

# METODI ED ALGORITMI DI OTTIMIZZAZIONE PER IL PROBLEM SOLVING

Docente:  
Aristide Mingozzi

Adattamento:  
Edoardo Rosa

---

# INDICE

<b>1</b>	<b>Modelli e formulazioni matematiche</b>	<b>1</b>
1.1	The Traveling Salesman Problem . . . . .	2
1.1.1	Formulazioni Matematiche del TSP . . . . .	3
1.1.2	Eliminazione subtours di Miller, Tucker, Zemlin (1960) . . . . .	4
1.1.3	Il Traveling salesman problem con time windows (TSPTW) . . . . .	5
1.2	Project scheduling with resource constraints (PSR) . . . . .	7
1.2.1	Esempio di PSR . . . . .	7
1.2.2	Formulazione del PSR . . . . .	8
1.3	Fixed Charge Transportation Problem (FCTP) . . . . .	9
1.3.1	Descrizione del FCTP . . . . .	9
1.3.2	Formulazione del FCTP . . . . .	9
1.4	Assegnamento dei veicoli alle baie di carico . . . . .	11
1.4.1	Formulazione matematica . . . . .	11
1.5	Lot Sizing Problem . . . . .	13
1.5.1	Lot sizing senza vincoli di capacità . . . . .	13
<b>2</b>	<b>Introduzione alla programmazione lineare a numeri interi</b>	<b>17</b>
2.1	Arrotondamento ad una soluzione non-intera . . . . .	18
2.2	Unimodularità . . . . .	21
2.3	Metodo dei piani di taglio . . . . .	23
2.3.1	Piani di taglio . . . . .	23
2.3.2	Gomory cuts . . . . .	24
2.4	Metodi Branch and Bound . . . . .	29
2.4.1	Principio base dei metodi Branch and Bound . . . . .	29
2.4.2	Tipi di Branching . . . . .	31
2.4.3	Bounds . . . . .	32
2.4.4	Rilassamento lineare e surrogato . . . . .	38
2.5	Assegnamento Generalizzato . . . . .	39
2.5.1	Formulazione matematica . . . . .	39

2.5.2	Rilassamento lagrangiano	40
2.5.3	Algoritmo Branch & Bound	45
<b>3</b>	<b>Rilassamento Lagrangiano per il calcolo di lower bounds</b>	<b>46</b>
3.1	Rilassamento Lagrangiano di $P$ rispetto ai vincoli $Ax \geq b$	47
3.1.1	Esempio	47
3.2	Validità e importanza di $RL_u$	48
3.2.1	Esempio	48
3.3	TEOREMA: Dualità Lagrangiana debole	50
3.3.1	Dimostrazione	50
3.4	Lagrangiano Duale	51
3.5	Duality Gap	52
3.5.1	Esempio	52
3.6	TEOREMA: Dualità Lagrangiana Forte	53
3.6.1	Dimostrazione	53
3.6.2	Osservazioni	53
3.7	Caratterizzazione del Lagrangiano Duale	55
3.7.1	Definizione	55
3.7.2	TEOREMA	55
3.8	Lagrangiano Duale e Rilassamento Lineare	58
3.8.1	TEOREMA	58
3.8.2	Dimostrazione	58
3.8.3	TEOREMA: $L(u)$ è concava	58
3.9	Subgradiente di $L(u)$	60
3.9.1	Metodo del subgradiente	60
3.9.2	Vincoli Misti	62
3.9.3	Subgradiente per vincoli mist	62
3.10	Traveling Salesman Problem	63
3.10.1	Costi Simmetrici	63
3.10.2	Fomulazione Matematica (TSP Simmetrico)	63
3.10.3	Calcolo di $L(\lambda^0)$ per $\lambda^0 = 0$	65
3.10.4	Calcolo Penalità Lagrangiane	65
3.10.5	Rilassamento 1-TREE	69
3.10.6	Regola di branching TSP simmetrico	69
<b>4</b>	<b>Programmazione dinamica</b>	<b>71</b>
4.1	Motivazioni	72
4.1.1	Osservazione 1	72
4.1.2	Osservazione 2	72
4.1.3	Osservazione 3	72
4.2	Algoritmo	74
4.3	Algoritmo Forward (grafi aciclici)	75
4.4	Algoritmo di Bellman	76
4.4.1	Schema dell'algoritmo cammini minimi da 1 ad ogni $j \in V$	76
4.5	Knapsack 0-1	77
4.5.1	Esempio	77

4.5.2	Osservazione 1	77
4.5.3	Grafo dello spazio degli stati	79
4.5.4	Esempio	80
4.5.5	Ricorsione Froward - Knapsack 0-1	80
4.6	Programmazione a numeri interi	81
4.7	Programmazione Dinamica	82
4.8	Ricorsione di Programmazione Dinamica	83
4.9	Programmazione Dinamica: il TSP	84
4.9.1	Ricorsione per il calcolo di $f(\mathbf{S}, \mathbf{x}_i)$	84
4.9.2	Considerazioni computazionali	84
4.9.3	Esempio del TSP con 5 città	85
4.10	Ricorsione Forward per il TSP	89
4.11	Taglio 2-dimensionale a ghigliottina	90
4.11.1	Calcolo di $\mathbf{F}(\mathbf{x}, \mathbf{y})$	90
4.11.2	Inizializzazione	91
4.11.3	Esempio	91
4.11.4	Normal Cuts	92
4.12	Rilassamento dello spazio degli stati	93
4.12.1	Come ridurre lo spazio degli stati	93
4.12.2	Lower bound al cammino minimo	93
4.12.3	Sistema discreto multistadio	95
4.12.4	Rilassamento dello spazio degli stati per il TSP	96
4.12.5	Funzioni di mapping $\mathbf{g}(\cdot)$ per il TSP	97
4.12.6	Rilassamento q-path	97
4.12.7	Eliminazione dei loops di 2 vertici	98
4.12.8	Revers function per il TSP	99
4.12.9	Algoritmo DP+Lower Bound per il TSP	100
4.12.10	Algoritmo di programmazione dinamica per il TSP	100
<b>5</b>	<b>Metodi di decomposizione</b>	<b>102</b>
5.1	Dantzig-Wolfe Decomposition	103
5.1.1	Metodo di soluzione di $\mathbf{P}'$	103
5.1.2	Lower Bound	104
5.1.3	Esempio	105
5.1.4	Inizializzazione	106
5.2	Struttura Diagonale a blocchi di X	111
5.2.1	Metodo di soluzione	112
5.2.2	Lower Bound	112
5.3	Assegnamento Generalizzato	114
5.3.1	Decomposizione Dantzig-Wolfe del GAP	114
5.3.2	Duale di DP'	116
5.3.3	Algoritmo per risolvere LP'	117
5.4	Introduzione al simplesso Revisionato	119
5.4.1	Caso Semplice	119
5.5	Metodo del Simpleso Revisionato	123
5.5.1	Metodo del Simpleso in sintesi	123

5.6	Simpleso revisionato e metodo due fasi . . . . .	124
5.6.1	Fase 1 . . . . .	124
5.6.2	Fase 2 . . . . .	126
5.7	Simpleso tableau e Simpleso revisionato . . . . .	127
5.7.1	Occupazione di memoria . . . . .	127
5.7.2	Numero operazioni . . . . .	127

---

## ELENCO DELLE FIGURE

1.1	Grafo orientato . . . . .	4
1.2	Grafo H delle precedenze . . . . .	7
1.3	Esempio della rete di flusso (modello di Wagner-Whitin) . . . . .	15
1.4	. . . . .	15
1.5	. . . . .	16
2.3	* Problemi risolti . . . . .	32
4.1	Solo alcuni degli archi sono raffigurati . . . . .	88
4.2	Esempio di taglio a ghigliottina . . . . .	90
4.3	Sono equivalenti . . . . .	92
4.4	Il costo del cammino del grafo $G = (X, A)$ da $s$ a $t$ è 8 . . . . .	93

Copertina: [http://commons.wikimedia.org/wiki/File:Minimum\\_spanning\\_tree.svg](http://commons.wikimedia.org/wiki/File:Minimum_spanning_tree.svg)

---

## ELENCO DELLE TABELLE

2.1	Tableau ottimo. Soluzione continua!	26
2.2	Tableau ottimo.	27
5.1	Tableau di $P'$	117
5.2	1° Tableau	119
5.3	Tableau iterazione $t$	119





---

---

# 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 perché è la struttura di molti problemi pratici.

Si è soliti modellare 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.

$$\text{Min} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \quad (1.1)$$

$$\text{s.t.} \sum_{i \in N} x_{ij} = 1, \quad \forall j \in N \quad (1.2)$$

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

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

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

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:

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

#### 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  $(\alpha_l, \beta_l)$  con  $\alpha_l < \beta_l$ .

$A_i$  è il sottoinsieme degli indici degli archi che incidono sul vertice  $i$ :

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

Per una dato  $S \subset N$  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 indice  $l$  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.

$$\text{Min} \sum_{l=1} d_l x_l \quad (1.6)$$

$$\text{s.t.} \sum_{l \in A_i} x_l = 2, \forall i \in N \quad (1.7)$$

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

$$x_l \in \{0, 1\}, \quad l = 1, \dots, m \quad (1.9)$$

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

Sia  $u_i$  una variabile intera il cui valore rappresenta 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  $u_1 = 1, u_2 = 5, u_3 = 4, u_4 = 2, u_5 = 3$

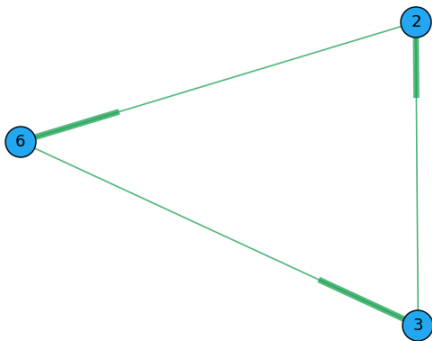
Miller, Tucker e Zemlin propongono in alternativa a:

$$\sum_{i \in S} \sum_{j \in N \setminus S} x_{ij} \geq 1, \quad \forall S \subset N \quad (1.10)$$

i seguenti vincoli:

$$u_i - u_j + n x_{ij} \leq n - 1, \quad i = 1, \dots, n, \quad j = 2, \dots, n \quad (1.11)$$

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



$$u_2 - u_6 + n \cdot x_{2,6} \leq n - 1$$

$$u_6 - u_3 + n \cdot x_{6,3} \leq n - 1$$

$$u_3 - u_2 + n \cdot x_{3,2} \leq n - 1$$

↓

$$3n \leq 3(n - 1)$$

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 ( $V = \{0, 1, \dots, n\}$ ).

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 visitato 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  $[r_i, d_i]$ ,  $\forall i \in V$ .

Si consideri la sequenza  $(0, i, \dots, i_{k-1}, i_k, \dots, i_n, 0)$  e sia  $t_{i_k}$  il tempo di arrivo al vertice  $i_k$ ,  $k = 0, 1, \dots, n + 1$ .

I tempi di arrivo sono calcolati come:

$$t_0 = 0 \quad (1.12)$$

$$t_{i_k} = \max\{(t_{i_{k-1}} + \theta_{i_{k-1} \cdot i_k}), r_{i_k}\} \quad (1.13)$$

#### 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  $j$  e  $x_{ij} = 0$  altrimenti.

$$\text{Min} \quad \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (1.14)$$

$$\text{s.t.} \quad \sum_{i \in A_j^-} x_{ij} = 1, \quad \forall j \in V \quad (1.15)$$

$$\sum_{j \in A_i^+} x_{ij} = 1, \quad \forall i \in V \quad (1.16)$$

$$t_i + \theta_{ij} - t_j \leq M(1 - x_{ij}), \quad \forall (i, j) \in A, j \neq 0 \quad (1.17)$$

$$t_i \leq d_i, \quad \forall i \in V \quad (1.18)$$

$$t_i \geq r_i, \quad \forall i \in V \quad (1.19)$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A \quad (1.20)$$

$$t_i \in \mathbb{N}^+, \quad \forall i \in V \quad (1.21)$$

dove

$$A_i^+ = \{j \in V : (i, j) \in A\}$$

$$A_i^- = \{j \in V : (i, j) \in A\}$$

$M$  un intero grande a piacere

$$r_0 = d_0 = 0$$

## 1.2 Project scheduling with resource constraints (PSR)

È dato un insieme  $\mathbb{X} = \{1, \dots, n\}$  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.

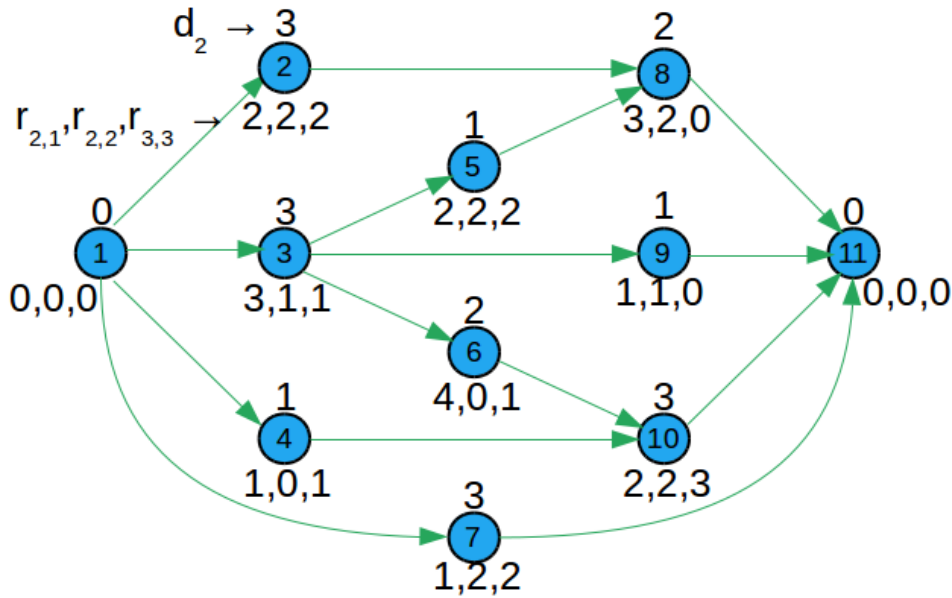
È dato un grafo  $G = (X, H)$  di precedenze, dove ogni arco  $(i, j) \in H$  impone che il job  $j$  può iniziare solo dopo che il job  $i$  è stato completato.

- 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.

Figura 1.2: Grafo  $H$  delle precedenze



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

### 1.2.2 Formulazione del PSR

Sia  $\xi_{it}$  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.

$$\text{Min} \sum_{t=1}^{T_{max}} t \xi_{nt} \quad (1.22)$$

$$\text{s.t.} \sum_{t=1}^{T_{max}} t \xi_{it} = 1, \quad i = 1, \dots, n \quad (1.23)$$

$$\sum_{t=1}^{T_{max}} t \xi_{jt} - \sum_{t=1}^{T_{max}} t \xi_{it} \geq d_i, \quad \forall (i, j) \in H \quad (1.24)$$

$$\sum_{i=1}^n r_{ik} \sum_{\tau=t-d_i+1}^t \xi_{i\tau} \leq b_k, \quad t = 1, \dots, T_{max} \text{ e } k = 1, \dots, m \quad (1.25)$$

$$\xi_{it} \in \{0, 1\}, \quad i = 1, \dots, n \text{ e } t = 1, \dots, T_{max} \quad (1.26)$$

Si osservi che:

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

#### 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:

$$\sum_{\tau=t-d_i+1}^t \xi_{i\tau} = 1 \text{ per } t = 3, 4, 5, 6 \text{ e } \sum_{\tau=t-d_i+1}^t \xi_{i\tau} = 0 \text{ per } t < 3 \text{ e } t > 6$$



### 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  $S = 1, 2, \dots, m$  è un insieme di  $m$  sorgenti e  $T = 1, 2, \dots, n$  è un insieme di  $n$  destinazioni.

Per ogni sorgente  $i \in S$  è disponibile 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  $f_{ij}$  per 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 matematica del FCTP è:

$$z(F0) = \min \sum_{i \in S} \sum_{j \in T} (c_{ij}x_{ij} + f_{ij}y_{ij}) \quad (1.27)$$

$$s.t. \quad \sum_{j \in T} x_{ij} = a_i, \quad i \in S \quad (1.28)$$

$$\sum_{i \in S} x_{ij} = b_j, \quad j \in T \quad (1.29)$$

$$x_{ij} \leq m_{ij}y_{ij}, \quad (i, j) \in A \quad (1.30)$$

$$x_{ij} \geq 0, \quad (i, j) \in A \quad (1.31)$$

$$y_{ij} \in \{0, 1\} \quad (1.32)$$

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 accettabile dei vincoli 1.28 e 1.29, 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;
- il tempo 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  $\tau = t, \dots, t + d_{ij} - 1$  (ovvero  $\tau \in T_j$  per ogni  $\tau = t, \dots, t + d_{ij} - 1$ ).

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  $\tau = t, \dots, t + d_{ij} - 1$ .

Sia  $c_{ijt}$  il costo per iniziare lo scarico del veicolo  $i \in N$  sulla linea  $j \in L_i$  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

Per ogni  $i \in N$ ,  $j \in L_i$  e  $t \in I_{ij}$  poniamo  $\delta_{ijt\tau} = 1$  per  $\tau = t, \dots, t + d_{ij} - 1$  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  $\tau$ .

La formulazione matematica  $F$  del problema è la seguente.

$$z(F) = \min \sum_{j \in L} \sum_{i \in N_j} \sum_{t \in I_{ij}} c_{ijt} + x_{ijt} + \sum_{j \in L} \sum_{\tau \in T_j} g_{j\tau} s_{j\tau} \quad (1.33)$$

$$s.t. \quad \sum_{j \in L_i} \sum_{t \in I_{ij}} x_{ijt} = 1, \quad i \in N \quad (1.34)$$

$$\sum_{i \in N_j} \sum_{t \in I_{ij}} \delta_{ijt\tau} x_{ijt} + s_{j\tau} = 1, \quad j \in L, \tau \in T_j \quad (1.35)$$

$$x_{ijt} \in 0, 1, \quad i \in N, j \in L_i, t \in I_{ij} \quad (1.36)$$

$$s_{j\tau} \in 0, 1, \quad j \in L, \tau \in T_j \quad (1.37)$$

Il vincolo 1.34 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 1.35 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 purpose.

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  $t = 1, \dots, T$  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 prodotte al fine di soddisfare la domanda in ciascun periodo.

Si suppone che la quantità prodotta nel periodo  $t$  sia subito disponibile e che la quantità non venduta alla fine di ogni mese venga 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  $t = 1, \dots, T$ :

$x_t$  quantità prodotta all'inizio del periodo  $t$ ;

$I_t$  livello del magazzino alla fine del periodo  $t$ ;

$y_t \in \{0, 1\} : y_t = 1$  se nel periodo  $t$  vi è produzione,  $y_t = 0$  altrimenti.

$$\text{Min } z = \sum_{t=1}^T (p_t x_t + h_t I_t + A_t y_t) \quad (1.38)$$

$$x_t + I_{t-1} = I_t + d_t, \quad t = 1, \dots, T \quad (1.39)$$

$$x_t \leq M y_t, \quad t = 1, \dots, \quad (1.40)$$

$$x_t, I_t \geq 0, \quad t = 1, \dots, T \quad (1.41)$$

$$y_t \in \{0, 1\}, \quad t = 1, \dots, T \quad (1.42)$$

$$\text{dove } M = \sum_{t=1}^T d_t \text{ e, per semplicità, si suppone che } I_0 = 0. \quad (1.43)$$

### 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.

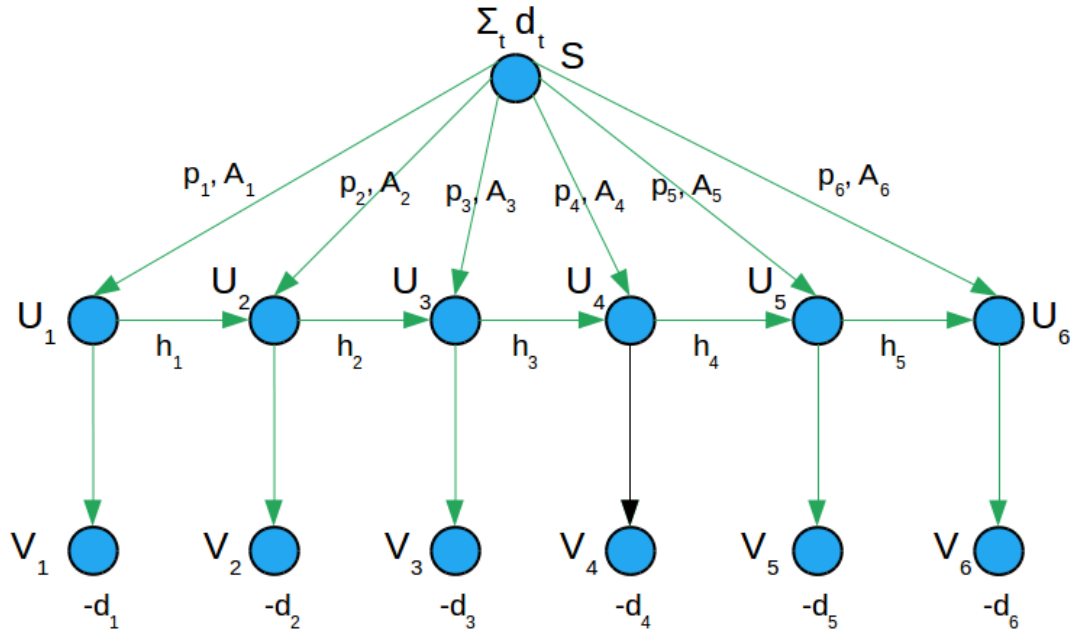
Per ciascun periodo  $t = 1, \dots, T$  vi sono gli archi:

$(S, U_t)$  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)



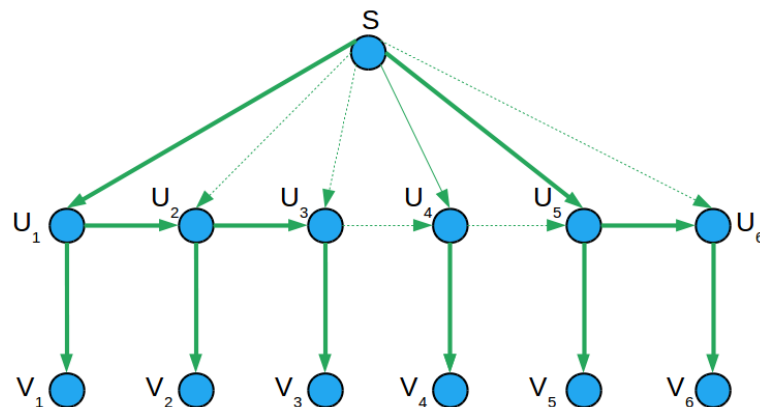
### 1.5.1.3 Proprietà della soluzione ottima

**Teorema.**

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

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

Figura 1.4



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

Si costruisca un grafo aciclico di  $T + 1$  vertici.

