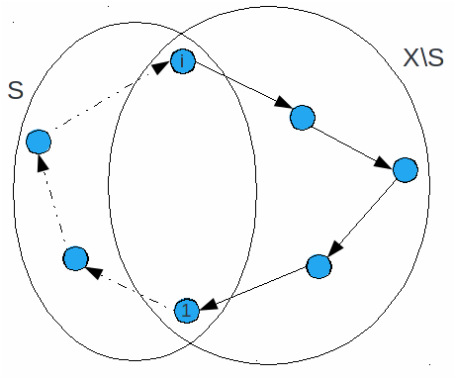
$\dashrightarrow$ cammino $f(S, i)$

Per calcolare $f'(S, i)$ si può usare la stessa ricorsione utilizzata per calcolare $f(S, i)$ ma usando la trasposta della matrice $[c_{ij}]$.
In modo simile si possono calcolare le funzioni $f'(k, 1)$, $f'(q, i)$.

4.12.9 Algoritmo DP+Lower Bound per il TSP

È la combinazione della "ricorsione forward" con la reverse function $f'(k,1)$ per eliminare stati che non possono condurre ad alcuna soluzione ottima.
Al posto di $f'(k,i)$ si può usare $f'(q,i)$ o qualsiasi altra funzione che derivi da un diverso rilassamento dello spazio degli stati.

Sia z^* il costo del TSP ottimo.
Se lo stato (S,i) fa parte della soluzione ottima di costo z^* allora Sia z_{UB} un upper bound a z^*



$$f(S,i) + f'(X \setminus S, i) = z^* \tag{4.72}$$

calcolato con un euristico. Sia $k = |S|$ per cui $|X \setminus S| = n - k$.
Lo stato (S,i) non può far parte del TSP ottimo se:

$$f(S,i) + \begin{cases} f'(n-k,i), \text{ se } \pi'(n-k,i) \in S \\ \phi'(n-k,i) \text{ se } \pi'(n-k,i) \in S \end{cases} \geq z_{UB} \tag{4.73}$$

4.12.10 Algoritmo di programmazione dinamica per il TSP

1. Poni $\mathscr{L}_i\{(\{i\}),1\}$, $f(\{i\},i) = 0$, $p(\{i\},i) = 1$ e $\mathscr{L}_r = \emptyset$, $r = 2,\ldots,n$. Sia z_{UB} un upper bound al TSP. Poni $k = 2$;

2. Espandi ogni stato del set $\mathscr{L}_{k-1}$:
   per ogni $(S,i) \in \mathscr{L}_{k-1}$ ripeti lo step 3;

3. Genera gli stati di $\mathscr{L}_k$ raggiungibili da (S,i):
   per ogni $j \in \Gamma^{-1} \setminus S$ considera lo stato (S',j) dove $S' = S \cup \{j\}$. Poni $h = f(S,i) + c_{ij}$.
   Lo stato (S',j) deve essere "rigettato" nei seguenti casi:

   - se $h + f'(n-k,i) \geq z_{UB}$ qualora $\pi'(n-k,i) \notin S'$;
   - se $h + \phi'(n-k,j) \geq z_{UB}$ qualora $\pi'(n-k,j) \in S'$;
   - se $(S',j) \in \mathscr{L}_k$ e $f(S',j) \leq h$.

   Se $(S',j) \notin \mathscr{L}_k$ allora poni $\mathscr{L}_k = \mathscr{L}_k \cup \{(S',j)\}$, $f(S',j) = h$ e $p(S',j) = (S,i)$.
   Se $(S',j) in \mathscr{L}_k$ e $f(S',j) > h$ allora poni $f(S',j) = h$ e $p(S',j) = (S,i)$;

4. Poni $k = k + 1$; se $k \leq n$ vai allo step 2;
5. L'ottimo è dato da $z^* = \underset{(X,j)\in\mathscr{L}_k}{Min}[f(X, j) + c_{ji}]$.

APPENDICE A

PROVA

A.1 Pippo