

Contents

1	Programmazione lineare	3
1.1	Modelli matematici	3
1.2	Problema di programmazione matematica	6
1.3	Problema di programmazione lineare	9
1.4	Teorema fondamentale	12
1.5	Procedimento del cardine	16
1.6	Algoritmo del simplesso - Dantzig (1947)	18
1.6.1	Ricerca della soluzione ottimale	20
1.6.2	Interpretazione geometrica	23
1.7	Terminazione	24
1.7.1	Ottimalità	24
1.7.2	Funzione obiettivo superiormente illimitata	25
1.8	Convergenza	27
1.9	Ricerca di una tabella ammissibile	28
1.10	Casi particolari	32
1.10.1	Soluzione degenera e ciclaggio	32
1.10.2	Infinite soluzioni ottimali	32
1.10.3	Problemi con vincoli di uguaglianza	33
1.10.4	Scelta della variabile uscente	34
2	Dualità	35
2.1	Problema duale	35
2.2	Proprietà di un problema e del suo duale	36
2.3	Interpretazione economica del problema duale	42
3	Programmazione lineare a numeri interi	51
3.1	Problemi lineari interi	51
3.2	Metodi branch and bound	54
3.2.1	Rilassamento	55
3.2.2	Separazione	56
3.2.3	Eliminazione	57
3.2.4	Algoritmo branch and bound (Land e Doig, 1960 - Little, 1963)	58
3.3	Metodi cutting plane	60
3.3.1	Algoritmo elementare (Dantzig, 1959)	61
3.3.2	Algoritmo di Gomory (1958)	61
4	Modelli lineari a numeri interi	64
4.1	Problema dello zaino	64
4.2	Problema del trasporto	65
4.3	Problema del magazzino	66
4.4	Problema dell'assegnazione	67
4.5	Problema del commesso viaggiatore	68

5	Applicazioni del metodo Branch and Bound	69
5.1	Problema dello zaino (Problema del massimo)	69
5.2	Problema dell'assegnazione (problema di minimo)	75
5.3	Problema con variabili intere qualsiasi (PLI misto)	80
5.4	Rilassamenti	84
5.4.1	Rilassamento continuo	84
5.4.2	Eliminazione di vincoli	85
5.4.3	Rilassamento lagrangiano	86
5.4.4	Rilassamento surrogato	87
5.5	Risoluzione per ispezione	88
6	Complementi di programmazione lineare	91
6.1	Variazione dei coefficienti iniziali	91
6.1.1	Applicazioni	93
6.2	Algoritmi greedy	94
6.2.1	Algoritmo greedy per il problema dello zaino	95
6.2.2	Algoritmo greedy per il problema del commesso viaggiatore	97
7	Teoria delle reti	98
7.1	Grafi	98
7.2	Reti	101
7.2.1	Flusso su una rete	102
7.2.2	Problema del flusso di costo minimo	103
7.3	Minimo Spanning Tree	104
7.4	Cammino minimo	110
7.4.1	SPP da un nodo v_s a tutti gli altri nodi	111
7.4.2	SPP tra qualsiasi coppia di nodi	117
7.5	Flusso massimo	118
7.6	Algoritmo del minimo cammino aumentante (Edmonds e Karp, 1972)	131
7.7	Complessità computazionale degli algoritmi di flusso massimo	134
7.8	Problema di produzione e magazzino	136
7.9	Problema di routing e scheduling	139
7.10	Problema di progettazione	140
7.10.1	CPM (Roy, 1960)	142
7.10.2	PERT (Malcom, Roseboom, Clark, Fazar, 1959)	146
8	Problema del flusso di costo minimo	150
8.1	Formulazione del problema	150
8.2	Problema di circolazione	153
8.3	Algoritmo Out of Kilter (Minty, 1960)	154

1 Programmazione lineare

1.1 Modelli matematici

Modelli matematici per la logistica (la separazione tra i due gruppi non è netta):

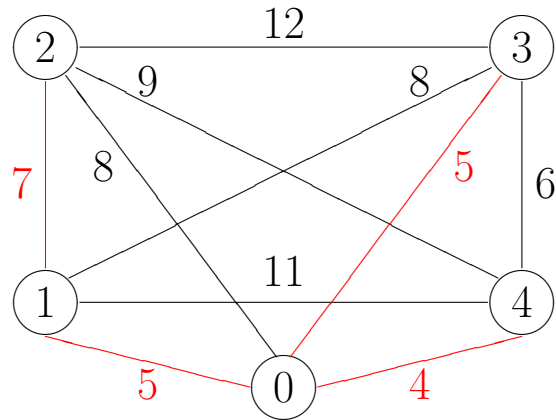
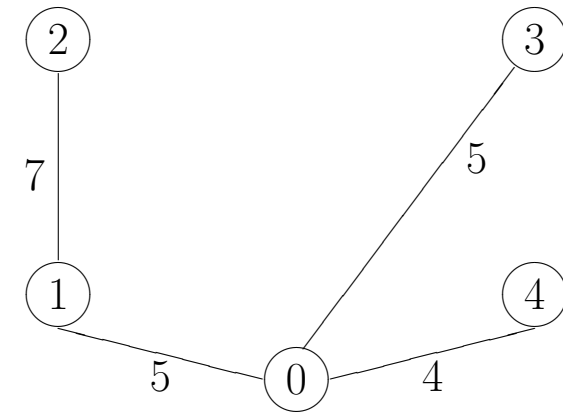
- Modelli di programmazione matematica
 - Produzione
 - Bin packing (Zaino)
 - Trasporto
 - Magazzino
 - Assegnazione
 - Commesso viaggiatore
 - Scheduling
 - Supply Chain

- Modelli su reti
 - Albero di costo minimo
 - Cammino minimo
 - Flusso massimo
 - Routing e scheduling
 - Localizzazione (*Location*)
 - PERT (*Project Evaluation and Review Techniques*)

I modelli possono essere utilizzati in fase progettuale e/o gestionale

Esempio 1.1 (Connessione - Progettazione e gestione)

Un'azienda ha 4 impianti (1, 2, 3, 4) che devono essere collegati ad una centrale elettrica (0):

PROGETTAZIONE**GESTIONE**

1.2 Problema di programmazione matematica

- elementi del problema da determinare: *variabili decisionali*
- relazioni (funzioni matematiche) intercorrenti tra le variabili
 - *funzione obiettivo*: valore del problema
 - *vincoli*: definiscono i valori ammissibili delle variabili

Forma generale

$$\max \{f(x) \text{ s.t. } x \in \mathbb{R}^n, \quad g_i(x) \leq 0, \quad i = 1, \dots, m\}$$

oppure:

$$\begin{array}{ll} \max & f(x) \\ \text{s.t.} & g_i(x) \leq 0, \quad i = 1, \dots, m \end{array}$$

dove $x \in \mathbb{R}^n$ sono le variabili decisionali

$f : \mathbb{R}^n \rightarrow \mathbb{R}$ è la funzione obiettivo

$g_i(x) \leq 0, g_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, m$ sono i vincoli

Riduzione alla forma generale

1. $\min \{f(x)\} = -\max \{-f(x)\}$
2. $g(x) \geq 0 \Leftrightarrow -g(x) \leq 0$
3. $g(x) = 0 \Leftrightarrow g(x) \leq 0; -g(x) \leq 0$

Soluzioni ammissibili

S_a = insieme dei valori di x per cui i vincoli sono soddisfatti

= insieme delle soluzioni ammissibili o insieme ammissibile o regione ammissibile

Può essere vuoto, limitato o illimitato

Soluzioni ottimali

S_{ott} = insieme dei valori di S_a in cui la funzione obiettivo assume il valore massimo

= insieme delle soluzioni ottimali o insieme ottimale o regione ottimale

Può essere vuoto, limitato o illimitato

Soluzione di un problema di programmazione matematica

Determinare sia una soluzione ottimale (punto di massimo) che il valore della funzione obiettivo (valore massimo)

- ricerca una strategia ottimale $\rightarrow$ punto di massimo
- confronto tra differenti strategie $\rightarrow$ valore massimo

Se l'insieme ammissibile non è chiuso può non esistere il massimo, ma solo l'estremo superiore

1.3 Problema di programmazione lineare

E' un problema di programmazione matematica in cui la funzione obiettivo e i vincoli sono lineari, o più esattamente affini

Forma canonica

$$\begin{aligned} \max \quad & z = c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

dove c (vettore dei costi = gradiente della funzione z) e $x \in \mathbb{R}^n$, A (matrice dei vincoli) è una matrice $m \times n$ e b (vettore dei termini noti) $\in \mathbb{R}^m$

Se nella forma canonica $b \geq 0$ il problema ammette la forma normale

Riduzione alla forma canonica

1. $\min z = c^T x \Leftrightarrow -\max -z = (-c)^T x$
2. $Ax \geq b \Leftrightarrow (-A)x \leq (-b)$
3. $Ax = b \Leftrightarrow Ax \leq b; \quad (-A)x \leq (-b)$
4. $x \leq 0 \Leftrightarrow (-x) \geq 0$
5. x non vincolata $\Leftrightarrow x = x' - x''; \quad x' \geq 0; \quad x'' \geq 0$

Soluzioni ammissibili

S_a è convesso in quanto intersezione dei semispazi $(Ax)_j \leq b_j, j = 1, \dots, m; x_i \geq 0, i = 1, \dots, n$ e può essere:

- vuoto (vincoli incompatibili)
- limitato (poliedro convesso)
- illimitato (troncone convesso)

$(Ax)_j = b_j, j = 1, \dots, m; x_i = 0, i = 1, \dots, n$ si dicono iperpiani generatori

Un punto di S_a intersezione unica di almeno n iperpiani generatori si dice vertice di S_a

Un punto di S_a intersezione unica di più di n iperpiani generatori si dice vertice degenero

L'intersezione di almeno $n - 1$ iperpiani generatori di cui $n - 1$ linearmente indipendenti è una retta

La parte di retta appartenente ad S_a si dice spigolo di S_a

I due vertici estremi di uno spigolo limitato si dicono adiacenti

Soluzioni ottimali

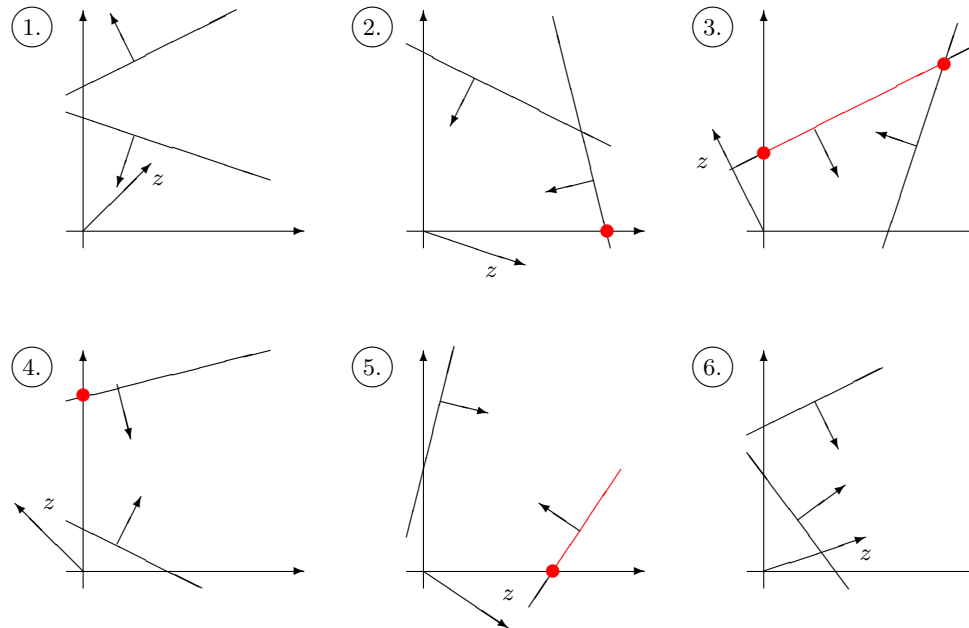
S_{ott} è insieme convesso ed è contenuto nella frontiera di S_a , se z non è costante, e può essere:

- vuoto
- costituito da un unico punto
- costituito da infiniti punti
 - limitato (poliedro convesso)
 - illimitato (troncone convesso)

Se S_a è vuoto non esistono soluzioni ottimali

Se S_a è un poliedro convesso esistono sempre soluzioni ottimali (teorema di Weierstrass)

Se S_a è un troncone convesso o esistono soluzioni ottimali o la funzione obiettivo è superiormente illimitata



1.4 Teorema fondamentale

Teorema 1.1 (Teorema fondamentale della programmazione lineare)

Se un problema lineare ha soluzioni ottimali almeno una di esse è un vertice di S_a

Dimostrazione

Si possono distinguere due casi:

1. S_a costituito da un poliedro convesso K
2. S_a costituito da un troncone convesso T

Caso 1 Siano $x_1, \dots, x_p$ i vertici di K

Sia x^* una soluzione ottimale del problema lineare scritta come combinazione convessa dei vertici di K :

$$x^* = \sum_{i=1, \dots, p} \lambda_i x_i \quad \text{con} \quad \sum_{i=1, \dots, p} \lambda_i = 1; \lambda_i \geq 0, i = 1, \dots, p$$

Posto $z_i = z(x_i)$ e con $z^* = z(x^*)$, per la linearità di z si ha:

$$z^* = \sum_{i=1, \dots, p} \lambda_i z_i$$

Supponendo per assurdo che nessun vertice di K sia ottimale si ha $z_i < z^*, i = 1, \dots, p$ e quindi:

$$z^* = \sum_{i=1, \dots, p} \lambda_i z_i < \sum_{i=1, \dots, p} \lambda_i z^* = z^* \sum_{i=1, \dots, p} \lambda_i = z^*$$

Caso 2 T può essere scritto come somma di un poliedro convesso K_1 , avente gli stessi vertici di T , e un cono poliedrico K_0 , avente gli spigoli paralleli agli spigoli illimitati di T :

$$T = K_1 + K_0$$

Siano $u_1, \dots, u_p$ i vertici di K_1 e $v_1, \dots, v_q$ i generatori di K_0

Sia x^* una soluzione ottimale del problema lineare che può essere scritta come:

$$x^* = u + v \text{ con } u \in K_1 \text{ e } v \in K_0$$

A loro volta u e v possono essere scritti come:

$$\begin{aligned} u &= \sum_{i=1, \dots, p} \lambda_i u_i \text{ con } \sum_{i=1, \dots, p} \lambda_i = 1, \lambda_i \geq 0, i = 1, \dots, p \\ v &= \sum_{j=1, \dots, q} \mu_j v_j \text{ con } \mu_j \geq 0, j = 1, \dots, q \end{aligned}$$

quindi:

$$x^* = \sum_{i=1, \dots, p} \lambda_i u_i + \sum_{j=1, \dots, q} \mu_j v_j$$

Posto $z_i = z(u_i)$, $z'_j = z(v_j)$ e $z^* = z(x^*)$ per la linearità di z si ha:

$$z^* = \sum_{i=1, \dots, p} \lambda_i z_i + \sum_{j=1, \dots, q} \mu_j z'_j$$

I valori z'_j sono tutti non positivi, altrimenti al crescere di μ_j anche z^* crescerebbe, quindi:

$$\sum_{j=1, \dots, q} \mu_j z'_j \leq 0$$

Inoltre supponendo per assurdo che nessun vertice di T sia ottimale si ha $z_i < z^*$, $i = 1, \dots, p$ e quindi:

$$z^* = \sum_{i=1, \dots, p} \lambda_i z_i + \sum_{j=1, \dots, q} \mu_j z'_j < \sum_{i=1, \dots, p} \lambda_i z^* = z^* \sum_{i=1, \dots, p} \lambda_i = z^*$$



- Il teorema precedente riduce la ricerca della soluzione ottimale ai soli vertici di S_a e costituisce il fondamento dell'algoritmo del simplesso
- L'enunciato riportato sottintende che il problema sia in forma canonica
Si consideri il seguente problema:

$$\max \quad z = x_1 + x_2$$

$$s.t. \quad x_1 + x_2 \leq 1$$

$z^* = 1$, $S_{ott} = \{(x_1, x_2) \in \mathbb{R}^2 | x_1 + x_2 = 1\}$ ma non esistono vertici

- Esiste un teorema, analogo ma meno importante, relativo alle soluzioni ammissibili:
Se un problema lineare ha soluzioni ammissibili almeno una di esse è un vertice di S_a

Un problema lineare presenta uno dei seguenti casi mutualmente esclusivi:

1. S_a è vuoto
2. esistono soluzioni ottimali
3. la funzione z è superiormente illimitata

1 e 3 si dicono casi di impossibilità

1.5 Procedimento del cardine

Siano dati n vettori $v_1, \dots, v_n$, non necessariamente linearmente indipendenti e m vettori $a_1, \dots, a_m$ appartenenti allo spazio generato da $v_1, \dots, v_n$

$a_1, \dots, a_m$ possono essere espressi come combinazione lineare di $v_1, \dots, v_n$:

$$a_h = \lambda_{h1}v_1 + \dots + \lambda_{hn}v_n \quad h = 1, \dots, m$$

Se $\lambda_{ij} \neq 0$ è possibile scambiare il vettore a_i e il vettore v_j :

$$v_j = -\frac{\lambda_{i1}}{\lambda_{ij}}v_1 - \dots - \frac{\lambda_{i,j-1}}{\lambda_{ij}}v_{j-1} + \frac{1}{\lambda_{ij}}a_i - \frac{\lambda_{i,j+1}}{\lambda_{ij}}v_{j+1} - \dots - \frac{\lambda_{in}}{\lambda_{ij}}v_n$$

e sostituendo si ha:

$$\begin{aligned} a_h = & \left(\lambda_{h1} - \lambda_{i1} \frac{\lambda_{hj}}{\lambda_{ij}} \right) v_1 + \dots + \left(\lambda_{h,j-1} - \lambda_{i,j-1} \frac{\lambda_{hj}}{\lambda_{ij}} \right) v_{j-1} + \\ & + \frac{\lambda_{hj}}{\lambda_{ij}} a_i + \left(\lambda_{h,j+1} - \lambda_{i,j+1} \frac{\lambda_{hj}}{\lambda_{ij}} \right) v_{j+1} + \dots + \left(\lambda_{hn} - \lambda_{in} \frac{\lambda_{hj}}{\lambda_{ij}} \right) v_n \quad h \neq i \end{aligned}$$

Lo scambio di a_i e v_j è detto *procedimento del cardine* e l'elemento λ_{ij} è detto *cardine* o *pivot*

Se i vettori $v_1, \dots, v_n$ costituiscono una base anche i vettori a_i e v_k , $k \neq j$ costituiscono una base per lo stesso spazio

Rappresentazione tabellare:

	v_1	$\dots$	v_j	$\dots$	v_n
a_1	λ_{11}	$\dots$	λ_{1j}	$\dots$	λ_{1n}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
a_i	λ_{i1}	$\dots$	λ_{ij}	$\dots$	λ_{in}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
a_m	λ_{m1}	$\dots$	λ_{mj}	$\dots$	λ_{mn}

Facendo cardine su λ_{ij} si perviene alla tabella:

	v_1	$\dots$	a_i	$\dots$	v_n
a_1	$\lambda_{11} - \lambda_{i1} \frac{\lambda_{1j}}{\lambda_{ij}}$	$\dots$	$\frac{\lambda_{1j}}{\lambda_{ij}}$	$\dots$	$\lambda_{1n} - \lambda_{in} \frac{\lambda_{1j}}{\lambda_{ij}}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
v_j	$-\frac{\lambda_{i1}}{\lambda_{ij}}$	$\dots$	$\frac{1}{\lambda_{ij}}$	$\dots$	$-\frac{\lambda_{in}}{\lambda_{ij}}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
a_m	$\lambda_{m1} - \lambda_{i1} \frac{\lambda_{mj}}{\lambda_{ij}}$	$\dots$	$\frac{\lambda_{mj}}{\lambda_{ij}}$	$\dots$	$\lambda_{mn} - \lambda_{in} \frac{\lambda_{mj}}{\lambda_{ij}}$

1.6 Algoritmo del simplesso - Dantzig (1947)

Dato un problema lineare in forma canonica:

$$\begin{aligned}
 \max \quad & z = c_1x_1 + \dots + c_nx_n + c_0 \\
 \text{s.t.} \quad & a_{11}x_1 + \dots + a_{1n}x_n \leq b_1 \\
 & \dots\dots\dots \\
 & a_{m1}x_1 + \dots + a_{mn}x_n \leq b_m \\
 & x_i \geq 0 \qquad \qquad \qquad i = 1, \dots, n
 \end{aligned}$$

Introducendo le variabili $u_j, j = 1, \dots, m$ si ottiene:

$$\begin{aligned}
 \max \quad & z = c_1x_1 + \dots + c_nx_n + c_0 \\
 \text{s.t.} \quad & u_1 = -a_{11}x_1 - \dots - a_{1n}x_n + b_1 \\
 & \dots\dots\dots \\
 & u_m = -a_{m1}x_1 - \dots - a_{mn}x_n + b_m \\
 & x_i \geq 0 \qquad \qquad \qquad i = 1, \dots, n \\
 & u_j \geq 0 \qquad \qquad \qquad j = 1, \dots, m
 \end{aligned}$$

Rappresentazione tabellare:

	x_1	$\dots$	x_n	
u_1	$-a_{11}$	$\dots$	$-a_{1n}$	b_1
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
u_m	$-a_{m1}$	$\dots$	$-a_{mn}$	b_m
z	c_1	$\dots$	c_n	c_0

L'intersezione dei semispazi $x_i \geq 0, i = 1, \dots, n, u_j \geq 0, j = 1, \dots, m$ definisce la regione ammissibile

Le variabili x_i formano una base di $\mathbb{R}^n$ in quanto associate ai versori di $\mathbb{R}^n$

Per definizione si associa alla tabella il vertice intersezione degli iperpiani ottenuti annullando le variabili della base corrente

L'ultima colonna rappresenta il valore delle variabili non in base corrente e il coefficiente c_0 rappresenta il valore di z

Se i coefficienti dell'ultima colonna, escluso c_0 , sono tutti non negativi il punto risulta ammissibile

1.6.1 Ricerca della soluzione ottimale

Si cerca una soluzione ottimale spostandosi da un vertice ammissibile ad un altro vertice ammissibile adiacente, in modo che nel nuovo vertice z abbia un valore non peggiore

Si opera uno scambio di variabili col procedimento del cardine tra una variabile in base detta *variabile uscente* e una variabile fuori base detta *variabile entrante*

L'algoritmo del simplesso assegna un criterio per determinare le variabili da scambiare

Sia data la seguente tabella di un problema lineare ad una generica iterazione dell'algoritmo del simplesso, relativa ad un vertice ammissibile:

	y_1	$\dots$	y_i	$\dots$	y_n	
y_{n+1}	α_{11}	$\dots$	α_{1i}	$\dots$	α_{1n}	β_1
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
y_{n+j}	α_{j1}	$\dots$	α_{ji}	$\dots$	α_{jn}	β_j
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
y_{n+m}	α_{m1}	$\dots$	α_{mi}	$\dots$	α_{mn}	β_m
z	γ_1	$\dots$	γ_i	$\dots$	γ_n	γ_0

Scelta della variabile uscente

Facendo cardine su α_{ji} il valore di z risulta $\gamma_0 - \gamma_i \frac{\beta_j}{\alpha_{ji}}$

$y'_i = -\frac{\beta_j}{\alpha_{ji}}$ deve essere non negativo, quindi per ottenere un incremento della funzione obiettivo γ_i deve essere positivo

Scelta della variabile entrante

Scegliendo y_{n+j} come variabile entrante essendo y_i la variabile uscente si ha:

$$\begin{aligned} y'_i &= -\frac{\beta_j}{\alpha_{ji}} \\ y'_{n+k} &= \beta_k - \beta_j \frac{\alpha_{ki}}{\alpha_{ji}} \quad k \neq j \end{aligned}$$

Per l'ammissibilità:

$$\begin{aligned} -\frac{\beta_j}{\alpha_{ji}} &\geq 0 \\ \beta_k - \beta_j \frac{\alpha_{ki}}{\alpha_{ji}} &\geq 0 \quad k \neq j \end{aligned}$$

La prima è vera solo se $\alpha_{ji} < 0$

La seconda è vera se $\alpha_{ki} \geq 0$, mentre per $\alpha_{ki} < 0$ equivale a:

$$\frac{\beta_j}{\alpha_{ji}} \geq \frac{\beta_k}{\alpha_{ki}}$$

che è vera se:

$$j \in \underset{\alpha_{ki} < 0}{argmax} \left\{ \frac{\beta_k}{\alpha_{ki}} \right\} = \underset{\alpha_{ki} < 0}{argmin} \left\{ \left| \frac{\beta_k}{\alpha_{ki}} \right| \right\}$$

- Il criterio di scelta della variabile uscente non è rigoroso
In fase di implementazione è necessario quindi precisare il criterio (il massimo dei γ_i , il primo da sinistra, ecc.)
- Il criterio di scelta della variabile entrante è rigoroso tranne nei casi in cui esistano più indici per cui si ottiene il minimo in valore assoluto; in questo caso in fase di implementazione è necessario precisare il criterio (il primo dall'alto, ecc.)