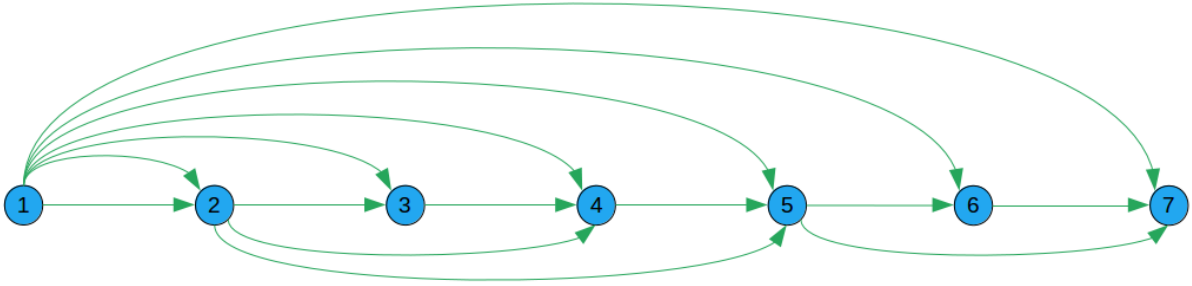
Si definiscano gli archi  $(j, k)$  per  $j = 0, \dots, T - 1$  e  $k = j + 1, \dots, T$ .

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

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:

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

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.

$$\begin{array}{ll} \text{Min} & cx \\ & Ax = b \\ & x \geq 0 \\ & x \text{ intero} \end{array}$$

Le variabili devono assumere valori interi:

*Es :*  $x_i$  = Numero di uomini che devono essere assegnati al lavoro  $i$ .  
      = Numero di automezzi che devono operare il trasporto lungo la "tratta  $i$ "  
      = Numero di macchine da utilizzare nella lavorazione  $i$

## 2.1 Arrotondamento ad una soluzione non-intera

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

$$\begin{aligned}
 \text{Es : } \quad & \text{Min } z = -2x_1 + 3x_2 \\
 & x_1 + x_2 \geq 3 \\
 & 3x_1 + x_2 \leq 6 \\
 & x_2 \leq 5 \\
 & x_1, x_2 \geq 0 \text{ ed intere}
 \end{aligned}$$

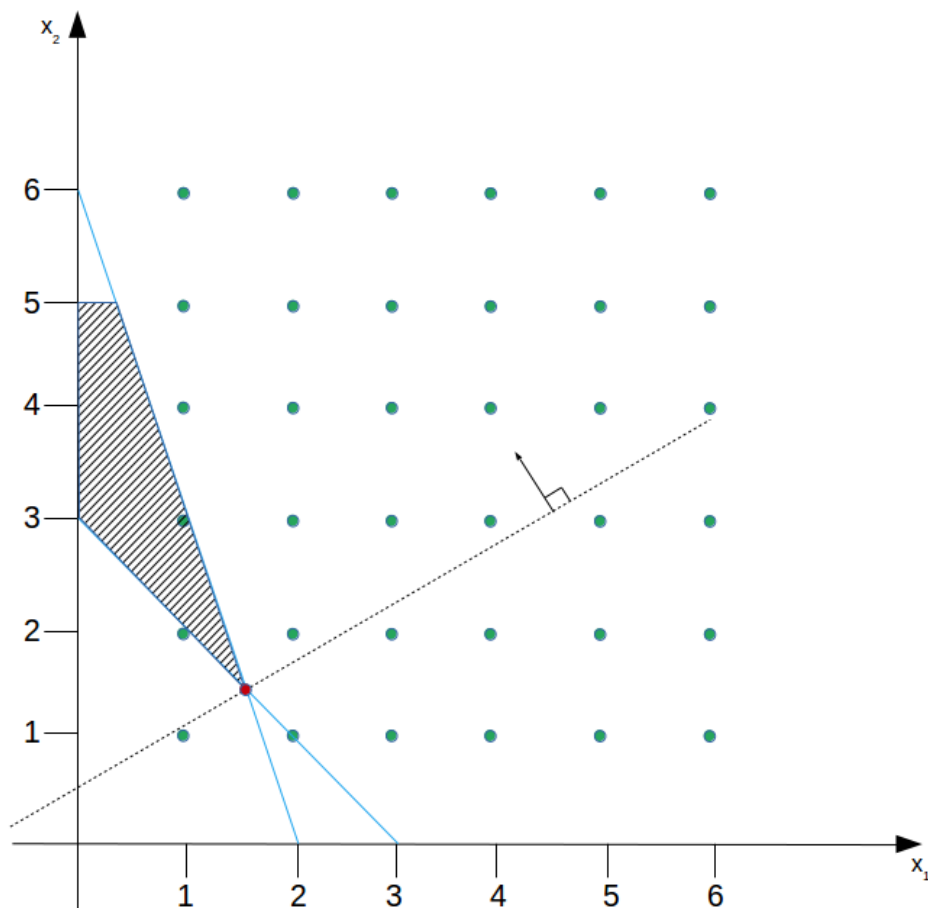


Figura 2.1:  
 Soluzione continua:  $z = \frac{3}{4}$ ;  $x_1 = \frac{3}{2}$ ,  $x_2 = \frac{3}{2}$   
 Soluzione intera:  $z = 4$ ;  $x_1 = 1$ ,  $x_2 = 2$

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

$$\begin{aligned} \text{Es : } \quad & \text{Min } z = 8x_1 + 6x_2 \\ & 4x_1 + 3x_2 \geq 6 \\ & x_1, x_2 \geq 0 \text{ ed intere} \end{aligned}$$

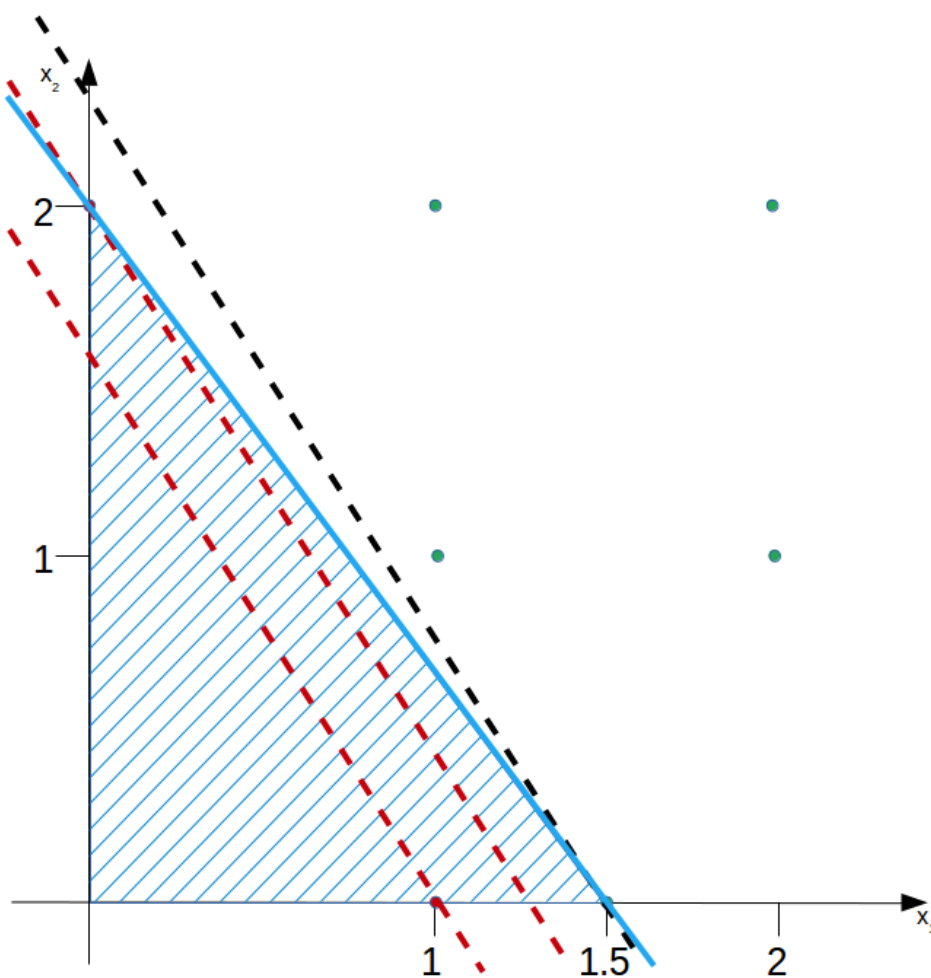


Figura 2.2:

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.

The diagram shows a 2D coordinate system with axes  $x_1$  and  $x_2$ . A grid of green dots represents integer points. A blue hatched quadrilateral represents the feasible region of the continuous problem. A red dot at the top-right vertex of this region is labeled "Soluzione ottima continua" (Continuous optimal solution). A blue dot on the upper boundary of the feasible region is labeled "Soluzione ottima intera" (Integer optimal solution). An arrow points from the text "La funzione decresce" (The function decreases) towards the top-right, indicating the direction of the objective function.

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:

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

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 elementi 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.

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

$$\begin{array}{l} R = 1, 2, 3, 4 \\ R_1 = 1, 2, 3, 4 \\ R_2 = \emptyset \end{array} \quad \begin{array}{l} R = 1, 2, 3, 4, 5 \\ R_1 = 1, 2, 3 \\ R_2 = 4, 5 \end{array}$$

La totale unimodularità della matrice  $A$  è **condizione sufficiente** affinché la soluzione ottima  $x^*$  sia intera per

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

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

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

Supponiamo  $A, c, b$  interi.

Si consideri il problema rilassato che si ottiene da  $ILP$  ignorando i vincoli di interezza.

Indichiamo tale problema con  $LP$ .

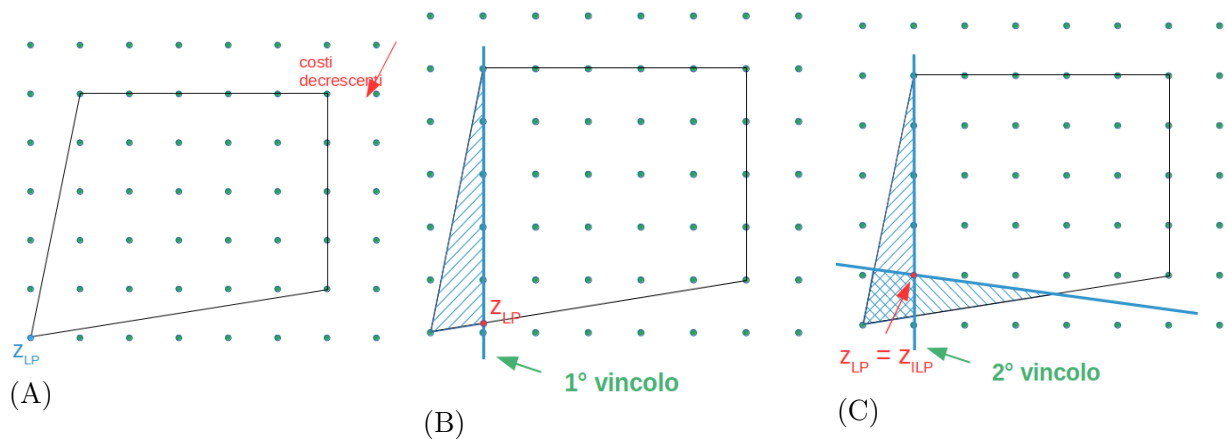
$$LP \begin{cases} z_{LP} = \text{Min } cx \\ ax = b \\ x \geq 0 \end{cases}$$

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:

$$x_r + \sum_{j=m+1}^n y_r^j x_j = \bar{b}_r \quad (2.1)$$

Poniamo:

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

Inoltre:

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

Sostituendo, la 2.1 diviene:

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

o anche:

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

Ne segue:

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

La soluzione corrente non soddisfa il vincolo 2.1 in quanto  $x_j = 0, j = m+1, \dots, n$  mentre  $F_r > 0$  poiché  $\bar{b}_r$  si è supposto non intero.

Se il vincolo 2.1 viene aggiunto al problema LP allora la soluzione corrente risulterà non ammissibile.

Per determinare una nuova soluzione che soddisfi il vincolo 2.1 può essere impiegato il *Simplesso Duale* partendo dal tableau ottimo relativo alla soluzione corrente.



Al tableau va aggiunto il vincolo:

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

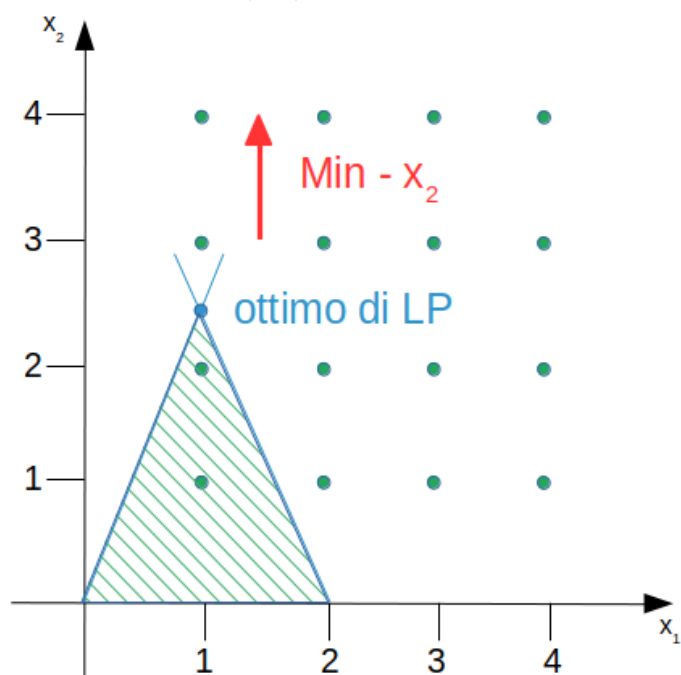
La nuova variabile slack  $s$  è una nuova variabile base.

Il nuovo tableau è non ammissibile per il Primale ma duale ammissibile.

**Esempio.**

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

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:

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

Dalla riga di  $x_2$  si ha la seguente equazione:

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

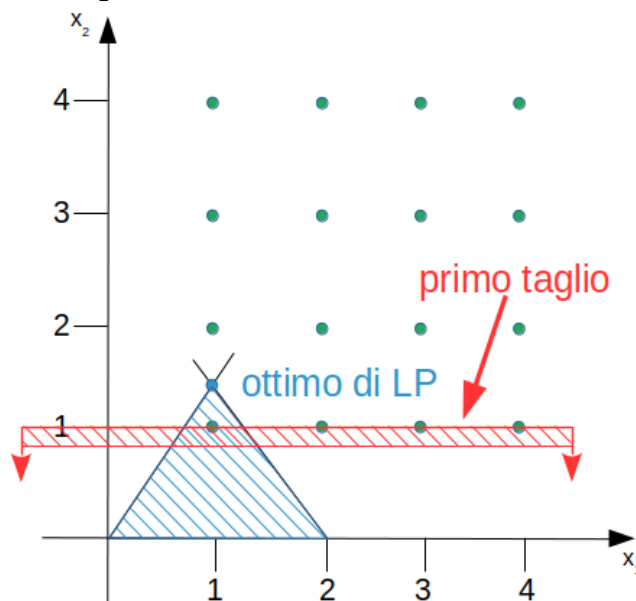
da cui:

$$-\frac{1}{4}x_3 - \frac{1}{4}x_4 + s_1 = -\frac{1}{2} \quad (2.2)$$

Si osservi che per definizione di  $x_3$  e  $x_4$  si ha

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

Sostituendo in 2.2 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 il cut:

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

Il nuovo tableau, quindi, diviene:

		$x_1$	$x_2$	$x_3$	$x_4$	$s_1$	$s_2$	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

e ottimizzando con il simplesso duale:

		$x_1$	$x_2$	$x_3$	$x_4$	$s_1$	$s_2$	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.

L'algoritmo converge in un numero finito di passi purché venga impiegata un'appropriata regola lessicografica per la scelta del pivot.

**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

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

$$S_0 = \{x : Ax = b, x \geq 0 \text{ intero}\}$$

### 2.4.1 Principio base dei metodi Branch and Bound

Suddividere il problema  $P_0$  nei sottoproblemi  $P_1, P_2, \dots, P_k$  a cui corrispondono gli insiemi di soluzioni ammissibili  $S_1, S_2, \dots, S_k$ . La suddivisione è tale per cui:

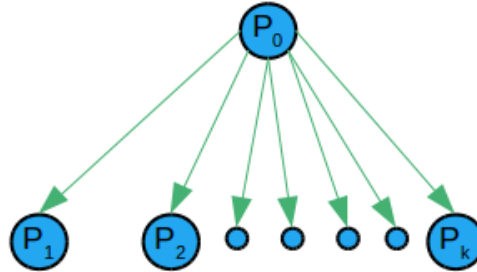
$$S_1 \cup S_2 \cup \dots \cup S_k = S_0$$

È evidente che:

$$\min_{x \in S_0} cx = \text{MIN} \left\{ \min_{x \in S_1} cx, \min_{x \in S_1} cx, \dots, \min_{x \in S_k} cx \right\}$$

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

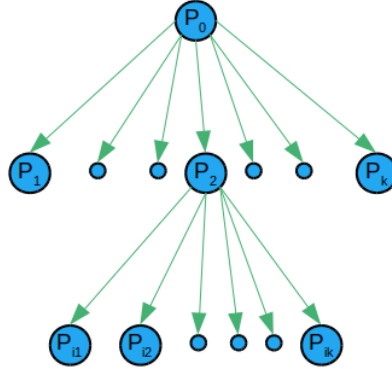
Possiamo rappresentare la suddivisione di  $P_0$  in  $P_1, P_2, \dots, P_k$  mediante un albero



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  $P_{i1}, P_{i2}, \dots, P_{ik}$  a cui corrispondono gli insiemi di soluzioni ammissibili  $S_{i1}, S_{i2}, \dots, S_{ik}$  tali che  $S_{i1} \cup S_{i2} \cup \dots \cup S_{ik} = S_i$

Si ha il seguente albero



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

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  $P_{i1}, P_{i2}, \dots, P_{ir}$  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.

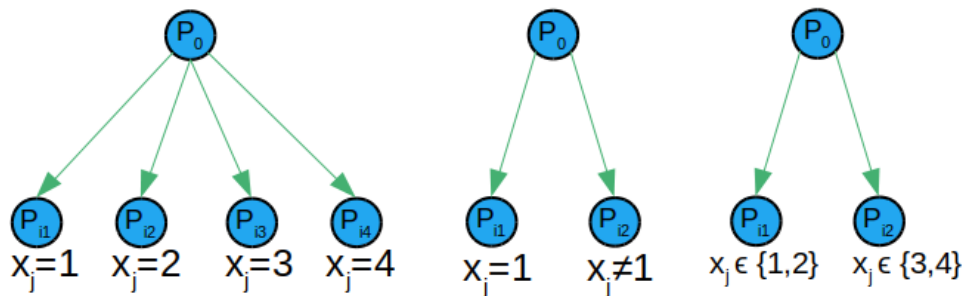
$$S_{i\alpha} \cap S_{i\beta} = \emptyset$$

Se la condizione sopra è soddisfatta allora  $\{S_{i1}, \dots, S_{ir}\}$  è una **PARTIZIONE** di  $S_i$ .

Si noti che la condizione non è *necessaria* ma rende computazionalmente efficiente il processo di branching.

### 2.4.1.1 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.

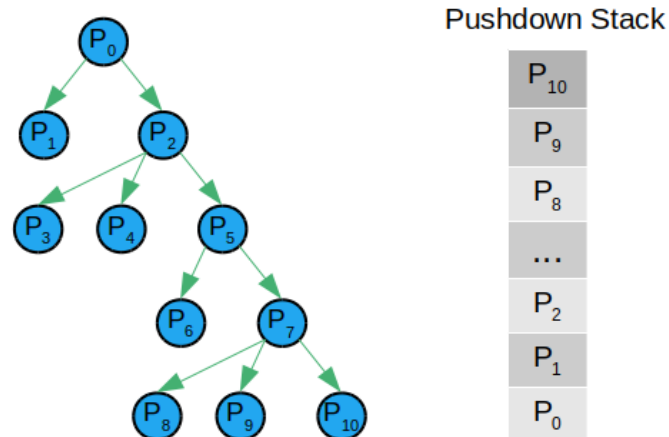


### 2.4.2 Tipi di Branching

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

#### 2.4.2.1 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 penultimo sottoproblema generato e su questo viene effettuato il branching.

#### 2.4.2.2 Breadth-first search

Il branching procede da livello a livello, ovvero il problema  $P_0$  è suddiviso in  $P_1, P_2, \dots, P_k$  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  $K - 1$ .

