Modelli Matematici per la Logistica

Vito Fragnelli

A.A. 2010-2011

Capitolo 1

Programmazione lineare

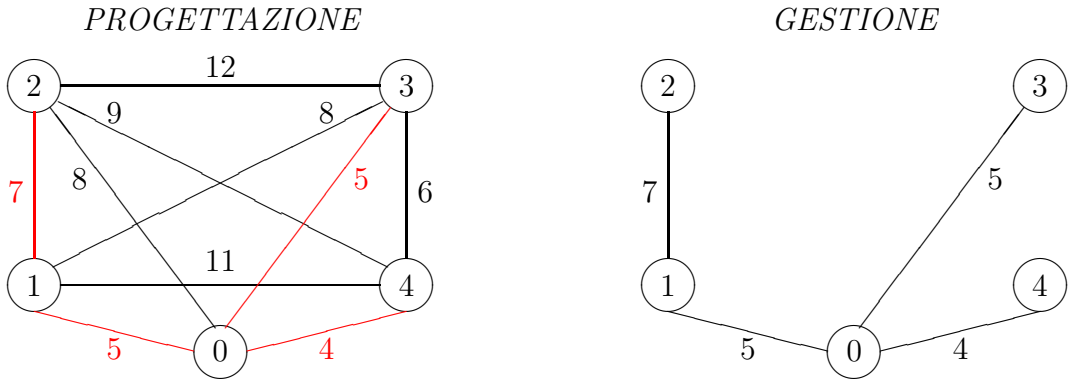
1.1 Modelli matematici

Lo studio della logistica utilizza opportuni modelli matematici. Possiamo distinguere due gruppi di modelli, anche se la separazione 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. Il seguente esempio può chiarire la differenza tra progettazione e gestione.

Esempio 1.1.1 (Connessione - Progettazione e gestione) *Un'azienda ha 4 impianti (1, 2, 3, 4) che devono essere collegati ad una centrale elettrica (0):*



Nel primo caso il problema consiste nel determinare le connessioni di costo minimo, mentre nel secondo caso il problema è come ripartire i costi. ♢

1.2 Problema di programmazione matematica

Rappresentare un problema tramite un modello di programmazione matematica consiste nell'associare agli elementi da determinare delle variabili, dette *variabili decisionali*, e alle relazioni intercorrenti tra esse delle funzioni matematiche, una delle quali, che rappresenta il valore del problema, è detta *funzione obiettivo* e le altre, che definiscono quali valori delle variabili sono ammissibili, sono dette *vincoli*.

La forma più generale di un problema di programmazione matematica è la seguente:

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

oppure:

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

dove x rappresenta un vettore di $\mathbb{R}^n$ le cui componenti corrispondono alle variabili decisionali, la funzione $f : \mathbb{R}^n \to \mathbb{R}$ rappresenta la funzione obiettivo e le relazioni $g_i(x) \leq 0, g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$ rappresentano i vincoli del problema.

Tutti i problemi possono essere scritti in questa forma in quanto si ha:

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

L'insieme dei valori di x per cui i vincoli sono soddisfatti è detto *insieme delle soluzioni ammissibili* o *insieme ammissibile* o *regione ammissibile* e si indica con S_a. L'insieme ammissibile può essere vuoto, limitato o illimitato.

Soluzioni ottimali

L'insieme dei valori di x per cui i vincoli sono soddisfatti e la funzione obiettivo assume il valore massimo è detto *insieme delle soluzioni ottimali* o *insieme ottimale* o *regione ottimale* e si indica con S_{ott}. L'insieme ottimale può essere vuoto, limitato o illimitato.

Risolvere un problema di programmazione matematica vuol dire determinare sia una soluzione ottimale (punto di massimo) che il valore della funzione obiettivo (valore massimo); a seconda delle situazioni può essere più rilevante l'una o l'altra. In particolare il punto di massimo è importante se si ricerca una strategia ottimale, mentre il valore massimo acquista maggiore importanza nel confronto tra differenti strategie.

1.3 Problema di programmazione lineare

Un problema di programmazione lineare o problema lineare è un problema di programmazione matematica in cui la funzione obiettivo e i vincoli sono lineari.

Definizione 1.3.1 *Un problema lineare si dice in forma canonica se è scritto nella forma:*

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

dove c e x sono vettori di $\mathbb{R}^n$, A è una matrice $m \times n$ e b è un vettore di $\mathbb{R}^m$.
c rappresenta il gradiente della funzione z ed è detto vettore dei costi, A è detta matrice dei vincoli e b è detto vettore dei termini noti.