Una volta determinate le variabili da scambiare si applica il procedimento del cardine

1.6.2 Interpretazione geometrica

Scelta dell'iperpiano generatore uscente

I coefficienti della funzione obiettivo rappresentano le componenti del gradiente secondo la base corrente z è crescente lungo gli spigoli con y_i crescente se $\gamma_i \geq \frac{1}{2}$ positivo

Scelta dell'iperpiano generatore entrante

Scegliendo $y_i = 0$ come iperpiano generatore uscente di base lo spostamento dal vertice corrente al vertice ammissibile adiacente lungo lo spigolo $y_h = 0, h \neq i$ si ottiene facendo entrare in base il primo iperpiano generatore non in base. Il punto di intersezione dell'iperpiano generatore $y_{n+k} = 0, k = 1, \dots, m$ con lo spigolo $y_h = 0, h \neq i$ ha coordinate, nella base corrente:

$$\begin{aligned} y_h &= 0 & h &\neq i \\ y_i &= -\frac{\beta_k}{\alpha_{ki}} (\Leftarrow y_{n+k} = \alpha_{ki}y_i + \beta_k = 0) \end{aligned}$$

Se $y_i < 0$ lo spostamento è avvenuto nella direzione decrescente per y_i e quindi non è ammissibile altrimenti l'iperpiano generatore che viene incontrato per primo è quello per cui il valore di y_i è minimo

1.7 Terminazione

1.7.1 Ottimalità

Dall'ultima riga si ricava:

$$z = \gamma_1 y_1 + \cdots + \gamma_i y_i + \cdots + \gamma_n y_n + \gamma_0$$

Se $\gamma_i \leq 0, i = 1, \dots, n$ allora in tutta la regione ammissibile il valore di z è non migliore di quello corrente

Si definisce *tabella ottimale* una tabella che ha l'ultima riga, tranne al più l'ultimo elemento, tutta non positiva e l'ultima colonna, tranne al più l'ultimo elemento, tutta non negativa

1.7.2 Funzione obiettivo superiormente illimitata

Dalla tabella si ricava:

$$y_{n+k} = \alpha_{k1}y_1 + \dots + \alpha_{ki}y_i + \dots + \alpha_{kn}y_n + \beta_k \quad k = 1, \dots, m$$

per cui se in corrispondenza di un coefficiente $\gamma_i > 0$ si ha $\alpha_{ki} \geq 0, k = 1, \dots, m$ allora le variabili y_{n+k} rimangono ammissibili al crescere di y_i , per cui $\gamma_i y_i$ può crescere indefinitamente e z risulta superiormente illimitata

- Regione ammissibile illimitata

Indipendentemente dal valore di γ_i la condizione $\alpha_{ki} \geq 0, k = 1, \dots, m$ vuol dire che la regione ammissibile ammette uno spigolo illimitato, cioè risulta essere un troncone

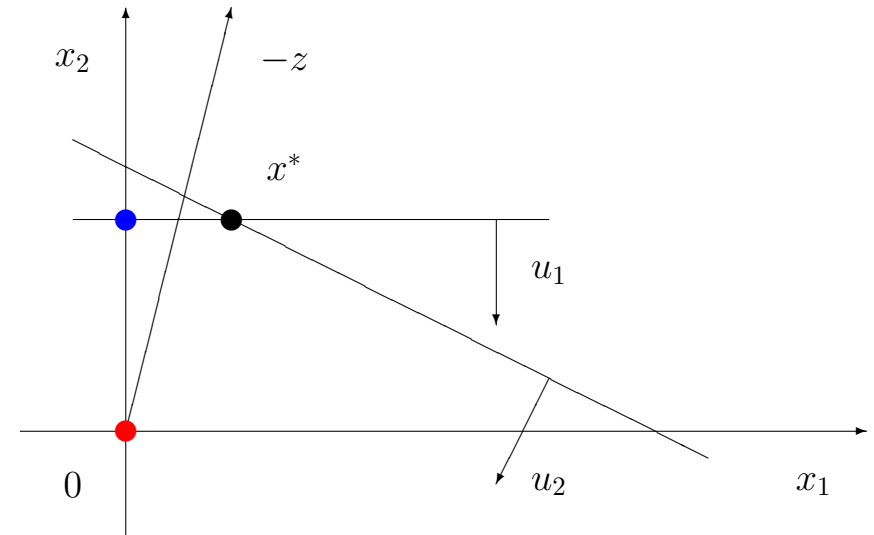
Esempio 1.2 Risolvere con l'algoritmo del simplesso il seguente problema di programmazione lineare:

$$\begin{aligned} \min \quad & z = -x_1 - 4x_2 \\ \text{s.t.} \quad & x_2 \leq 2 \\ & x_1 + 2x_2 \leq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Riportando il problema in forma canonica, la tabella iniziale è data da:

	x_1	x_2	
u_1	0	-1	2
u_2	-1	-2	5
$-z$	1	4	0
$x = (0, 0), z = 0$			
	x_1	u_1	
x_2	0	-1	2
u_2	-1	2	1
$-z$	1	-4	8
$x = (0, 2), z = -8$			
	u_2	u_1	
x_2	0	-1	2
x_1	-1	2	1
$-z$	-1	-2	9

La tabella è ottimale e la soluzione è $x^* = (1, 2), z^* = -9$



1.8 Convergenza

L'algoritmo del simplesso converge in quanto i vertici sono in numero finito e, tranne nei casi degeneri, ad ogni iterazione si determina un vertice in cui la funzione obiettivo è non peggiore dei precedenti, per cui non si ritorna su uno stesso vertice

Quindi in un numero finito di passi si determina un vertice ottimale, se esiste, oppure si determina un vertice che è origine di uno spigolo illimitato su cui la funzione obiettivo è superiormente illimitata

1.9 Ricerca di una tabella ammissibile

Se il problema lineare non ammette la forma normale la tabella iniziale associata all'origine non è ammissibile, pertanto è necessario determinare una tabella ammissibile, se la regione ammissibile è non vuota

In questo caso si applica un diverso criterio per determinare le variabili da scambiare

Per la non ammissibilità esiste almeno un $\beta_j < 0$

Si cerca un $\alpha_{ji} > 0$ e si determina così la variabile uscente y_i

Scegliendo come cardine $\alpha_{ki} < 0$ con $\beta_k > 0$ la variabile entrante è quella per cui si ha il minimo di $\left| \frac{\beta_k}{\alpha_{ki}} \right|$, in modo che tutti i termini noti non negativi restino non negativi

Facendo cardine $\alpha_{ki} > 0$ con $\beta_k < 0$ la variabile entrante è quella per cui si ha il massimo di $\left| \frac{\beta_k}{\alpha_{ki}} \right|$, in modo che tutti i termini noti negativi corrispondenti a coefficienti positivi nella colonna della variabile uscente diventino non negativi

Il criterio converge poichè ad ogni passo i termini negativi diminuiscono di numero o diminuisce il valore assoluto del più negativo

- Il criterio di scelta della variabile uscente non è rigoroso
In fase di implementazione è necessario quindi precisare il criterio (il massimo degli α_{ji} , il primo da sinistra, ecc.)
- Il criterio di scelta della variabile entrante è anche meno rigoroso
Si noti che il caso $\alpha_{ji} > 0$ e $\beta_j < 0$ esiste sempre
In fase di implementazione è necessario precisare il criterio

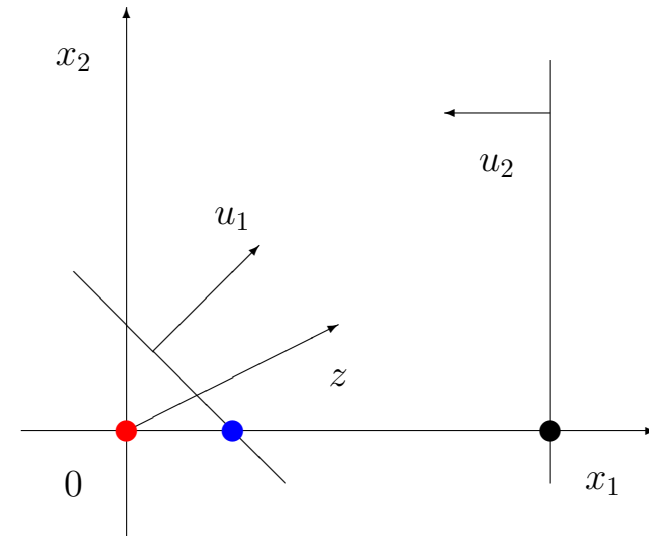
Esempio 1.3 Risolvere con l'algoritmo del simplesso il seguente problema di programmazione lineare:

$$\begin{aligned} \max \quad & z = 2x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 1 \\ & x_1 \leq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Riportando il problema in forma canonica, la tabella iniziale è data da:

	x_1	x_2	
u_1	1	1	-1
u_2	-1	0	4
z	2	1	0
$x = (0, 0), z = 0$			
	u_1	x_2	
x_1	1	-1	1
u_2	-1	1	3
z	2	-1	2
$x = (1, 0), z = 2$			
	u_2	x_2	
x_1	-1	0	4
u_1	-1	1	3
z	-2	1	8
$x = (4, 0), z = 8$			

La colonna di x_2 indica che la funzione obiettivo è superiormente illimitata



Regione ammissibile vuota

Se a $\beta_j < 0$ corrispondono $\alpha_{ji} \leq 0, i = 1, \dots, n$ allora non esistono soluzioni ammissibili perchè la condizione:

$$y_{n+j} = \alpha_{j1}y_1 + \dots + \alpha_{ji}y_i + \dots + \alpha_{jn}y_n + \beta_j \geq 0$$

non può essere soddisfatta

- Regione ammissibile limitata

Se ad un termine $\beta_j \geq 0$ corrispondono coefficienti $\alpha_{ji} < 0, i = 1, \dots, n$ vuol dire che la regione ammissibile è limitata in quanto da:

$$\alpha_{j1}y_1 + \dots + \alpha_{ji}y_i + \dots + \alpha_{jn}y_n + \beta_j \geq 0$$

si ricava:

$$y_i \leq -\frac{\beta_j}{\alpha_{ji}} \quad i = 1, \dots, n$$

- Se $\beta_j = 0$ e $\alpha_{ji} < 0, i = 1, \dots, n$ la regione ammissibile si riduce ad un punto

1.10 Casi particolari

1.10.1 Soluzione degenerare e ciclaggio

Se in una tabella esiste $\beta_j = 0$, il corrispondente iperpiano $y_{n+j} = 0$ pur non formando la base, passa ugualmente per il vertice rappresentato dalla tabella, che risulta degenerare

Se y_{n+j} viene scelta come variabile entrante i valori delle variabili restano invariati e la nuova tabella rappresenta lo stesso vertice

Oltre a rimanere nello stesso vertice, possono presentarsi ciclicamente le stesse basi

Questo fenomeno detto ciclaggio può essere evitato utilizzando diversi metodi, ad esempio la *regola di Bland* e il *metodo delle perturbazioni di Wolfe*

1.10.2 Infinite soluzioni ottimali

Se $\gamma_i = 0$ incrementando la variabile y_i la funzione obiettivo non varia; pertanto se la tabella è ottimale scegliendo y_i come variabile uscente è possibile determinare un nuovo vertice ottimale (salvo nel caso di vertice degenerare)

Per la linearità del problema vuol dire che tutti i punti dello spigolo sono ottimali

Se tutti gli $\alpha_{ji}, j = 1, \dots, m$ sono non negativi allora esiste uno spigolo illimitato di punti ottimali

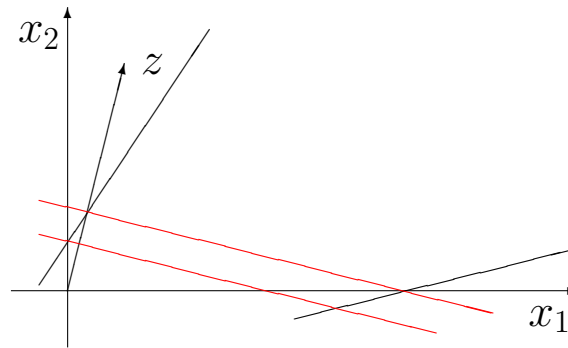
1.10.3 Problemi con vincoli di uguaglianza

Sono possibili differenti metodologie:

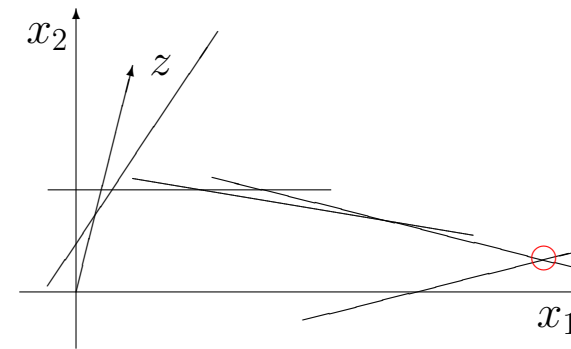
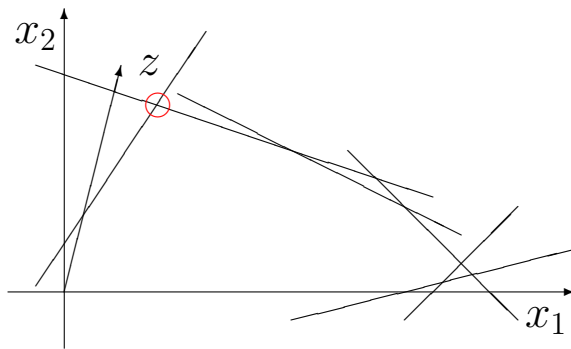
- ricondurlo alla forma canonica
- definire un problema associato in cui tutti i vincoli di uguaglianza $Ax = b$ sono scritti in forma di disuguaglianza $Ax \leq b$ e la funzione obiettivo è data dal valore dei vincoli $1^T(Ax - b)$
se esiste una soluzione ottimale di valore nullo per il problema associato si ritorna al problema dato
le condizioni di ottimalità escludono le u relative ai vincoli di uguaglianza
- "forzare" i vincoli di uguaglianza ad entrare in base (poichè devono avere valore nullo), verificando che quelli eventualmente rimasti fuori base abbiano anch'essi valore nullo
le condizioni di ottimalità escludono le u relative ai vincoli di uguaglianza

1.10.4 Scelta della variabile uscente

La scelta dell'elemento su cui “fare cardine” riveste un ruolo molto importante, ma non è possibile fissare regole generali, visto che la tabella rappresenta una situazione “locale”



Componente maggiore del gradiente o incremento maggiore della funzione obiettivo?
Dipende ...



2 Dualità

2.1 Problema duale

Dato il problema lineare (primale):

$$\begin{array}{ll} \max & z = c^T x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array}$$

È sempre possibile definire un problema ad esso associato (duale):

$$\begin{array}{ll} \min & w = b^T u \\ \text{s.t.} & A^T u \geq c \\ & u \geq 0 \end{array}$$

- A matrice $m \times n \Rightarrow x \in \mathbb{R}^n, u \in \mathbb{R}^m$

2.2 Proprietà di un problema e del suo duale

1. Il duale del problema duale coincide con il problema dato

Dimostrazione

Riportando il problema duale in forma canonica si ha:

$$\begin{aligned} -\max \quad & -w = -b^T u \\ \text{s.t.} \quad & -A^T u \leq -c \\ & u \geq 0 \end{aligned}$$

il cui duale è:

$$\begin{aligned} -\min \quad & -t = -c^T y \\ \text{s.t.} \quad & -(A^T)^T y \geq -b \\ & y \geq 0 \end{aligned}$$

che può essere riscritto come:

$$\begin{aligned} \max \quad & t = c^T y \\ \text{s.t.} \quad & Ay \leq b \\ & y \geq 0 \end{aligned}$$

che è equivalente al problema dato



2. Se un problema lineare e il suo duale hanno rispettivamente soluzioni ammissibili x^0 e u^0 , allora vale:

$$c^T x^0 \leq b^T u^0$$

e inoltre entrambi hanno soluzioni ottimali

Dimostrazione

Per l'ammissibilità di x^0 e u^0 si ha:

$$\begin{aligned} Ax^0 \leq b, \quad u^0 \geq 0 &\Rightarrow (Ax^0)^T u^0 \leq b^T u^0 \\ A^T u^0 \geq c, \quad x^0 \geq 0 &\Rightarrow (A^T u^0)^T x^0 \geq c^T x^0 \end{aligned}$$

Osservando che $(Ax^0)^T u^0 = (A^T u^0)^T x^0$ si ha la tesi



3. Se un problema lineare e il suo duale hanno rispettivamente soluzioni ammissibili x^* e u^* per cui vale $c^T x^* = b^T u^*$, allora x^* e u^* sono soluzioni ottimali rispettivamente del primale e del duale

Dimostrazione

Dalla proprietà 2 si ha:

$$\begin{aligned} c^T x &\leq b^T u^* = c^T x^* \Rightarrow x^* \text{ è ottimale} \\ b^T u &\geq c^T x^* = b^T u^* \Rightarrow u^* \text{ è ottimale} \end{aligned}$$



4. I teorema della dualità

Uno dei due problemi ha soluzioni ottimali se e solo se ne ha anche l'altro

In questo caso $\max z = \min w$

Dimostrazione

Dati un problema e il suo duale in forma canonica:

$$\begin{array}{ll} \max & z = c^T x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array} \quad \begin{array}{ll} -\max & -w = (-b)^T u \\ \text{s.t.} & (-A)^T u \leq (-c) \\ & u \geq 0 \end{array}$$

le corrispondenti tabelle iniziali sono:

$$\begin{array}{c|c|c} & x^T & \\ \hline u & -A & b \\ \hline z & c^T & 0 \end{array} \quad \begin{array}{c|c|c} & u^T & \\ \hline x & A^T & -c \\ \hline -w & -b^T & 0 \end{array}$$

Le due tabelle sono una la trasposta cambiata di segno dell'altra; applicando il metodo del simplesso alla prima tabella, si supponga di far cardine su $-(A)_{ji}$, allora nella seconda tabella si fa cardine su $(A^T)_{ij}$, così le tabelle successive sono ancora una la trasposta cambiata di segno dell'altra, fino alla tabella ottimale di uno dei due problemi

Allora anche la sua trasposta cambiata di segno è ottimale

Dalla tabella ottimale del problema primale si ha:

$$\max \quad z = \gamma_0$$

e dalla tabella ottimale del problema duale si ha:

$$-\max \quad -w = -\gamma_0$$

cioè:

$$\min \quad w = \gamma_0$$

per cui $\max \quad z = \min \quad w$



5. Se uno dei due problemi ha soluzione illimitata allora l'altro non ha soluzioni ammissibili

Dimostrazione

Se l'altro problema avesse una soluzione ammissibile il valore della funzione obiettivo in quel punto sarebbe un limitante per la funzione obiettivo del primo problema



6. Se uno dei due problemi non ha soluzioni ammissibili allora l'altro o ha soluzione illimitata o non ha soluzioni ammissibili

Dimostrazione

Se l'altro problema avesse soluzioni ottimali le avrebbe anche il primo problema



7. Il teorema della dualità

Due soluzioni ammissibili x^* e u^* sono soluzioni ottimali se e solo se valgono le relazioni:

$$\begin{aligned}(b - Ax^*)^T u^* &= 0 \\ (A^T u^* - c)^T x^* &= 0\end{aligned}$$

Dimostrazione

Se valgono le relazioni:

$$\begin{aligned}(b - Ax^*)^T u^* &= 0 \\ (A^T u^* - c)^T x^* &= 0\end{aligned}$$

sommando membro a membro si ottiene:

$$b^T u^* - (Ax^*)^T u^* + (A^T u^*)^T x^* - c^T x^* = 0$$

e ricordando che $(Ax^*)^T u^* = (A^T u^*)^T x^*$ si ha:

$$b^T u^* = c^T x^*$$

che per la proprietà 3 fornisce l'ottimalità di x^* e u^*

Viceversa se x^* e u^* sono ottimali, per il I teorema della dualità, si ha:

$$b^T u^* = c^T x^*$$

Aggiungendo e togliendo la quantità $u^{*T} Ax^*$ e riordinando si ha:

$$(b - Ax^*)^T u^* + (A^T u^* - c)^T x^* = 0$$

e per i vincoli dei due problemi si ricavano le relazioni cercate



- Le relazioni del II teorema della dualità vengono chiamate anche condizioni di complementarità
- Un vincolo di uno dei due problemi può essere soddisfatto come uguaglianza e la corrispondente variabile dell'altro problema può essere nulla
- La tabella ottimale di un problema fornisce la soluzione del problema duale:

$$w^* = z^*$$

$$u_i = 0 \quad \text{se la variabile primale } u_i \text{ è fuori base}$$

$$u_i = -\gamma_i \quad \text{se la variabile primale } u_i \text{ è in base}$$

2.3 Interpretazione economica del problema duale

Problema di produzione

$$\begin{array}{ll} \max & z = c^T x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array}$$

b vettore delle risorse

x vettore dei beni prodotti

A matrice tecnologica (unità di risorsa per unità di bene prodotto)

c vettore dei valori unitari dei beni prodotti

z valore dei beni prodotti

vincoli non si possono utilizzare più risorse di quelle disponibili
quantità di beni prodotti non negative

$$[A] = \frac{\text{unità di risorsa}}{\text{unità di bene}}$$

$$[c] = \frac{\text{denaro}}{\text{unità di bene}}$$

e dalla relazione $[A][u] = [c]$ si ricava:

$$[u] = \frac{\text{denaro}}{\text{unità di risorsa}}$$

Problema duale

- u costo-ombra delle risorse
(valore di una ulteriore unità di risorsa in quel processo produttivo)
- w costo-ombra globale delle risorse o valore-ombra del processo produttivo
- vincoli costi-ombra unitari non inferiori ai valori dei beni prodotti
costi-ombra non negativi

Le variabili duali non rappresentano il valore unitario delle risorse perchè è un dato e non una variabile, inoltre alcune risorse risulterebbero avere valore nullo

Il termine costo-ombra è dato dal fatto che non è legato al valore effettivo della risorsa, ma a quello che ha in quello specifico processo produttivo (matrice tecnologica e risorse)

Se una risorsa non viene completamente utilizzata il costo-ombra è nullo, altrimenti può essere positivo o nullo

Un bene non viene prodotto se il costo-ombra è superiore al suo valore, altrimenti può essere prodotto o meno

Se una risorsa non è completamente utilizzata non vi è nessun vantaggio ad averne una maggiore disponibilità, altrimenti il valore dei beni prodotti può essere incrementato disponendo di una ulteriore unità di risorsa (variabile duale non nulla) o può restare invariato (variabile duale nulla)

Se il costo-ombra delle risorse necessarie a produrre un bene è superiore al suo valore è svantaggioso produrlo, altrimenti può essere svantaggioso produrlo (variabile primale nulla) o può non esserlo (variabile primale non nulla)

Ulteriore interpretazione

Una ditta vuole acquistare tutte le risorse e deve determinare il valore ottimale dei prezzi u da offrire per ogni unità di risorsa

La ditta cerca di minimizzare la spesa di acquisto

$$\min b^T u$$

Il valore delle risorse necessarie a produrre una unità di bene deve essere non inferiore al suo valore unitario

$$A^T u \geq c$$

I prezzi devono essere non negativi

$$u \geq 0$$

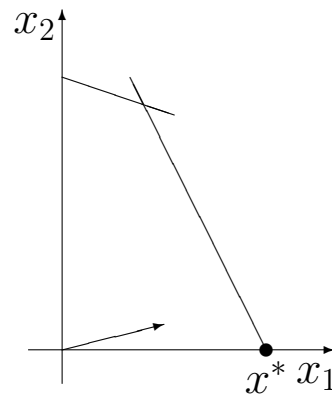
Esempio 2.1 (Modello di produzione lineare)

Due processi produttivi utilizzano due risorse, R_1 e R_2 , per produrre due beni, B_1 e B_2 ; una unità di B_1 richiede 1 unità di R_1 e 2 unità di R_2 , mentre una unità di B_2 richiede 3 unità di R_1 e 1 unità di R_2 . Si supponga di disporre di 12 unità di R_1 e 6 unità di R_2 e che una unità di B_1 abbia valore 4 e una unità di B_2 abbia valore 1

Problema primale

$$\begin{aligned} \max \quad & z = 4x_1 + x_2 \\ \text{s.t.} \quad & x_1 + 3x_2 \leq 12 \\ & 2x_1 + x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