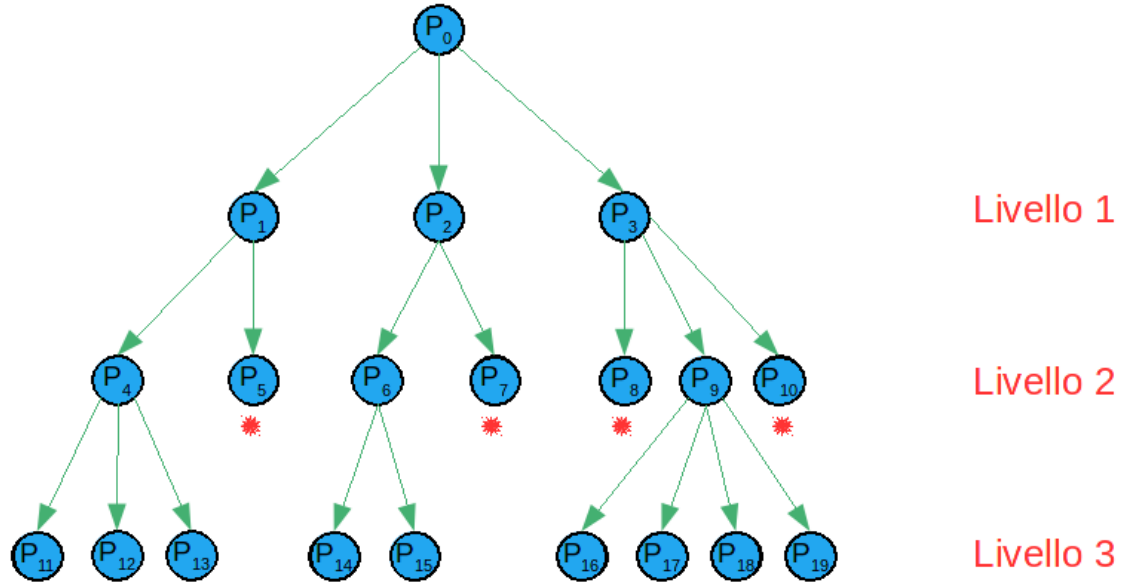


Figura 2.3: \* Problemi risolti

### 2.4.3 Bounds

La ricerca della soluzione ottima di  $P_0$  è completa quando sono stati risolti tutti i sottoproblemi generati.

Questo processo può essere migliorato calcolando, per ogni sottoproblema  $P_j$  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

$$LB_i \leq \min_{x \in S_i} \{cx\}$$

**Upper Bound:** diremo che  $UB_i$  è un upper bound al sottoproblema  $P_0$  se

$$UB_i \geq \min_{x \in S_0} \{cx\}$$

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



## 2.4.3.1 Calcolo del Lower Bound

Sia dato il problema

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

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

I seguenti metodi producono validi Lower Bounds a  $P_0$ .

## 2.4.3.2 Rilassamento continuo

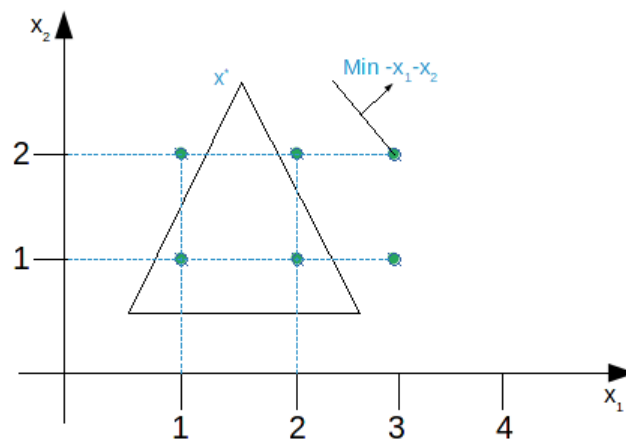
Si ignori il vincolo  $x$  intero; il problema risultante è risolvibile con la programmazione lineare.

Sia  $z_{LP}^*$  il costo di tale soluzione; si ha

$$z_{LP}^* \leq z^*$$

- 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.

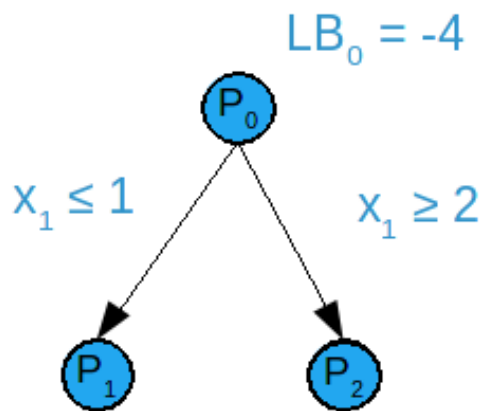
**Esempio.**



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

$P_0$  è suddiviso in due sottoproblemi  $P_1$  e  $P_2$

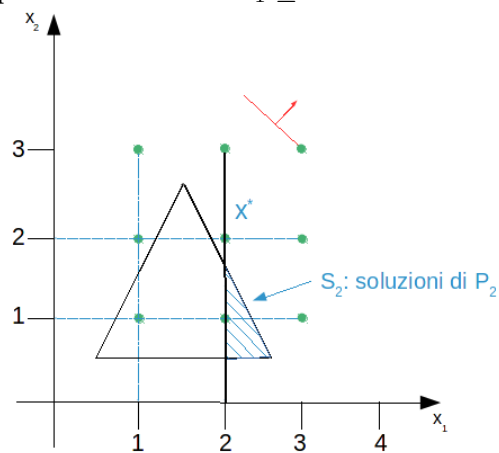
- $P_1$  imponiamo che  $x_1 \leq 1$
- $P_2$  imponiamo che  $x_1 \geq 2$



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

### Esame del sottoproblema $P_2$

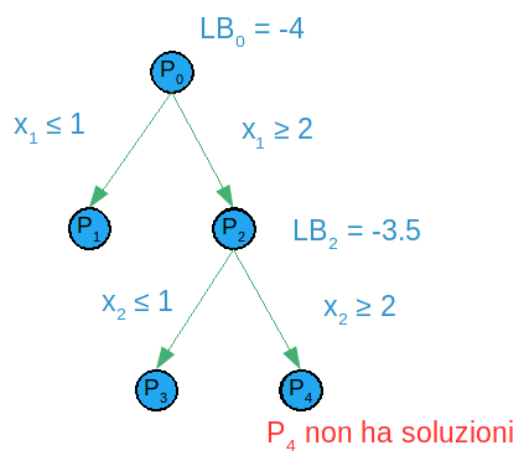
Il lower bound si ottiene imponendo il vincolo  $x_1 \geq 2$ .



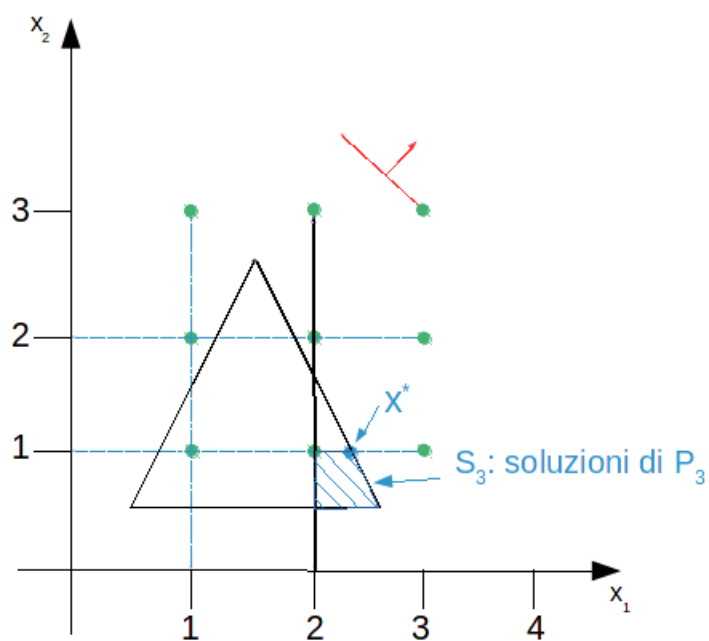
L'ottimo per  $P_2$  è  $z_{LP}^* = -3.5$  con componenti  $x_1^* = 2$  e  $1 < x_2^* < 2$

Il problema  $P_2$  viene suddiviso in  $P_3$  e  $P_4$  dove

- $P_3$  imponiamo  $x_1 \geq 2$  e  $x_2 \leq 1$
- $P_4$  imponiamo  $x_1 \geq 2$  e  $x_2 \geq 2$



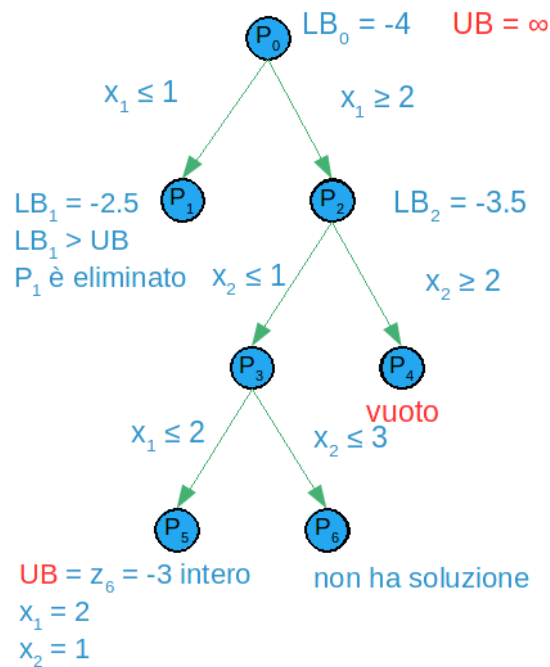
Esame del sottoproblema  $P_3$



L'ottimo di  $P_3$  è  $z_{LP}^* = -3.25$  con componenti  $x_2^* = 1$  e  $2 < x_1^* < 3$ .

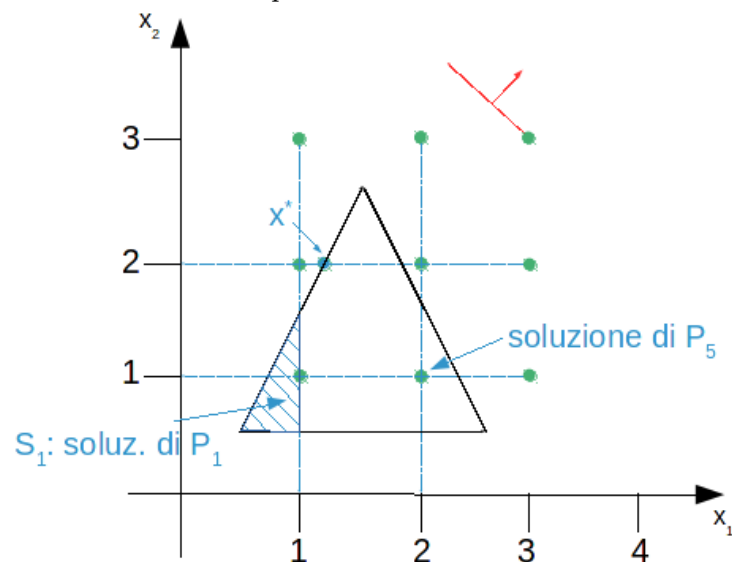
Il problema  $P_3$  viene suddiviso in  $P_5$  e  $P_6$  dove

- $P_5$  imponiamo  $x_1 \geq 2$ ,  $x_2 \leq 1$  e  $x_1 \leq 2$
- $P_6$  imponiamo  $x_1 \geq 2$ ,  $x_2 \leq 2$  e  $x_1 \geq 3$



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.

## 2.4.3.3 Eliminazione di alcuni vincoli

Si consideri il problema

$$P_0 \left\{ \begin{array}{l} \text{Min } z = cx \\ Ax = b \quad m_1 \text{ vincoli} \\ Dx = h \quad m_2 \text{ vincoli} \\ x \geq 0 \quad \text{intero} \end{array} \right.$$

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

$$RP \left\{ \begin{array}{l} \text{Min } z_{RP} = cx \\ Dx = h \\ x \geq 0 \text{ e intero} \end{array} \right.$$

Sia  $z_{RP}^*$  il valore ottimo di RP; si ha

$$z_{RP}^* \leq z^* \quad (z^* \text{ ottimo di } P_0)$$

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

## 2.4.3.4 Rilassamento Surrogato

Sia dato il problema

$$P_0 \left\{ \begin{array}{l} \text{Min } z = cx \\ \sum_{j=1}^n a_{ij}x_j \geq b_i \quad i = 1, \dots, m_1 \\ Dx = h \quad m_2 \text{ vincoli} \\ x \geq 0 \quad \text{intero} \end{array} \right.$$

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

$$SP \left\{ \begin{array}{l} \text{Min } z_{SP} = cx \\ \sum_{j=1}^{m_1} \pi_i \sum_{j=1}^n a_{ij}^j \geq \sum_{i=1}^{m_1} \pi_i b_i \quad \pi_i \geq 0 \\ Dx = h \quad m_2 \text{ vincoli} \\ x \geq 0 \quad \text{intero} \end{array} \right.$$

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

### 2.4.4 Rilassamento lineare e surrogato

$$P \left\{ \begin{array}{l} z^* = \text{Min } cx \\ \text{s.t. } Ax \geq b \\ x \geq 0 \text{ intero} \end{array} \right.$$

Rilassamento lineare

$$LP \left\{ \begin{array}{ll} z_{LP}^* = \text{Min } cx & = \text{Max } \omega b \\ \text{s.t. } Ax \geq b & \omega A \leq c \\ x \geq 0 \text{ intero} & \omega \geq 0 \end{array} \right.$$

Sia  $\omega^*$  la soluzione duale ottima.

$$SP \left\{ \begin{array}{l} z_{SP}^* = \text{Min } cx \\ (\omega^* A)x \geq \omega^* b \\ x \geq 0 \text{ intero} \end{array} \right.$$

Ottenuto tramite **rilassamento surrogato**.

**Teorema.**

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

$$z_{LP}^* = \omega^* b \quad \text{e} \quad c - \omega^* A x^*$$

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

$$(c - \omega^* A)x^* \geq 0 \quad \text{ovvero} \quad cx^* \geq \omega^* Ax^* \quad (2.3)$$

ma anche (da SP):

$$\omega^* Ax^* \geq \omega^* b \quad (2.4)$$

Da 2.3 e 2.4:

$$cx^* \geq \omega^* Ax^* \geq \omega^* b = z_{LP}^*$$

ovvero

$$z_{SP}^* = cx^* \geq z_{LP}^*$$

□

## 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  $i$ ,  $i = 1, \dots, m$

$a_{ij}$  : 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

$$\text{Min} \quad \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (2.5)$$

$$\sum_{i=1}^m x_{ij}, \quad j = 1, \dots, n \quad (2.6)$$

$$\sum_{j=1}^n a_{ij} x_{ij}, \quad i = 1, \dots, m \quad (2.7)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \quad (2.8)$$

Dove:

**2.6** ogni oggetto  $j$  deve essere assegnato ad un solo contenitore

**2.7** 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 2.6

$$\begin{aligned}
 L(u) &= \text{Min} \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} - \overbrace{\sum_{j=1}^n u_j \left( \sum_{i=1}^m x_{ij} - 1 \right)}^{-u(Ax-b)} = \\
 L(u) &= \text{Min} \sum_{i=1}^m \left( \sum_{j=1}^n (c_{ij} - u_j) x_{ij} \right) + \sum_{j=1}^n u_j \\
 &\quad \text{s.t.} \sum_{j=1}^n a_{ij} x_{ij} \leq b_i, \quad i = 1, \dots, m \\
 &\quad x_{ij} \in \{0, 1\}
 \end{aligned}$$

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

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

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)

$$L(u^0) = \text{Min}_x \sum_{i=1}^m \left( \sum_{j=1}^n (c_{ij} - u_j^0) x_{ij} \right) + \sum_{j=1}^n u_j^0$$

$$L(u^0) = \text{Min}_x 3x_{11} + 3x_{12} + 4x_{13} + 9x_{14} + 2x_{21} + 6x_{22} - 9x_{23} + 3x_{24} + 0$$

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

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

$$x_{11}, \dots, x_{24} \in \{0, 1\}$$



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

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

Soluzione ottima:  $z_1 = 0$ ;  $x_{11}^0 = x_{12}^0 = x_{13}^0 = x_{14}^0 = 0$

$$2^0 \text{ problema } \begin{cases} z_2 = \text{Min } 2x_{21} + 6x_{22} - 9x_{23} + 3x_{24} \\ 3x_{21} + x_{22} + 6x_{23} + 4x_{24} \leq 12 \end{cases}$$

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

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

La soluzione di  $L(u^0)$  non soddisfa i vincoli 2.6; infatti:

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

$$x^0 = (0, 0, 0, 0, 0, 0, 1, 0)$$

$$\text{Vincoli 2.6} \begin{cases} x_{11} + x_{21} = 1 & (j=1) \\ x_{12} + x_{22} = 1 & (j=2) \\ x_{13} + x_{23} = 1 & (j=3) \\ x_{14} + x_{24} = 1 & (j=4) \end{cases}$$

Si noti che la soluzione di  $L(u^0)$  viola i vincoli 2.6 per  $j = 1, 2, 4$  mentre soddisfa quello per  $j = 3$ .

**Aggiornamento delle penalità  $\{u\}$**

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

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

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

$$u_1^1 = u_1^0 - 2 \cdot \frac{12}{3} \cdot (-1) = 0 + 8 = 8$$

$$u_2^1 = u_2^0 - 2 \cdot 4 \cdot (-1) = 0 + 8 = 8$$

$$u_3^1 = u_3^0 - 2 \cdot 4 \cdot (0) = 0 + 0 = 0$$

$$u_4^1 = u_4^0 - 2 \cdot 4 \cdot (-1) = 0 + 8 = 8$$

Calcolo di  $L(u^1)$

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

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

$$u^1 = (8, 8, 0, 8)$$

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

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

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

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

Soluzione ottima:  $z_2 = -17$ ;  $x_{21}^1 = x_{22}^1 = x_{23}^1 = 1$ ;  $x_{24}^1 = 0$ .

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

I vincoli 2.6 sono violati dalla soluzione  $x^1$  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  $L(u^1) > L(u^0)$

$$u_j^k = u_j^{k-1} - \alpha_2 \cdot \frac{(z^* - L(u^{k-1}))}{\sum_{j=1}^n (\sum_{i=1}^m x_{ij} - 1)^2} \cdot (\sum_{i=1}^m x_{ij} - 1) \quad (2.9)$$

$$u_1^2 = u_1^1 - 2 \cdot \frac{6}{3} \cdot (1) = 8 - 4 = 4 \quad (2.10)$$

$$u_2^2 = u_2^1 - 2 \cdot 2 \cdot (1) = 8 - 4 = 4 \quad (2.11)$$

$$u_3^2 = u_3^1 - 2 \cdot 2 \cdot (0) = 0 - 0 = 0 \quad (2.12)$$

$$u_4^2 = u_4^1 - 2 \cdot 2 \cdot (-1) = 8 + 4 = 12 \quad (2.13)$$

Calcolo di  $L(u^2)$

$$L(u^2) = \text{Min} \sum_{i=1}^m \left( \sum_{j=1}^n (c_{ij} - u_j^2) x_{ij} \right) + \sum_{j=1}^n u_j^1 \quad (2.14)$$

$$L(u^2) = z_1 + z_2 + 20 \quad (2.15)$$

dove  $z_1$  e  $z_2$  sono i valori ottimi dei problemi sequenti

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

$$u^2 = (4, 4, 0, 12)$$

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

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

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

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

$$u_j^3 = u_j^2 - \alpha_3 \cdot \frac{(z^* - L(u^2))}{\sum_{j=1}^n (\sum_{i=1}^m x_{ij} - 1)^2} \cdot \left( \sum_{i=1}^m x_{ij} - 1 \right) \quad (2.16)$$

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

$$u_1^3 = u_1^2 - 6 \cdot (0) = 4 \quad (2.17)$$

$$u_2^3 = u_2^2 - 6 \cdot (0) = 4 \quad (2.18)$$

$$u_3^3 = u_3^2 - 6 \cdot (0) = 0 \quad (2.19)$$

$$u_4^3 = u_4^2 - 6 \cdot (1) = 6 \quad (2.20)$$

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

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

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

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

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

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 2.7

$$\begin{aligned} L(u) &= \text{Min} \overbrace{\sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij} - \sum_{i=1}^m \lambda_i \left( \sum_{j=1}^n a_{ij}x_{ij} - b_i \right)}^{cx - \lambda(Ax - b)} = \\ L(u) &= \text{Min} \sum_{j=1}^n \left( \sum_{i=1}^m (c_{ij} - \lambda_i a_{ij})x_{ij} \right) + \sum_{i=1}^m \lambda_i b_i \\ &\quad \sum_{i=1}^m x_{ij} = 1 \\ &\quad x_{ij} \in \{0, 1\} \end{aligned}$$

L'ottimo si ottiene ponendo, per ogni  $j$

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

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

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

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

$b = (30, 12)$

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

$$\begin{aligned} L(\lambda^0) &= \text{Min} \sum_{j=1}^n \left( \sum_i (c_{ij} - \lambda_i a_{ij})x_{ij} + \sum_{i=1}^m \lambda_i b_i \right) \\ L(\lambda^0) &= 3x_{11} + 3x_{12} + 4x_{13} + 9x_{14} + 2x_{21} + 6x_{22} - 9x_{23} + 3x_{24} + 0 \\ &\quad \begin{cases} x_{11} & & + x_{21} & & = 1 \\ & x_{12} & & + x_{22} & = 1 \\ & & x_{13} & & + x_{23} & = 1 \\ & & & x_{14} & & + x_{24} & = 1 \end{cases} \end{aligned}$$

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

Dal 1° vincolo  $x_{11} = 0, x_{21} = 1$

Dal 2° vincolo  $x_{12} = 1, x_{22} = 0$

Dal 3° vincolo  $x_{13} = 0, x_{23} = 1$

Dal 4° vincolo  $x_{14} = 0, x_{24} = 1$

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

Soluzione ottenuta per  $L(u^0)$

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

$x^1 = (0, 1, 0, 0, 1, 0, 1, 1)$

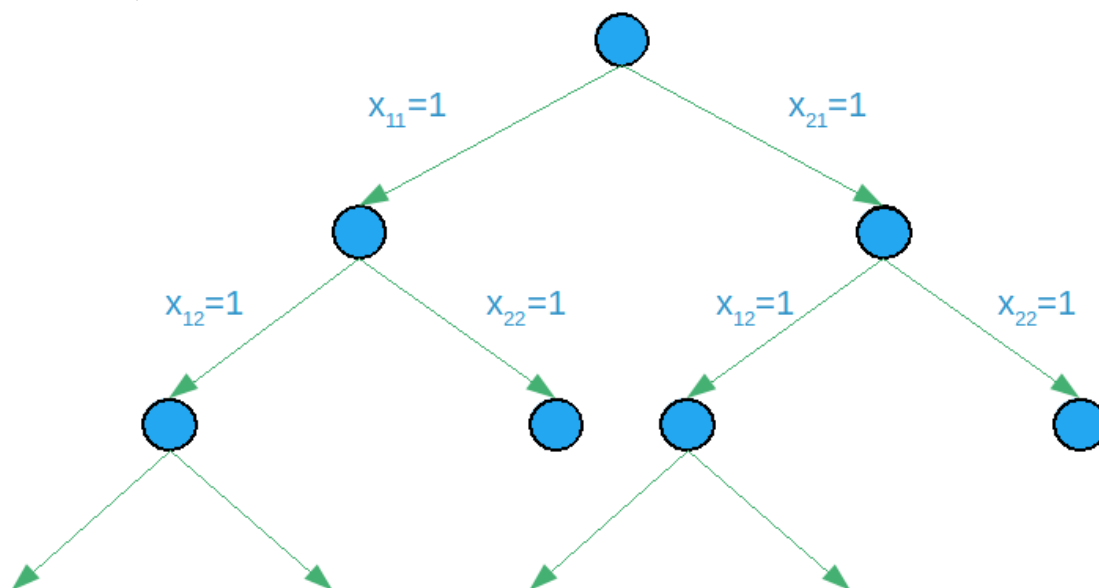
$5x_{11} + 7x_{12} + 4x_{13} + 2x_{14} \leq 30$  soddisfatto

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

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

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  $0 \leq x \leq 1$ :

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

In molti casi:

- è proibitivo 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**.

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

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

#### 3.1.1 Esempio

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

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

### 3.2 Validità e importanza di $RL_u$

Possiamo dimostrare che  $L(u) \leq z(P)$ ,  $\forall u \geq 0$  e quindi  $\max_{u \geq 0} [L(u)] \leq z(P)$ .

In certe condizioni la soluzione ottima di  $RL_u$  è anche la soluzione ottima di  $P$ .

#### 3.2.1 Esempio

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

La soluzione ottima è  $x_1 = x_2 = 1$ ,  $x_3 = x_4 = 0$  e  $z(P) = 5$ .

Il rilassamento lagrangiano dei tre vincoli richiede tre moltiplicatori  $u_1, u_2, u_3$ ; quindi

$$(RL_u) \quad L(u) = \text{Min } 2x_1 + 3x_2 + 4x_3 + 5x_4 - u_1(x_1 + x_3 - 1) \quad (3.2)$$

$$- u_2(x_1 + x_4 - 1) \quad (3.3)$$

$$- u_3(x_2 + x_3 + x_4 - 1) \quad (3.4)$$

$$\text{s.t. } x_i \in \{0, 1\}, i = 1, \dots, 4 \quad (3.5)$$

ma anche

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

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

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  $u_3 = 2.2$

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

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  $u_3 = 3$

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

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

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

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

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

#### 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

$$cx^* - u(Ax^* - b) \geq L(u) \quad (3.8)$$

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

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

## 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:

$$D_L \quad z(D_L) = \max_{u \geq 0} [L(u)] \quad (3.10)$$

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:

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

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

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

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

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

Per calcolare  $z(D_L) = \max_{u \geq 0} [L(u)]$  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:

$$A\bar{x} \geq b \quad (3.12)$$

$$\bar{u}(A\bar{x} - b) = 0 \quad (3.13)$$

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

#### 3.6.1 Dimostrazione

Dimostriamo 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

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

Per il teorema della dualità Lagrangiana debole si ha:

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

Quindi da 3.14 e 3.15 si ottiene

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

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:

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

Abbiamo dimostrato che se valgono 3.12 e 3.13 allora

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

Quindi da 3.17 e 3.18 si ottiene

$$z(D_L) = z(P) \quad (3.19)$$

□

#### 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.

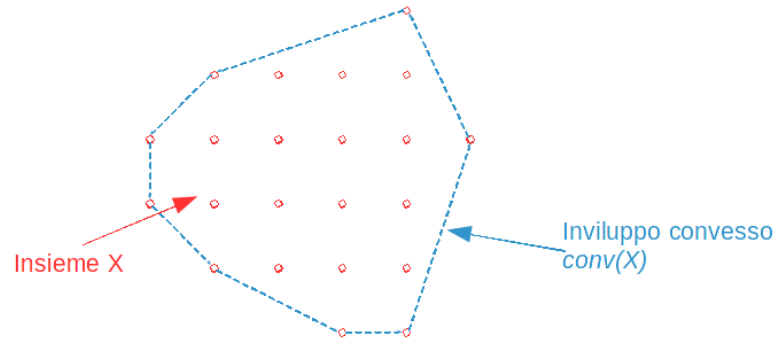
- Che relazione esiste tra  $z(D_L)$  e  $z(LP)$  il valore del Rilassamento Lineare di  $P$ ?

## 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  $\text{conv}(X)$  l'involuppo convesso di tutti i punti di  $X$  (ovvero,  $\text{conv}(X)$  è l'intersezione di tutti gli insiemi convessi che contengono  $X$ ).



Si osservi che l'ottimo del problema Lagrangiano:

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

corrisponde ad un punto estremo di  $\text{conv}(X)$

### 3.7.2 TEOREMA

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

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

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

#### 3.7.2.1 Dimostrazione

Si ricordi che:

$$D_L \quad z(D_L) = \text{Max}_{u \geq 0} [L(u)]$$

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

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

Quindi, il problema  $D_L$  può essere scritto come:

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

o anche

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

poichè  $L(u)$  raggiunge l'ottimo in un punto estremo di  $\text{conv}(X)$ .

Indichiamo con  $x^i$ ,  $i = 1, \dots, t$  i punti estremi di  $\text{conv}(X)$ . Il problema  $D_L$  può essere scritto come:

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

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

$$D_L \quad \left\{ \begin{array}{l} z(D_L) = \text{Max } v \\ \text{s.t. } v \leq cx^i - u(Ax^i - b), \quad i = 1, \dots, t \\ v \text{ qualsiasi} \\ u \geq 0 \end{array} \right.$$

Il duale di questo problema è il seguente

$$DD_L \quad \left\{ \begin{array}{l} z(D_L) = \text{Min } \sum_{i=1}^t \lambda_i (cx^i) \\ \text{s.t. } \sum_{i=1}^t \lambda_i = 1 \\ \sum_{i=1}^t \lambda_i (Ax^i - b) \geq 0 \\ \lambda_i \geq 0, \quad i = 1, \dots, t \end{array} \right.$$



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

$$DD_L \left\{ \begin{array}{l} z(D_L) = \text{Min } c(\sum_{i=1}^t \lambda_i x^i) \\ s.t. \sum_{i=1}^t \lambda_i = 1 \\ A(\sum_{i=1}^t \lambda_i x^i) \geq b(\sum_{i=1}^t \lambda_i) \\ \lambda_i \geq 0, i = 1, \dots, t \end{array} \right. \quad \begin{array}{l} (3.20) \\ (3.21) \\ (3.22) \\ (3.23) \end{array}$$

Si osservi che, per ogni t-pla  $\lambda_1, \dots, \lambda_t$  che soddisfa i vincoli 3.21 e 3.23, il punto  $x = \sum_{i=1}^t \lambda_i x^i$  appartiene a  $\text{conv}(X)$ . Quindi il problema  $DD_L$  può essere riscritto come

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

□

## 3.8 Lagrangiano Duale e Rilassamento Lineare

### 3.8.1 TEOREMA

$$z(D_L) \geq z(LP) \quad (3.25)$$

### 3.8.2 Dimostrazione

Il rilassamento lineare  $LP$  è definito come

$$LP \left\{ \begin{array}{l} z(LP) = \text{Min } cx \\ s.t. \ Ax \geq b \\ \quad Bx \geq d \\ \quad 0 \leq x \leq 1 \end{array} \right. \quad (3.26)$$

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

$$LP \left\{ \begin{array}{l} z(LP) = \text{Min } cx \\ s.t. \ Ax \geq b \\ \quad x \in \bar{X} \end{array} \right. \quad (3.27)$$

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

$$\text{conv}(X) \subseteq \bar{X} \quad (3.28)$$

e poichè abbiamo dimostrato che

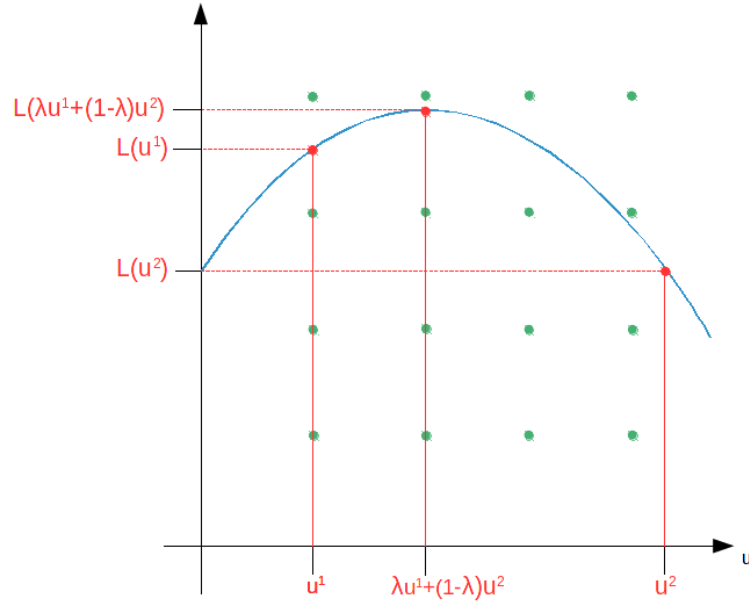
$$D_L \left\{ \begin{array}{l} z(D_L) = \text{Min } cx \\ s.t. \ Ax \geq b \\ \quad x \in \text{conv}(X) \end{array} \right. \quad (3.29)$$

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  $RL_{u^0}$ :

$$L(u^0) = cx^0 - u^0(Ax^0 - b) \quad (3.30)$$

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

$$L(u^1) \leq cx^0 - u^1(Ax^0 - b) \quad (3.31)$$

$$L(u^2) \geq cx^0 - u^2(Ax^0 - b) \quad (3.32)$$

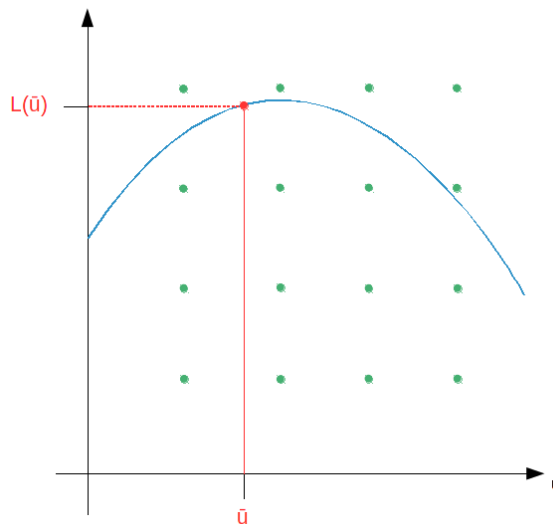
Moltiplicando la 3.31 per  $\lambda$ , la 3.32 e sommando:

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

### 3.9 Subgradiente di $L(u)$

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

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



**Come calcolare  $y$ ?**

Sia  $\bar{x}$  tale che

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

Per ogni  $u \geq 0$  si ha

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

Sottraendo dalla 3.36 la 3.35 si ottiene

$$L(u) - L(\bar{u}) \leq -(A\bar{x} - b)(u - \bar{u}) \quad (3.37)$$

ma anche

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

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

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

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

$$z(D_L) = \max_{u \in \{u^1, u^2, \dots, u^k\}} [L(u)]$$

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

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

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

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

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  $u^r$  affinché 3.40 sia verificata?

Supponiamo che  $A$  abbia  $m$  righe e quindi  $u = (u_1, \dots, u_m)$  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

$$-(a_i^{r-1} - b_i)(u_i^r - u_i^{r-1}) > 0, \forall i = 1, \dots, m \quad (3.41)$$

Da cui seguono i seguenti 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  $u_i^r < u_i^{r-1}$  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:

$$L(u^r) \begin{cases} \text{Min } cx - u^r(Ax - b) \\ \text{s.t.}; Bx \geq d \\ x \in (0, 1) \end{cases}$$

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  $u^{r+1}$

$$u_i^{r+1} = \text{Max}[0, u_i^r - \alpha \cdot \frac{z_{UB} - L(u^r)}{\sum_{i=1}^m \tilde{y}_i^2} \cdot \tilde{y}_i], \forall i$$

dove  $\tilde{y}_i = a^i x^r - b_i$  e  $\alpha$  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  $\alpha$  ( $\alpha \leftarrow \alpha/2$ ) se per  $\delta$  iterazioni consecutive  $L(u) \leq LB$
6. I valori di  $\alpha$  e  $\delta$  vanno determinati sperimentalmente: tipicamente  $\alpha = 2$  e  $\delta = 30$ .

### 3.9.2 Vincoli Misti

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

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

ma anche:

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

### 3.9.3 Subgradiente per vincoli misti

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  $t = \alpha \frac{z_{UB} - L(u)}{\sum_{i=1}^m y_i^2}$

$$u_i^1 \leftarrow \max[0, u_i^1 - t y_i^1], \quad i = 1, \dots, m_1$$

$$u_i^2 \leftarrow u_i^2 - t y_i^2, \quad i = 1, \dots, m_2$$

$$u_i^3 \leftarrow \min[0, u_i^3 - t y_i^3], \quad i = 1, \dots, m_3$$

## 3.10 Traveling Salesman Problem

### 3.10.1 Costi Simmetrici

$n$  vertici,  $m$  archi.

$G = (N, A)$ : grafo non-orientato.

$N = \{1, \dots, n\}$  insieme dei vertici.

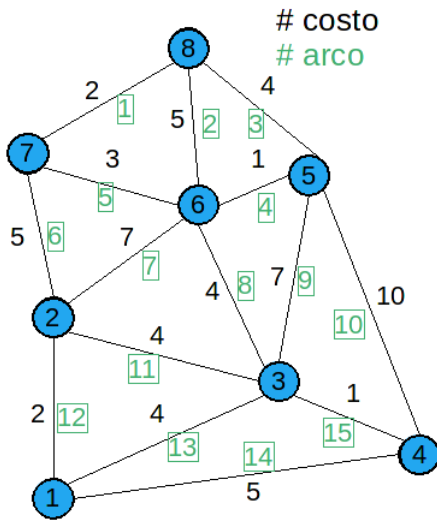
$A = \{1, \dots, m\}$  insieme degli archi.

**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  $(\alpha_l, \beta_l)$  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 Formulazione Matematica (TSP Simmetrico)

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

$x_l = 0$ , altrimenti.

$$\left\{ \begin{array}{l} \text{Min } z = \sum_{l=1}^m c_l x_l \end{array} \right. \quad (3.42)$$

$$\left\{ \begin{array}{l} \sum_{l \in B_i} x_l = 2; \quad i = 1, \dots, n \end{array} \right. \quad (3.43)$$

$$\left\{ \begin{array}{l} \sum_{l \in K_t} x_l \geq 1; \quad \forall K_t = (S_t, N \setminus S_t) \quad S_t \subset N, S_t \neq \emptyset, |S_t| \geq 2 \end{array} \right. \quad (3.44)$$

$$\left\{ \begin{array}{l} x_l \in \{0, 1\}; \quad l = 1, \dots, m \end{array} \right. \quad (3.45)$$

### 3.10.2.1 1° Rilassamento Lagrangiano (SST)

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

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

$$\sum_{l=1}^m x_l = n \quad (3.47)$$

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

$$x_l \in \{0, 1\} \quad l = 1, \dots, m \quad (3.49)$$

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

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

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

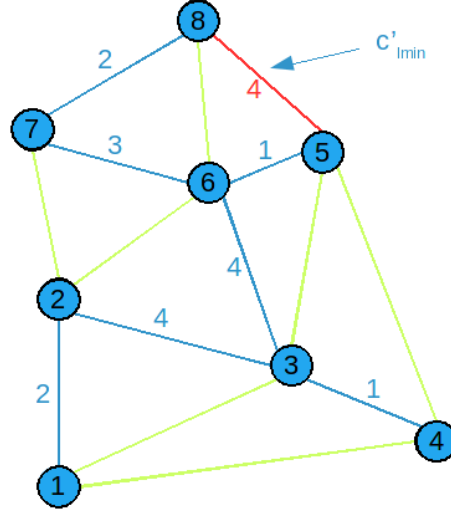
$$\begin{aligned} \sum_{i=1}^n \sum_{l \in B_i} \lambda_i x_l &= \sum_{l=1}^m (\lambda_{\alpha_l} + \lambda_{\beta_l}) x_l \\ L(\lambda) &= \min \sum_{l=1}^m \underbrace{(c_l - \lambda_{\alpha_l} - \lambda_{\beta_l})}_{c'_l} x_l + 2 \sum_{i=1}^n \lambda_i \\ \sum_{l=1}^m x_l &= n \\ \sum_{l \in K_t} x_l &\geq 1; \quad \forall K_t \equiv (S_t, N \setminus S_t) \quad S_t \subset N, S_t \neq \emptyset \\ x_l &\in \{0, 1\} \end{aligned}$$

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

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

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  $c'_{lmin} = 4$

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

## 3.10.4 Calcolo Penalità Lagrangiane

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

$$\sum_{l \in B_i} x_l = 2; \quad i = 1, \dots, n \quad (3.52)$$

diviene

$$d_i = 2; \quad i = 1, \dots, n \quad (3.53)$$

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

$$d^0 = (1, 2, 3, 1, 2, 3, 2, 2) \quad (3.54)$$

$$\lambda^k = \lambda^{k-1} - t_k(Ax^{k-1} - b)$$

dove

$$t_k = \lambda_k \cdot \frac{(z^* - L(\lambda^{k-1}))}{\|Ax^{k-1} - b\|^2}$$

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

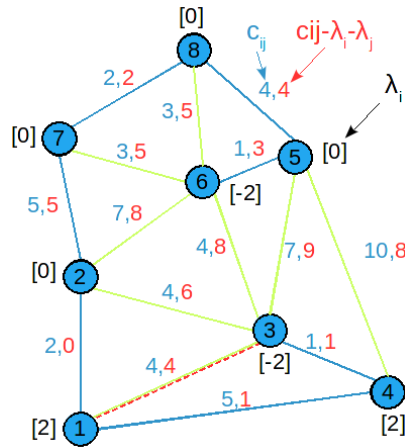
$$t_1 = 2 \cdot \frac{(25 - 21)}{\sum_i (d_i^0 - 2)^2} = 2 \cdot \frac{4}{4} = 2$$

$$\lambda_i^1 = 0 - 2(d_i^0 - 2) \quad i = 1, \dots, n$$

Quindi

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

### 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 $k = 2$

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

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

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

Quindi

$$\lambda_1^2 = 2 - \frac{5}{2} - 1 = -\frac{1}{2}$$

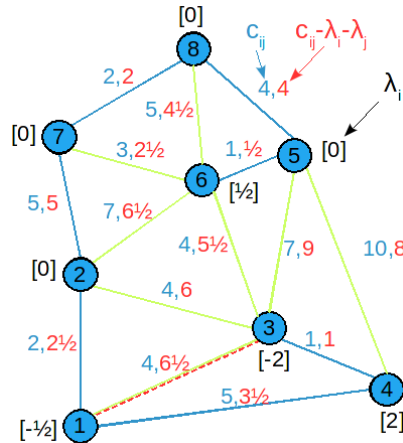
$$\lambda_i^2 = \lambda_i^1, \quad i = 2, 3, 4, 5$$

$$\lambda_6^2 = -2 - \frac{5}{2} \cdot (-1) = -2 + \frac{5}{2} = \frac{1}{2}$$

$$\lambda_7^2 = \lambda_7^1, \quad \lambda_8^2 = \lambda_8^1$$

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

#### 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

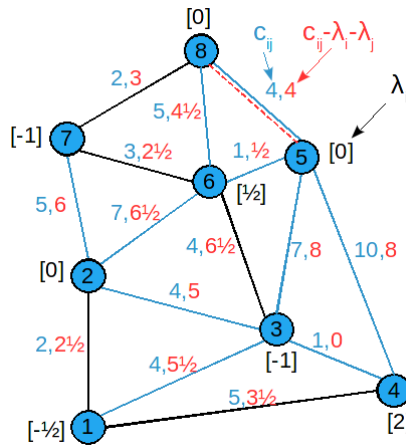
$$\lambda_3^3 = -2 - 1 \cdot (-1) = -1$$

$$\lambda_7^3 = 0 - 1 \cdot 1 = -1$$

$$\text{altrimenti } \lambda_i^3 = \lambda_i^2, \quad \forall i \neq 3, 7$$

$$\lambda^3 = (-\frac{1}{2}, 0, -1, +2, 0, \frac{1}{2}, -1, 0)$$

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:

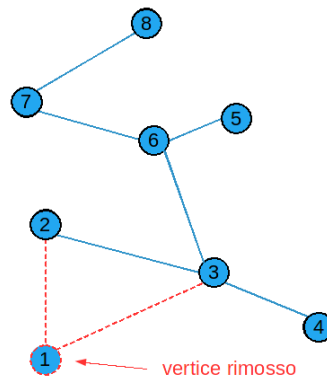
$$\lambda_i^4 = \lambda_i^3 - \frac{1}{2} \cdot (d_i^3 - 2)$$

*continuare per esercizio...*

### 3.10.5 Rilassamento 1-TREE

*Held and Karp*

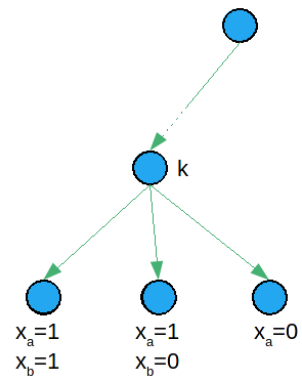
- Si rimuova dal grafo un vertice;
- Si calcoli lo shortest spanning tree (SST) sul grafo rimanente;
- Si aggiungano i due links di costo minimo che incidono sul vertice rimosso;
- Il lower bound è dato dalla somma del costo dello *SST* e dei costi dei due link aggiunti



### 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 e due links  $a$  e  $b$  che nello *SST* incidono su  $i$  e liberi.