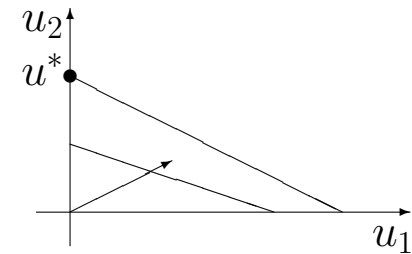
$x^* = (3, 0); z^* = 12$



Problema duale

$$\begin{aligned} \min \quad & w = 12u_1 + 6u_2 \\ \text{s.t.} \quad & u_1 + 2u_2 \geq 4 \\ & 3u_1 + u_2 \geq 1 \\ & u_1, u_2 \geq 0 \end{aligned}$$

$u^* = (0, 2); w^* = 12$



$u_1^* = 0$ in accordo col fatto che la risorsa R_1 non è completamente utilizzata

$u = (1, 0)$ porterebbe allo stesso valore di w , ma in questo caso si potrebbero vendere solo 9 unità di R_1 , ottenendo 9, e utilizzare le risorse trattenute per produrre tre unità di B_1 con un valore 12, migliorando il risultato $\diamond$

Esempio 2.2 (Significato delle variabili duali)

Sia $u_i^* \neq 0$, cioè $\sum_{j=1, \dots, n} a_{ij} x_j^* = b_i$

Se $b_i^0 = b_i + 1$ si ha:

$$w^{0*} = \sum_{h \neq j} b_h u_h^* + b_i^0 u_i^* = w^* + u_i^*$$

e quindi $z^{0*} = z^* + u_i^*$

**Esempio 2.3 (Significato dei vincoli duali)**

Sia $\sum_{j=1, \dots, n} a_{ij} x_j^* = b_i$, $u_i^* \neq 0$ e $u_k^* = 0$, $k \neq i$

I vincoli duali si riducono a $a_{ij} u_i^* \geq c_j$ da cui si ricava:

$$u_i^* \geq \frac{c_j}{a_{ij}}, \quad j = 1, \dots, n$$

dove $\frac{c_j}{a_{ij}}$ rappresenta il valore unitario fornito dalla risorsa i se è utilizzata per produrre il bene j

La soluzione è $u_i^* = \max_j \left\{ \frac{c_j}{a_{ij}} \right\}$ cioè il costo-ombra della risorsa i è uguale al massimo valore unitario che può essere fornito dalla risorsa



Esempio 2.4 (Paradosso dell'anellino) *Un artigiano stravagante fabbrica anellini di plastica con diamanti autentici*

Dispone di un anellino del costo di 1 dollaro e di due diamanti del costo di 900 dollari

Un anellino con diamante viene venduto per 1000 dollari

$$\begin{array}{lll}
 \max & z = 99x & \text{profitto} \\
 \text{s.t.} & x \leq 1 & \text{vincolo sugli anellini} \\
 & x \leq 2 & \text{vincolo sui diamanti} \\
 & x \geq 0 &
 \end{array}$$

	x			u_a	
u_a	-1^*	1	x	-1	1
u_d	-1	2	u_d	1	1
z	99	0	z	-99	99

$$x^* = 1, z^* = 99$$

si produce un anellino con un profitto di 99 dollari

$$u_a^* = 99, u_d^* = 0, w^* = 99$$

l'artigiano è disposto a pagare fino a 99 dollari in più per un altro anellino, ma non è disposto a pagare alcunchè per un altro diamante



APPENDICE - Formulazione del problema duale col problema primale in forma non canonica

Problema primale di massimo

coefficienti della funzione obiettivo

termini noti dei vincoli

colonna j dei coefficientiriga i dei coefficienti

$$x_j \geq 0$$

$$x_j \leq 0$$

 x_j libera

$$\sum_{j=1,\dots,n} a_{ij}x_j \leq b_i$$

$$\sum_{j=1,\dots,n} a_{ij}x_j \geq b_i$$

$$\sum_{j=1,\dots,n} a_{ij}x_j = b_i$$

Problema duale di minimo

termini noti dei vincoli

coefficienti della funzione obiettivo

riga j dei coefficienticolonna i dei coefficienti

$$\sum_{i=1,\dots,m} a_{ij}u_i \geq c_j$$

$$\sum_{i=1,\dots,m} a_{ij}u_i \leq c_j$$

$$\sum_{i=1,\dots,m} a_{ij}u_i = c_j$$

$$u_i \geq 0$$

$$u_i \leq 0$$

 u_i libera

Problema primale di minimo

coefficienti della funzione obiettivo

termini noti dei vincoli

colonna j dei coefficientiriga i dei coefficienti

$$x_j \geq 0$$

$$x_j \leq 0$$

 x_j libera

$$\sum_{j=1,\dots,n} a_{ij}x_j \leq b_i$$

$$\sum_{j=1,\dots,n} a_{ij}x_j \geq b_i$$

$$\sum_{j=1,\dots,n} a_{ij}x_j = b_i$$

Problema duale di massimo

termini noti dei vincoli

coefficienti della funzione obiettivo

riga j dei coefficienticolonna i dei coefficienti

$$\sum_{i=1,\dots,m} a_{ij}u_i \leq c_j$$

$$\sum_{i=1,\dots,m} a_{ij}u_i \geq c_j$$

$$\sum_{i=1,\dots,m} a_{ij}u_i = c_j$$

$$u_i \leq 0$$

$$u_i \geq 0$$

 u_i libera

3 Programmazione lineare a numeri interi

3.1 Problemi lineari interi

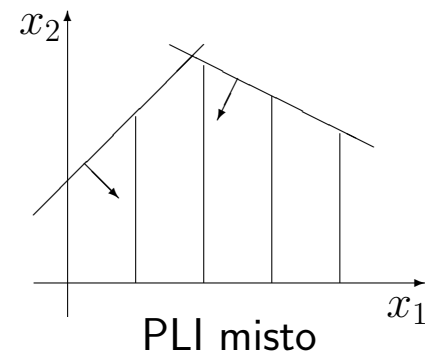
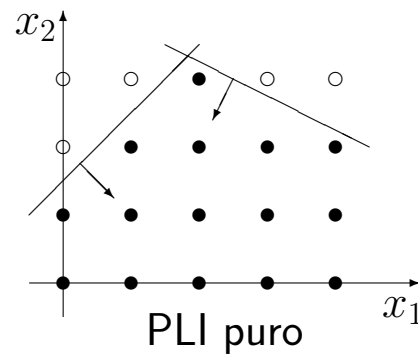
Dato il problema lineare ordinario (PLO):

$$\begin{aligned} \max \quad & z = c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

aggiungendo la condizione di integrità:

$$x_i \in \mathbb{N}, i \in I$$

si ottiene un problema lineare a numeri interi (PLI) ad esso associato



Proprietà generali

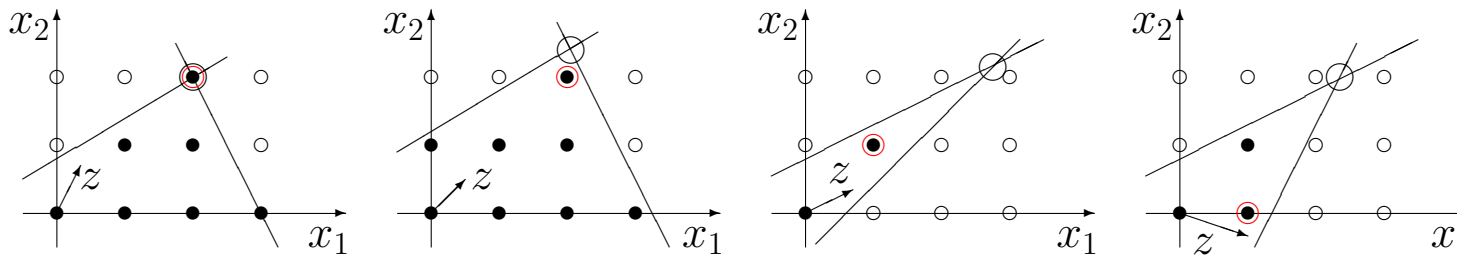
- $S'_a \subseteq S_a$ per cui

$$S_a = \emptyset \Rightarrow S'_a = \emptyset$$

•

$$z'(x) = z(x), x \in S'_a \Rightarrow \sup z' \leq \sup z$$

La soluzione di un PLI può risultare complessa poichè non esiste alcuna relazione con l'insieme delle soluzioni ottimali del PLO



Classi di metodi risolutivi:

Metodi approssimati

- Arrotondamento o troncamento della soluzione del PLO
Non garantisce nè l'ottimalità, nè l'ammissibilità
- Punto ammissibile più vicino alla soluzione del PLO
Può essere complesso determinarlo e non garantisce l'ottimalità

Metodi esatti

- Metodi branch and bound (separazione e valutazione)
- Metodi cutting plane (piani di taglio o iperpiani secanti)

3.2 Metodi branch and bound

Si suddivide il problema assegnato in sottoproblemi più semplici da risolvere e si utilizzano le soluzioni di questi per ricavare la soluzione del problema dato

I concetti fondamentali sono rilassamento, separazione e eliminazione

Notazioni

Dato un problema P qualsiasi siano:

- $S_a(P)$ l'insieme delle soluzioni ammissibili
- $z(P)$ la funzione obiettivo da massimizzare
- $x^*(P)$ la soluzione ottimale
- $z^*(P)$ il valore ottimale
- $z_s(P)$ il valore approssimato (stima o limitazione)

3.2.1 Rilassamento

Un problema P' si dice rilassamento di un problema P se:

$$\begin{aligned} S_a(P) &\subset S_a(P') \\ z(P')(x) &\geq z(P)(x) \quad x \in S_a(P) \end{aligned}$$

Proprietà del rilassamento

- $S_a(P') = \emptyset \Rightarrow S_a(P) = \emptyset$
- $z^*(P') \geq z^*(P)$
- $x^*(P') \in S_a(P), z(P')(x^*(P')) = z(P)(x^*(P')) \Rightarrow (x^*(P'), z^*(P')) = (x^*(P), z^*(P))$

3.2.2 Separazione

Un problema P si dice separato nei sottoproblemi $P_1, \dots, P_n$ se:

- $S_a(P) = \bigcup_{i=1, \dots, n} S_a(P_i)$
 $S_a(P_i) \cap S_a(P_j) = \emptyset \quad i \neq j$
- $z(P_i) = z(P) \quad i = 1, \dots, n$
- È opportuno non avere una partizione troppo fine per non dover risolvere troppi sottoproblemi

Proprietà della separazione

- $z^*(P) = \max_{i=1, \dots, n} z^*(P_i) = z^*(P_{i^*}) \Rightarrow x^*(P) = x^*(P_{i^*})$

3.2.3 Eliminazione

Un problema P si dice eliminato se un suo rilassamento P' verifica una delle tre condizioni seguenti:

- $E1 \quad S_a(P') = \emptyset$
 - $E2 \quad z^*(P') \leq z^* \quad \text{dove } z^* \text{ è il valore della soluzione ottimale corrente}$
 - $E3 \quad x^*(P') \in S_a(P)$
- Se si utilizza una stima $z_s(P') \geq z^*(P')$ non è possibile verificare la condizione $E2$ in quanto da $z_s(P') \leq z_s^*$, dove z_s^* è la migliore stima corrente, non discende $z^*(P') \leq z^*$, poichè $z_s^* \geq z^*$. D'altra parte $z_s(P') \leq z^*$ è una condizione di eliminazione
 - È conveniente che il valore della stima $z_s(P')$ sia il più vicino possibile al valore $z^*(P')$, purchè questo non comporti una eccessiva difficoltà e/o troppo tempo
 - È importante che le condizioni di eliminazione siano assegnate per il rilassamento P' invece che per P , in quanto nel corso dell'algoritmo si risolvono i problemi P'

Proprietà dell'eliminazione

- $E1 \Rightarrow S_a(P) = \emptyset$
- $E2 \Rightarrow z^*(P) \leq z^*$
- $E3 \text{ e } z(P')(x^*(P')) = z(P)(x^*(P')) \Rightarrow x^*(P') = x^*(P)$

3.2.4 Algoritmo branch and bound (Land e Doig, 1960 - Little, 1963)

- a) inizializzare la lista dei problemi da risolvere col problema assegnato;
- b) se nella lista c'è un problema P da risolvere estrarlo, considerare un rilassamento P' e andare a c); altrimenti andare ad h);
- c) risolvere il problema P' (oppure determinare $z_s(P')$);
- d) se $S_a(P') = \emptyset$ allora $S_a(P) = \emptyset$, eliminare il problema P per il criterio $E1$ e tornare a b);
- e) se $z^*(P') \leq z^*$ (oppure $z_s(P') \leq z^*$) allora $z^*(P) \leq z^*$, eliminare il problema P per il criterio $E2$ e tornare a b);
- f) se $x^*(P') \in S_a(P)$ e $z(P')(x^*(P')) = z(P)(x^*(P'))$ allora $x^*(P') = x^*(P)$, eliminare P per il criterio $E3$ e tornare a b); se $z^*(P) > z^*$ allora porre $(x^*(P), z^*(P))$ nuova soluzione del problema, porre $z^* = z^*(P)$ e tornare a b);
- g) se P non è stato eliminato decidere se abbandonare il problema;
se si abbandona separare P , aggiornare la lista dei problemi da risolvere e tornare a b);
altrimenti considerare un nuovo rilassamento P' e tornare a c);
- h) se è stata trovata una soluzione $(x^*(P), z^*(P))$ questa è la soluzione ottimale;
altrimenti il problema non ammette soluzioni;

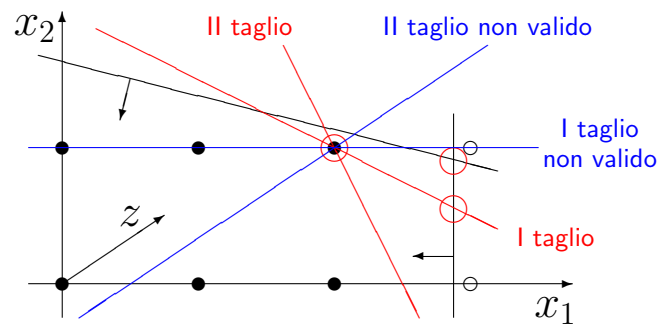
- Quello presentato è l'algoritmo base; di volta in volta è necessario precisare i criteri di rilassamento, di valutazione e di separazione

Gestione della lista

- Esistono diverse strategie di gestione della lista dei problemi da risolvere
- Per semplificare la gestione della memoria si può utilizzare la strategia depth-first o LIFO
- Le limitazioni $z_s(P')$ permettono di utilizzare la strategia highest-first; per il criterio E2, si possono eliminare alcuni dei problemi presenti nella lista
- La lista può essere gestita in modo casuale, o pseudocasuale nei casi di informazioni particolarmente imprecise
- Si può utilizzare un albero in cui ogni foglia rappresenta un sottoproblema della lista con associata la sua limitazione

3.3 Metodi cutting plane

Si determina una soluzione del problema ottenuto eliminando i vincoli di integrità (problema rilassato); se la soluzione trovata non è intera, si introduce un ulteriore vincolo detto *iperpiano secante* o *taglio* che la renda non ammissibile senza eliminare nessuna soluzione intera, ripetendo il procedimento finché non si determina una soluzione ottimale intera



Si noti che qualunque coppia di tagli (I, II) permette di determinare la soluzione ottimale

I metodi si differenziano per il modo in cui si generano gli iperpiani secanti

È importante che il criterio di generazione degli iperpiani secanti permetta di trovare la soluzione ottimale in un numero finito di passi

Notazioni

y_i	$i = 1, \dots, n$	variabili in base
y_{n+j}	$j = 1, \dots, m$	variabili fuori base

3.3.1 Algoritmo elementare (Dantzig, 1959)

La soluzione ottimale del problema lineare rilassato è caratterizzata come intersezione degli iperpiani generatori $y_i = 0, i = 1, \dots, n$; pertanto, se le variabili fuori base non sono intere, qualsiasi soluzione intera ha qualche y_i non nullo e quindi maggiore o uguale a 1

Il vincolo:

$$y_1 + \dots + y_n \geq 1$$

costituisce un iperpiano secante valido

Questo metodo è efficiente ma non garantisce la convergenza

3.3.2 Algoritmo di Gomory (1958)

La tabella iniziale deve essere a coefficienti interi

Se nella tabella ottimale del problema rilassato un termine β_k non è intero, si considera il vincolo:

$$y_{n+k} = \alpha_{k1}y_1 + \dots + \alpha_{ki}y_i + \dots + \alpha_{kn}y_n + \beta_k \geq 0$$

e si genera il vincolo:

$$\tilde{y}_{n+k} = f_{k1}y_1 + \dots + f_{ki}y_i + \dots + f_{kn}y_n - f_k \geq 0$$

dove $f_{ki}, i = 1, \dots, n$ è la mantissa di $-\alpha_{ki}$ e f_k è la mantissa di β_k e sono sempre non negative

Teorema 3.1 *Il vincolo $\tilde{y}_{n+k} = f_{k1}y_1 + \dots + f_{ki}y_i + \dots + f_{kn}y_n - f_k \geq 0$ costituisce un iperpiano secante valido, cioè non è soddisfatto dalla soluzione ottimale corrente e non esclude nessuna altra soluzione intera*

Dimostrazione

Nel caso della soluzione ottimale corrente, essendo $y_i = 0, i = 1, \dots, n$ si ha:

$$\tilde{y}_{n+k} = -f_k$$

che non verifica il vincolo essendo negativo

Per ogni soluzione ammissibile intera $y_i, i = 1, \dots, n, y_{n+j}, j = 1, \dots, m$ è intera la quantità:

$$y_{n+k} + \tilde{y}_{n+k} = -q_{k1}y_1 - \dots - q_{ki}y_i - \dots - q_{kn}y_n + q_k$$

dove $q_{ki}, i = 1, \dots, n$ è la parte intera di $-\alpha_{ki}$ e q_k è la parte intera di β_k

$$\begin{array}{l} -\alpha_{ki} = q_{ki} + f_{ki} \Rightarrow -q_{ki} = \alpha_{ki} + f_{ki} \\ \beta_k = q_k + f_k \Rightarrow q_k = \beta_k - f_k \end{array}$$

Dovendo essere y_{n+k} intera per ipotesi, risulta intera anche $\tilde{y}_{n+k}$; allora essendo:

$$\tilde{y}_{n+k} + f_k = f_{k1}y_1 + \dots + f_{ki}y_i + \dots + f_{kn}y_n \geq 0$$

ed essendo $0 < f_k < 1$ si ha:

$$\tilde{y}_{n+k} \geq 0$$



- Dopo aver introdotto il nuovo vincolo si cerca la nuova soluzione ottimale utilizzando il simplesso duale, essendo la tabella ottimale duale ammissibile
- Esiste un secondo algoritmo di Gomory che permette di risolvere i problemi di programmazione intera mista generando differentemente l'iperpiano secante
- Il metodo converge perchè i coefficienti degli iperpiani introdotti sono dati dal rapporto dei divisori dei coefficienti della tabella iniziale, che generano un gruppo ciclico finito

4 Modelli lineari a numeri interi

4.1 Problema dello zaino

n oggetti di peso p_i e valore c_i

Si ha a disposizione uno zaino di capienza assegnata P

Massimizzare il valore trasportato

$$\begin{aligned} \max \quad & \sum_{i=1, \dots, n} c_i x_i \\ \text{s.t.} \quad & \sum_{i=1, \dots, n} p_i x_i \leq P \\ & x_i \in \{0, 1\} \quad i = 1, \dots, n \end{aligned}$$

4.2 Problema del trasporto

n centri di produzione con capacità produttiva di p_i unità e m centri di distribuzione con richiesta di r_j unità
 c_{ij} è il costo unitario di trasporto dal centro di produzione i al centro di distribuzione j

Minimizzare il costo di trasporto

$$\begin{aligned}
 \min \quad & \sum_{i=1, \dots, n} \sum_{j=1, \dots, m} c_{ij} x_{ij} \\
 \text{s.t.} \quad & \sum_{i=1, \dots, n} x_{ij} = r_j & j = 1, \dots, m \\
 & \sum_{j=1, \dots, m} x_{ij} \leq p_i & i = 1, \dots, n \\
 & x_{ij} \in N & i = 1, \dots, n; j = 1, \dots, m
 \end{aligned}$$

Condizione di capacità produttiva

$$\sum_{j=1, \dots, m} r_j \leq \sum_{i=1, \dots, n} p_i$$

4.3 Problema del magazzino

n periodi consecutivi per i quali sono noti la domanda d_i , la capacità produttiva massima C_i e il costo di produzione unitario c_i

Si può utilizzare un magazzino di capacità H_i , dove H_0 è la disponibilità iniziale e il costo di magazzinaggio unitario è h_i

Minimizzare il costo di produzione

$$\begin{aligned} \min \quad & \sum_{i=1, \dots, n} (c_i x_i + h_i y_i) \\ \text{s.t.} \quad & x_i + y_{i-1} = d_i + y_i \quad i = 1, \dots, n \\ & 0 \leq x_i \leq C_i \quad i = 1, \dots, n \\ & 0 \leq y_i \leq H_i \quad i = 1, \dots, n \end{aligned}$$

dove $y_0 = H_0$

Condizione di disponibilità

$$\begin{aligned} \sum_{j=1, \dots, i} d_j &\leq H_0 + \sum_{j=1, \dots, i} C_j, \quad i = 1, \dots, n \\ d_i &\leq C_i + H_{i-1}, \quad i = 1, \dots, n \end{aligned}$$

4.4 Problema dell'assegnazione

n macchine a cui assegnare un operaio tra n

c_{ij} è il costo di assegnazione dell'operaio i alla macchina j

Minimizzare i costi di assegnazione

$$\begin{aligned} \min \quad & \sum_{i=1, \dots, n} \sum_{j=1, \dots, n} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i=1, \dots, n} x_{ij} = 1 & j = 1, \dots, n \\ & \sum_{j=1, \dots, n} x_{ij} = 1 & i = 1, \dots, n \\ & x_{ij} \in \{0, 1\} & i, j = 1, \dots, n \end{aligned}$$

4.5 Problema del commesso viaggiatore

n città da visitare senza mai ripassare da alcuna città

c_{ij} è il costo di un tragitto, non necessariamente simmetrico

Minimizzare il costo (circuito hamiltoniano di costo minimo)

$$\begin{array}{ll}
 \min & \sum_{i=1, \dots, n} \sum_{j=1, \dots, n} c_{ij} x_{ij} \\
 \text{s.t.} & \sum_{i=1, \dots, n} x_{ij} = 1 & j = 1, \dots, n \\
 & \sum_{j=1, \dots, n} x_{ij} = 1 & i = 1, \dots, n \\
 & y_i - y_j + (n+1)x_{ij} \leq n & i, j = 1, \dots, n \\
 & x_{ij} \in \{0, 1\} & i, j = 1, \dots, n
 \end{array}$$

5 Applicazioni del metodo Branch and Bound

5.1 Problema dello zaino (Problema del massimo)

<i>oggetto</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
<i>peso</i>	22	18	13	7
<i>valore</i>	43	32	26	12
<i>peso massimo trasportabile</i> = 30				

Rapporti valore/peso degli oggetti:

<i>oggetto</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
<i>valore/peso</i>	1,955	1,777	2,000	1,714

Ordine decrescente:

<i>oggetto</i>	<i>C</i>	<i>A</i>	<i>B</i>	<i>D</i>
<i>peso</i>	13	22	18	7
<i>valore</i>	26	43	32	12
<i>variabile associata</i>	x_1	x_2	x_3	x_4
<i>peso massimo trasportabile</i> = 30				

Problema lineare intero a variabili 0-1:

$$\begin{aligned} \max \quad & 26x_1 + 43x_2 + 32x_3 + 12x_4 \\ \text{s.t.} \quad & 13x_1 + 22x_2 + 18x_3 + 7x_4 \leq 30 \end{aligned}$$

Bound di Dantzig

Prendere gli oggetti secondo l'ordine dato, senza superare il peso massimo trasportabile e una frazione (eventualmente nulla) dell'oggetto critico

$$L = \lfloor 26 + (\frac{17}{22})43 \rfloor = 59$$

Portare C ($x_1 = 1$); non portare C ($x_1 = 0$)

$$\begin{aligned} (1) \quad & \max \quad 26 + 43x_2 + 32x_3 + 12x_4 \\ & \text{s.t.} \quad 22x_2 + 18x_3 + 7x_4 \leq 17 \quad L(1) = \lfloor 26 + (\frac{17}{22})43 \rfloor = 59 \\ (0) \quad & \max \quad 43x_2 + 32x_3 + 12x_4 \\ & \text{s.t.} \quad 22x_2 + 18x_3 + 7x_4 \leq 30 \quad L(0) = \lfloor 43 + (\frac{8}{18})32 \rfloor = 57 \end{aligned}$$

Portare C e A ($x_1 = 1, x_2 = 1$); portare C e non portare A ($x_1 = 1, x_2 = 0$)

$$\begin{aligned} (11) \quad & \max \quad 26 + 43 + 32x_3 + 12x_4 \\ & \text{s.t.} \quad 18x_3 + 7x_4 \leq -5 \quad S_a = \emptyset \\ (10) \quad & \max \quad 26 + 32x_3 + 12x_4 \\ & \text{s.t.} \quad 18x_3 + 7x_4 \leq 17 \quad L(10) = \lfloor 26 + (\frac{17}{18})32 \rfloor = 56 \end{aligned}$$

Non portare C e portare A ($x_1 = 0, x_2 = 1$); non portare C e A ($x_1 = 0, x_2 = 0$)

$$\begin{aligned}
 (01) \quad & \max \quad 43 + 32x_3 + 12x_4 \\
 & s.t. \quad 18x_3 + 7x_4 \leq 8 \qquad L(01) = \lfloor 43 + (\frac{8}{18})32 \rfloor = 57 \\
 (00) \quad & \max \quad 32x_3 + 12x_4 \\
 & s.t. \quad 18x_3 + 7x_4 \leq 30 \qquad L(00) = \lfloor 32 + 12 \rfloor = \underline{44}
 \end{aligned}$$

La soluzione del problema (00) è ammissibile con valore uguale alla limitazione

Non portare C e portare A e B ($x_1 = 0, x_2 = 1, x_3 = 1$); non portare C e B e portare A ($x_1 = 0, x_2 = 1, x_3 = 0$)

$$\begin{aligned}
 (011) \quad & \max \quad 43 + 32 + 12x_4 \\
 & s.t. \quad 7x_4 \leq -10 \qquad S_a = \emptyset \\
 (010) \quad & \max \quad 43 + 12x_4 \\
 & s.t. \quad 7x_4 \leq 8 \qquad L(010) = \lfloor 43 + 12 \rfloor = \underline{55}
 \end{aligned}$$

La soluzione del problema (010) è ammissibile con valore uguale alla limitazione

Portare C e B e non portare A ($x_1 = 1, x_2 = 0, x_3 = 1$); portare C e non portare A e B ($x_1 = 1, x_2 = 0, x_3 = 0$)

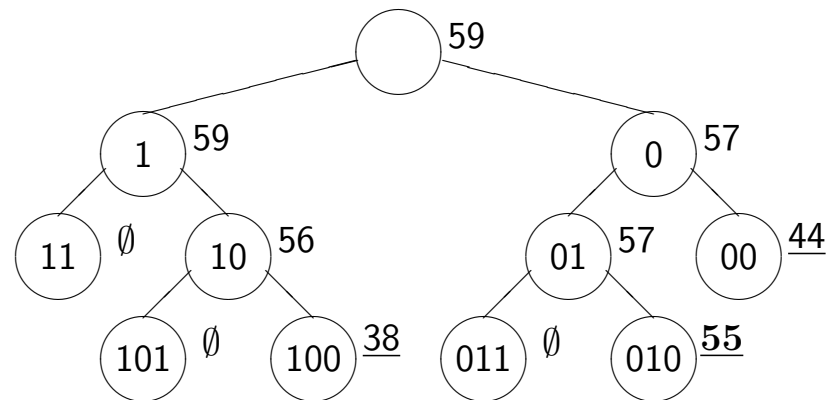
$$\begin{aligned} (101) \quad & \max \quad 26 + 32 + 12x_4 \\ & \text{s.t.} \quad 7x_4 \leq -1 \qquad S_a = \emptyset \end{aligned}$$

$$\begin{aligned} (100) \quad & \max \quad 26 + 12x_4 \\ & \text{s.t.} \quad 7x_4 \leq 17 \qquad L(100) = \lfloor 26 + 12 \rfloor = \underline{38} \end{aligned}$$

La soluzione del problema (100) è ammissibile con valore uguale alla limitazione

La soluzione del problema (010) è ottimale, avendo valore migliore di tutte le limitazioni correnti

Si portano gli oggetti A e D con valore complessivo 55 e peso complessivo 29

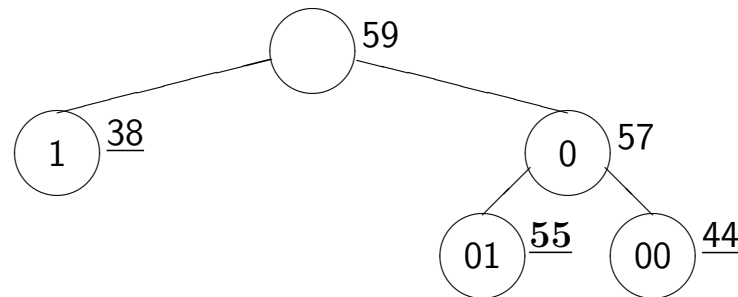


Miglioramento del calcolo della limitazione

Gli oggetti che superano il peso residuo trasportabile non potranno essere trasportati

$$\begin{array}{ll}
 \max & 26x_1 + 43x_2 + 32x_3 + 12x_4 \\
 \text{s.t.} & 13x_1 + 22x_2 + 18x_3 + 7x_4 \leq 30 \quad L = \lfloor 26 + (\frac{17}{22})43 \rfloor = 59 \\
 (1) \quad \max & 26 + 43x_2 + 32x_3 + 12x_4 \\
 \text{s.t.} & 22x_2 + 18x_3 + 7x_4 \leq 17 \quad L(1) = \lfloor 26 + 12 \rfloor = \underline{38} \\
 (0) \quad \max & 43x_2 + 32x_3 + 12x_4 \\
 \text{s.t.} & 22x_2 + 18x_3 + 7x_4 \leq 30 \quad L(0) = \lfloor 43 + (\frac{8}{18})32 \rfloor = 57 \\
 (01) \quad \max & 43 + 32x_3 + 12x_4 \\
 \text{s.t.} & 18x_3 + 7x_4 \leq 8 \quad L(01) = \lfloor 43 + 12 \rfloor = \underline{55} \\
 (00) \quad \max & 32x_3 + 12x_4 \\
 \text{s.t.} & 18x_3 + 7x_4 \leq 30 \quad L(00) = \lfloor 32 + 12 \rfloor = \underline{44}
 \end{array}$$

La soluzione del problema (01) è ottimale e coincide con la soluzione precedente



- Nel caso di più soluzioni ottimali con pesi differenti, il metodo le considera equivalenti

- Si consideri il seguente problema dello zaino:

<i>oggetto</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>valore</i>	20	15	5	3	3
<i>peso</i>	5	5	3	2	2
peso massimo trasportabile = 9					

Risolvendolo con l'algoritmo Branch and Bound, utilizzando il bound di Dantzig e le tecniche di accelerazione e osservando che gli oggetti sono già ordinati, si ha:

$$L = \lfloor 20 + (\frac{4}{5})15 \rfloor = 32$$

$$L(1) = \lfloor 20 + 5 + (\frac{1}{2})3 \rfloor = 26$$

$$L(0) = \lfloor 15 + 5 + (\frac{1}{2})3 \rfloor = 21$$

$$L(11) = S_a = \emptyset$$

$$L(10) = \lfloor 20 + 5 + (\frac{1}{2})3 \rfloor = 26$$

$$L(101) = \lfloor 20 + 5 \rfloor = \underline{25}$$

$$L(100) = \lfloor 20 + 3 + 3 \rfloor = \underline{\mathbf{26}}$$

Se nel calcolo di $L(1)$ si applica una metodologia greedy si ha $L(1) = \lfloor 20 + 5 \rfloor = 25$ che non è un bound

5.2 Problema dell'assegnazione (problema di minimo)

	a	b	c	d
A	18	18	17	12
B	16	17	16	12
C	17	15	15	11
D	14	13	15	18

Ogni operaio deve essere assegnato ad una macchina $\rightarrow$ sottrarre ad ogni riga il suo minimo:

	a	b	c	d
A	6	6	5	0
B	4	5	4	0
C	6	4	4	0
D	1	0	2	5

Ad ogni macchina deve essere assegnato un operaio $\rightarrow$ sottrarre ad ogni colonna il suo minimo.

La somma dei minimi sottratti costituisce il *costo fisso della matrice* e la limitazione L

La separazione viene fatta scegliendo una assegnazione di costo nullo con il *massimo costo di alternativa*:

	a	b	c	d
A	5	6	3	0^3
B	3	5	2	0^2
C	5	4	2	0^2
D	0^3	0^4	0^2	5

$$L = 12 + 12 + 11 + 13 + 1 + 2 = 51$$

Assegnare l'operaio D alla macchina b ; non assegnarlo

- (*not* $D - b$)

$$L = 51 + 4 = 55$$

- ($D - b$)

Si eliminano la riga di D e la colonna di b e si calcola il nuovo costo fisso:

	a	c	d
A	2	1	0^1
B	0^2	0^0	0^0
C	2	0^0	0^0

$$L = 51 + 3 + 2 = 56$$

Si riprende il problema (*not* $D - b$)

Si attribuisce costo M all'assegnazione $D - b$:

	a	b	c	d
A	5	2	3	0^2
B	3	1	2	0^1
C	5	0^1	2	0^0
D	0^3	M	0^2	5

Assegnare l'operaio D alla macchina a ; non assegnarlo

- (*not* $D - b$)(*not* $D - a$)

$$L = 55 + 3 = 58$$

- (*not* $D - b$)($D - a$)

	b	c	d
A	2	1	0^1
B	1	0^0	0^0
C	0^1	0^0	0^0

$$L = 55 + 2 = 57$$

Si riprende il problema $(D - b)$

Assegnare l'operaio B alla macchina a ; non assegnarlo

- $(D - b)(\text{not } B - a)$

$$L = 56 + 2 = 58$$

- $(D - b)(B - a)$

	c	d
A	1	0^1
C	0^1	0^0

$$L = 56$$

Si riprende il problema $B - a$

Assegnare l'operaio A alla macchina d ; non assegnarlo (*equivalentemente: Assegnare l'operaio C alla macchina c ; non assegnarlo*)

- $D - b)(B - a)(\text{not } A - d)$

$$L = 56 + 1 = 57$$

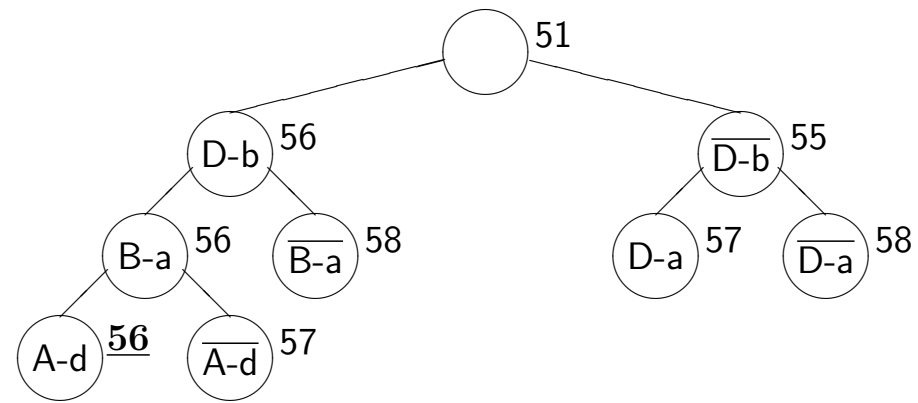
- $(D - b)(B - a)(A - d)$

	c
C	0

$$L = 56$$

La soluzione del problema $(D - b)(B - a)(A - d)$ è ammissibile ed ottimale

La soluzione è data dalle assegnazioni $A/d, B/a, C/c, D/b$, aventi costo rispettivamente 12, 16, 15, 13 e costo complessivo 56



5.3 Problema con variabili intere qualsiasi (PLI misto)

Il metodo branch and bound è utile per un PLI misto, quando non si può usare l'algoritmo di Gomory

Si risolve con l'algoritmo del simplesso il rilassamento ottenuto eliminando il vincolo di integrità, determinando una prima limitazione

Se la variabile intera x_i assume il valore non intero x_i^* si separa il problema aggiungendo i vincoli $x_i \leq \lfloor x_i^* \rfloor$ e $x_i \geq \lceil x_i^* \rceil$

Si risolvono con l'algoritmo del simplesso o con l'algoritmo duale i sottoproblemi ottenuti, determinando le corrispondenti limitazioni, fino ad ottenere una soluzione intera per un sottoproblema; il sottoproblema è eliminato (condizione E3) e si possono eliminare anche tutti i sottoproblemi pendenti che hanno una limitazione peggiore (condizione E2)

Si prosegue finchè la lista dei problemi è vuota; la soluzione trovata è ottimale

- I vincoli aggiunti eliminano soluzioni ammissibili per il problema rilassato, ma non per il PLI assegnato

Esempio 5.1 (PLI misto) Sia dato il seguente PLI misto:

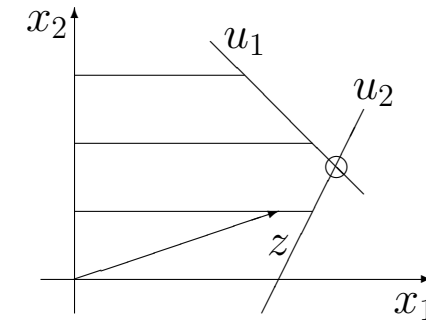
$$\begin{aligned}
 P : \max \quad & 3x_1 + x_2 \\
 \text{s.t.} \quad & x_1 + x_2 \leq \frac{11}{2} \\
 & 2x_1 - x_2 \leq 6 \\
 & x_1, x_2 \geq 0; x_2 \text{ intero}
 \end{aligned}$$

	x_1	x_2	
u_1	-1	-1	$\frac{11}{2}$
u_2	-2*	1	6
z	3	1	0

	u_2	x_2	
u_1	$\frac{1}{2}$	$-\frac{3}{2}$ *	$\frac{5}{2}$
x_1	$-\frac{1}{2}$	$\frac{1}{2}$	3
z	$-\frac{3}{2}$	$\frac{5}{2}$	9

	u_2	u_1	
x_2	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{5}{3}$
x_1	$-\frac{1}{3}$	$-\frac{1}{3}$	$\frac{23}{6}$
z	$-\frac{2}{3}$	$-\frac{5}{3}$	$\frac{79}{6}$

La soluzione $x^* = (\frac{23}{6}, \frac{5}{3})$, $z^* = \frac{79}{6}$ non è ammissibile



$$P1 = \{P \cup x_2 \leq \lfloor \frac{5}{3} \rfloor\} = \{P \cup x_2 \leq 1\}$$

$$P2 = \{P \cup x_2 \geq \lceil \frac{5}{3} \rceil\} = \{P \cup x_2 \geq 2\}$$

Dalla tabella finale si ricava $x_2 = \frac{1}{3}u_2 - \frac{2}{3}u_1 + \frac{5}{3}$:

$$P1 = \{P \cup -\frac{1}{3}u_2 + \frac{2}{3}u_1 - \frac{2}{3} \geq 0\}$$

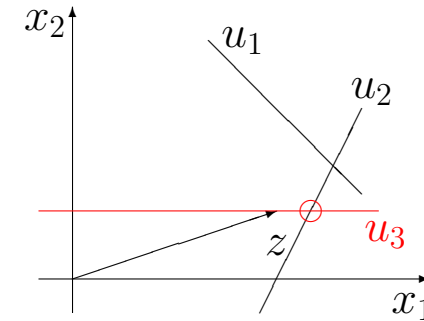
$$P2 = \{P \cup \frac{1}{3}u_2 - \frac{2}{3}u_1 - \frac{1}{3} \geq 0\}$$

Risolvendo P1 con l'algoritmo duale si ottiene:

	u_2	u_1	
x_2	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{5}{3}$
x_1	$-\frac{1}{3}$	$-\frac{1}{3}$	$\frac{23}{6}$
u_3	$-\frac{1}{3}$	$\frac{2}{3}^*$	$-\frac{2}{3}$
z	$-\frac{2}{3}$	$-\frac{5}{3}$	$\frac{79}{6}$

	u_2	u_3	
x_2	0	-1	1
x_1	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{7}{2}$
u_1	$\frac{1}{2}$	$\frac{3}{2}$	1
z	$-\frac{3}{2}$	$-\frac{5}{2}$	$\frac{23}{2}$

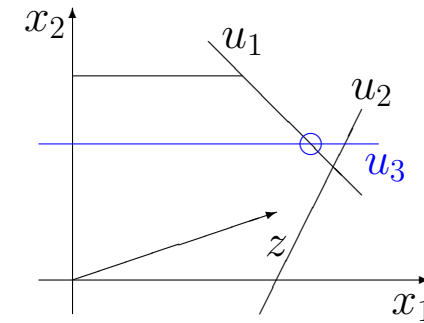
La soluzione $x^* = (\frac{7}{2}, 1)$, $z^* = \frac{23}{2}$ è ammissibile



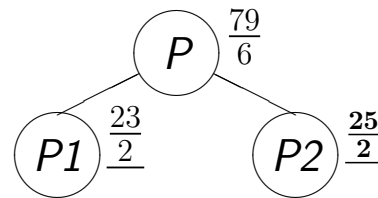
Risolvendo $P2$ con l'algoritmo duale si ottiene:

	u_2	u_1	
x_2	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{5}{3}$
x_1	$-\frac{1}{3}$	$-\frac{1}{3}$	$\frac{23}{6}$
u_3	$\frac{1}{3}^*$	$-\frac{2}{3}$	$-\frac{1}{3}$
z	$-\frac{2}{3}$	$-\frac{5}{3}$	$\frac{79}{6}$

	u_3	u_1	
x_2	1	0	2
x_1	-1	-1	$\frac{7}{2}$
u_2	3	2	1
z	-2	-3	$\frac{25}{2}$



La soluzione $x^* = (\frac{7}{2}, 2)$, $z^* = \frac{25}{2}$ è ammissibile ed ottimale per il problema P



5.4 Rilassamenti

- I rilassamenti possono essere applicati parzialmente e possono essere combinati tra loro
- La soluzione del problema rilassato costituisce in generale una limitazione della soluzione ottimale ma può anche essere la soluzione ottimale se vale la condizione E3

5.4.1 Rilassamento continuo

$$\begin{aligned} P : \quad & \max \quad z = c^T x \\ & \text{s.t.} \quad Ax \leq b \\ & \quad \quad x \geq 0 \\ & \quad \quad x_j \in N \quad \quad j \in I \end{aligned}$$

$$\begin{aligned} P' : \quad & \max \quad z = c^T x \\ & \text{s.t.} \quad Ax \leq b \\ & \quad \quad x \geq 0 \end{aligned}$$

5.4.2 Eliminazione di vincoli

Problema ben strutturato

$$\begin{aligned} P : \quad & \max \quad z = c^T x \\ & s.t. \quad A_1 x \leq b_1 \\ & \quad \quad A_2 x \leq b_2 \\ & \quad \quad x \geq 0 \\ & \quad \quad x_j \in N \quad \quad j \in I \end{aligned}$$

$$\begin{aligned} P' : \quad & \max \quad z = c^T x \\ & s.t. \quad A_1 x \leq b_1 \\ & \quad \quad x \geq 0 \\ & \quad \quad x_j \in N \quad \quad j \in I \end{aligned}$$

5.4.3 Rilassamento lagrangiano

Si inseriscono i vincoli nella funzione obiettivo, eventualmente con un peso non negativo per penalizzare le soluzioni non ammissibili, riducendo le dimensioni del problema

$$\begin{aligned}
 P : \quad & \max \quad z = \sum_{j=1, \dots, n} c_j x_j \\
 & s.t. \quad \sum_{j=1, \dots, n} a_{ij} x_j \leq b_i \quad i = 1, \dots, m \\
 & \quad \quad x_j \geq 0 \quad j = 1, \dots, n \\
 & \quad \quad x_j \in N \quad j \in I
 \end{aligned}$$

$$\begin{aligned}
 P' : \quad & \max \quad z_L = \sum_{j=1, \dots, n} c_j x_j + \sum_{i=1, \dots, m} \lambda_i \left(b_i - \sum_{j=1, \dots, n} a_{ij} x_j \right) \\
 & s.t. \quad x_j \geq 0 \quad j = 1, \dots, n \\
 & \quad \quad x_j \in N \quad j \in I
 \end{aligned}$$

5.4.4 Rilassamento surrogato

Si sostituisce ai vincoli la somma degli stessi, eventualmente con un peso non negativo, riducendo le dimensioni del problema

$$\begin{aligned}
 P : \quad & \max \quad z = \sum_{j=1, \dots, n} c_j x_j \\
 & s.t. \quad \sum_{j=1, \dots, n} a_{ij} x_j \leq b_i \quad i = 1, \dots, m \\
 & \quad \quad x_j \geq 0 \quad j = 1, \dots, n \\
 & \quad \quad x_j \in N \quad j \in I \\
 P' : \quad & \max \quad z = \sum_{j=1, \dots, n} c_j x_j \\
 & s.t. \quad \sum_{j=1, \dots, n} \left(\sum_{i=1, \dots, m} \pi_i a_{ij} \right) x_j \leq \sum_{i=1, \dots, m} \pi_i b_i \\
 & \quad \quad x_j \geq 0 \quad j = 1, \dots, n \\
 & \quad \quad x_j \in N \quad j \in I
 \end{aligned}$$

5.5 Risoluzione per ispezione

I problemi strutturalmente molto semplici possono essere risolti efficientemente per *ispezione*

Si considerano caratteristiche peculiari del problema rilassato per ottenere rapidamente la soluzione ottimale o una buona limitazione

Non esistono regole generali

Esempio 5.2 (Rilassamento lagrangiano di un problema dello zaino)

$$\begin{aligned}
& \max \sum_{j=1, \dots, n} c_j x_j \\
& \text{s.t.} \quad \sum_{j=1, \dots, n} a_j x_j \leq C \quad \& \\
& \quad \quad x_j \in \{0, 1\} \quad \quad \quad j = 1, \dots, n
\end{aligned}$$

Applicando il rilassamento lagrangiano:

$$\begin{aligned}
& \max \sum_{j=1, \dots, n} c_j x_j + \lambda \left(C - \sum_{j=1, \dots, n} a_j x_j \right) \\
& \quad \quad x_j \in \{0, 1\} \quad \quad \quad j = 1, \dots, n
\end{aligned}$$

cioè:

$$\begin{aligned}
& \max \lambda C + \sum_{j=1, \dots, n} (c_j - \lambda a_j) x_j \\
& \quad \quad x_j \in \{0, 1\} \quad \quad \quad j = 1, \dots, n
\end{aligned}$$

che risolto per ispezione fornisce la soluzione ottimale:

$$x_j = \begin{cases} 1 & \text{se } c_j - \lambda a_j > 0 \\ 0 & \text{se } c_j - \lambda a_j < 0 \\ \text{indifferente} & \text{se } c_j - \lambda a_j = 0 \end{cases}$$



Esempio 5.3 (Rilassamento surrogato di un problema a variabili 0-1)

$$\begin{aligned}
& \max \sum_{j=1, \dots, n} c_j x_j \\
& \text{s.t.} \quad \sum_{j=1, \dots, n} a_j x_j \leq b_i \quad i = 1, \dots, m \\
& \quad \quad x_j \in \{0, 1\} \quad j = 1, \dots, n
\end{aligned}$$

Applicando il rilassamento surrogato:

$$\begin{aligned}
& \max \sum_{j=1, \dots, n} c_j x_j \\
& \text{s.t.} \quad \sum_{j=1, \dots, n} \left(\sum_{i=1, \dots, m} \pi_i a_{ij} \right) x_j \leq \sum_{i=1, \dots, m} \pi_i b_i \\
& \quad \quad x_j \in \{0, 1\} \quad j = 1, \dots, n
\end{aligned}$$

cioè un problema dello zaino in cui gli oggetti hanno peso $\sum_{i=1, \dots, m} \pi_i a_{ij}$ e il peso massimo trasportabile è $\sum_{i=1, \dots, m} \pi_i b_i$ ◇

6 Complementi di programmazione lineare

6.1 Variazione dei coefficienti iniziali

Tabella iniziale:

	x^T	
u	A	b
z	c^T	0

Tabella finale:

	ξ^T	
ν	α	β
z	χ^T	χ_0

u' variabili fuori base nella tabella iniziale e successivamente entrate in base

x'' variabili in base nella tabella iniziale e rimaste in base

x' variabili in base nella tabella iniziale e successivamente uscite

u'' variabili fuori base nella tabella iniziale e rimaste fuori base

Tabella finale partizionata

	u'^T	x''^T	
x'	α_{11}	α_{12}	β_1
u''	α_{21}	α_{22}	β_2
z	χ_1^T	χ_2^T	χ_0

Tabella iniziale con la partizione indotta

	x'^T	x''^T	
u'	A_{11}	A_{12}	b_1
u''	A_{21}	A_{22}	b_2
z	c_1^T	c_2^T	0

Utilizzando la matrice A_{11} come cardine matriciale:

	u'^T	x''^T	
x'	$(A_{11})^{-1}$	$-(A_{11})^{-1}A_{12}$	$-(A_{11})^{-1}b_1$
u''	$A_{21}(A_{11})^{-1}$	$A_{22}-A_{21}(A_{11})^{-1}A_{12}$	$b_2-A_{21}(A_{11})^{-1}b_1$
z	$c_1^T(A_{11})^{-1}$	$c_2^T - c_1^T(A_{11})^{-1}A_{12}$	$-c_1^T(A_{11})^{-1}b_1$