Genera 3 nodi come segue



---

---

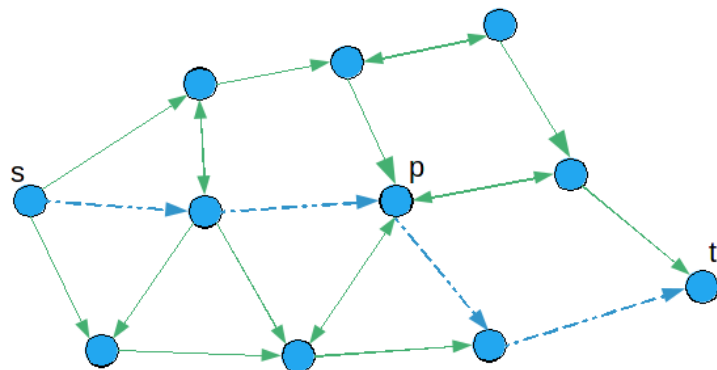
## CAPITOLO 4

---

# 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'osservazione 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:

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

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  $d(1), d(2), \dots, d(k)$  dei cammini minimi dal vertice 1 ai vertici  $(1, 2, \dots, k) \subset V$ .

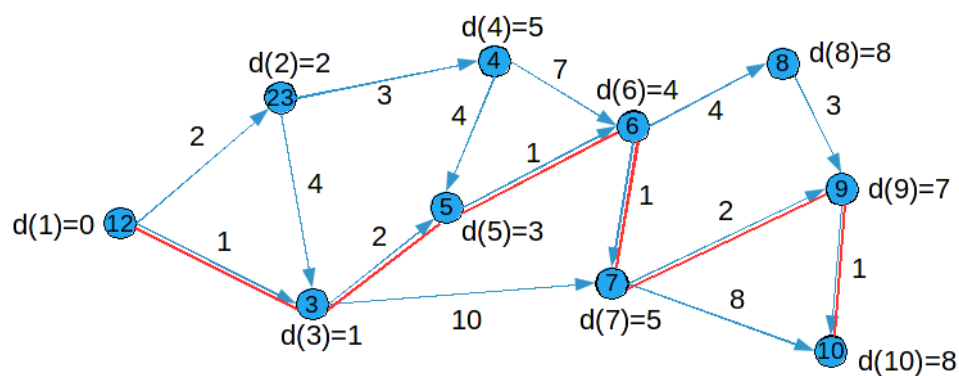


Allora, per come è stato definite il grafo aciclico  $G$ , sono noti i costi  $d(i)$ ,  $\forall i \in \Gamma_{k+1}^{-1}$  e quindi

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

## 4.2 Algoritmo

1. Poni  $d(1) = 0, d(2) = \dots = d(n) = \infty$ ;
2. Per  $v = 2, \dots, n$  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:

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

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

$$D(k, j) = D(k - 1, j) \quad (4.4)$$

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

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

Partendo dalla 4.5 si a che la ricorsione è:

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

La ricorsione 4.6 impone un ordine implicito di calcolo:

prima  $D(1, j)$ ,  $\forall j \in V$ , poi  $D(2, j)$ ,  $D(3, j)$ ,  $\dots$ ,  $D(n - 1, j)$ .

### 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  $k = 2, \dots, n - 1$  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  $d(j) = D(n - 1, j)$ ,  $\forall j \in V$

## 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

$$KP \left\{ \begin{array}{l} z = \max \sum_{j=1}^n c_j x_j \\ \sum_{j=1}^n a_j x_j \leq b \\ x_j \in \{0, 1\}, \quad j = 1, \dots, n \end{array} \right. \quad (4.7)$$

$$\sum_{j=1}^n a_j x_j \leq b \quad (4.8)$$

$$x_j \in \{0, 1\}, \quad j = 1, \dots, n \quad (4.9)$$

### 4.5.1 Esempio

$$\left\{ \begin{array}{l} z = \max 10x_1 + 7x_2 + 25x_3 + 24x_4 \\ 2x_1 + 1x_2 + 6x_3 + 5x_4 \leq 7 \\ x_j \in \{0, 1\}, \quad j = 1, \dots, 4 \end{array} \right.$$

La soluzione ottima è  $x_1^* = 1, x_2^* = 0, x_3^* = 0, x_4^* = 1$ .

### 4.5.2 Osservazione 1

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

$$KP_k(q) \left\{ \begin{array}{l} f_k(q) = \max \sum_{j=1}^k c_j x_j \\ \sum_{j=1}^k a_j x_j \leq q \\ x_j \in \{0, 1\}, \quad j = 1, \dots, k \\ \text{dove } q = \sum_{j=1}^k a_j x_j^* \end{array} \right. \quad (4.10)$$

$$\sum_{j=1}^k a_j x_j \leq q \quad (4.11)$$

$$x_j \in \{0, 1\}, \quad j = 1, \dots, k \quad (4.12)$$

$$\text{dove } q = \sum_{j=1}^k a_j x_j^* \quad (4.13)$$

### 4.5.2.1 Esempio

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

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

La dimostrazione dell'**osservazione 1** è ovvia!

Si consideri la famiglia degli  $n(b+1)$  sottoproblemi

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

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  $(x_1^*, \dots, x_{k-1}^*, x_k^k)$  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  $((x_1^*, \dots, x_{k-1}^*))$  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  $(x_1^*, \dots, x_{k-1}^*)$  è la soluzione ottima di  $KP_{k-1}(q - a_k)$  ovvero  $f_k(q) = f_{k-1}(q - a_k) + c_k$

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

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

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  $f_2(\cdot)$ ,  $\dots$ , infine,  $f_{n-1}(\cdot)$  consente di calcolare  $f_n(\cdot)$ .

### 4.5.2.2 Esempio

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

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

	$f_1$	$f_2$	$f_3$	$f_4$
$q = 0$	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:

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

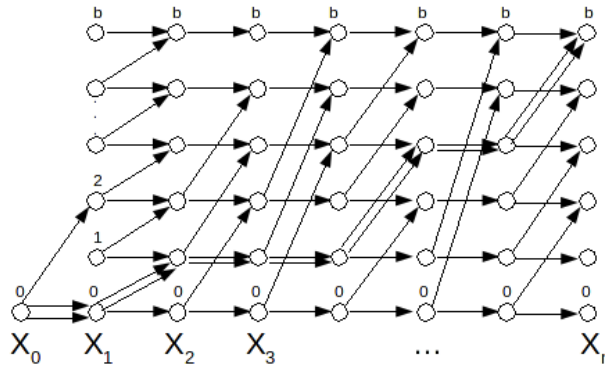
### 4.5.3 Grafo dello spazio degli stati

Alla ricorsione

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

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

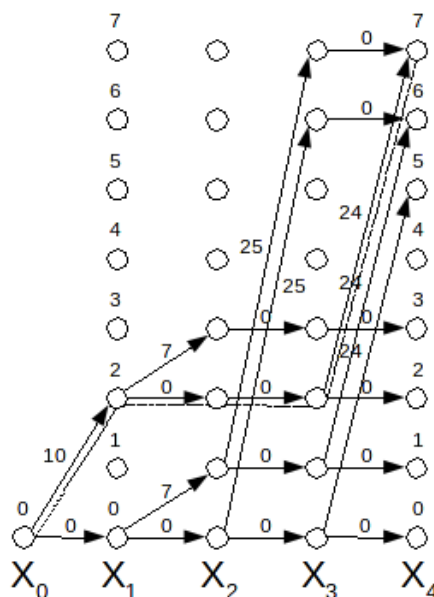
- $X$  si compone di  $n + 1$  partizioni  $X_0, X_1, X_2, \dots, X_n$ .  
 $X_0 = (0)$  e ogni altra partizione  $X_k$ ,  $k = 1, \dots, n$  contiene  $b + 1$  vertici corrispondenti agli stati  $(0, 1, 2, \dots, b)$ ;
- Su ogni vertice  $q \in X_k$  terminano al più due archi: l'arco avente vertice 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

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



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  $f_2(q)$ ,  $q = 4, \dots, 7$ , 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  $f_0(q) = -\infty$ ,  $q = 1, \dots, b$ . Sia  $k = 0$ ;
2. inizializza  $f_{k+1}(q) = -\infty$ ,  $q = 0, 1, \dots, b$ ;
3. Per ogni  $q = 0, 1, \dots, b$  tale che  $f_k(q) \geq 0$  aggiorna

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

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

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

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



## 4.6 Programmazione a numeri interi

$$F^* = \text{Min} \sum_{j=1}^n C_j x_j \quad (4.20)$$

$$\sum_{j=1}^n a_{1j} x_j = b_1 \quad (4.21)$$

$$\sum_{j=1}^n a_{2j} x_j = b_2 \quad (4.22)$$

$$\vdots \quad \vdots \quad (4.23)$$

$$\sum_{j=1}^n a_{mj} x_j = b_m \quad (4.24)$$

$$x_j \geq 0 \text{ e intero} \quad (4.25)$$

Per semplicità assumiamo che

$$\begin{aligned} a_{ij} &\geq 0 & \forall i, j \\ b_i &\geq 0 & \forall i \end{aligned}$$

## 4.7 Programmazione Dinamica

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

$$\sum_{j=1}^k a_{1j} x_j = v_1 \quad (4.27)$$

$$\sum_{j=1}^k a_{2j} x_j = v_2 \quad (4.28)$$

$$\vdots \quad \vdots \quad (4.29)$$

$$\sum_{j=1}^k a_{mj} x_j = v_m \quad (4.30)$$

$$x_1, x_2, \dots, x_k \geq 0 \text{ e intero} \quad (4.31)$$

$F_k(V)$  venga calcolato per  $k = 1, \dots, n$ ;  $\forall \underline{v} \leq \underline{b}$  e  $\underline{v} \geq 0$ .

La soluzione ottima è data da

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

## 4.8 Ricorsione di Programmazione Dinamica

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

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

per  $k = 1, \dots, n$ ;  $\forall \underline{v} \leq \underline{b}$ .

La ricorsione richiede di conoscere  $F_0(\underline{v})$

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

$k$  rappresenta lo **stadio** della ricorsione mentre  $\underline{v}$  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 arco  $(x_i, x_j) \in A$  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 $f(S, x_i)$

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

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

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

#### 4.9.1.1 Calcolo del valore $z^*$ della soluzione ottima

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

È richiesta la seguente inizializzazione:

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

### 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  $f(\{x_2, x_3, x_4x_5\}, x_2)$

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

e quindi bisogna conoscere:

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

Le **variabili di stato** sono rappresentate da

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

Lo stadio può essere definito in modi diversi; deve corrispondere al seguente 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  $|S| = 2$ .

$\vdots$

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:

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

Inizializzazione:

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

$$f(\{2\}, 2) = 6$$

$$f(\{3\}, 3) = 11$$

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

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

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

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

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

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

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

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

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

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

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. $|S| = 3$ ; $\forall x_i \in S$

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

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

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

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

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

da cui

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

Similarmente

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

#### 4.9.3.4 Soluzione ottima

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

## 4.9.3.5 Spazio degli stati del TSP

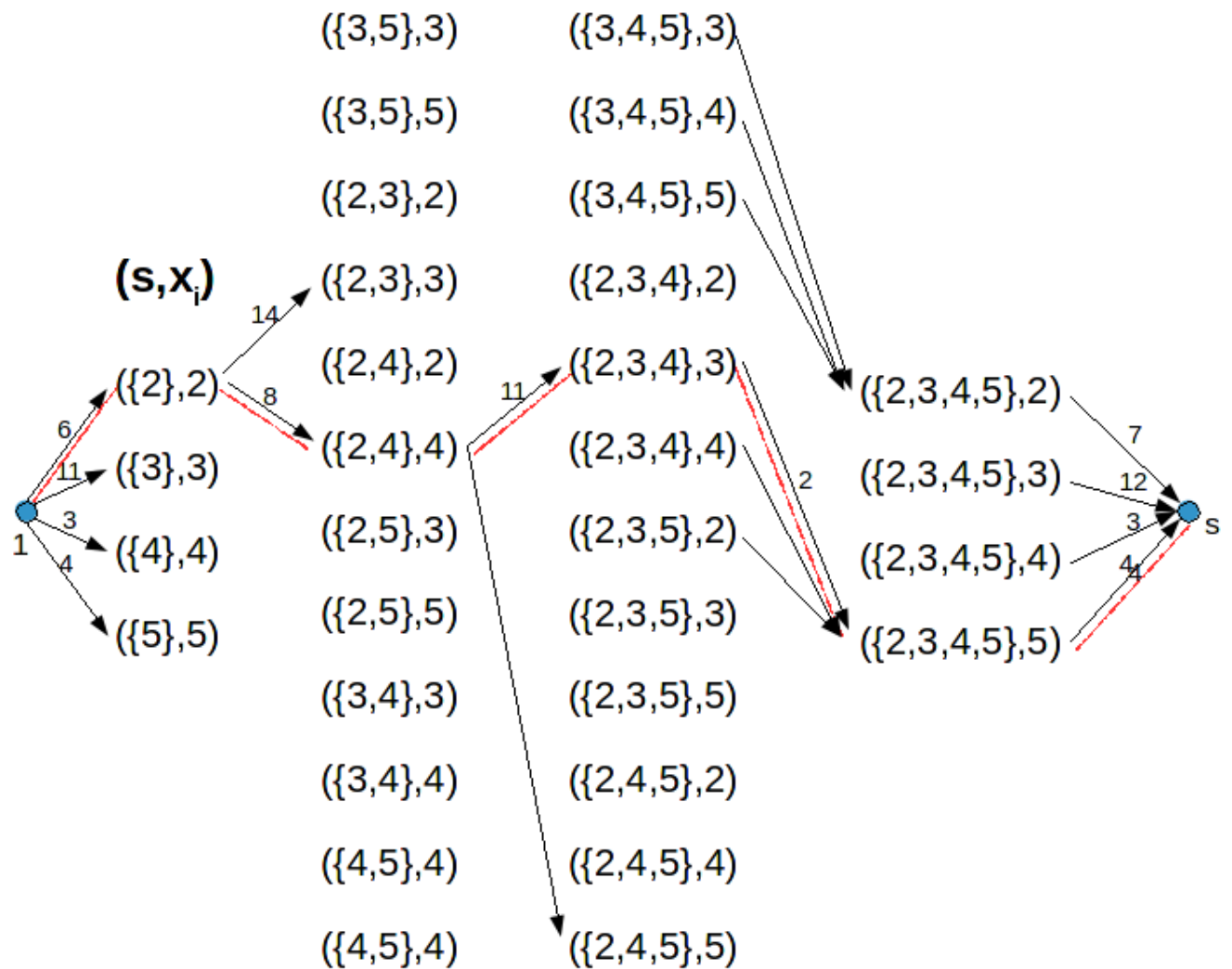


Figura 4.1: Solo alcuni degli archi sono raffigurati

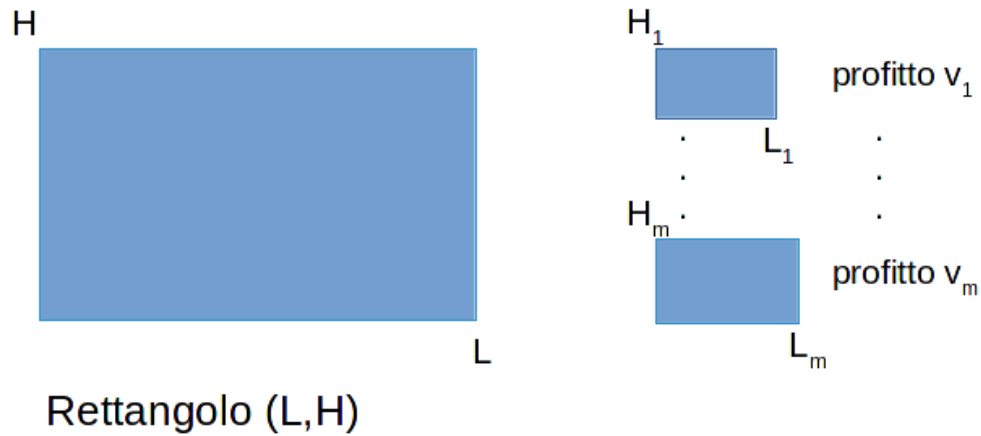


## 4.10 Ricorsione Forward per il TSP

1.  $\mathcal{L}_0 = \{(\{1\}, 1)\}$ ,  $f(\{1\}, 1) = 0$ ,  $p(\{1\}, 1) = 0$ . Inizializza  $\mathcal{L}_r = \emptyset$ ,  $r = 1, \dots, n$ . Definisci  $k = 1$ .
2. Espansione degli stati del set  $\mathcal{L}_{k-1}$ : per ogni stato  $(S, i) \in \mathcal{L}_{k-1}$  ripeti lo step 3.
3. Genera/Aggiorna gli stati di  $\mathcal{L}_k$  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:
  - $(S', j) \in \mathcal{L}_k$ , allora  $\mathcal{L}_k = \mathcal{L}_k \cup \{f(S', j)\}$  e  $f(S', j) = h$ . Poni  $p(S', j) = (S, i)$ ;
  - $(S', j) \in \mathcal{L}_k$  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:

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

## 4.11 Taglio 2-dimensionale a ghigliottina



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

**Obiettivo:** massimizzare il profitto totale dei pezzi tagliati.

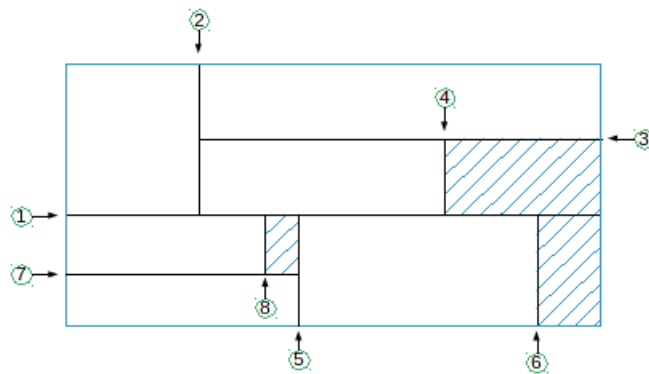


Figura 4.2: Esempio di taglio a ghigliottina

Indichiamo con:

$F(x, y)$  il profitto massimo per tagliare un rettangolo di dimensione  $(x, y)$  con  $x \leq L$  e  $y \leq H$ .

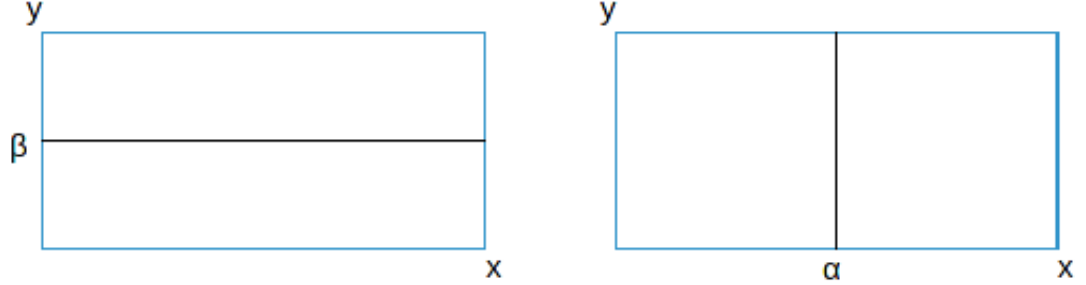
$F(L, H)$  il valore della soluzione ottima.

### 4.11.1 Calcolo di $F(x, y)$

Si hanno tre casi:

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

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  $\beta$  parallelo all'altezza in posizione  $\alpha$ .