La tabella ottenuta è quella finale espressa in funzione dei coefficienti iniziali

6.1.1 Applicazioni

- **Stabilità della soluzione ottimale**

Verificare se al variare di qualche coefficiente la soluzione data resta ottimale ed eventualmente come cambiano le coordinate della soluzione ottimale e il valore ottimale

- **Aggiunta di un vincolo**

Un ulteriore vincolo, ad esempio nel metodo degli iperpiani secanti, può essere scritto in funzione delle variabili in base nella tabella iniziale:

$$u = a_1^T x' + a_2^T x'' + a \geq 0$$

e in funzione delle variabili in base nella tabella finale:

$$u = [a_1^T (A_{11})^{-1}] u' + [a_2^T - a_1^T (A_{11})^{-1} A_{12}] x'' + [a - a_1^T (A_{11})^{-1} b_1] \geq 0$$

In questa forma cui può essere aggiunto alla tabella e proseguire con il simplesso

6.2 Algoritmi greedy

Costituiscono una classe di algoritmi approssimati caratterizzati da bassa approssimazione ma elevata velocità di esecuzione

Si analizzano le variabili decisionali, secondo un test di ottimalità locale

La soluzione trascura gli aspetti globali del problema e risente fortemente dell'ordine con cui vengono analizzate le variabili

Trovano applicazione soprattutto nei problemi decisionali a variabili 0-1 e se la velocità di calcolo e una buona approssimazione sono più importanti della soluzione esatta

6.2.1 Algoritmo greedy per il problema dello zaino

Si ordinano gli oggetti in ordine decrescente di rapporto valore/peso

Si prende un oggetto se può essere preso

L'algoritmo richiede $O(n)$ operazioni, dove n è il numero di oggetti

L'approssimazione è discreta se i pesi degli oggetti non sono molto differenti tra loro e sono piccoli rispetto al peso massimo trasportabile

Esempio 6.1 (Caso peggiore)

Oggetto	A	B
Valore	2	M
Peso	1	M
Peso massimo trasportabile = M		

L'algoritmo greedy prende il primo oggetto e scarta il secondo fornendo $z_s = 2$, mentre la soluzione ottimale è prendere il secondo oggetto e scartare il primo, con $z^* = M$, per cui:

$$\frac{z_s}{z^*} = \frac{2}{M} \rightarrow 0 \quad \text{se} \quad M \rightarrow +\infty$$



Esempio 6.2 (Soluzione approssimata con l'algoritmo greedy)

Oggetto	A	B	C	D	E	F	G	H
Valore	60	58	62	51	50	49	39	32
Peso	5	5	6	5	5	5	4	4
Peso massimo trasportabile = 20								

Applicando l'algoritmo Branch and Bound:

$$L = \left\lfloor 60 + 58 + 62 + \frac{4}{5}51 \right\rfloor = 220$$

per cui $z^* \leq 220$

L'algoritmo greedy prende gli oggetti A, B, C, G con $z_s = 219$

$$\frac{z_s}{z^*} \geq \frac{z_s}{L} = \frac{219}{220} \approx 0.995$$

Si può verificare che la soluzione trovata è esatta



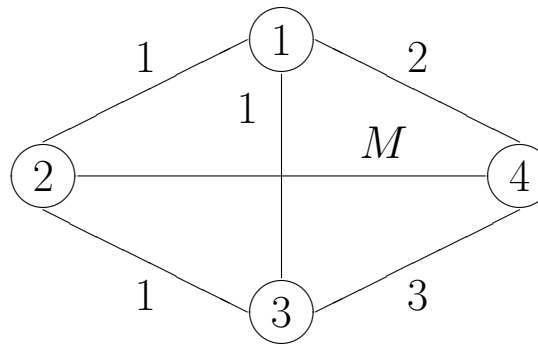
6.2.2 Algoritmo greedy per il problema del commesso viaggiatore

Da ogni città si va nella città più vicina, in termini di costi, dove non si è ancora stati (*nearest unvisited*) iniziando dalla città 1

L'algoritmo richiede $O(n^2)$ operazioni, dove n è il numero di città

L'approssimazione è discreta se le città non sono collocate disordinatamente rispetto alla città 1 e i costi rispettano le disuguaglianze triangolari

Esempio 6.3 (Caso peggiore)



L'algoritmo greedy determina il percorso $1 - 3 - 2 - 4 - 1$ fornendo $z_s = 4 + M$, mentre il percorso ottimale è $1 - 2 - 3 - 4 - 1$ con $z^* = 7$, per cui:

$$\frac{z^*}{z_s} = \frac{7}{4 + M} \rightarrow 0 \quad \text{se} \quad M \rightarrow +\infty$$



7 Teoria delle reti

7.1 Grafi

Intuitivamente un grafo è un insieme finito di punti (*nodi* o *vertici*) ed un insieme di frecce (*archi*) che uniscono coppie di punti

Il verso della freccia assegna un *orientamento* all'arco

Se non si tiene conto dell'orientamento degli archi il grafo è detto *multigrafo* o *non orientato* e gli archi sono detti *spigoli*

p numero massimo di archi orientati nello stesso verso presenti tra due nodi (*p-grafo*)

n numero di nodi (*ordine*)

m numero di archi

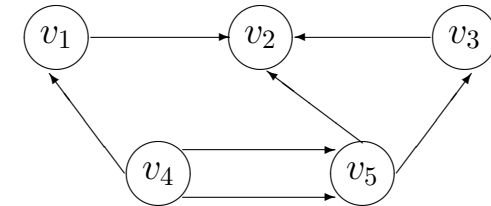
Definizione 7.1 *Un grafo è una coppia $G(N, A)$, dove $N = \{v_1, \dots, v_n\}$ è un insieme di elementi detti nodi o vertici e $A = \{a_{ij} = (v_i, v_j) | v_i, v_j \in N\}$ è una famiglia di elementi del prodotto cartesiano $N \times N$ detti archi.*

Esempio 7.1 (Elementi di un grafo)

$$N = \{v_1, v_2, v_3, v_4, v_5\}$$

$$A = \{a_{12}, a_{32}, a_{41}, a_{45}, a_{45}, a_{52}, a_{53}\}$$

$$p = 2; n = 5; m = 7$$



Definizione 7.2

- Dato un arco $a_{ij} = (v_i, v_j)$ il nodo v_i è detto nodo iniziale, primo nodo o predecessore del nodo v_j ; il nodo v_j è detto nodo finale, secondo nodo o successore del nodo v_i . I nodi v_i e v_j sono detti adiacenti. L'arco a_{ij} è detto uscente da v_i ed entrante in v_j
- Un arco $a_{ii} = (v_i, v_i)$, cioè in cui il nodo iniziale e finale coincidono è detto cappio
- Due archi che hanno un nodo in comune sono detti adiacenti
- L'insieme dei secondi nodi degli archi uscenti da un nodo v_i è detto insieme dei successori di v_i e si indica con $\omega^+(i)$ o semplicemente $\omega(i)$
- L'insieme dei primi nodi degli archi entranti in un nodo v_i è detto insieme dei predecessori di v_i e si indica con $\omega^-(i)$.

- Una sequenza di archi aventi ciascuno un nodo in comune con l'arco precedente e l'altro nodo in comune con l'arco seguente è detto cammino
- Un cammino in cui nessun arco viene percorso più di una volta è detto semplice; se nessun nodo viene incontrato più di una volta è detto elementare
- Un cammino semplice in cui il primo e l'ultimo nodo coincidono è detto ciclo; se il cammino è elementare il ciclo è detto elementare.
- Un cammino o un ciclo in cui tutti gli archi sono percorsi secondo il proprio orientamento è detto orientato, altrimenti è detto non orientato
- Se esiste un cammino tra i nodi v_i e v_j , v_j è detto accessibile da v_i e viceversa
- Un grafo $G(N, A)$ privo di cappi e in cui esiste al più un arco tra ogni coppia di nodi è detto semplice
- Un grafo $G(N, A)$ è detto connesso se ogni nodo è accessibile dagli altri
- Un multigrafo $G(N, A)$ connesso e privo di cicli elementari è detto albero
- Dato un grafo $G(N, A)$ e un sottoinsieme $A' \subset A$, il grafo $G(N, A')$ ottenuto eliminando dal grafo G gli archi dell'insieme $A \setminus A'$ è detto grafo parziale generato da A' .
- Dato un grafo connesso $G(N, A)$ un grafo parziale $G(N, A')$ connesso e privo di cicli elementari è detto spanning tree o albero ricoprente o albero parziale.
- Dato un grafo $G(N, A)$ e una bipartizione N' e N'' dell'insieme N , l'insieme degli archi aventi un estremo in N' e l'altro in N'' è detto taglio.

7.2 Reti

Definizione 7.3 *Una rete è un grafo $G(N, A)$ nel quale ad ogni arco $a_{ij} \in A$ si associano tre valori interi l_{ij}, u_{ij}, c_{ij} detti rispettivamente capacità minima, capacità massima e costo unitario dell'arco e ad ogni nodo v_i si associa un valore intero b_i ; se $b_i > 0$ il nodo v_i è detto sorgente e b_i è detta disponibilità, se $b_i < 0$ il nodo v_i è detto pozzo e b_i è detta domanda, se $b_i = 0$ il nodo v_i è detto di attraversamento*

- c_{ij} può indicare un costo, un peso, una lunghezza o qualsiasi altra cosa
- La condizione di integrità ha soprattutto motivazioni storiche

7.2.1 Flusso su una rete

Data una rete $G(N, A)$ si definisce *flusso sulla rete* una funzione

$$x : A \rightarrow \mathbb{Z}$$

dove x_{ij} è detto *flusso sull'arco* e sono soddisfatti i *vincoli di capacità degli archi* e i *vincoli di bilanciamento nei nodi*.
Al flusso x_{ij} si associa il costo $c_{ij}x_{ij}$; la somma dei costi di tutti gli archi è detta *costo associato al flusso x*

$$\begin{aligned} \text{vincoli di capacità degli archi :} & \quad l_{ij} \leq x_{ij} \leq u_{ij} & \quad \forall a_{ij} \in A \\ \text{vincoli di bilanciamento nei nodi :} & \quad \sum_{j \in \omega^+(i)} x_{ij} - \sum_{j \in \omega^-(i)} x_{ji} = b_i & \quad \forall v_i \in N \\ \text{costo associato al flusso :} & \quad \sum_{v_i \in N} \sum_{j \in \omega^+(i)} c_{ij} x_{ij} \end{aligned}$$

- Il vincolo di bilanciamento nei nodi è noto anche come legge di Kirchoff

7.2.2 Problema del flusso di costo minimo

$$\begin{aligned}
& \min \sum_{v_i \in N} \sum_{j \in \omega^+(i)} c_{ij} x_{ij} \\
& s.t. \quad \sum_{j \in \omega^+(i)} x_{ij} - \sum_{j \in \omega^-(i)} x_{ji} = b_i \quad \forall v_i \in N \\
& \quad \quad l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall a_{ij} \in A
\end{aligned}$$

In questa forma possono essere rappresentati tutti i problemi di reti

7.3 Minimo Spanning Tree

Trovare uno spanning tree di costo minimo

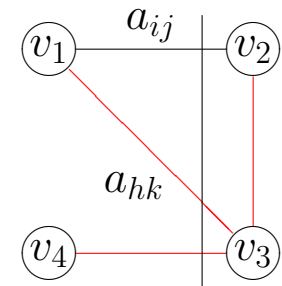
Se gli archi hanno tutti lo stesso costo la soluzione è un qualsiasi spanning tree

Teorema 7.1 *Dato un grafo $G(N, A)$ sia $N' \subseteq N$ un sottinsieme di vertici e sia $a_{ij} = (v_i, v_j)$ un arco di peso minimo appartenente al taglio generato da N' , cioè tale che $v_i \in N'$ e $v_j \in N \setminus N'$ oppure $v_i \in N \setminus N'$ e $v_j \in N'$; allora esiste un albero di peso minimo contenente a_{ij}*

Dimostrazione

Per assurdo sia T' un albero non contenente a_{ij} e di peso strettamente minore di ogni albero contenente a_{ij} e sia a_{hk} l'arco di T' appartenente al taglio generato da N' e facente parte con a_{ij} di un ciclo. Sostituendo a_{ij} ad a_{hk} si ottiene un albero di peso non superiore a T'

Assurdo

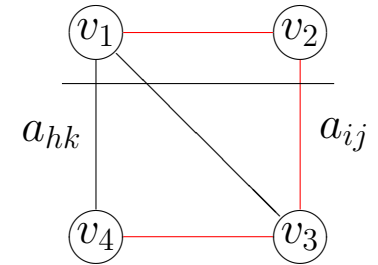


Teorema 7.2 *Dato un grafo $G(N, A)$ sia C un ciclo del grafo G e sia $a_{ij} = (v_i, v_j)$ un arco di peso massimo appartenente al ciclo C , allora esiste un albero di peso minimo non contenente a_{ij}*

Dimostrazione

Per assurdo sia T' un albero contenente a_{ij} e di peso strettamente minore di ogni albero non contenente a_{ij} , siano $N', N \setminus N' \subset N$ i sottoinsiemi del taglio di T' a cui appartiene a_{ij} e sia a_{hk} l'altro arco di C appartenente al taglio generato da N' . Sostituendo a_{hk} ad a_{ij} si ottiene un albero di peso non superiore a T'

Assurdo



Algoritmo di Prim (1957)

Costruisce uno spanning tree parziale $T'(N', A')$ aggiungendo ad ogni iterazione un arco ad A' e i relativi estremi a N' , fino a che $N' = N$

Algoritmo

- a) $A' = \emptyset; N' = \{v_1\};$
- b) determinare l'arco $a_{ij} = (v_i, v_j)$ di peso minimo tale che $v_i \in N'$ e $v_j \in N/N'$ o $v_i \in N/N'$ e $v_j \in N'$;
- c) $A' = A' \cup \{a_{ij}\}; N' = N' \cup \{v_i\} \cup \{v_j\};$
- d) se $N' \neq N$ tornare a b);
altrimenti STOP;

L'algoritmo è corretto in quanto ad ogni iterazione aggiunge l'arco di peso minimo appartenente al taglio generato da N' , in accordo col Teorema 7.1

Algoritmo di Kruskal (1956)

Costruisce uno spanning tree $T(N, A')$ aggiungendo ad ogni iterazione l'arco di peso minimo che non genera cicli ad A' , fino a che $\text{card}(A') = \text{card}(N) - 1$

Algoritmo

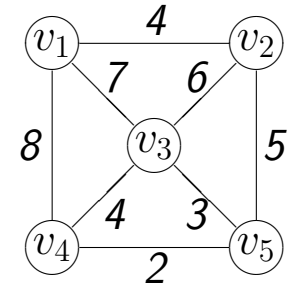
- a) $A' = \emptyset$;
- b) determinare l'arco $a_{ij} = (v_i, v_j)$ di peso minimo tale che $a_{ij} \in A/A'$ e non forma cicli;
- c) $A' = A' \cup \{a_{ij}\}$;
- d) se $\text{card}(A') < \text{card}(N) - 1$ tornare a b);
altrimenti STOP

L'algoritmo è corretto in quanto ciascun arco di G che non compare in T genererebbe un ciclo rispetto al quale sarebbe l'arco di peso massimo, in accordo col Teorema 7.2

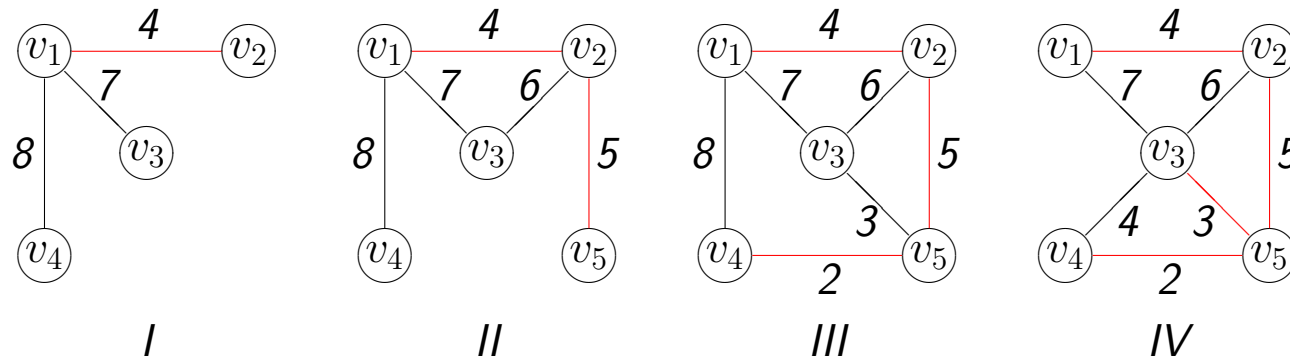
- Esiste una variante dell'algoritmo di Kruskal che elimina l'arco di peso massimo che non fa perdere la connessione

Esempio 7.2 (Minimo Spanning Tree)

Determinare uno spanning tree di peso minimo per il grafo a lato, esaminando gli archi e i vertici in ordine di indice crescente.



Algoritmo di Prim



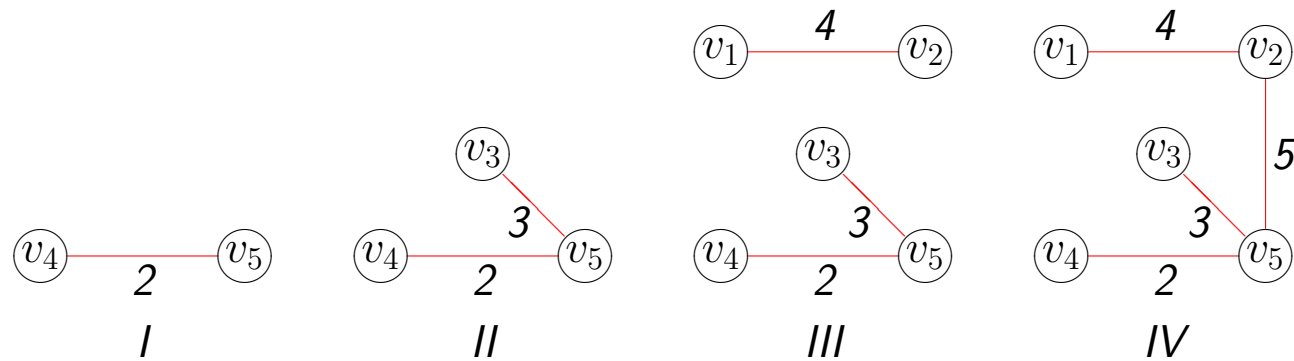
$$0 \ A' = \emptyset; N' = \{v_1\}$$

$$I \ A' = \{a_{12}\}; N' = \{v_1, v_2\}$$

$$II \ A' = \{a_{12}, a_{25}\}; N' = \{v_1, v_2, v_5\}$$

$$III \ A' = \{a_{12}, a_{25}, a_{45}\}; N' = \{v_1, v_2, v_5, v_4\}$$

$$IV \ A' = \{a_{12}, a_{25}, a_{45}, a_{35}\}; N' = \{v_1, v_2, v_5, v_4, v_3\}; \text{STOP } (N' = N)$$

Algoritmo di Kruskal

0 $A' = \emptyset$

I $A' = \{a_{45}\}$

II $A' = \{a_{45}, a_{35}\}$

III $A' = \{a_{45}, a_{35}, a_{12}\}$; l'arco a_{34} di peso 4 non viene inserito perchè formerebbe il ciclo $v_3 - v_4 - v_5 - v_3$

IV $A' = \{a_{45}, a_{35}, a_{12}, a_{25}\}$; STOP ($\text{card}(A') = \text{card}(N) - 1$)



7.4 Cammino minimo

Trovare il cammino orientato di costo minimo tra due nodi assegnati

Se gli archi hanno tutti lo stesso costo la soluzione è il cammino con il minimo numero di archi, altrimenti è possibile che un cammino con più archi risulti migliore

E' un problema semplice ma servono algoritmi efficienti perchè è un sottoproblema di molti altri problemi (reti di trasporto, reti di comunicazione, percorsi veicolari, progettazione di reti, ecc.)

Per memorizzare i cammini di costo minimo si utilizza un puntatore che per ogni nodo indica l'unico predecessore

7.4.1 SPP da un nodo v_s a tutti gli altri nodi

Ad ogni nodo v_i si assegnano un valore o etichetta $d(i)$ che indica il costo del cammino corrente da v_s a v_i e un puntatore $pred(i)$ che indica il predecessore di v_i nel cammino corrente

Si parte da un valore temporaneo per $d(i)$ che può essere modificato ad ogni iterazione fino ad ottenere il valore esatto

Esistono due classi di algoritmi:

- algoritmi label setting o di assegnazione
modificano le etichette in modo tale che ad ogni iterazione almeno una di esse diventi esatta; il più noto è l'algoritmo di Dijkstra
- algoritmi label correcting o di correzione
modificano le etichette in modo tale che solo all'ultima iterazione si è certi dell'esattezza di tutte le etichette; i più noti algoritmi sono quello di Ford e la variante di Bellman

Gli algoritmi di assegnazione sono più efficienti nei casi peggiori, invece gli algoritmi di correzione non necessitano di archi a costo non negativo

Algoritmo di Dijkstra (1959)

Algoritmo

- a) $d(s) = 0$;
- b) per $i = 1, \dots, n$ $d(i) = c_{si}$ e $pred(i) = s$;
- c) sia $d(h) = \min\{d(i) | d(i) \text{ non è esatta}\}$ ($d(h)$ diventa esatta);
- d) se $v_i \in \omega(h)$ e $d(i)$ non è esatta $d(i) = \min\{d(i), d(h) + c_{hi}\}$ eventualmente, se $d(i)$ è stata modificata, $pred(i) = h$;
- e) se tutti i valori $d(i)$ sono esatti STOP ($n - 1$ iterazioni);
altrimenti tornare a c);

L'algoritmo è corretto in quanto ad ogni iterazione le etichette temporanee (che rappresentano il costo del cammino minimo che utilizza solo nodi con etichetta esatta) sono tutte non minori di quelle esatte; per il nodo v_h con etichetta temporanea minima ogni altro cammino deve comprendere almeno un nodo v_k con etichetta non esatta, per cui il costo è non minore di quello da v_s a v_k più il costo c_{kh} . Se qualche arco ha costo negativo la prova di correttezza non sussiste

Algoritmo di Ford (1956)

Algoritmo

- a) $d(s) = 0$;
- b) per $i = 1, \dots, n$ $d(i) = M$ e $pred(i) = s$ (M maggiorante delle etichette esatte);
- c) se esiste a_{hi} con $d(i) > d(h) + c_{hi}$ porre $d(i) = d(h) + c_{hi}$ e $pred(i) = h$ e ripetere c);
altrimenti STOP (le etichette sono tutte esatte);

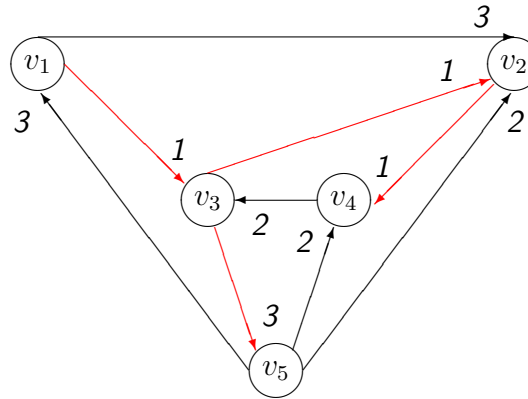
L'algoritmo è corretto in quanto dopo ogni aggiornamento l'etichetta $d(i)$ è ancora un maggiorante del cammino minimo da v_s a v_i , poichè lo è $d(h)$. D'altra parte al termine dell'algoritmo preso un qualsiasi cammino C da v_s a v_i si ha che $d(i) \leq \sum_{(v_k, v_l) \in C} c_{kl}$ per cui il valore finale di $d(i)$ è un minorante del cammino minimo da v_s a v_i e quindi $d(i)$ è esatta

Variante di Bellman (1958)

Si esaminano gli archi secondo un ordinamento prefissato

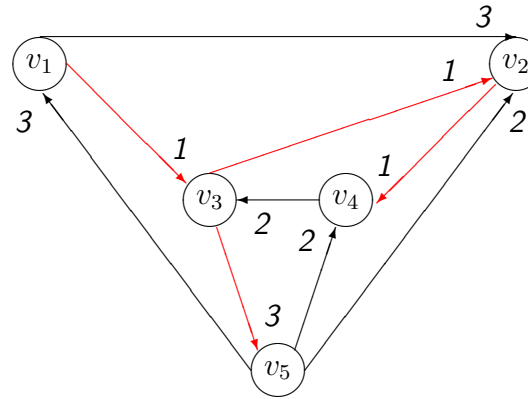
L'algoritmo termina dopo aver esaminato gli archi al più $n - 1$ volte

Determina l'esistenza di cicli di costo negativo, eseguendo una ulteriore iterazione (cammino composto da almeno n archi)

Esempio 7.3 (Grafo con archi di peso non negativo)Algoritmo di Dijkstra

d	0	99	99	99	99	pred	1	1	1	1	1	h	= 1
d	0	3	1	99	99	pred	1	1	1	1	1	h	= 3
d	0	2	1	99	4	pred	1	3	1	1	3	h	= 2
d	0	2	1	3	4	pred	1	3	1	2	3	h	= 4
d	0	2	1	3	4	pred	1	3	1	2	3	h	= 5
d	0	2	1	3	4	pred	1	3	1	2	3		

STOP



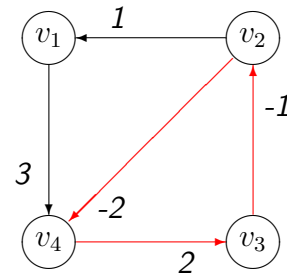
Algoritmo di Bellmann

Ordinamento per indici crescenti

d	0	99	99	99	99	pred	1	1	1	1	1	a_{12}
d	0	3	99	99	99	pred	1	1	1	1	1	a_{13}
d	0	3	1	99	99	pred	1	1	1	1	1	a_{24}
d	0	3	1	4	99	pred	1	1	1	2	1	a_{32}
d	0	2	1	4	99	pred	1	3	1	2	1	a_{35}
d	0	2	1	4	4	pred	1	3	1	2	3	$a_{43}, a_{51}, a_{52}, a_{54}$ (1) $a_{12}, a_{13}, a_{24},$
d	0	2	1	3	4	pred	1	3	1	2	3	$a_{32}, a_{35}, a_{43}, a_{51}, a_{52}, a_{54}$ (2) $a_{12}, a_{13}, a_{24}, a_{32}$
												$[a_{35}, a_{43}, a_{51}, a_{52}, a_{54}$ (3)]
d	0	2	1	3	4	pred	1	3	1	2	3	
STOP												



Esempio 7.4 (Grafo con ciclo negativo)



Algoritmo di Dijkstra

d 0 99 99 99 **pred** 1 1 1 1 **h** = 1

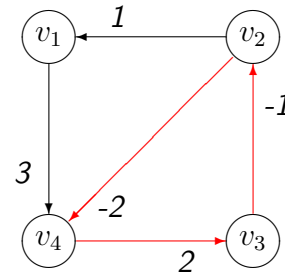
d 0 99 99 3 **pred** 1 1 1 1 **h** = 4

d 0 99 5 3 **pred** 1 1 4 1 **h** = 3

d 0 4 5 3 **pred** 1 3 4 1 **h** = 2

d 0 4 5 3 **pred** 1 3 4 1

STOP (non identificato ciclo negativo)



Algoritmo di Bellmann

Ordinamento per indici crescenti

d	0	99	99	99	pred	1	1	1	1	a_{14}
d	0	99	99	3	pred	1	1	1	1	a_{21}, a_{24}, a_{32}
d	0	98	99	3	pred	1	3	1	1	a_{43}
d	0	98	5	3	pred	1	3	4	1	(1) $a_{14}, a_{21}, a_{24}, a_{32}$
d	0	4	5	3	pred	1	3	4	1	a_{43} (2) a_{14}, a_{21}, a_{24}
d	0	4	5	2	pred	1	3	4	2	a_{32}, a_{43}
d	0	4	4	2	pred	1	3	4	2	(3) $a_{14}, a_{21}, a_{24}, a_{32}$

STOP (identificato ciclo negativo)



7.4.2 SPP tra qualsiasi coppia di nodi

Si può applicare uno degli algoritmi precedenti assumendo ogni volta un nodo iniziale diverso, oppure si possono usare algoritmi specifici, ad esempio l'algoritmo di Floyd (1962)

7.5 Flusso massimo

Data una rete di trasporto $G(N, A)$, cioè:

- senza cappi
- con una unica sorgente v_s priva di archi entranti e con un unico pozzo v_t privo di archi uscenti, per i quali non sono definite la disponibilità e la domanda
- con gli altri nodi di attraversamento (cioè con domanda nulla)

determinare un flusso x che soddisfa i vincoli e per cui sia massimo il *valore del flusso da v_s a v_t* , indicato con $F(v_s, v_t)$:

$$\sum_{i \neq s, t} \left(\sum_{j \in \omega^+(i)} x_{ij} - \sum_{j \in \omega^-(i)} x_{ji} \right) + \sum_{j \in \omega^+(s)} x_{sj} - \sum_{j \in \omega^-(t)} x_{jt} = 0$$

per la legge di Kirchoff la prima sommatoria è nulla, per cui si ha:

$$\sum_{j \in \omega^+(s)} x_{sj} = \sum_{j \in \omega^-(t)} x_{jt} = F(v_s, v_t)$$

E' utile anche per valutare l'affidabilità di una rete (indisponibilità di qualche arco)

Problema di programmazione lineare a numeri interi:

$$\begin{aligned}
 & \max \quad z = F \\
 & \text{s.t.} \quad \sum_{j \in \omega^+(s)} x_{sj} - F = 0 \\
 & \quad \sum_{j \in \omega^+(i)} x_{ij} - \sum_{j \in \omega^-(i)} x_{ji} = 0 \quad \forall v_i \in N \setminus \{v_s, v_t\} \\
 & \quad - \sum_{j \in \omega^-(t)} x_{jt} + F = 0 \\
 & \quad l_{ij} \leq x_{ij} \leq u_{ij}; x_{ij} \text{ intero} \quad \forall a_{ij} \in A
 \end{aligned}$$

Algoritmi di programmazione lineare o algoritmi specifici

Algoritmi di cammino aumentante

Costruiscono una sequenza di cammini dalla sorgente v_s al pozzo v_t sui quali incrementare il flusso fino a raggiungere il massimo

- Algoritmo del contrassegno (Ford e Fulkerson, 1956)
- Algoritmo del minimo cammino aumentante (Edmonds e Karp, 1972)
- Algoritmo delle reti stratificate (Dinic, 1970 - Ahuja e Orlin, 1991)

Algoritmi di preflusso

Sono algoritmi più flessibili che consentono di aumentare il flusso non su un cammino ma su un singolo arco con una operazione di invio (*push*)

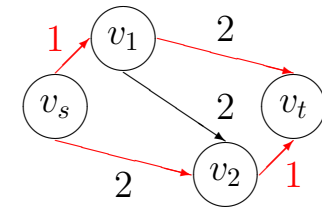
- Algoritmo di preflusso (Karzanov, 1974 - Goldberg e Tarjan, 1986)
- Variante (Goldberg e Tarjan, 1986)
- Algoritmo di scaling dell'eccesso (Ahuja e Orlin, 1991)

Algoritmo del contrassegno

- a) inizializzare x con un flusso ammissibile (se le capacità minime sono tutte nulle il flusso nullo è ammissibile);
- b) $V = \{v_s\}$ (V è l'insieme dei nodi contrassegnati);
- c) se esistono due nodi adiacenti $v_i \in V$ e $v_j \notin V$ contrassegnare v_j
 con $+i$ se $j \in \omega^+(i)$ e $x_{ij} < u_{ij}$ oppure
 con $-i$ se $j \in \omega^-(i)$ e $x_{ji} > l_{ji}$ e andare a d);
 altrimenti STOP (flusso massimo);
- d) se $v_t \in V$ esiste un cammino non orientato C da v_s a v_t sul quale si può variare il flusso di Δ andare a e);
 altrimenti tornare a c);
- e) costruire un nuovo flusso x ponendo:

$$x_{ij} = \begin{cases} x_{ij} & \text{se } a_{ij} \notin C \\ x_{ij} + \Delta & \text{se } a_{ij} \in C \text{ secondo l'orientamento} \\ x_{ij} - \Delta & \text{se } a_{ij} \in C \text{ non secondo l'orientamento} \end{cases}$$

tornare a b);



L'algoritmo è corretto in quanto preso $U \subset N$ qualsiasi con $v_s \in U$, $v_t \notin U$ si ha:

$$F(v_s, v_t) = \sum_{j \in \omega^+(s)} x_{sj} + \sum_{v_i \in U, i \neq s} \left\{ \sum_{j \in \omega^+(i)} x_{ij} - \sum_{j \in \omega^-(i)} x_{ji} \right\}$$

Eliminando i termini di segno opposto si ha:

$$F(v_s, v_t) = \sum_{v_i \in U, v_j \notin U} x_{ij} - \sum_{v_i \in U, v_j \notin U} x_{ji} \leq \sum_{v_i \in U, v_j \notin U} u_{ij} - \sum_{v_i \in U, v_j \notin U} l_{ji}$$

Preso $V = \{\text{nodì contrassegnati al termine dell'algoritmo}\}$ si ha:

$$F(v_s, v_t) = \sum_{v_i \in V, v_j \notin V} x_{ij} - \sum_{v_i \in V, v_j \notin V} x_{ji} = \sum_{v_i \in V, v_j \notin V} u_{ij} - \sum_{v_i \in V, v_j \notin V} l_{ji}$$

Definizione 7.4 *Data una rete di trasporto $G(N, A)$ e il taglio generato da un insieme $U \subset N$, con $v_s \in U$ e $v_t \notin U$ si chiama capacità del taglio la quantità $\sum_{v_i \in U, v_j \notin U} u_{ij} - \sum_{v_i \in U, v_j \notin U} l_{ji}$*

Teorema 7.3 (Teorema di Ford e Fulkerson (max flow - min cut, 1956)) *Data una rete di trasporto $G(N, A)$ si ha:*

$$\max F(v_s, v_t) = \min \left\{ \sum_{v_i \in U, v_j \notin U} u_{ij} - \sum_{v_i \in U, v_j \notin U} l_{ji} \mid U \subset N, v_s \in U, v_t \notin U \right\}$$

Problema del flusso compatibile

Definizione 7.5 Si definisce distanza di un flusso x dalle capacità l e u la quantità:

$$d(x) = \sum_{a_{ij} \in A} d(x_{ij})$$

$$\text{dove } d(x_{ij}) = \begin{cases} l_{ij} - x_{ij} & \text{se } x_{ij} < l_{ij} \\ 0 & \text{se } l_{ij} \leq x_{ij} \leq u_{ij} \\ x_{ij} - u_{ij} & \text{se } x_{ij} > u_{ij} \end{cases}$$

Si noti che $d(x) \geq 0$ e in particolare $d(x) = 0 \Leftrightarrow x$ è ammissibile

Algoritmo di Herz (1967)

- a) inizializzare x in modo che siano soddisfatti i vincoli di conservazione del flusso ma non (necessariamente) quelli di capacità (flusso nullo);
- b) se esiste $x_{hk} < l_{hk}$ contrassegnare v_k con $+h$ e andare a c);
- b') se esiste $x_{kh} > u_{kh}$ contrassegnare v_k con $-h$;
- c) se si contrassegna v_t si contrassegna anche v_s con $+t$;
- c') se si contrassegna v_s si contrassegna anche v_t con $-s$;
- d) se esistono due nodi adiacenti v_i contrassegnato e v_j non contrassegnato, si contrassegna v_j con $+i$ se $j \in \omega^+(i)$ e $x_{ij} < u_{ij}$ oppure con $-i$ se $j \in \omega^-(i)$ e $x_{ji} > l_{ji}$ e andare a e); altrimenti STOP (non esiste un flusso ammissibile);
- e) se si contrassegna v_h esiste un ciclo non orientato C sul quale si può variare il flusso di Δ e andare a f); altrimenti tornare a c);
- f) costruire un nuovo flusso x ponendo:

$$x_{ij} = \begin{cases} x_{ij} & \text{se } a_{ij} \notin C \\ x_{ij} + \Delta & \text{se } a_{ij} \in C \text{ secondo l'orientamento} \\ x_{ij} - \Delta & \text{se } a_{ij} \in C \text{ non secondo l'orientamento} \end{cases}$$
- g) se $d(x) = 0$ STOP (il flusso x è ammissibile) altrimenti tornare a b);

L'algoritmo converge poichè ad ogni iterazione la distanza $d(x)$ si riduce

Teorema 7.4 (Teorema di Hoffman, 1960)

Data una rete di trasporto $G(N, A)$ esiste un flusso ammissibile x se e solo se preso comunque $U \subset N$, con $v_s, v_t \in U$ oppure $v_s, v_t \notin U$ si ha:

$$\sum_{v_i \in U, v_j \notin U} l_{ij} \leq \sum_{v_i \in U, v_j \notin U} u_{ji}$$

L'insieme dei nodi non contrassegnati al termine dell'algoritmo quando non esiste il flusso ammissibile non verifica la condizione del teorema

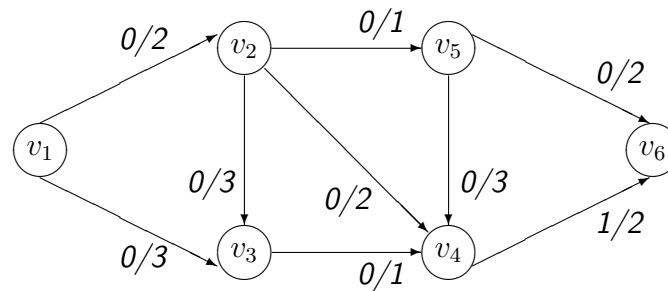
Esempio 7.5 Si considerino le seguenti due situazioni di flusso impossibile:



Nel caso a) dopo aver determinato il flusso 2 su entrambi gli archi, si contrassegnano v_s e v_t e l'insieme $\{v_1\}$ non verifica la condizione del teorema

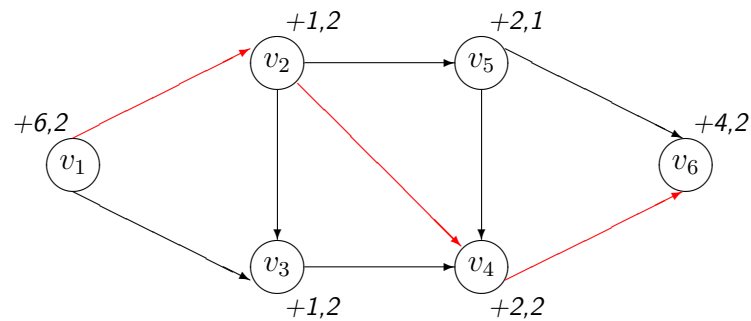
Nel caso b) dopo aver determinato il flusso 1 su entrambi gli archi, si contrassegna v_1 e l'insieme $\{v_s, v_t\}$ non verifica la condizione del teorema ◇

Esempio 7.6 (Flusso massimo)



- si esaminano nodi e archi secondo l'ordine crescente degli indici
- si contrassegnano tutti i nodi possibili
- al contrassegno si aggiunge il massimo incremento corrente

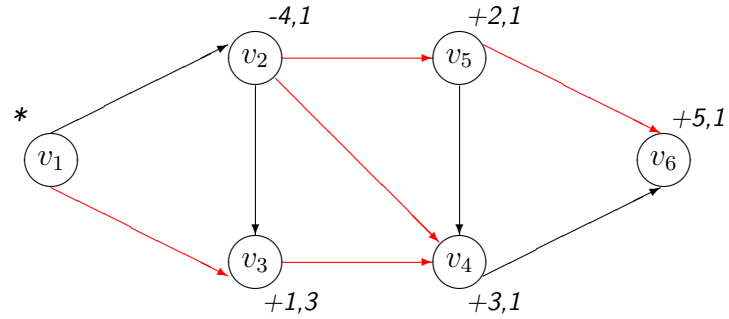
Algoritmo di Herz dall'arco a_{46}



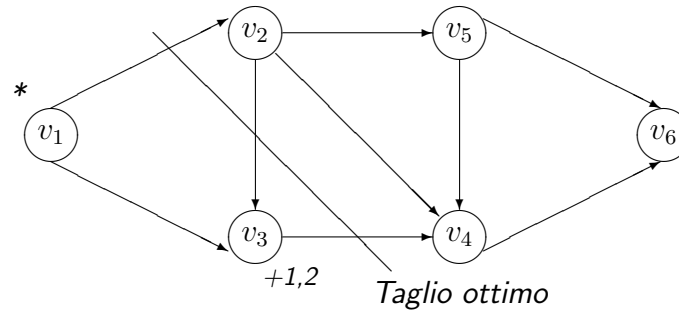
Ciclo: $v_4 - v_6 - v_1 - v_2 - v_4$; $\Delta = 2$

Flusso ammissibile

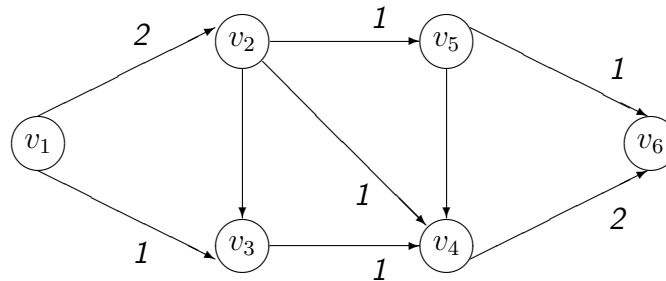
Algoritmo di Ford e Fulkerson



Cammino aumentante: $v_1 - v_3 - v_4 - v_2 - v_5 - v_6$; $\Delta = 1$



Flusso massimo



Taglio minimo $(v_1, v_2), (v_2, v_3), (v_3, v_4)$; capacità 3



Relazione di dualità tra flusso massimo e taglio minimo

Dato il problema di flusso massimo:

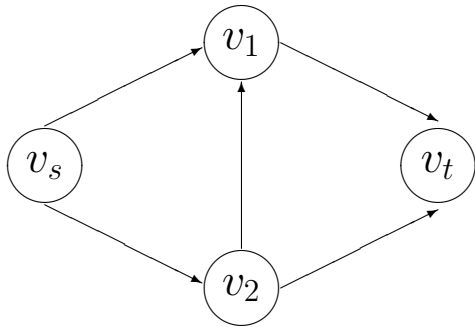
$$\begin{aligned}
 \max \quad & z = F \\
 \text{s.t.} \quad & \sum_{j \in \omega^+(s)} x_{sj} - F = 0 \\
 & \sum_{j \in \omega^+(i)} x_{ij} - \sum_{j \in \omega^-(i)} x_{ji} = 0 \quad \forall v_i \in N \setminus \{v_s, v_t\} \\
 & - \sum_{j \in \omega^-(t)} x_{jt} + F = 0 \\
 & l_{ij} \leq x_{ij} \leq u_{ij}; x_{ij} \text{ intero} \quad \forall a_{ij} \in A
 \end{aligned}$$

trascurando il vincolo di integrità del flusso, il duale è:

$$\begin{aligned}
 \min \quad & w = \sum_{a_{ij} \in A} u_{ij} \alpha_{ij} + \sum_{a_{ij} \in A} l_{ij} \beta_{ij} \\
 \text{s.t.} \quad & y_i - y_j + \alpha_{ij} + \beta_{ij} = 0 \quad \forall a_{ij} \in A \\
 & y_t - y_s = 1 \\
 & \alpha_{ij} \geq 0 \quad \forall a_{ij} \in A \\
 & \beta_{ij} \leq 0 \quad \forall a_{ij} \in A
 \end{aligned}$$

- y_i sono libere e possono essere considerate variabili 0-1 col significato $y_i = 0$ se v_i appartiene all'insieme U che genera il taglio e $y_i = 1$ altrimenti
- $y_t - y_s = 1$ assicura $y_t = 1$ e $y_s = 0$, cioè $v_s \in U, v_t \notin U$

Esempio 7.7 (Max flow - min cut)



	x_{s1}	x_{s2}	x_{1t}	x_{21}	x_{2t}	F	min
y_s	1	1				-1	= 0
y_1	-1		1	-1			= 0
y_2		-1		1	1		= 0
y_t			-1		-1	1	= 0
α_{s1}	1						$\leq u_{s1}$
α_{s2}		1					$\leq u_{s2}$
α_{1t}			1				$\leq u_{1t}$
α_{21}				1			$\leq u_{21}$
α_{2t}					1		$\leq u_{2t}$
β_{s1}	1						$\geq l_{s1}$
β_{s2}		1					$\geq l_{s2}$
β_{1t}			1				$\geq l_{1t}$
β_{21}				1			$\geq l_{21}$
β_{2t}					1		$\geq l_{2t}$
	=	=	=	=	=	=	
max	0	0	0	0	0	1	

Dato un taglio generato da un insieme $U \subset N$, per un arco $a_{ij} \in A$ sono possibili i seguenti casi:

- 1. $v_i \in U, v_j \notin U \Rightarrow y_i = 0, y_j = 1$; allora per l'ammissibilità si ha $\alpha_{ij} + \beta_{ij} = 1$ e per l'ottimalità si ha $\alpha_{ij} = 1, \beta_{ij} = 0$*
- 2. $v_i \notin U, v_j \in U \Rightarrow y_i = 1, y_j = 0$; allora per l'ammissibilità si ha $\alpha_{ij} + \beta_{ij} = -1$ e per l'ottimalità si ha $\alpha_{ij} = 0, \beta_{ij} = -1$*
- 3. $v_i, v_j \in U$ oppure $v_i, v_j \notin U \Rightarrow y_i = y_j$; allora per l'ammissibilità si ha $\alpha_{ij} + \beta_{ij} = 0$ e per l'ottimalità si ha $\alpha_{ij} = 0, \beta_{ij} = 0$*

Quindi il valore della funzione obiettivo è la capacità del taglio e la soluzione ottimale è un taglio di capacità minima ◇

Il Teorema di Ford e Fulkerson è un caso particolare del I Teorema della dualità

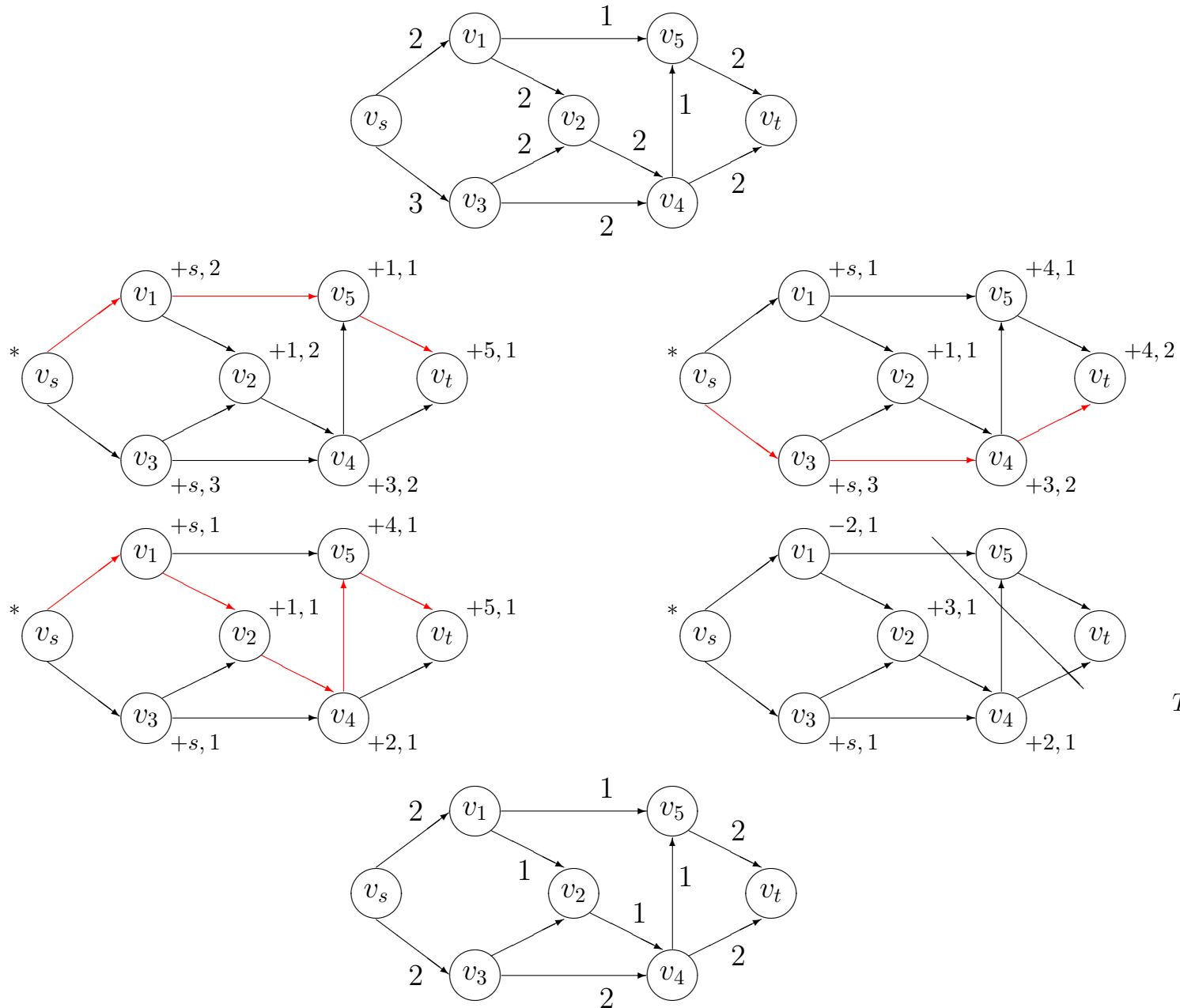
7.6 Algoritmo del minimo cammino aumentante (Edmonds e Karp, 1972)