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

#### 4.11.2 Inizializzazione

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

$$F^0(x, y) = \max [0, v_i : l_i \leq x, h_i \leq y, i = 1, \dots, m] \quad (4.44)$$

- 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  $x = 1, \dots, L$  e  $y = 1, \dots, H$  calcola

$$F(x, y) = \max [F^0(x, y); F(x, \beta) + F(x, y - \beta), \beta = 1, \dots, y - 1, F(\alpha, y) + F(x - \alpha, y), \alpha = 1, \dots, x - 1] \quad (4.45)$$

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

Si può migliorare la ricorsione sostituendo

$$\begin{aligned} \beta &= 1, \dots, y - 1 \text{ con } 1 \leq \beta \leq y/2 \\ \alpha &= 1, \dots, x - 1 \text{ con } 1 \leq \alpha \leq x/2 \end{aligned}$$

#### 4.11.3 Esempio

Sia  $y = 5$

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

Si noti che  $T_1 \equiv T_4$  e  $T_2 \equiv T_3$  è ciò giustifica la sostituzione di  $\beta = 1, \dots, y - 1$  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

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

##### Posizioni possibili per tagli paralleli alla lunghezza

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

Definiamo

$$\begin{aligned} p(x) &= \max [0, \alpha : \alpha \leq x, \alpha \in L_0], \quad x = 1, \dots, L-1 \\ p(y) &= \max [0, \beta : \beta \leq y, \beta \in H_0], \quad y = 1, \dots, H-1 \end{aligned}$$

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

$$\begin{aligned} F(x, y) &= \max [F^0(x, y); F(x, \beta) + F(x, y - \beta), \beta \in H_0, \beta \leq y/2, \\ &\quad F(\alpha, y) + F(p(x - \alpha), y), \alpha \in L_0, \alpha \leq x/2] \end{aligned}$$

## 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 seguente semplice idea che consente di calcolare un lower bound al costo del cammino minimo in un grafo. I vertici sono clusterizzati in

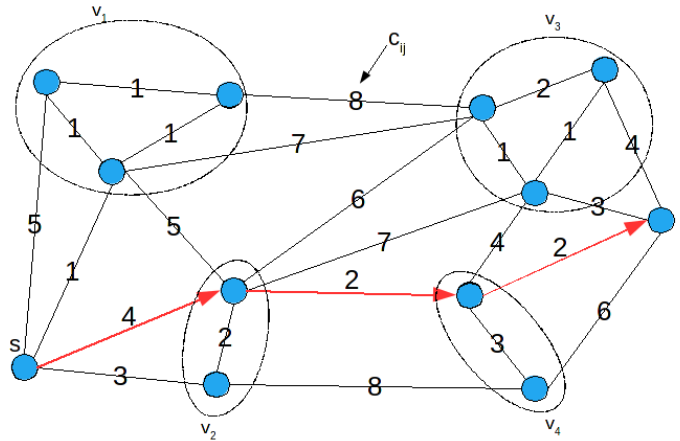
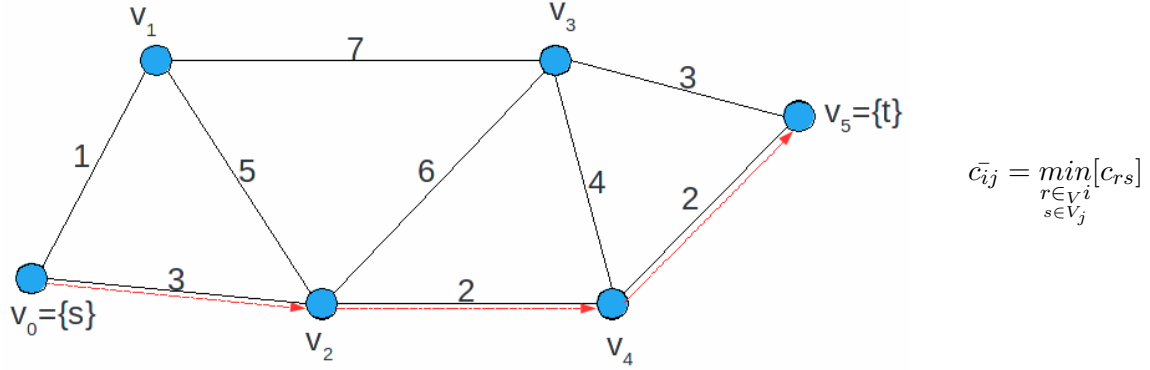


Figura 4.4: Il costo del cammino del grafo  $G = (X, A)$  da  $s$  a  $t$  è 8

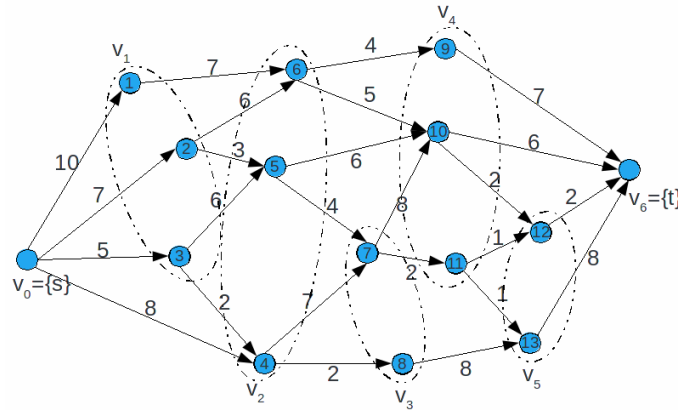
4 cluster come mostrato (non è importante il criterio di clustering per quanto segue). Ogni  $v_k \subset X$ ,  $k = 1, \dots, 4$  e  $v_k \cap v_j = \emptyset$   $j \neq k = 1 \dots 4$ .

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



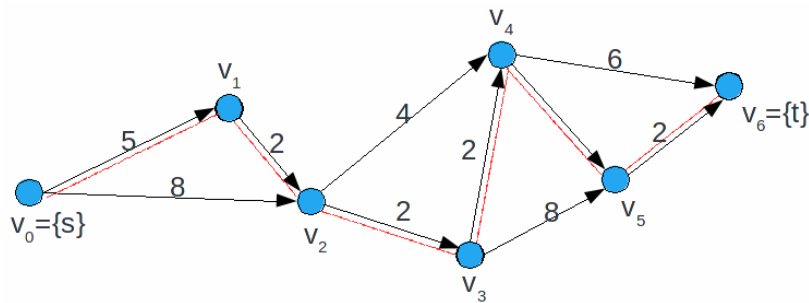
Il costo del cammino minimo da  $v_0$  a  $v_5 \in \bar{G}$  ( $= 7$ ) è un valido lower bound.

#### 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

$s = (s_1, \dots, s_m)$ : variabile di stato

$\mathcal{L}_k$ : 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  $s \in \mathcal{L}_k$  allora stadio  $k$ .

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

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)

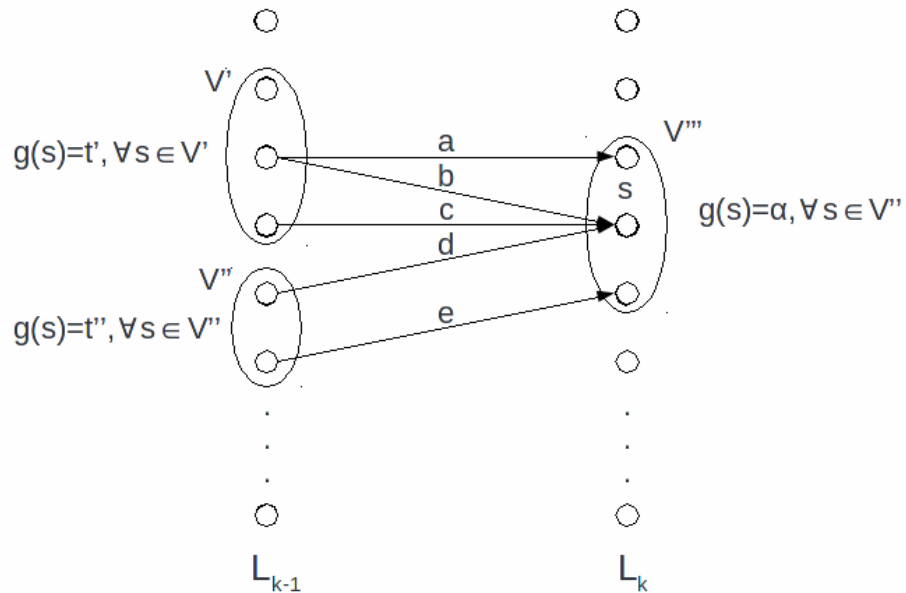
$$s = (S, i)$$

$$\mathcal{L}_k = \{(S, i) : S \subset X \text{ t.c. } 1 \in S, |S| = k, i \in S\}$$

$$\Delta^{-1}(S, i) = \{(S \setminus \{i\}, j) : j \in S \setminus \{i, 1\}\}$$

$$v((S \setminus \{i\}, j), (S, i)) = c_{ji}$$

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



Più stati di  $\mathcal{L}$  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:  $F^{-1}(\alpha) = \{t', t''\}$ ).

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

### Esempio

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

Nello spazio  $R$  la ricorsione diviene

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

Si noti che  $f_k(g(s)) \leq f_k(s)$ ,  $\forall s \in \mathcal{L}$ , 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

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

$(s, i)$ : variabile di stato

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

$$\Delta^{-1}(s, i) = \{(S \setminus i, j) : j \in S \setminus \{i, 1\}\} \quad (4.51)$$

$$F^{-1}(g(s), i) = \{(g(S \setminus \{i\}), j) : j \in S \setminus \{i, 1\}\} \subseteq \{g(S \setminus \{i\}, j) : j \in \Gamma^{-1}\} \quad (4.52)$$

$$v((S \setminus \{i\}, j), (s, i)) = c_{ji} \quad (4.53)$$

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

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

La ricorsione nello spazio rilassato diviene

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

Inizializzazione

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



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

#### 4.12.5.1 Rilassamento $n$ -path

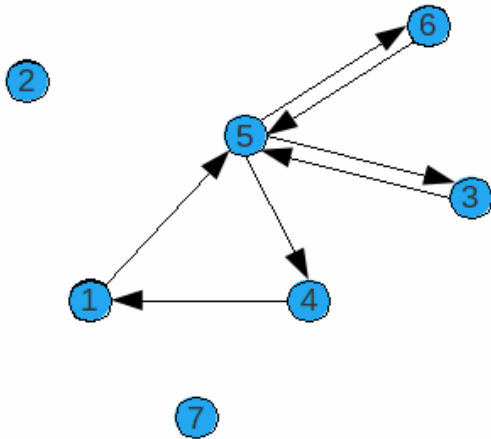
$g(S) = |S| : (S, i) \rightarrow (k, i)$  dove  $k = |S|$

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

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 associi un peso  $q_i \leq 1$  e  $q_1 = 0$

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

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

Inizializza:

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

$$\text{dove } Q = \sum_{i=1}^k q_i \quad (4.61)$$

Il lower bound al *TSP* è dato da

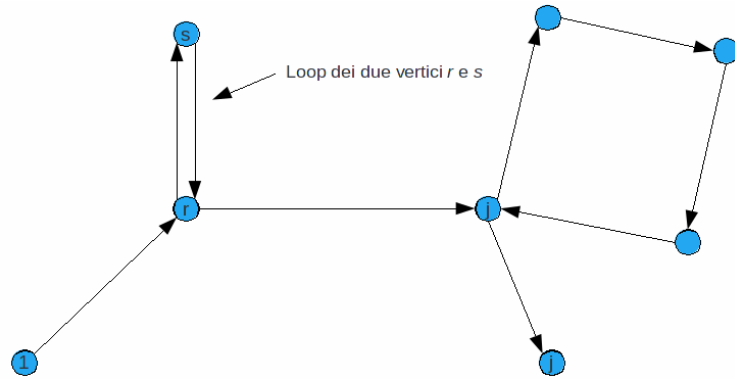
$$z^* = \min_i [f(Q, i) + c_{j1}] \quad (4.62)$$

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 esattamente 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 esem-



pio)

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 vertice 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

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

Quindi calcola per ogni vertice  $i$

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

$$\pi(q, i) = j^* \quad (4.65)$$

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

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

Inizializza

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

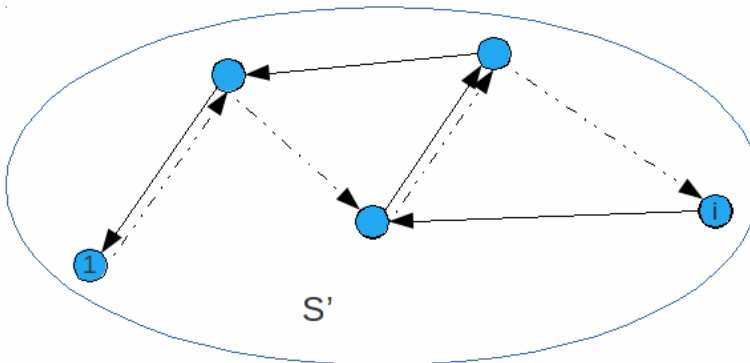
$$\pi(q_i, i) = 1 \quad (4.69)$$

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

$$\gamma(q, i) = 0 \quad (4.71)$$

#### 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  $f'(k, 1)$ ,  $f'(q, i)$ .

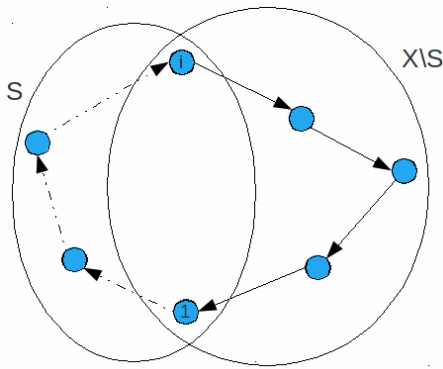
### 4.12.9 Algoritmo DP+Lower Bound per il TSP

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  $z^*$



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

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:

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

### 4.12.10 Algoritmo di programmazione dinamica per il TSP

1. Poni  $\mathcal{L}_1\{(\{i\}, 1)\}$ ,  $f(\{i\}, i) = 0$ ,  $p(\{i\}, i) = 1$  e  $\mathcal{L}_r = \emptyset$ ,  $r = 2, \dots, n$ . Sia  $z_{UB}$  un upper bound al *TSP*. Poni  $k = 2$ ;
2. Espandi ogni stato del set  $\mathcal{L}_{k-1}$ :  
per ogni  $(S, i) \in \mathcal{L}_{k-1}$  ripeti lo step 3;
3. Genera gli stati di  $\mathcal{L}_k$  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  $(S', j) \in \mathcal{L}_k$  e  $f(S', j) \leq h$ .

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

Se  $(S', j) \in \mathcal{L}_k$  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  $z^* = \underset{(X,j) \in \mathcal{L}_k}{Min} [f(X,j) + c_{ji}]$ .

---

---

## CAPITOLO 5

---

### METODI DI DECOMPOSIZIONE

## 5.1 Dantizg-Wolfe Decomposition

Si consideri il seguente problema di programmazione lineare:

$$(P) \begin{cases} \text{Min } cx \\ Ax = b \quad x \in X \end{cases} \quad A \text{ è } (m \times n) \quad (5.1)$$

dove  $X$  è un insieme *poliedrico convesso limitato* che rappresenta vincoli aventi una "speciale" struttura.

Per il teorema della rappresentazione, essendo per ipotesi  $X$  un insieme limitato, detti  $x^1, x^2, \dots, x^t$  i punti estremi di  $X$  allora ogni  $x \in X$  può essere rappresentato come:

$$\begin{aligned} x &= \sum_{j=1}^t \lambda_j x^j \\ \text{s.t. } \sum_{j=1}^t \lambda_j &= 1 \\ \lambda_j &\geq 0, \quad j = 1, \dots, t \end{aligned}$$

Sostituendo  $x$  così definito in  $P$  si ottiene la seguente formulazione equivalente di  $P$  nelle variabili  $\lambda_1, \lambda_2, \dots, \lambda_t$ .

$$P' \begin{cases} z' = \text{Min } \sum_{j=1}^t (cx^j) \lambda_j & (5.2) \\ \text{s.t. } \sum_{j=1}^t (Ax^j) \lambda_j = b & (5.3) \\ \sum_{j=1}^t \lambda_j = 1 & (5.4) \\ \lambda_j \geq 0, \quad j = 1, \dots, t & (5.5) \end{cases}$$

### 5.1.1 Metodo di soluzione di $P'$

$P'$  non può essere risolto direttamente poichè il numero  $t$  di punti estremi di  $X$  è (di solito) molto grande e tali punti non possono essere enumerati a priori.

Si cerca quindi di risolvere  $P'$  senza dover generare tutti i punti estremi di  $X$ .

### 5.1.1.1 Schema dell'algoritmo

#### 1. Problema Master

- Si risolva  $P'$  usando un insieme limitato di  $k$  punti estremi  $x^1, \dots, x^k$  dove  $k \ll t$ .
- Sia  $(w, \alpha)$  la soluzione duale ottima di  $P'$  usando i  $k$  punti estremi generati.  
 $w = (w_1, w_2, \dots, w_t)$  variabile duale dei vincoli 5.3 e  $\alpha$  variabile duale del vincolo 5.4.

$$D' \begin{cases} \text{Max } wb + \alpha & (5.6) \\ \text{s.t. } w(Ax^j) + \alpha \leq cx^j, \quad j = 1, \dots, k & (5.7) \\ w \in \mathcal{R}^m, \quad \alpha \in \mathcal{R} & (5.8) \end{cases} \quad (5.9)$$

#### 2. Ottimalità della soluzione del Master $P'$

La soluzione ottima di  $P'$  è ottima per l'intero problema se e solo se  $(w, \alpha)$  soddisfa i vincoli duali dei punti estremi  $x^{k+1}, x^{k+2}, \dots, x^t$  non considerati di  $P'$ , ovvero:

$$w(Ax^j) + \alpha \leq cx^j, \quad j = k+1, \dots, t \quad (5.10)$$

In altri termini, la soluzione di  $P'$  è ottima e  $(w, \alpha)$  è ottima per  $D'$  se

$$z_j - c_j = w(Ax^j) + \alpha - cx^j = (wA - c)x^j + \alpha \leq 0, \quad j = k+1, \dots, t \quad (5.11)$$

Per verificare se le 5.11 sono soddisfatte o violate è sufficiente cercare (se esiste) il punto estremo di  $X$  che viola le 5.11.

Tale punto estremo, se esiste, sarà la soluzione ottima di costo positivo del seguente sottoproblema  $SP$ :

$$SP \begin{cases} z_{SP} = \text{Max } (wA - c)x + \alpha & (5.12) \\ \text{s.t. } x \in X & (5.13) \end{cases}$$

sia  $x^*$  la soluzione ottima di  $SP$

3. Se  $z_{SP} > 0$  allora la corrente soluzione duale  $(w, \alpha)$  viola il vincolo 5.11 per il punto estremo  $x^*$  di  $X$ . Poni  $k = k+1$ ,  $x^k = x^*$  e ritorna allo step 1.  
 Se  $z_{SP} = 0$  allora  $(w, \alpha)$  soddisfa i vincoli duali 5.11 per i punti estremi non considerati da  $P'$ , quindi la soluzione di  $P'$  è ottima: *stop*.

Potrebbe risultare computazionalmente proibitivo raggiungere la condizione di ottimalità a causa dell'elevato numero di punti estremi.

### 5.1.2 Lower Bound

Ad ogni interazione è possibile calcolare un lower bound  $LB$  al costo ottimo  $z^*$  di  $P$  e quindi si può terminare l'algoritmo quando il costo di  $z'$  di  $P'$  limitato a  $k$  punti estremi è sufficientemente



"vicino" a  $LB$ ; ad esempio, quando

$$\frac{z' - LB}{LB} \leq TOL \quad (5.14)$$

dove  $TOL$  è definito a-priori dall'utente.

### 5.1.2.1 Calcolo del lower bound $LB$

Si noti che per come è definito  $SP$  si ha che

$$(wA - c)x + \alpha \leq z_{SP}, \quad \forall x \in X \quad (5.15)$$

o anche

$$wAx - cx + \alpha \leq z_{SP}, \quad \forall x \in X \quad (5.16)$$

Si consideri una qualunque soluzione  $x$  di  $P$  (ovvero  $x \in X$  e  $Ax = b$ ), dalla 5.16 si ha:

$$wb - cx + \alpha \leq z_{SP} \quad (5.17)$$

o anche

$$cx \geq \overbrace{wb + \alpha}^{z'} - z_{SP} = z' - z_{SP} \quad (5.18)$$

Quindi  $LB = wb + \alpha - z_{SP}$  è un lower bound valido e l'algoritmo può terminare qualora la soluzione  $z'$  del problema master verifichi la 5.14.

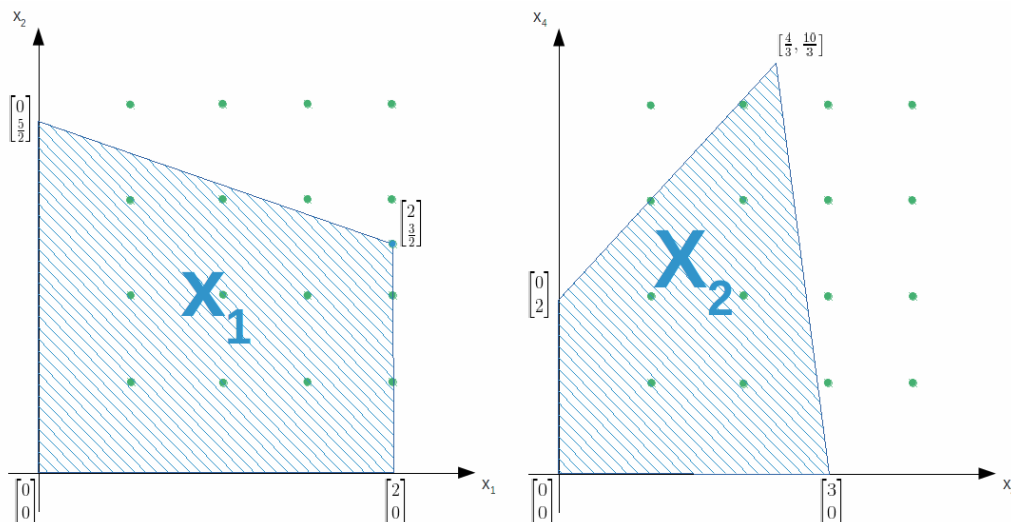
### 5.1.3 Esempio

$$\begin{aligned} \text{Min} \quad & -2x_1 - x_2 - x_3 + x_4 \\ \text{s.t.} \quad & \left. \begin{aligned} x_1 + x_3 &\leq 2 \\ x_1 + x_2 + 2x_4 &\leq 3 \end{aligned} \right\} Ax \leq b \\ & \left. \begin{aligned} x_1 &\leq 2 \\ x_1 + 2x_2 &\leq 5 \\ -x_3 + x_4 &\leq 2 \\ 2x_3 + x_4 &\leq 6 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned} \right\} x \in X \end{aligned}$$

Quindi, forlamente,  $P$  può essere scritto come

$$P \left\{ \begin{aligned} \text{Min} \quad & cx \\ & Ax + s = b \\ & x \in X, s \geq 0 \end{aligned} \right. \quad (5.19)$$

Ogni  $(x_1, x_2, x_3, x_4) \in X$  ha le prime due componenti in  $X_1$  e le ultime due in  $X_2$  come mostrato in seguito.



$$X_1 = \{(x_1, x_2) : x_1 \leq 2, x_1 + 2x_2 \leq 5; x_1, x_2 \geq 0\}$$

$$X_2 = \{(x_3, x_4) : -x_3 + x_4 \leq 2, 2x_3 + x_4 \leq 6, x_3, x_4 \geq 0\}$$

#### 5.1.4 Inizializzazione

Siano  $x^1, x_2, \dots, x_t$  i punti estremi di  $X$ .