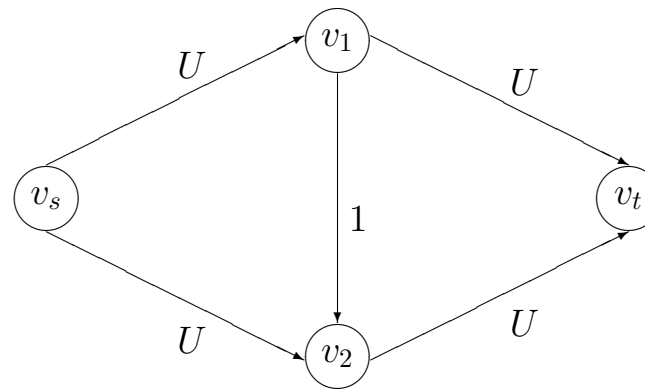
Si analizzano i nodi etichettati in ampiezza (ordine di etichettatura)

E' sufficiente modificare il passo c) nel modo seguente:

c') se $V \neq \emptyset$ detto v_i il primo nodo di V secondo l'ordine di etichettatura, contrassegnare tutti i nodi adiacenti $v_j \notin V$ con "+i" se $j \in w^+(i)$ e $x_{ij} < u_{ij}$ oppure con "-i" se $j \in w^-(i)$ e $x_{ij} > l_{ij}$; eliminare v_i da V e andare a d);

Esempio 7.8 (Algoritmo di Edmonds e Karp)



Esempio 7.9 (Confronto di complessità)

L'algoritmo di Edmonds e Karp determina i cammini aumentanti $v_s - v_1 - v_t$ e $v_s - v_2 - v_t$, entrambi con incremento U

L'algoritmo di Ford e Fulkerson può determinare sequenzialmente i cammini aumentanti $v_s - v_1 - v_2 - v_t$ e $v_s - v_2 - v_1 - v_t$ con incremento unitario, richiedendo complessivamente $2U$ cammini $\diamond$

7.7 Complessità computazionale degli algoritmi di flusso massimo

Si consideri una rete con n nodi e m archi:

Algoritmo di Ford e Fulkerson

- Analisi degli archi per ogni aggiornamento = $O(m)$
- Aggiornamento del flusso = $O(n)$ - Ogni cammino contiene al più $n - 1$ archi
- Numero degli aggiornamenti = $O(nU)$ - U è il massimo delle capacità massime degli archi; il flusso viene aumentato di almeno un'unità e il taglio generato da v_s e $N \setminus \{v_s\}$ contiene al più n archi

Complessivamente si ha $O((n + m)nU)$ e quindi l'algoritmo ha complessità $O(nmU)$

Algoritmo di Edmonds e Karp

- Analisi degli archi per ogni aggiornamento = $O(m)$
- Aggiornamento del flusso = $O(n)$ - Ogni cammino contiene al più $n - 1$ archi
- Numero dei cammini minimi = $O(nm)$ - Per ogni lunghezza ci sono al più $O(m)$ cammini; infatti un arco a_{ij} viene saturato due volte con cammini di lunghezza k se esiste un cammino di lunghezza k contenente l'arco a_{ij} nel verso opposto all'orientamento, ma allora esisterebbe un cammino di lunghezza minore di k .

Complessivamente si ha $O((n + m)nm)$ e quindi l'algoritmo ha complessità $O(nm^2)$

Reti sparse ($m = O(n)$) e dense ($m = O(n^2)$):

algoritmo	rete sparsa	rete densa
Ford e Fulkerson	$O(n^2U)$	$O(n^3U)$
Edmonds e Karp	$O(n^3)$	$O(n^5)$

- Se $U = O(n)$ i due algoritmi sono equivalenti per le reti sparse
- Il Teorema della decomposizione del flusso: *Dati due flussi ammissibili, relativi alla stessa rete, è sempre possibile passare dall'uno all'altro decomponendo il flusso in al più $n + m$ cammini e cicli di cui al più m cicli* fornisce un limite inferiore per il numero di iterazioni di un algoritmo nel caso peggiore. Dal punto di vista pratico il Teorema assicura l'esistenza, ma non dà indicazioni sul come determinare i cammini e i cicli

7.8 Problema di produzione e magazzino

Problema di produzione e magazzino

Una azienda produce per n periodi consecutivi; per ciascun periodo $i = 1, \dots, n$ sono noti:

d_i domanda

c_i costo di produzione unitario

h_i costo di immagazzinamento unitario

C_i capacità produttiva massima

H_i capacità del magazzino

$$c_{pi} = c_i \quad i = 1, \dots, n$$

$$c_{ir} = 0 \quad i = 1, \dots, n$$

$$c_{i,i+1} = h_i \quad i = 1, \dots, n - 1$$

$$l_{ij} = 0 \quad a_{ij} \in A$$

$$u_{pi} = C_i \quad i = 1, \dots, n$$

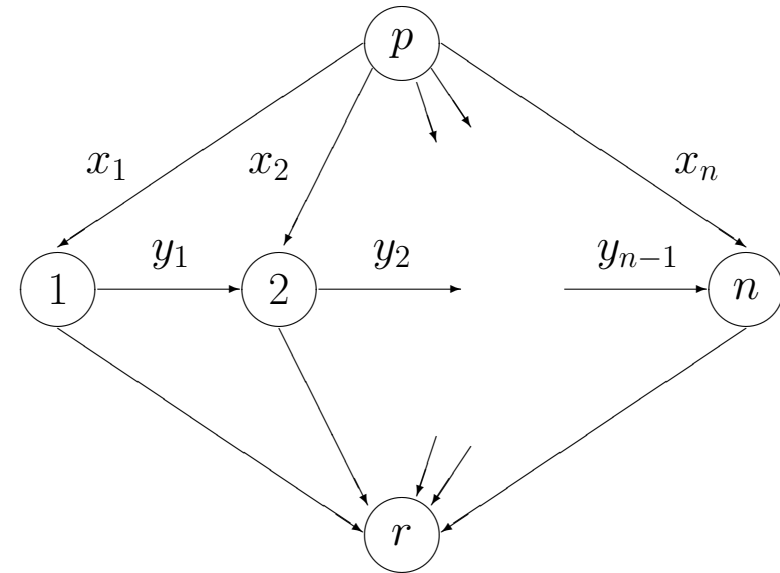
$$u_{ir} = d_i \quad i = 1, \dots, n$$

$$u_{i,i+1} = H_i \quad i = 1, \dots, n - 1$$

$$b_p = \sum_{i=1, \dots, n} d_i$$

$$b_r = - \sum_{i=1, \dots, n} d_i$$

$$b_i = 0 \quad i = 1, \dots, n$$



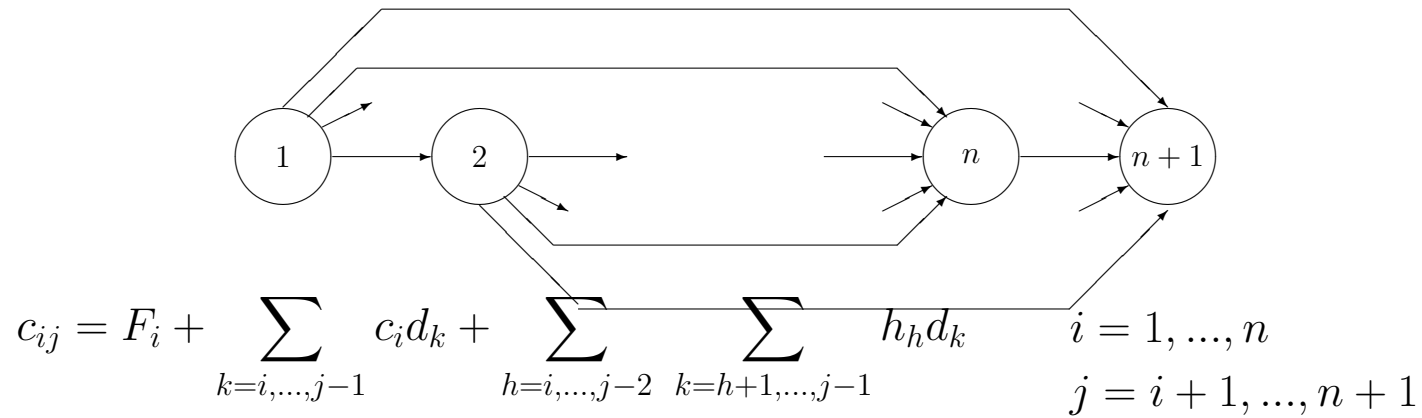
I vincoli di bilanciamento nel nodo i sono i vincoli dinamici del problema del magazzino:

$$x_i + y_{i-1} = d_i + y_i \quad i = 1, \dots, n$$

Problema di flusso di costo minimo da p a r

Se la capacità produttiva e la capacità del magazzino non sono limitate si ottiene un problema di cammino minimo da p a ciascun i : per ciascun periodo si attinge al magazzino o alla produzione

Se per ogni periodo $i = 1, \dots, n$ esiste un costo fisso di produzione F_i si costruisce la seguente rete:



Problema di cammino minimo da 1 a $n + 1$

Gli archi facenti parte della soluzione indicano in quali periodi produrre e per quanti periodi

7.9 Problema di routing e scheduling

Variante del problema del trasporto con itinerari reali e operazioni di carico e scarico

Si suppone un'unica origine o deposito (*depot*) ma più destinazioni

E' possibile che un unico viaggio non consenta di soddisfare tutte le richieste

Se le operazioni presso le destinazioni devono rispettare vincoli temporali (*finestre*) si ha un problema di routing e scheduling

La complessità del problema può diventare molto elevata

7.10 Problema di progettazione

Dato un progetto composto da n attività, le durate, $d_i, i = 1, \dots, n$, e le relazioni di precedenza, si vogliono determinare:

- la durata minima del progetto
- l'istante $t_i, i = 1, \dots, n$ in cui ogni attività può cominciare
- l'istante $T_i, i = 1, \dots, n$ entro cui ogni attività deve cominciare

Esempio 7.10 (Descrizione di un progetto) *Sia dato il progetto rappresentato dal seguente schema:*

<i>Attività</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>
<i>Durata</i>	6	7	2	3	5	4	8	3
<i>Precedenze</i>	<i>B</i>	<i>D</i>	<i>E</i>	<i>E</i>	<i>H</i>	<i>H</i>		
	<i>C</i>	<i>E</i>	<i>F</i>	<i>G</i>				
		<i>G</i>	<i>H</i>					



Modelli

- CPM (Critical Path Method): le attività corrispondono ai nodi del grafo
- PERT (Project Evaluation and Review Technique) le attività corrispondono agli archi del grafo; ulteriori archi (*attività fittizie*) rappresentano le precedenze

7.10.1 CPM (Roy, 1960)

● **Dati**

- all'attività i si associa il nodo v_i del grafo
- l'arco (v_i, v_j) indica che l'attività i precede l'attività j ($i \prec j$)
- all'arco (v_i, v_j) si associa il costo d_i corrispondente alla durata dell'attività i

● **Ipotesi**

- si aggiungono un'attività iniziale α e un'attività finale ω , entrambe di durata nulla; il nodo v_α viene collegato tramite archi uscenti ai nodi corrispondenti alle attività che non devono essere precedute da altre e il nodo v_ω viene collegato tramite archi entranti ai nodi corrispondenti alle attività che non sono seguite da altre
- il grafo è aciclico per costruzione (un'attività non può iniziare dopo il suo termine)

• Definizioni

- t_ω è detto *durata minima del progetto*
- $m_i = T_i - t_i$ è detto *ritardo massimo ammissibile*
- le attività per cui $T_i = t_i$ sono dette *attività critiche*
- un cammino da v_α a v_ω che passa solo per nodi corrispondenti ad attività critiche è detto *cammino critico*

• Calcolo di t_i e T_i

- t_i è il costo del cammino più lungo da v_α a v_i

$$t_\alpha = 0$$

$$t_i = \max \{t_j + d_j; v_j \in \omega^-(i)\}$$

- $T_\omega - T_i$ è il costo del cammino più lungo da v_i a v_ω

$$T_\omega = t_\omega$$

$$T_i = \min \{T_j - d_i; v_j \in \omega^+(i)\}$$

• Cammino critico

- si determinano i ritardi massimi $m_i = T_i - t_i$, si identificano le attività critiche e i cammini critici

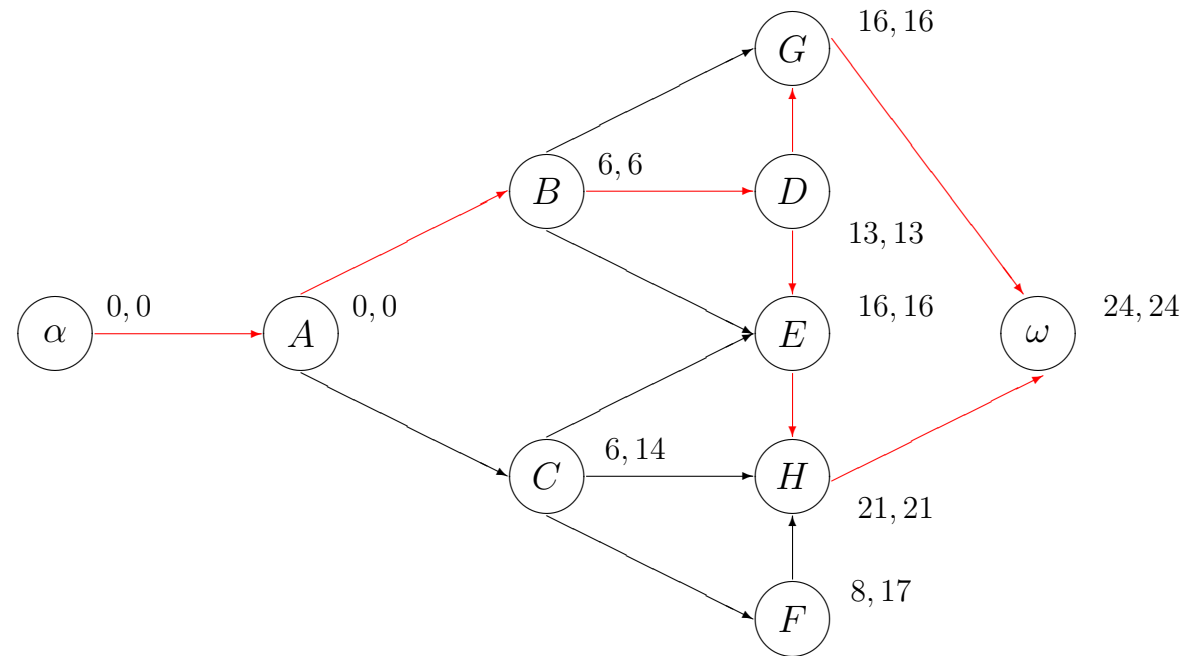
- Il calcolo di t_i e T_i può essere accelerato numerando le attività in modo da rispettare le precedenze (*topological sorting*):

$$t_i = \max \{t_j + d_j; j < i\}$$

$$T_i = \min \{T_j - d_i; j > i\}$$

- Nella modellizzazione non si tiene conto della possibilità che un'attività abbia una durata superiore al previsto, almeno a priori, in quanto una maggior durata può essere vista, a posteriori, come un inizio ritardato

Esempio 7.11 (Cammino critico) Riferendosi all'Esempio 7.10 si ottengono i seguenti risultati:



Tempo minimo: 24

Attività critiche: A, B, D, E, G, H

Cammini critici: $A - B - D - G$ e $A - B - D - E - H$

Anche i cammini $A - B - G$ e $A - B - E - H$ passano solo per nodi corrispondenti ad attività critiche



7.10.2 PERT (Malcom, Roseboom, Clark, Fazar, 1959)

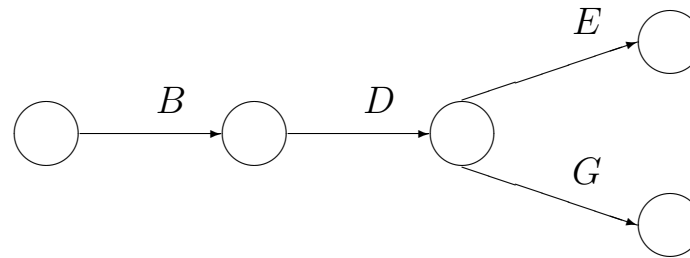
● Definizioni

- se $T_i = t_i$ l'evento i è detto *critico*
 - un'attività è detta *critica* se l'arco corrispondente (v_i, v_j) congiunge due eventi critici e $t_j - t_i$ è uguale alla durata dell'attività
 - un'attività è detta *fittizia* se non corrisponde a nessuna attività del progetto; le attività fittizie vengono inserite per rappresentare correttamente le relazioni di precedenza
- Il calcolo di t_i e T_i viene fatto nel modo visto nel modello CPM
 - In generale per limitare la complessità, si cerca di minimizzare il numero di attività fittizie
 - Talvolta le attività fittizie vengono introdotte per ottenere un grafo semplice

Esempio 7.12 (PERT) *Riferendosi all'Esempio 7.10 si può osservare che:*

$$\left. \begin{array}{l} B \prec D, E, G \\ D \prec E, G \end{array} \right\} \Rightarrow B \prec D \prec E, G$$

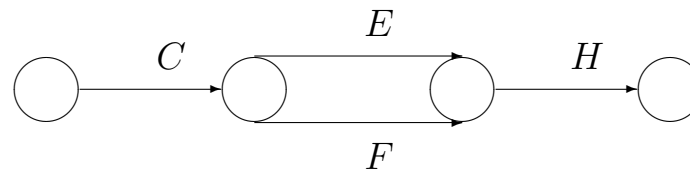
cioè:



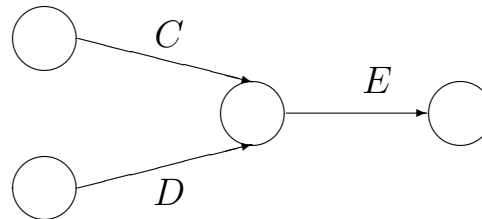
Inoltre si ha:

$$\left. \begin{array}{l} C \prec E, F, H \\ E \prec H \\ F \prec H \end{array} \right\} \Rightarrow C \prec E, F \prec H$$

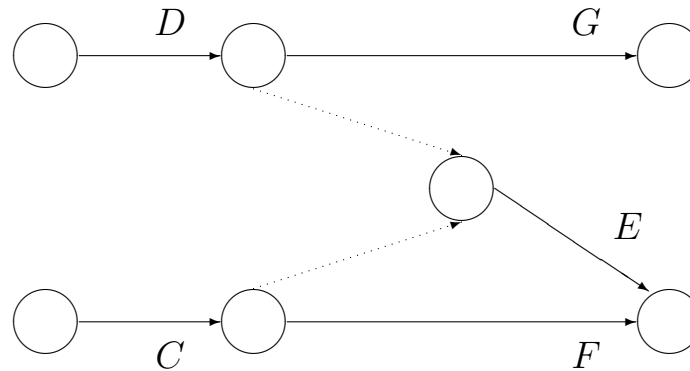
cioè:



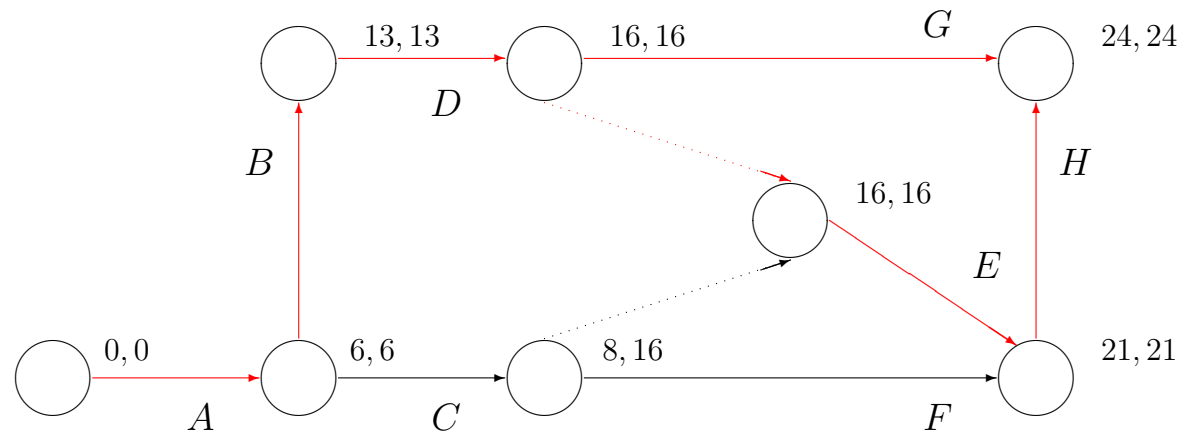
Le relazioni $C \prec E$ e $D \prec E$ non possono essere unificate come:



poichè $D \prec G$ e $C \prec F$ ma non si hanno le precedenze $C \prec G$ e $D \prec F$; inserendo due attività fittizie si ottiene:



A questo punto si può ottenere il grafo finale:



Il tempo minimo, le attività critiche e i cammini critici sono gli stessi



- Il modello PERT è in generale più complesso da realizzare (attività fittizie)
Risulta più semplice tenere conto di variazioni nella durata delle attività
- Il modello CPM è più vantaggioso se è necessario introdurre vincoli non standard (archi di costo opportuno):
 - un vincolo del tipo *l'attività j deve cominciare quando è stata completata una percentuale p dell'attività i* si esprime introducendo un arco (v_i, v_j) di costo pd_i
 - un vincolo del tipo *l'attività j deve cominciare dopo un tempo t da quando è stata completata l'attività i* si esprime introducendo un arco (v_i, v_j) di costo $d_i + t$
 - un vincolo del tipo *l'attività j deve cominciare dopo un tempo t dall'inizio del progetto* si esprime introducendo un arco (v_α, v_j) di costo t

8 Problema del flusso di costo minimo

8.1 Formulazione del problema

E' il problema più generale tra i problemi su reti

$$\begin{aligned}
 \min \quad & z = \sum_{i \in N} \sum_{j \in \omega^+(i)} c_{ij} x_{ij} \\
 \text{s.t.} \quad & \sum_{j \in \omega^+(i)} x_{ij} - \sum_{j \in \omega^-(i)} x_{ji} = b_i \quad \forall v_i \in N \quad (P1) \\
 & x_{ij} \geq l_{ij} \quad \forall a_{ij} \in A \quad (P2) \\
 & x_{ij} \leq u_{ij} \quad \forall a_{ij} \in A \quad (P3) \\
 & x_{ij} \text{ intero} \quad \forall a_{ij} \in A \quad (P4)
 \end{aligned}$$

$$\bullet \sum_{i \in N} b_i = 0$$

L'algoritmo più generale è il simplesso su reti

L'algoritmo *Out of Kilter* si basa sulle condizioni di ottimalità del II Teorema della dualità

Problema duale (a meno del vincolo P4):

$$\begin{aligned}
 \max \quad z &= \sum_{i \in N} b_i y_i + \sum_{i \in N} \sum_{j \in \omega^+(i)} l_{ij} s_{ij} + \sum_{i \in N} \sum_{j \in \omega^+(i)} u_{ij} t_{ij} \\
 s.t. \quad y_i - y_j + s_{ij} - t_{ij} &= c_{ij} & \forall a_{ij} \in A & \quad (D1) \\
 s_{ij} &\geq 0 & \forall a_{ij} \in A & \quad (D2) \\
 t_{ij} &\leq 0 & \forall a_{ij} \in A & \quad (D3)
 \end{aligned}$$

- I vincoli duali (D1) sono di uguaglianza in quanto le variabili primali x_{ij} sono libere
- Le variabili duali y_i associate ai nodi, dette *potenziali dei nodi*, sono libere in quanto i vincoli primali (P1) sono di uguaglianza