Poniamo  $\hat{c}_j = cx_j$  il costo del punto estremo  $x^j$ ,  $j = 1 \text{ dots } t$

$$P' \left\{ \begin{array}{l} \text{Min } \sum_{j=1}^t \hat{c}_j \lambda_j \\ \sum_{j=1}^t (Ax^j) \lambda_j + s = b \\ \sum_{j=1}^t \lambda_j = 1 \\ \lambda_j \geq 0, j = 1, \dots, t \end{array} \right. \quad \begin{array}{l} (5.20) \\ (5.21) \\ (5.22) \\ (5.23) \end{array}$$

Si può partire con il punto estremo  $x^1 = (0, 0, 0, 0)$  di costo  $\hat{c}_1 = 0$

$$P' \left\{ \begin{array}{ll} z^1 = \text{Min } 0\lambda_1 & (5.24) \\ s_1 = 2 & w_1 \\ s_3 = 3 & w_2 \\ \lambda_1 = 1 & \alpha \\ s_1, s_2, \lambda_1 \geq 0 & (5.28) \end{array} \right. \quad \begin{array}{l} (5.25) \\ (5.26) \\ (5.27) \end{array}$$

Ottimo:  $\lambda_1 = 1$ ,  $s_1 = 2$ ,  $s_2 = 3$ ,  $z^1 = 0$  e  $(w_1, w_2, \alpha) = (0, 0, 0, 0)$ .

La soluzione primale di  $P$  di costo  $z^1 = 0$  è  $x = \lambda_1 x^1 = 0x^1 = (0, 0, 0, 0)$ .

#### 5.1.4.1 Iterazione 1

$$SP \quad \text{Max } (wA - c)x + \alpha \quad (5.29)$$

$$s.t. \quad x \in X \quad (5.30)$$

Poichè  $(w_1, w_2, \alpha) = 0$

$$SP \quad \text{Max } 2x_1 + x_2 + x_3 - x_4 + 0 \quad (5.31)$$

$$x \in X \text{ o } (x_1, x_2) \in X_1, (x_3, x_4) \in X_2 \quad (5.32)$$

$SP$  è separabile nei vettori  $(x_1, x_2) \in X_1$  e  $(x_3, x_4) \in X_2$  per cui, come si vede in 5.1.3, si ha che una soluzione ottima di  $SP$  è data dal punto estremo

$$x^2 = (2, \frac{3}{2}, 3, 0) \quad (5.33)$$

$$z_2 - \hat{c}_2 = (wA - c)x^2 + \alpha = -cx^2 = \frac{17}{2} > 0$$

$$\text{Lower bound } z^1 - (z_2 - \hat{c}_2) = -\frac{17}{2} = -8.5 \text{ e } x_2 \text{ entra nel Master}$$

#### 5.1.4.2 Nuovo Master

$$P' \left\{ \begin{array}{l} \text{Min } \hat{c}_1 \lambda_1 + \hat{c}_2 \lambda_2 \\ (Ax^1) \lambda_1 + (Ax_2) \lambda_2 + s = b \\ \lambda_1 + \lambda_2 = 1 \\ \lambda_1 + \lambda_2 \geq 0, s \geq 0 \end{array} \right. \quad (5.34)$$

$$(Ax^1) \lambda_1 + (Ax_2) \lambda_2 + s = b \quad (5.35)$$

$$\lambda_1 + \lambda_2 = 1 \quad (5.36)$$

$$\lambda_1 + \lambda_2 \geq 0, s \geq 0 \quad (5.37)$$

Nel calcolo in esame  $Ax^1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ,  $Ax_2 = \begin{bmatrix} 5 \\ 7 \\ \frac{7}{2} \end{bmatrix}$ , quindi

$$P' \left\{ \begin{array}{l} z^1 = \text{Min } 0 \cdot \lambda_1 + (-\frac{17}{2}) \lambda_2 \\ 0 \cdot \lambda_1 + 5 \lambda_2 + s_1 = 2 \\ 0 \cdot \lambda_1 + \frac{7}{2} \lambda_2 + s_2 = 3 \\ \lambda_1 + \lambda_2 = 1 \\ \lambda_1, \lambda_2, s_1, s_2 \geq 0 \end{array} \right.$$

Ottimo:  $\lambda_1 = \frac{3}{5}$ ,  $\lambda_2 = \frac{2}{5}$ ,  $s_1 = 0$ ,  $s_2 = \frac{2}{5}$ ,  $z^1 = -\frac{17}{5}$ .

$$(w_1, w_2, \alpha) = (-\frac{17}{10}, 0, 0)$$

La soluzione primale di  $P$  di costo  $z^1 = -\frac{17}{6} = -3 \cdot 4$  è

$$x = \lambda_1 x^1 + \lambda_2 x^2 = \frac{5}{3} + \frac{2}{5} x^2 = (\frac{4}{5}, \frac{3}{5}, \frac{6}{5}, 0) \quad (5.38)$$

### 5.1.4.3 Iterazione 2

$$SP \begin{cases} \text{Max} & (wA - c)x + \alpha \\ \text{s.t.} & x \in X \end{cases}$$

$$(wA - c) = (\frac{3}{10}, 1, -\frac{7}{10}, -1)$$

$$SP \begin{cases} \text{Max} & \frac{3}{10}x_1 + x_2 - \frac{7}{10}x_3 - x_4 + 0 \\ \text{s.t.} & (x_1, x_2) \in X_1, (x_3, x_4) \in X_2 \end{cases}$$

Ottimo  $x^3 = (0, \frac{5}{2}, 0, 0)$  e  $z - \hat{c}_3 = \frac{5}{2} > 0$ .

Lower bound  $z^1 - (z_3 - \hat{c}_3) = -\frac{17}{5} - \frac{5}{2} = -5.4$

### 5.1.4.4 Nuovo Master

$$P' \begin{cases} \text{Min} & \hat{c}_1 \lambda_1 + \hat{c}_2 \lambda_2 + \hat{c}_3 \lambda_3 \\ & (Ax^1) \lambda_1 + (Ax^2) \lambda_2 + (Ax^3) \lambda_3 + s = b \\ & \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ & \lambda_1, \lambda_2, \lambda_3, s \geq 0 \end{cases}$$

$$\hat{c}_3 = cx^3 = -\frac{5}{2}, Ax^3 = \begin{pmatrix} 0 \\ 5 \\ 2 \end{pmatrix}, \text{ quindi}$$

$$P' \begin{cases} z^1 = \text{Min} & 0 \cdot \lambda_1 - \frac{17}{2} \lambda_2 - \frac{5}{2} \lambda_3 \\ & 0 \cdot \lambda_1 + 5 \lambda_2 + 0 \cdot \lambda_3 + s_1 = 2 \\ & 0 \cdot \lambda_1 + \frac{7}{2} \lambda_2 + \frac{5}{2} \lambda_3 + s_2 = 3 \\ & \lambda_1 + \lambda_2 + \lambda_3 = 1 \end{cases}$$

Ottimo:  $\lambda_1 = 0, \lambda_2 = \frac{2}{5}, \lambda_3 = \frac{3}{5}, s_1 = 0, s_2 = \frac{1}{10}$  e  $z^1 = -4.9$ , inoltre,  $(w_1, w_2, \alpha) = (-\frac{6}{5}, 0, -\frac{5}{2})$ .

La soluzione di  $P$  di costo  $-4.9$  è  $x = \lambda_2 x^2 + \lambda_3 x^3 = (\frac{4}{5}, 21, \frac{6}{5}, 0)$

#### 5.1.4.5 Iterazione 3

$$SP \begin{cases} \text{Max } (wA - c)x + \alpha \\ \text{s.t. } x \in X \end{cases}$$

$$(wA - c) = (\frac{4}{5}, 1, -\frac{1}{5}, -1)$$

$$SP \begin{cases} z_{SP} = \text{Max } \frac{4}{5}x_1 + x_2 - \frac{1}{5}x_3 - x_4 - \frac{5}{2} \\ (x_1, x_2) \in X_1, (x_3, x_4) \in X_2 \end{cases}$$

$$x^4 = (2, \frac{3}{2}, 0, 0) \text{ e } z_{SP} = (z_4 - \hat{c}_4) = \frac{3}{5}.$$

$$\text{Lower Bound } z^1 - (z_4 - \hat{c}_4) = -4.9 - \frac{3}{5} = -5.5$$

#### 5.1.4.6 Nuovo Master

$$Ax^4 = \begin{Bmatrix} 2 \\ 7 \\ \frac{7}{2} \end{Bmatrix}$$

$$cx^4 = -\frac{11}{2}$$

$$P \begin{cases} z^1 = \text{Min } 0 \cdot \lambda_1 - \frac{17}{2}\lambda_2 - \frac{5}{2}\lambda_3 - \frac{11}{2}\lambda_4 \\ 0 \cdot \lambda_1 + 5\lambda_2 + 0 \cdot \lambda_3 + 2\lambda_4 + s_1 = 2 \\ 0 \cdot \lambda_1 + \frac{7}{2}\lambda_2 + \frac{6}{2}\lambda_3 + \frac{7}{2}\lambda_4 + s_2 = 3 \\ \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1 \\ \lambda_1, \dots, \lambda_4, s_1, s_2 \geq 0 \end{cases}$$

$$\text{Ottimo } \lambda_1 = 0, \lambda_2 = \frac{1}{3}, \lambda_3 = \frac{1}{6}, \lambda_4 = \frac{1}{2}, z^1 = -5.$$

$$(w_1, w_2, \alpha) = (-1, -1, 0)$$

$$\text{La soluzione di } P \text{ di costo } -5 \text{ è } x = \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4 = \frac{1}{3}x^2 + \frac{1}{6}x^3 + \frac{1}{2}x^4 = (1, 2, 1, 0)$$

#### 5.1.4.7 Iterazione 4

$$SP \begin{cases} \text{Max } (wA - c)x + \alpha \\ x \in X \end{cases}$$

$$(wA - c) = (0, 0, 0, -3)$$

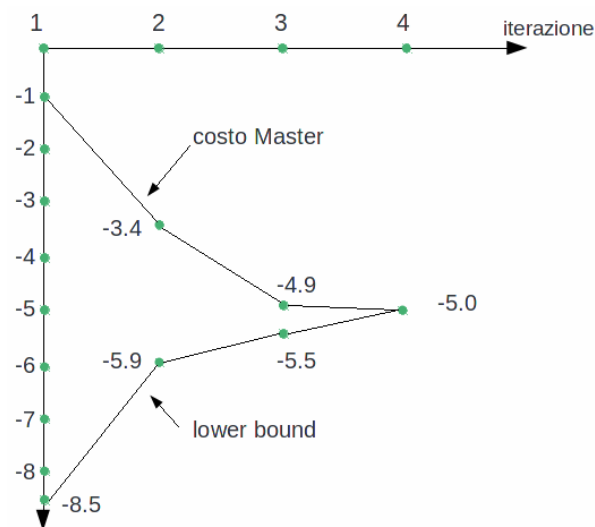
$$SP \begin{cases} z_{SP} = \text{Max } 0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3 - 3x_4 + 0 \\ (x_1, x_2) \in X_1, (x_3, x_4) \in X_2 \end{cases}$$

Ottimo  $z_{SP} = 0$ : *stop*.

Lower Bound  $z^1 - z_{SP} = z^1 = -s$ .

Quindi la soluzione ottima del problema originario è

$$x = \frac{1}{3}x^2 + \frac{1}{6}x^3 + \frac{1}{2}x^4 = (1, 2, 1, 0)$$



## 5.2 Struttura Diagonale a blocchi di X

È il caso in cui  $X$  può essere decomposto in  $T$  sottoinsiemi  $X_1, X_2, \dots, X_T$  e il vettore delle variabili in  $T$  vettori  $x_1, x_2, \dots, x_T$  tali che  $x_k \in X_k$ ,  $k = 1, \dots, T$ .

Indichiamo con  $c_k$  il vettore dei costi e con  $A_k$  la sottomatrice di  $A$  relativa alle variabili  $x_k$ ,  $k = 1, \dots, T$ .

Il problema  $P$  si può scrivere come segue:

$$P \left\{ \begin{array}{l} \text{Min } c_1x_1 + c_2x_2 + \dots + c_Tx_T = \text{Min } \sum_{k=1}^T c_kx_k \\ A_1x_1 + A_2x_2 + \dots + A_Tx_T = b \quad \sum_k A_kx_k = b \\ B_1x_1 = b_1 \\ \quad B_2x_2 = b_2 \\ \quad \quad \ddots \quad \vdots \\ \quad \quad \quad B_Tx_T = b_T \\ x_1, x_2, \dots, x_T \geq 0 \end{array} \right.$$

dove  $X_i = \{x_i : B_i x_i = b_i, x_i \geq 0\}$ ,  $i = 1, \dots, T$ .

L'algoritmo descritto in precedenza può essere ulteriormente specializzato per strutturare la particolare struttura che definisce  $x \in X$ .

Supponiamo che ogni insieme  $X_i$  sia limitato.

Per ogni sottoinsieme  $X_k$  (limitato) indichiamo con  $x_k^1, x_k^2, \dots, x_k^{t_k}$  i punti estremi, quindi  $x_k \in X_k$  può essere rappresentato come

$$\left. \begin{array}{l} x_k = \sum_{j=1}^{t_k} \lambda_{kj} x_k^j \\ \sum_{j=1}^{t_k} \lambda_{kj} = 1 \\ \lambda_{kj} \geq 0, \quad j = 1, \dots, t_k \end{array} \right\} k = 1, \dots, T$$

Sostituendo ogni  $x_k$  con  $k = 1, \dots, T$  così definito in  $P$  si ha

$$P' \left\{ \begin{array}{l} z' = \text{Min } \sum_{k=1}^T \left( \sum_{j=1}^{t_k} (c_k x_k^j) \lambda_{kj} \right) \quad (5.39) \\ \sum_{k=1}^T \left( \sum_{j=1}^{t_k} (A_k x_k^j) \lambda_{kj} \right) = b \quad (5.40) \\ \sum_{j=1}^{t_k} \lambda_{kj} = 1, \quad k = 1, \dots, T \quad (5.41) \\ \lambda_{kj} \geq 0, \quad j = 1, \dots, t_k, \quad k = 1, \dots, T \quad (5.42) \end{array} \right.$$

Siano  $w = (w_1, \dots, w_m)$  e  $\alpha = (\alpha_1, \dots, \alpha_T)$  le variabili duali, rispettivamente dei vincoli 5.40 e 5.41.

$$D' \left\{ \begin{array}{l} \text{Max } wb + \sum_{k=1}^T \alpha_k \\ w(A_k x_k^j) + \alpha_k \leq c_k x_k^j, \quad j = 1, \dots, t_k, \quad k = 1, \dots, T \\ w \in \mathcal{R}^m, \quad \alpha \in \mathcal{R}^T \end{array} \right. \quad (5.43)$$

$$(5.44)$$

$$(5.45)$$

### 5.2.1 Metodo di soluzione

Si fa uso della particolare struttura di  $X$ .

#### 1. Master problem

Si risolva  $P'$  usando un numero limitato  $r_k \ll t_k$  dei punti estremi di ciascun insieme  $X_k$ ,  $k = 1, \dots, T$ .

Sia  $(w, \alpha)$  la soluzione del duale del suddetto master.

#### 2. Ottimalità della soluzione del Master

La soluzione del Master è ottima se  $(w, \alpha)$  soddisfa i vincoli duali per i punti estremi non generati; ovvero, se per ogni  $k = 1, \dots, T$

$$w(A_k x_k^j) + \alpha_k \leq c_k x_k^j, \quad j = r_k + 1, \dots, t_k \quad (5.46)$$

ovvero se

$$(wA_k - c_k)x_k^j + \alpha_k \leq 0, \quad j = r_k + 1, \dots, t_k \quad (5.47)$$

Si risolva quindi per ogni  $k = 1, \dots, T$

$$SP_k \left\{ \begin{array}{l} z_{SP} = \text{Max } (wA_k - c_k)x_k + \alpha_k \\ \text{s.t. } B_k x_k = b_k \\ x_k \geq 0 \end{array} \right.$$

Sia  $x_k^*$  la soluzione ottima.

3. Se  $z_{SP}^k \geq 0$ ,  $\forall k$  : stop la soluzione è ottima; altrimenti per ogni  $k$  per cui  $z_{SP}^k > 0$  poni  $r_k = r_k + 1$ ,  $x_k^{r_k} = x_k$  e quindi ritorna al punto 1.

### 5.2.2 Lower Bound

Come descritto in precedenza, si può terminare l'algoritmo quando il costo  $z'$  del Master non dista "troppo" dall'ottimo ovvero quando

$$\frac{z' - LB}{LB} \leq TOL$$



dove  $LB$  può essere così calcolato. Per come è definito ogni  $SP_k$ ,  $k = 1, \dots, T$ , si ha che

$$(wA_k - c_k)x_k + \alpha_k \leq z_{SP}^k, \quad k = 1, \dots, T \quad (5.48)$$

o anche

$$c_k x_k \geq wA_k x_k + \alpha_k - z_{SP}^k, \quad k = 1, \dots, T \quad (5.49)$$

Sommando

$$\sum_{k=1}^T c_k x_k \geq w \sum_{k=1}^T A_k x_k + \sum_{k=1}^T A_k x_k + \sum_{k=1}^T \alpha_k - \sum_{k=1}^T z_{SP}^k \quad (5.50)$$

Si considero un qualunque  $x = (x_1, \dots, x_T)$  che soddisfi  $x_k \in X_k$ ,  $\forall k$  e  $A_1 x_1 + A_2 x_2 + \dots, A_T x_T = b$  si ha

$$\sum_{k=1}^T c_k x_k \geq wb + \sum_{k=1}^T \alpha_k - \sum_{k=1}^T z_{SP}^k \quad (5.51)$$

Quindi  $LB = wb + \underbrace{\sum_k \alpha_k}_{z'} - \sum_k z_{SP}^k$ .

$LB$  è un limite inferiore al costo della soluzione ottima.

### 5.3 Assegnamento Generalizzato

$m$  contenitori di capacità  $Q_1, \dots, Q_m$

$n$  oggetti che devono essere caricati nei contenitori

$c_{oj}$  costo per caricare l'oggetto  $i$  nel contenitore  $j$

$q_{ji}$  spazio del contenitore  $j$  occupato da  $i$  se  $i$  viene caricato nel contenitore  $j$ .

Si vuole caricare tutti gli oggetti nei contenitori minimizzando il costo complessivo.

$$P \left\{ \begin{array}{l} \text{Min} \sum_{i=1}^n \sum_{j=1}^m c_{ij} x_{ij} \end{array} \right. \quad (5.52)$$

$$P \left\{ \begin{array}{l} \text{s.t.} \sum_{j=1}^m x_{ij} = 1, \quad i = 1, \dots, n \end{array} \right. \quad (5.53)$$

$$P \left\{ \begin{array}{l} \sum_{i=1}^n q_{ij} x_{ij} \leq Q, \quad j = 1, \dots, m \end{array} \right. \quad (5.54)$$

$$P \left\{ \begin{array}{l} x_{ij} \in \{0, 1\}, \quad \forall i, j \end{array} \right. \quad (5.55)$$

$x_{ij} = 1$  se  $i$  è caricato nel contenitore  $j$ ;  $x_{ij} = 0$  altrimenti.

#### 5.3.1 Decomposizione Dantzig-Wolfe del GAP

Indichiamo con  $X_j$  l'insieme delle soluzioni intere del vincolo 5.54 per il contenitore  $j$ ,  $j = 1, \dots, m$

$$X_j = \{x_j : \sum_{i=1}^n q_{ij} x_{ij} \leq Q, \quad x_{ij} \in \{0, 1\}, \quad i = 1, \dots, n\}$$

dove  $x_j = (x_{1j}, x_{2j}, \dots, x_{nj})^T$ . Siano  $x_j^1, x_j^2, \dots, x_j^{t_j}$  i punti estremi di  $\text{conv}(X_j)$ .  
Per il teorema della rappresentazione abbiamo:

$$\left. \begin{array}{l} x_{ij} = \sum_{r=1}^{t_j} \lambda_{jr} x_{ij}^r, \quad i = 1, \dots, n \\ \sum_{r=1}^{t_j} \lambda_{jr} = 1 \\ \lambda_{jr} \geq 0 \end{array} \right\} j = 1, \dots, m \quad (5.56)$$

Sostituendo in  $P$ , definito da 5.52 - 5.55,  $x_{ij}$  secondo 5.56:

$$P' \left\{ \begin{array}{l} \text{Min} \sum_{i=1}^n \sum_{j=1}^m c_{ij} \sum_{r=1}^{t_j} \lambda_{jr} x_{ij}^r \end{array} \right. \quad (5.57)$$

$$s.t. \sum_{j=1}^m \sum_{r=1}^{t_j} \lambda_{jr} x_{ij}^r = 1, \quad i = 1, \dots, n \quad (5.58)$$

$$\sum_{r=1}^{t_j} \lambda_{jr} = 1, \quad j = 1, \dots, m \quad (5.59)$$

$$\lambda_{jr} \in \{0, 1\}, \quad j = 1, \dots, m, \quad r = 1, \dots, t_j \quad (5.60)$$

I vincoli 5.60 derivano dal fatto che  $x_{ij} \in \{0, 1\}$ .

Perché  $\lambda_{ji} \in \{0, 1\}$  invece di  $\lambda_{jr} \geq 0$ ?

Il motivo è che le soluzioni frazionarie di  $\lambda_{jr}$  per un dato  $j$  inducono nei 5.60 soluzioni frazionarie di  $x_{ij}$ .

Si consideri il seguente esempio dei 5.60 per un dato  $j$  per il quale vengono mostrati  $t_j = 5$  punti estremi di  $\text{conv}(X_j)$ .

	$\lambda_{j1}$	$\lambda_{j2}$	$\lambda_{j3}$	$\lambda_{j4}$	$\lambda_{j5}$	
$x_{1j}$	1	1	0	0	1	
$x_{2j}$	0	1	1	0	1	
$\cdot$	1	0	1	1	0	
$\cdot$	0	1	0	1	0	
$\cdot$	1	0	0	1	0	
$x_{6j}$	0	0	1	0	1	
	1	1	1	1	1	$= 1$

$$\lambda_{j1}, \dots, \lambda_{j5} \geq 0$$

Ogni soluzione frazionaria di  $\lambda$  induce una soluzione frazionaria in  $x_{ij}$ .