Condizioni di ottimalità del II Teorema della dualità:

$$\left(- \sum_{j \in \omega^+(i)} x_{ij} + \sum_{j \in \omega^-(i)} x_{ji} + b_i \right) y_i = 0 \quad \forall v_i \in N \quad (C1)$$

$$(x_{ij} - l_{ij}) s_{ij} = 0 \quad \forall a_{ij} \in A \quad (C2)$$

$$(-x_{ij} + u_{ij}) t_{ij} = 0 \quad \forall a_{ij} \in A \quad (C3)$$

$$(y_i - y_j + s_{ij} - t_{ij} - c_{ij}) x_{ij} = 0 \quad \forall a_{ij} \in A \quad (C4)$$

- I vincoli (P1) e (D1) sono di uguaglianza per cui le condizioni (C1) e (C4) sono soddisfatte se le variabili x, y, s e t sono ammissibili

Definizione 8.1 La quantità $c_{ij} - y_i + y_j$ è detta costo ridotto dell'arco a_{ij} ; si indica con c_{ij}^*

Il problema si riduce all'analisi di 3 casi per ogni arco:

$$\text{Caso 1 } x_{ij} = l_{ij} \quad \Rightarrow t_{ij} = 0 \quad \Rightarrow y_i - y_j + s_{ij} = c_{ij} \quad \Rightarrow c_{ij}^* \geq 0$$

$$\text{Caso 2 } l_{ij} < x_{ij} < u_{ij} \quad \Rightarrow s_{ij} = t_{ij} = 0 \quad \Rightarrow y_i - y_j = c_{ij} \quad \Rightarrow c_{ij}^* = 0$$

$$\text{Caso 3 } x_{ij} = u_{ij} \quad \Rightarrow s_{ij} = 0 \quad \Rightarrow y_i - y_j + t_{ij} = c_{ij} \quad \Rightarrow c_{ij}^* \leq 0$$

Il problema di flusso di costo minimo equivale al sistema:

$$\left\{ \begin{array}{ll} \sum_{j \in \omega^+(i)} x_{ij} - \sum_{j \in \omega^-(i)} x_{ji} = b_i & \forall v_i \in N \quad (S1) \\ l_{ij} \leq x_{ij} \leq u_{ij} & \forall a_{ij} \in A \quad (S2) \\ c_{ij}^* \geq 0 & \text{se } x_{ij} = l_{ij} \quad (S3) \\ c_{ij}^* = 0 & \text{se } l_{ij} < x_{ij} < u_{ij} \quad (S4) \\ c_{ij}^* \leq 0 & \text{se } x_{ij} = u_{ij} \quad (S5) \end{array} \right.$$

8.2 Problema di circolazione

I nodi sono tutti di attraversamento ($b_i = 0, \forall i \in N$)

Ogni problema di flusso equivale ad un problema di circolazione:

- Si aggiungono una supersorgente s^* , collegata ad ogni sorgente v_h con un arco (s^*, v_h) avente $l_{s^*h} = u_{s^*h} = b_h$ e $c_{s^*h} = 0$, e un superpozzo t^* , collegato ad ogni pozzo v_k con un arco (v_k, t^*) avente $l_{kt^*} = u_{kt^*} = -b_k$ e $c_{kt^*} = 0$
- Si inserisce un arco (t^*, s^*) , avente $l_{t^*s^*} = u_{t^*s^*} = \sum_{b_h > 0} b_h = - \sum_{b_k < 0} b_k$ e $c_{t^*s^*} = 0$ e si pone $b_{s^*} = b_{t^*} = 0$

8.3 Algoritmo Out of Kilter (Minty, 1960)

Si risolve il sistema (S1-S5) partendo da un flusso iniziale che soddisfa le equazioni (S1) e un insieme qualsiasi di valori per i potenziali

Ad ogni iterazione si modificano il flusso, conservando le equazioni (S1), o i potenziali, fino a soddisfare le equazioni (S2-S5)

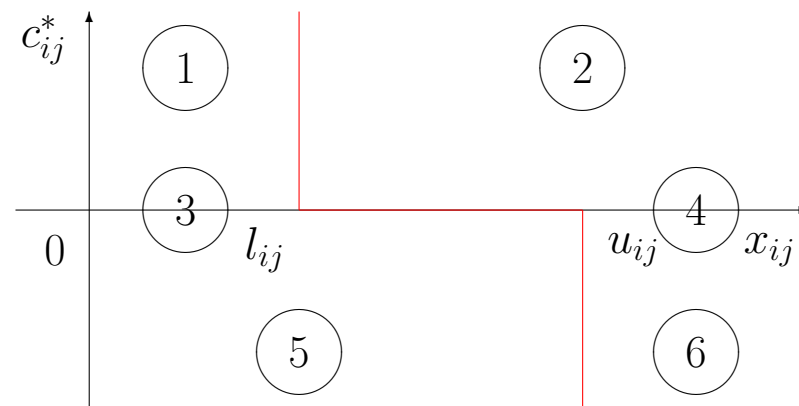
Se il problema è di circolazione il flusso nullo soddisfa le equazioni (S1)

A seconda dei valori di c_{ij}^* le equazioni (S2-S5) possono essere violate nei seguenti casi:

- | | | |
|------------------------------------|------------------------------------|------------------------------------|
| 1) $c_{ij}^* > 0, x_{ij} < l_{ij}$ | 3) $c_{ij}^* = 0, x_{ij} < l_{ij}$ | 5) $c_{ij}^* < 0, x_{ij} < u_{ij}$ |
| 2) $c_{ij}^* > 0, x_{ij} > l_{ij}$ | 4) $c_{ij}^* = 0, x_{ij} > u_{ij}$ | 6) $c_{ij}^* < 0, x_{ij} > u_{ij}$ |

- I casi 1) e 2) derivano dalla condizione (S3), i casi 3) e 4) dalla condizione (S4), i casi 5) e 6) dalla condizione (S5); le condizioni (S2) sono soddisfatte indirettamente

Kilter diagram



Definizione 8.2 Se un arco soddisfa le condizioni di ottimalità (S2-S5) è detto ammissibile (In Kilter) altrimenti è detto non ammissibile (Out of Kilter)

Per un arco (v_i, v_j) non ammissibile si possono valutare la violazione k_{ij} (Kilter number) e la massima variazione possibile del flusso Δ_{ij} per ciascuno dei 6 casi precedenti:

1)	$k_{ij} = l_{ij} - x_{ij}$	$\Delta_{ij} = l_{ij} - x_{ij}$
2)	$k_{ij} = c_{ij}^*(x_{ij} - l_{ij})$	$\Delta_{ij} = x_{ij} - l_{ij}$
3)	$k_{ij} = l_{ij} - x_{ij}$	$\Delta_{ij} = u_{ij} - x_{ij}$
4)	$k_{ij} = x_{ij} - u_{ij}$	$\Delta_{ij} = x_{ij} - l_{ij}$
5)	$k_{ij} = c_{ij}^*(x_{ij} - u_{ij})$	$\Delta_{ij} = u_{ij} - x_{ij}$
6)	$k_{ij} = x_{ij} - u_{ij}$	$\Delta_{ij} = x_{ij} - u_{ij}$

La violazione è positiva per gli archi non ammissibili e si pone nulla per gli archi ammissibili

L'algoritmo cerca di modificare il flusso sugli archi e se questo non è possibile modifica i potenziali

L'obiettivo è ridurre il valore della violazione globale, data dalla somma dei Kilter number, in modo da soddisfare le condizioni di ottimalità o determinare l'impossibilità del problema

Modifica del flusso

Dato (v_i, v_j) non ammissibile, si cerca di aumentare il flusso nei casi 1), 3), 5) o di diminuirlo nei casi 2), 4), 6). Nei primi tre casi si etichetta v_j con "+i", invece nei secondi tre si etichetta v_i con "-j". Si cerca di etichettare l'altro nodo di (v_i, v_j) secondo la seguente procedura:

- se v_k è etichettato e v_h non è etichettato si etichetta v_h con "+k" se esiste l'arco (v_k, v_h) e se $c_{kh}^* > 0$ e $x_{kh} < l_{kh}$ oppure $c_{kh}^* \leq 0$ e $x_{kh} < u_{kh}$ (nel caso in cui $c_{kh}^* = 0$ e $x_{kh} < u_{kh}$ si pone $\Delta_{kh} = u_{kh} - x_{kh}$);
- se v_k è etichettato e v_h non è etichettato si etichetta v_h con "-k" se esiste l'arco (v_h, v_k) e se $c_{hk}^* < 0$ e $x_{hk} > u_{hk}$ oppure $c_{hk}^* \geq 0$ e $x_{hk} > l_{hk}$ (nel caso in cui $c_{hk}^* = 0$ e $x_{hk} > l_{hk}$ si pone $\Delta_{hk} = x_{hk} - l_{hk}$);

Se si etichetta l'altro nodo di (v_i, v_j) si modifica il flusso sul ciclo C di $\Delta = \min \{\Delta_{ij}\}$

Si assegna a C l'orientamento indotto da (v_i, v_j) e si pone:

$$x_{hk} = \begin{cases} x_{hk} & \text{se } a_{ij} \notin C \\ x_{hk} + \Delta & \text{se } a_{hk} \in C \text{ secondo l'orientamento} \\ x_{hk} - \Delta & \text{se } a_{hk} \in C \text{ non secondo l'orientamento} \end{cases}$$

- Le regole di etichettatura impediscono che i due estremi dell'arco (v_i, v_j) possano etichettarsi reciprocamente, creando un ciclo non elementare
- Come per l'algoritmo di Ford e Fulkerson, Δ può essere inserito come seconda etichetta e inoltre esistono differenti strategie di etichettatura

Modifica dei potenziali

Se non si etichetta l'altro nodo di (v_i, v_j) si cerca di modificare i potenziali di $c^0 = \min \{c_1, -c_2\}$, dove:

$$c_1 = \min \{c_{hk}^* > 0 \text{ s.t. } v_h \text{ etichettato e } v_k \text{ non etichettato con } x_{hk} \leq u_{hk}\}$$

$$c_2 = \max \{c_{hk}^* < 0 \text{ s.t. } v_h \text{ non etichettato e } v_k \text{ etichettato con } x_{hk} \geq l_{hk}\}$$

I nuovi potenziali sono:

$$y_h = y_h + c^0 \text{ se } v_h \text{ è etichettato}$$

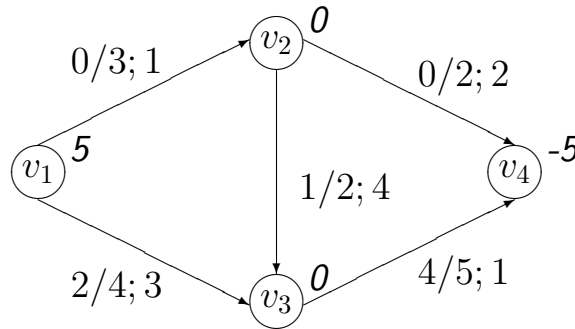
- Dopo aver modificato i potenziali o si ottiene un arco ammissibile in più oppure è possibile etichettare almeno un nuovo nodo. Infatti:
 - se $c^0 = c_1 = c_{h^*k^*}^*$ il nuovo valore di $c_{h^*k^*}^*$ è nullo, per cui se $x_{h^*k^*} = u_{h^*k^*}$ l'arco (v_{h^*}, v_{k^*}) diventa ammissibile, mentre se $x_{h^*k^*} < u_{h^*k^*}$ il nodo $v_{k^*}^*$ può essere etichettato
 - se $c^0 = -c_2 = -c_{h^*k^*}^*$ il nuovo valore di $c_{h^*k^*}^*$ è nullo, per cui se $x_{h^*k^*} = l_{h^*k^*}$ l'arco (v_{h^*}, v_{k^*}) diventa ammissibile, mentre se $x_{h^*k^*} > l_{h^*k^*}$ il nodo $v_{h^*}^*$ può essere etichettato

Algoritmo

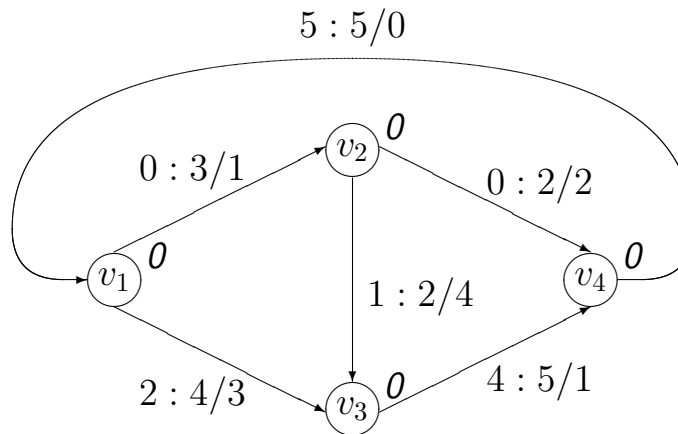
- a) porre $x = 0$ e $y = 0$;
 - b) se esiste un arco non ammissibile andare a c);
altrimenti STOP (flusso di costo minimo);
 - c) cercare un ciclo su cui modificare il flusso e andare a d);
 - d) se si determina un ciclo modificare il flusso e tornare a b);
altrimenti determinare c^0 e andare a e);
 - e) se $c^0 < +\infty$ modificare i potenziali e tornare a c);
altrimenti STOP (non ci sono soluzioni ammissibili);
- La presenza dei vincoli di capacità sugli archi esclude la possibilità che la funzione obiettivo sia inferiormente illimitata
 - L'algoritmo converge perchè ad ogni iterazione almeno un Kilter number si riduce ed essendo i dati interi si ha la terminazione in un numero finito di iterazioni, salvo nei casi degeneri

Esempio 8.1 (Algoritmo Out of Kilter)

Notazione per gli archi: $l/u; c$

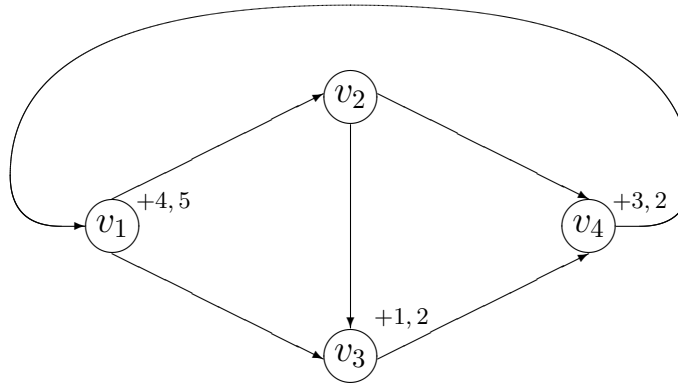


Problema di circolazione associato:



<i>flussi</i>	$x_{12} = 0$	$x_{13} = 0$	$x_{23} = 0$	$x_{24} = 0$	$x_{34} = 0$	$x_{41} = 0$
<i>potenziali</i>	$y_1 = 0$	$y_2 = 0$	$y_3 = 0$	$y_4 = 0$		
<i>costi ridotti</i>	$c_{12}^* = 1$	$c_{13}^* = 3$	$c_{23}^* = 4$	$c_{24}^* = 2$	$c_{34}^* = 1$	$c_{41}^* = 0$
<i>Kilter number</i>	$k_{12} = 0$	$k_{13} = 2$	$k_{23} = 1$	$k_{24} = 0$	$k_{34} = 4$	$k_{41} = 5$

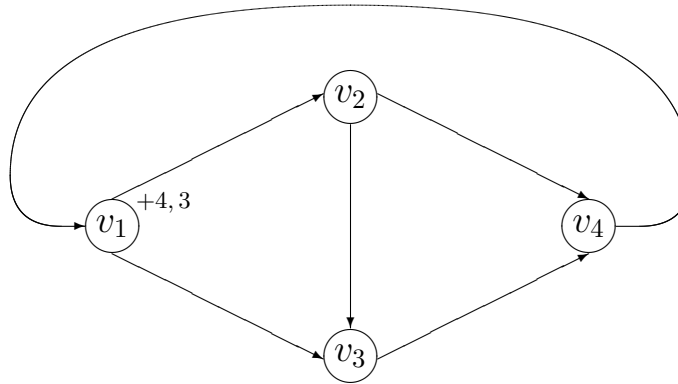
Scegliendo (v_4, v_1) con violazione massima:



Si modificano i flussi sul ciclo $v_4 - v_1 - v_3 - v_4$:

<i>flussi</i>	$x_{12} = 0$	$x_{13} = 2$	$x_{23} = 0$	$x_{24} = 0$	$x_{34} = 2$	$x_{41} = 2$
<i>potenziali</i>	$y_1 = 0$	$y_2 = 0$	$y_3 = 0$	$y_4 = 0$		
<i>costi ridotti</i>	$c_{12}^* = 1$	$c_{13}^* = 3$	$c_{23}^* = 4$	$c_{24}^* = 2$	$c_{34}^* = 1$	$c_{41}^* = 0$
<i>Kilter number</i>	$k_{12} = 0$	$k_{13} = 0$	$k_{23} = 1$	$k_{24} = 0$	$k_{34} = 2$	$k_{41} = 3$

Scegliendo ancora (v_4, v_1) con violazione massima:



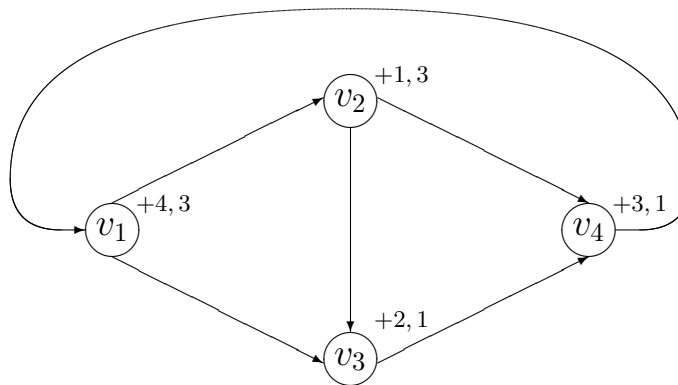
Non essendo stato determinato un ciclo si modificano i potenziali:

$$c_1 = \min \{c_{12}^*, c_{13}^*\} = 1; c_2 = \max \emptyset = -\infty; c^0 = \min \{1, +\infty\} = 1$$

$$\text{potenziali} \quad y_1 = 1 \quad y_2 = 0 \quad y_3 = 0 \quad y_4 = 0$$

$$\text{costi ridotti} \quad c_{12}^* = 0 \quad c_{13}^* = 2 \quad c_{23}^* = 4 \quad c_{24}^* = 2 \quad c_{34}^* = 1 \quad c_{41}^* = 1$$

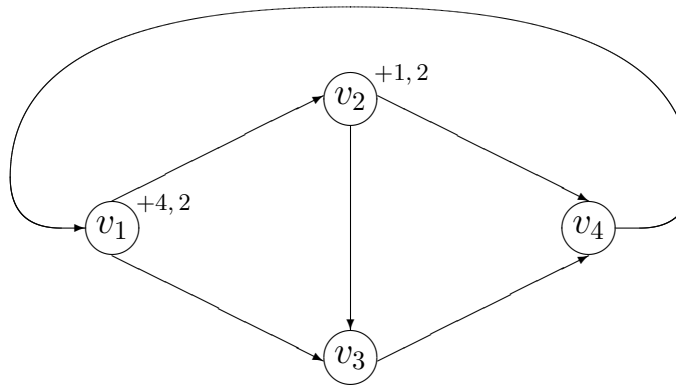
(v_4, v_1) rimane non ammissibile e si prosegue ad etichettare:



Si modificano i flussi sul ciclo $v_4 - v_1 - v_2 - v_3 - v_4$:

flussi	$x_{12} = 1$	$x_{13} = 2$	$x_{23} = 1$	$x_{24} = 0$	$x_{34} = 3$	$x_{41} = 3$
potenziali	$y_1 = 1$	$y_2 = 0$	$y_3 = 0$	$y_4 = 0$		
costi ridotti	$c_{12}^* = 0$	$c_{13}^* = 2$	$c_{23}^* = 4$	$c_{24}^* = 2$	$c_{34}^* = 1$	$c_{41}^* = 1$
Kilter number	$k_{12} = 0$	$k_{13} = 0$	$k_{23} = 0$	$k_{24} = 0$	$k_{34} = 1$	$k_{41} = 2$

Scegliendo ancora (v_4, v_1) con violazione massima:



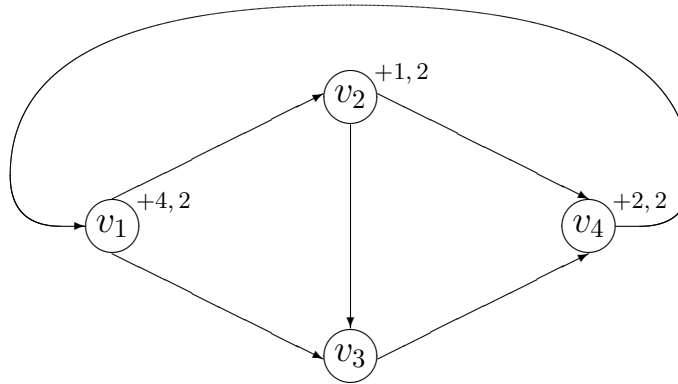
Non essendo stato determinato un ciclo si modificano i potenziali:

$$c_1 = \min \{c_{13}^*, c_{23}^*, c_{24}^*\} = 2; c_2 = \max \emptyset = -\infty; c^0 = \min \{2, +\infty\} = 2$$

$$\text{potenziali} \quad y_1 = 3 \quad y_2 = 2 \quad y_3 = 0 \quad y_4 = 0$$

$$\text{costi ridotti} \quad c_{12}^* = 0 \quad c_{13}^* = 0 \quad c_{23}^* = 2 \quad c_{24}^* = 0 \quad c_{34}^* = 1 \quad c_{41}^* = 3$$

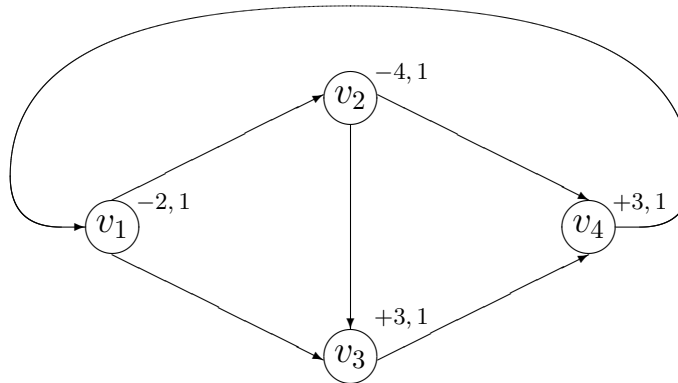
(v_4, v_1) rimane non ammissibile e si prosegue ad etichettare:



Si modificano i flussi sul ciclo $v_4 - v_1 - v_2 - v_4$:

flussi	$x_{12} = 3$	$x_{13} = 2$	$x_{23} = 1$	$x_{24} = 2$	$x_{34} = 3$	$x_{41} = 5$
potenziali	$y_1 = 3$	$y_2 = 2$	$y_3 = 0$	$y_4 = 0$		
costi ridotti	$c_{12}^* = 0$	$c_{13}^* = 0$	$c_{23}^* = 2$	$c_{24}^* = 0$	$c_{34}^* = 1$	$c_{41}^* = 3$
Kilter number	$k_{12} = 0$	$k_{13} = 0$	$k_{23} = 0$	$k_{24} = 0$	$k_{34} = 1$	$k_{41} = 0$

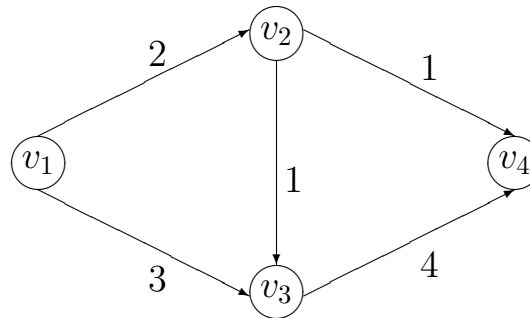
Scegliendo l'arco (v_3, v_4) con violazione massima:



Si modificano i flussi sul ciclo $v_3 - v_4 - v_2 - v_1 - v_3$:

flussi	$x_{12} = 2$	$x_{13} = 3$	$x_{23} = 1$	$x_{24} = 1$	$x_{34} = 4$	$x_{41} = 5$
potenziali	$y_1 = 3$	$y_2 = 2$	$y_3 = 0$	$y_4 = 0$		
costi ridotti	$c_{12}^* = 0$	$c_{13}^* = 0$	$c_{23}^* = 2$	$c_{24}^* = 0$	$c_{34}^* = 1$	$c_{41}^* = 3$
Kilter number	$k_{12} = 0$	$k_{13} = 0$	$k_{23} = 0$	$k_{24} = 0$	$k_{34} = 0$	$k_{41} = 0$

Le violazioni sono tutte nulle, quindi il flusso di costo minimo è:



Il costo è $2 \cdot 1 + 3 \cdot 3 + 1 \cdot 4 + 1 \cdot 2 + 4 \cdot 1 = 21$