**Esempio.**

$\lambda_{ji} = \frac{1}{2}$ ,  $\lambda_{j2} = \frac{1}{2}$  e  $\lambda_{j3} = \lambda_{j4} = \lambda_{j5} = 0$  induce  
 $x_{1j} = 1$ ,  $x_{2j} = \frac{1}{2}$ ,  $x_{3j} = \frac{1}{2}$ ,  $x_{4j} = \frac{1}{2}$  e  $x_{5j} = 0$ .

Affinchè soluzioni frazionarie  $\lambda$  inducano soluzioni intere  $x$  è necessario che i punti estremi  $x_j^r$  corrispondenti a  $\lambda_{jr} > 0$  coincidano. Ma ciò non avviene perchè i punti estremi di  $\text{conv}(X)$  sono ovviamente distinti.

Quindi l'unica possibilità affinchè  $x_{ij} \in \{0, 1\}$  è che anche  $\lambda_{jr} \in \{0, 1\}$ .

$P'$  può essere riscritto come segue:

$$P' \left\{ \begin{array}{l} \text{Min} \sum_{j=1}^m \sum_{r=1}^{t_j} c(x_j^r) \lambda_{jr} \\ \text{s.t.} \sum_{j=1}^m \sum_{r=1}^{t_j} x_{ij}^r \lambda_{jr} = 1, \quad i = 1, \dots, n \\ \sum_{r=1}^{t_j} \lambda_{jr} = 1, \quad j = 1, \dots, m \\ \lambda_{jr} \in \{0, 1\} \end{array} \right. \quad (5.61)$$

$$\quad \quad \quad (5.62)$$

$$\quad \quad \quad (5.63)$$

$$\quad \quad \quad (5.64)$$

dove  $c(x_j^r) = \sum_{i=1}^n c_{ij} x_{ij}^r$  è il costo del punto estremo  $x_j^r$ .

Indichiamo con  $LP'$  il rilassamento lineare di  $P'$  (sostituiamo ?? con  $\lambda_{jr} \geq 0$ ).  
 $LP'$  può essere risolto usando il metodo Dantzig-Wolfe.

### 5.3.2 Duale di DP'

$w_i, i = 1, \dots, n$ : variabili duali dell'equazione 5.62

$\alpha_j, j = 1, \dots, m$ : variabili duali dell'equazione 5.63

$$DLP' \left\{ \begin{array}{l} \text{Max} \sum_{i=1}^n w_i + \sum_{j=1}^m \alpha_j \\ \text{s.t.} \sum_{i=1}^n w_i + x_{ij}^r + \alpha_j \leq c(x_j^r), \quad r = 1, \dots, t_j, \quad j = 1, \dots, m \\ w_i \in \mathcal{R}, \quad i = 1, \dots, n \\ \alpha_j \in \mathcal{R}, \quad j = 1, \dots, m \end{array} \right. \quad (5.65)$$

$$\quad \quad \quad (5.66)$$

$$\quad \quad \quad (5.67)$$

$$\quad \quad \quad (5.68)$$

Tabella 5.1: Tableau di  $P'$ 

	$c(x_1^1)$	$c(x_1^2)$	$\dots$	$c(x_1^{t_1})$	$\dots$	$c(x_j^1)$	$c(x_j^r)$	$\dots$	$c(x_j^{t_j})$			
$w_1$	$x_{11}^1$	$x_{11}^2$	$\dots$	$x_{11}^{t_1}$	$\dots$	$x_{1j}^1$	$x_{1j}^r$	$\dots$	$x_{1j}^{t_j}$	$\dots$	$\dots$	$=1$
$w_2$	$x_{21}^1$	$x_{21}^2$		$x_{21}^{t_1}$		$x_{2j}^1$	$x_{2j}^r$	$\dots$	$x_{2j}^{t_j}$			$=1$
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$		$\vdots$			$\vdots$
$w_n$	$x_{n1}^1$	$x_{n1}^2$	$\dots$	$x_{n1}^{t_1}$		$x_{nj}^1$	$x_{nj}^r$	$\dots$	$x_{nj}^{t_j}$			$=1$
$\alpha_1$	1	1	$\dots$	1		$\ddots$						$=1$
$\alpha_2$									$=1$			
$\vdots$					1		$\dots$	1	$\dots$	1	$=1$	
$\vdots$									$\ddots$	$=1$		
$\alpha_j$									1	$\dots$	1	$=1$

### 5.3.3 Algoritmo per risolvere $LP'$

#### 1. Master iniziale

Si generino  $E_j$  punti estremi di  $\text{conv}(X_j)$ ,  $j = 1 \text{ dots } m$ .

#### 2. Si risolva $LP'$ usando i correnti $E_j$ punti estremi e sia $(w, \alpha)$ la soluzione del duale $DLP'$

#### 3. Ottimalità della soluzione

La soluzione è ottima se

$$\sum_{i=1}^n w_i x_{ij}^r + \alpha_j \leq c(x_j^r), \quad r = E_j, \dots, t_j, \quad j = 1, \dots, m$$

che ricordando la definizione di  $c(x_j^r)$ , diviene

$$\sum_{i=1}^n w_i x_{ij}^r + \alpha_j \leq \sum_i \alpha_{ij} x_{ij}^r$$

oppure

$$\sum_{i=1}^n (w_i - c_{ij}) x_{ij}^r + \alpha_j \leq 0$$

Si risolva per ogni  $j = 1, \dots, m$

$$SP_j \begin{cases} z_{SP}^j = \text{Max} \sum_{i=1}^n (w_i - c_{ij}) x_{ij} + \alpha_j \\ \text{s.t.} \sum_i q_{ij} x_{ij} \leq Q_j \\ x_{ij} \in \{0, 1\}, \quad i = 1, \dots, n \end{cases}$$

Sia  $x_{ij}^*$  la soluzione ottima

4. Se  $z_{SP}^j \leq 0, \forall j$ , allora STOP; altrimenti per ogni  $j$  per cui  $z_{SP}^j > 0$  poni  $E_j = E_j + 1$ ,  $x_{ij}^{E_j} = x_{ij}^*$  e ritorna al punto 1.

## 5.4 Introduzione al simplesso Revisionato

Il vettore  $w = c_B B^{-1}$  e la matrice  $B^{-1}$  posson essere indentificati nel Tableau nel modo seguente:

### 5.4.1 Caso Semplice

$$\begin{cases} \text{Min } z = cx \\ Ax \leq b \quad (b \geq 0) \\ x \geq 0 \end{cases}$$

Si aggiungano  $m$  variabili slack  $x_{n+1}, \dots, x_{n+m}$ .

La base iniziale è  $B = [a_{n+1}, \dots, a_{n+m}] \equiv I$ .

Tabella 5.2: 1° Tableau

	z	$x_1$	$x_2$	$\dots$	$x_n$	$x_{n+1}$	$x_{n+1}$	$\dots$	$x_{n+m}$
z	1	$-c_1$	$-c_2$	$\dots$	$-c_n$	0	0	$\dots$	0
$x_{n+1}$	0	A				1			$b_1$
$x_{n+2}$	0						1		$b_1$
$\vdots$	$\vdots$						$\ddots$		$\vdots$
$x_{n+1}$	0							1	$b_m$

Tabella 5.3: Tableau iterazione  $t$

	z	$x_1$	$x_2$	$\dots$	$x_n$	$x_{n+1}$	$x_{n+1}$	$\dots$	$x_{n+m}$
z	1	$z_1 - c_1$	$z_2 - c_2$	$\dots$	$z_n - c_n$	$w_1$	$w_2$	$\dots$	$w_m c_B b$
$x_{n+1}$	0	$(y^{-1})$				$B^{-1}$			
$x_{n+2}$	0								
$\vdots$	$\vdots$								
$x_{n+1}$	0								

All'iterazione  $t$  sia  $B^{-1}$  l'inversa della base  $B$ .

- $B^{-1}$  si trova in corrispondenza alle colonne delle variabili di scarto (ovvero nelle colonne  $n+1, \dots, n+m$  e nelle righe  $1, \dots, m$ )
- Le variabili  $w = c_B B^{-1}$  sono nella riga 0 in corrispondenza alle colonne  $n+1, \dots, n+m$ . Ogni  $w_i$  corrisponde al costo ridotto  $z_{n+i} - c_{n+1}$  della variabile di scarto  $x_{n+1}$ ; infatti

$$z_{n+i} - c_{n+i} = c_B B^{-1} a_{n+i} - c_{n+i}$$

essendo  $w = c_B B^{-1}$  e  $c_{n+i} = 0$  ( $x_{n+i}$  è variabile di scarto)

$$z_{n+i} = c_{n+i} = w a_{n+i}$$

ma per la variabile di scarto  $x_{n+i}$  si ha

$$a_{n+1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

ed essendo  $w = (w_1, w_2, \dots, w_m)$ , si ha

$$z_{n+i} - c_{n+i} = (w_1, w_2, \dots, w_m) \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{bmatrix} = w_i$$

Le ultime  $m+1$  colonne del tableau consentono di fare il simplesso anche senza conoscere le prime  $n$  colonne.

Si può procedere come segue:

- si calcola  $z_j - c_j = wa_j - c_j$  per ogni variabile non base  $j \in R$ . Ciò è possibile essendo noto  $w$ !
- Si calcoli  $z_k - c_k = \max_{j \in R} [z_j - c_j]$ . Se  $z_k - c_k \leq 0$  STOP, la soluzione è ottima; altrimenti si procede
- si calcoli  $y^k = B^{-1}a_k$ . Ciò è possibile essendo noto  $B^{-1}$ . Se  $y^k \leq 0$  allora STOP (soluzione illimitata); altrimenti si proceda dopo aver ricostruito la colonna  $k$ . e si effettui il pivoting

	$z_k - c_k$	$w_1 \ w_2 \ \dots \ w_m$	$c_B \bar{b}$
$x_{B_1}$	$y_1^k$	$B^{-1}$	$\bar{b}_1$
	$y_2^k$		
	$\vdots$		
$x_{B_r}$	$y_r^k$		$\bar{b}_r$
	$\vdots$		
$x_{B_m}$	$y_m^k$		$\bar{b}_m$

su  $y_r^k$  dove  $\frac{\bar{b}_r}{y_r^k} = \min \left[ \frac{\bar{b}_i}{y_i^k} : y_i^k > 0 \right]$  ignorando le colonne da 1 a  $n$  ad eccezione della colonna  $k$ .

Dopo il pivoting nelle colonne  $n+1 - n+m$  ci sarà l'inversa della nuova base  $B' = (a_{B_1}, \dots, a_k, \dots, a_{B_m})$  e nella riga 0 avremo  $w' = c'_B B'^{-1}$ .



## 5.4.1.1 Esempio

$$\begin{aligned}
\text{Min } z &= -x_1 - 2x_2 + x_3 - x_4 - 4x_5 + 2x_6 \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 &= 6 \\
2x_1 - x_2 - 2x_3 + x_4 + x_8 &= 4 \\
x_3 + x_4 + 2x_5 + x_6 + x_9 &= 4 \\
x_1, \dots, x_9 &\geq 0
\end{aligned}$$

base iniziale  $B = [a_7, a_8, a_9]$ ,  $w = c_B B^{-1} = (0, 0, 0, 0)$   $\bar{b} = (6, 4, 4)^T$  e  $c_B \bar{b} = 0$ .  
 Calcolo  $z_5 - c_5 = 4 \geq z_j - c_j$ ,  $j = 1, \dots, 6$ , quindi

$$\begin{array}{c} \downarrow \\ x_5 \end{array}
\begin{array}{c} \text{RHS} \\ \text{variabili} \\ \text{base} \end{array}$$

$z_5 - c_5$	4	0	0	0	0
$y^5 = B^{-1}a_5 \equiv a_5$	1	1	0	0	6
	0	0	1	0	4
	<u>2</u>	0	0	1	4

$x_7$   
 $x_8$   
 $x_9 \rightarrow$

Dopo aver fatto il pivoting si ottiene

$$\begin{array}{c} \text{Matrice inversa} \\ \text{di } B' = [a_7, a_8, a_5] \rightarrow \end{array}$$

0	0	-2	-8
1	0	$-\frac{1}{2}$	4
0	1	0	4
0	0	$\frac{1}{2}$	2

$\text{RHS}$  var. base  
 $x_7$   
 $x_8$   
 $x_5$

Calcolo  $z_j = c_j = wa_j - c_j$  per le variabili non base dove  $w = (0, 0, -2)$ .  
 Si ha:

$$\begin{array}{c|cccccc}
j & 1 & 2 & 3 & 4 & 6 & 9 \\
\hline
z_j - c_j & 1 & 2 & -3 & -1 & -4 & -2
\end{array}$$

Quindi  $x_2$  è candidata ad entrare in base per cui calcoliamo  $y^2 = B^{-1}a_2$  e

$$\begin{array}{c} \downarrow \\ z_2 - c_2 \end{array}$$

2	0	0	-2	-8
<u>1</u>	1	0	$-\frac{1}{2}$	4
-1	0	1	0	4
0	0	0	$\frac{1}{2}$	2

$x_7 \rightarrow$   
 $x_8$   
 $x_5$

$$y^2 = B^{-1}a_2$$

Dopo aver fatto il pivoting si ottiene: Calcolo  $z_j - c_j$  per le variabili non base usando  $w = (-2, 0, 1)$

-2	0	-1	-16	
1	0	$-\frac{1}{2}$	4	$x_2$
1	1	$-\frac{1}{2}$	8	$x_8$
0	0	$\frac{1}{2}$	2	$x_5$

La soluzione è ottima!

j	1	3	4	6	9
$z_j - c_j$	-1	-3	-2	-5	-1

## 5.5 Metodo del Simpleso Revisionato

È un metodo per implementare il Simpleso al fine di risparmiare spazio in "memoria" e anche tempo calcolo.

### 5.5.1 Metodo del Simpleso in sintesi

1. Sia  $B$  una base ammissibile (calcola  $B^{-1}$ )
2. Calcola  $x_B = B^{-1}b = \bar{b}$  e quindi  $z = c_B B^{-1}b$
3. Calcola  $w = c_B B^{-1}$  e quindi  $z_j - c_j = w a_j - c_j, \forall j$  non-base.  
Scegli  $x_k$  tale che  $z_k - c_k = \underset{j \text{ non base}}{\text{Max}} \{z_j - c_j\}$ .
  - Se  $z_k - c_k \leq 0$  STOP
4. Calcola  $y^k = B^{-1}a_k$ : se  $y^k \leq 0$  STOP; altrimenti  $x_r: \frac{\bar{b}_r}{y_r^k} = \underset{i}{\text{Min}} \{ \frac{\bar{b}_i}{y_i^k} : y_i^k > 0 \}$ .  
Sostituisci in  $B$  la colonna  $a_r$  con  $a_k$  e vai al passo 1.

Noto  $B^{-1}$  si può costruire la matrice...

$w = c_B B^{-1}$	$c_B \bar{b}$
$B^{-1}$	$\bar{b}$

Sia  $x_k$  la variabile entrante e  $x_r$  quella uscente.

Si aggiorni il tableau effettuando il pivoting su

$y_r^k$  Ad ogni iterazione devono essere ricalcolati:

Base inversa	RHS	$x_k$
w	$c_B \bar{b}$	$z_k - c_k$
$B^{-1}$	$\bar{b}_1$	$y_1^k$
	$\vdots$	$\vdots$
	$\bar{b}_r$	$\textcircled{y_r^k}$
	$\vdots$	$\vdots$
	$\bar{b}_m$	$y_m^k$

1.  $z_j - c_j = w a_j - c_j$ , per ogni variabile  $j$  non-base
2. il vettore  $y^k = B^{-1}a_k$ , corrispondente alla variabile entrante.

Si riducono gli errori di arrotondamento che si accumulano nel simpleso tableau.

## 5.6 Simplexso revisionato e metodo due fasi

Nella fase 1

$$\begin{aligned} \text{Min } z &= 1x_\alpha \\ Ax + x_\alpha &= b \quad (b > 0) \\ x, x_\alpha &\geq 0 \end{aligned}$$

La base iniziale è  $B = [a_{n+1}, \dots, a_{n+m}] \equiv I_{m \times m}$   $\bar{b} = b$ ,  $c_B \bar{b} = \sum_{i=1}^m b_i$  e  $w = c_B B^{-1} = (1, 1, \dots, 1)$

$I = (1, \dots, 1)$ .

Quindi Se alla fine della prima fase si ha che il costo ottimo è nullo allora bisogna ricostruire la

		RHS	var. base
$w \rightarrow$	$\begin{matrix} 1 & 1 & \dots & 1 \end{matrix}$	$\sum_i b_i$	$\leftarrow c_B \bar{b} = c_B b$
	$\begin{matrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 1 \end{matrix}$	$\begin{matrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{matrix}$	$\begin{matrix} x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+m} \end{matrix}$
$B^{-1} \equiv I \rightarrow$		$\uparrow$ $B^{-1}b = \bar{b} = b$	

riga 0, ovvero  $w = c_B B^{-1}$  e  $c_B \bar{b}$  usando la funzione originale  $z = cx$  e quindi effettuare la fase 2.

Si consideri l'esempio già visto in precedenza

$$\begin{aligned} \text{Min } z &= x_1 - 2x_2 \\ x_1 + x_2 - x_3 &= 2 \\ -x_1 + x_2 - x_4 &= 1 \\ x_2 + x_5 &= 3 \\ x_1, \dots, x_5 &\geq 0 \end{aligned}$$

### 5.6.1 Fase 1

$$\begin{aligned} \text{Min } z &= x_6 + x_7 \\ x_1 + x_2 - x_3 + x_6 &= 2 \\ -x_1 + x_2 - x_4 + x_7 &= 1 \\ x_2 + x_5 &= 3 \\ x_1, \dots, x_7 &\geq 0 \end{aligned}$$

$B = [a_6, a_7, a_5] = I_{3 \times 3}$  e  $c_B = (1, 1, 0)$  quindi  $w = c_B B^{-1} = (1, 1, 0)$ ,  $\bar{b} = b$  e  $c_B \bar{b} = 3$ .

RHS				
1	1	0	3	var. base
1	0	0	2	$x_6$
0	1	0	1	$x_7$
0	0	1	3	$x_5$

Calcolo  $z_j - c_j$  per le variabili non base  $x_1, \dots, x_4$

quindi  $x_2$  è candidata ad entrare

in base e  $y^2 = B^{-1}a_2 = Ia_2 =$

$a_2$

j	1	2	3	4
$z_j - c_j$	0	2	-1	-1

$\downarrow$					
$x_2$					
2	1	1	0	3	
1	1	0	0	2	$x_6$
<b>1</b>	0	1	0	1	$x_7 \rightarrow$
1	0	0	1	3	$x_5$

Dopo il pivoting si ottiene

1	-1	0	1	
1	-1	0	1	$x_6$
0	1	0	1	$x_2$
0	-1	1	2	$x_5$

Calcolo  $z_j - c_j$  per ogni variabile  $j$  non base

j	1	2	3	7
$z_j - c_j$	2	-1	1	2

entra  $x_1$  e  $y^1 = B^{-1} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$

$\downarrow$					
$x_1$					
2	1	-1	0	1	
<b>2</b>	1	-1	0	1	$x_6 \rightarrow$
-1	0	1	1	0	$x_2$
1	0	-1	1	2	$x_5$

Dopo il pivoting:

$$\begin{array}{c|ccc|c}
 w \rightarrow & 0 & 0 & 0 & 0 & \leftarrow c_B \bar{b} \\
 \hline
 & \frac{1}{2} & -\frac{1}{2} & 0 & \frac{1}{2} & x_1 \\
 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{3}{2} & x_2 \\
 & -\frac{1}{2} & -\frac{1}{2} & 1 & -\frac{3}{2} & x_5
 \end{array}$$

### 5.6.2 Fase 2

Si calcoli  $w = c_B B^{-1}$  usando i costi  $c_1, c_2$  e  $c_5$  della funzione obiettivo originaria

$$\text{Min } z = x_1 - 2x_2$$

Quindi  $c_B = (c_1, c_2, c_5) = (1, -2, 0)$

$$\Rightarrow w = \left(-\frac{1}{2}, -\frac{3}{2}, 0\right)$$

$$\Rightarrow c_B \bar{b} = (1, -2, 0) \left(\frac{1}{2}, \frac{3}{2}, \frac{3}{2}\right)^T = -\frac{5}{2}$$

Si parte quindi con:

$$\begin{array}{ccc|c}
 -\frac{1}{2} & -\frac{3}{2} & 0 & -\frac{5}{2} \\
 \hline
 \frac{1}{2} & -\frac{1}{2} & 0 & \frac{1}{2} & x_1 \\
 \frac{1}{2} & \frac{1}{2} & 0 & \frac{3}{2} & x_2 \\
 -\frac{1}{2} & -\frac{1}{2} & 1 & \frac{3}{2} & x_5
 \end{array}$$

Si calcolino i costi  $z_j - c_j$  per le variabili non base e si continui.

## 5.7 Simplex tableau e Simplex revisionato

Computazionalmente si ha il seguente confronto

### 5.7.1 Occupazione di memoria

- Simplex  $(m+1) \times (n+1)$
- Simplex revisionato  $(m+1) \times (m+2)$ : se  $n \gg m$  il risparmio in memoria può essere rilevante!

### 5.7.2 Numero operazioni

		Pivoting	$z_j - c_j$	Totale
Simplex	Molt.	$(m+1)(n-m+1)$		$m(n-m) + n + 1$
Tableau	Add	$m(n-m+1)$		$m(n-m+1)$
Simplex	Molt.	$(m+1)^2$	$m(n-m)$	$m(n-m) + (m+1)^2$
Revisionato	Add	$m(m+1)$	$m(n-m)$	$m(n+1)$

Sembra vantaggioso il Simplex Tableau.

Nei problemi reali  $d = \frac{\text{numero elementi } \neq 0 \text{ di } A}{m \times n} \approx 0.05$

il calcolo di  $z_c - c_j = \sum_{i=1}^m w_i a_{ij} - c_j$  può essere accelerato da  $\sum_{i \in R_j} w_i a_{ij} - c_j$  dove  $R_j = \{i : a_{ij} \neq 0, i = 1, \dots, m\}$ .

Complessivamente il calcolo di  $z_j - c_j$  è  $d \cdot m \cdot (n - m)$ .