Osservazione 1.3.1

- *Per la precisione i vincoli sono affini, per la presenza del termine non nullo.*

Se la forma canonica verifica la relazione $b \geq 0$ il problema ammette la *forma normale*. Non tutti i problemi possono essere scritti in forma normale, mentre tutti i problemi possono essere scritti in forma canonica in quanto si ha:

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 $\Leftrightarrow x = x' - x''$; $x' \geq 0$; $x'' \geq 0$

Soluzioni ammissibili

S_a è un insieme convesso in quanto intersezione dei semispazi $(Ax)_j \leq b_j, j = 1, \ldots, m$; $x_i = 0, i = 1, \ldots, n$ e può essere vuoto, limitato (*poliedro convesso*) oppure illimitato (*troncone convesso*).

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

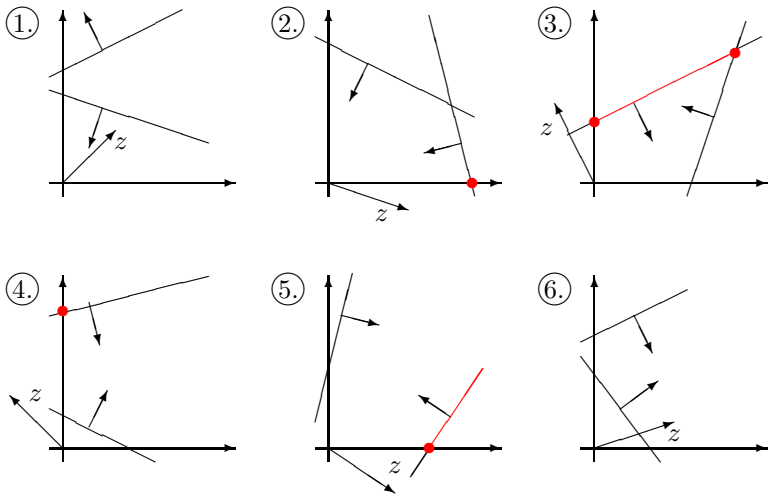
Se un punto di S_a è intersezione unica di almeno n iperpiani generatori si dice *vertice di* S_a; se è intersezione unica di più di n iperpiani generatori si dice *vertice degenere*.

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; se uno spigolo è limitato i due vertici estremi si dicono *adiacenti*.

Soluzioni ottimali

S_{ott} è un insieme convesso ed è contenuto nella frontiera di S_a, salvo che la funzione z sia costante, e può essere vuoto, costituito da un unico punto o da infiniti punti, in quanto se contiene due punti distinti contiene tutto il segmento che ha per estremi i due punti; se è costituito da infiniti punti può essere limitato, cioè è un poliedro convesso, oppure illimitato, cioè è un 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.



Nella figura precedente il vettore z rappresenta il gradiente della funzione obiettivo, gli altri vettori rappresentano i gradienti dei vincoli e sono stati omessi quelli relativi ai vincoli di non negatività; sono rappresentati i seguenti casi:

1. $S_{ott} = \emptyset$ in quanto $S_a = \emptyset$;

2. S_{ott} costituito da un unico punto con S_a limitato;

3. S_{ott} costituito da un insieme limitato di infiniti punti con S_a limitato;

4. S_{ott} costituito da un unico punto con S_a illimitato;

5. S_{ott} costituito da un insieme illimitato di infiniti punti con S_a illimitato;

6. $S_{ott} = \emptyset$ in quanto la funzione obiettivo è superiormente illimitata con S_a illimitato.

1.4 Teorema fondamentale

Teorema 1.4.1 *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 $x_1, \ldots, x_p$ i vertici del poliedro K.
Sia x^* una soluzione ottimale del problema lineare che può essere scritta come combinazione convessa dei vertici del poliedro K:

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

Indicando con z_i il valore della funzione obiettivo nel vertice x_i e con z^* il valore della funzione obiettivo nel punto x^*, per la linearità di z si ha:

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

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

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

Caso 2 Il troncone convesso 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, cioè:

$$T = K_1 + K_0$$

Siano $u_1, \ldots, u_p$ i vertici del poliedro K_1 e $v_1, \ldots, v_q$ i generatori del cono 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:

$$\begin{array}{rcl} u & = & \sum_{i=1,\ldots,p} \lambda_i u_i \quad \text{con} \quad \sum_{i=1,\ldots,p} \lambda_i = 1,\ \lambda_i \geq 0, i = 1, ..., p \\ v & = & \sum_{j=1,\ldots,q} \mu_j v_j \quad \text{con} \quad \mu_j \geq 0, j = 1, ..., q \end{array}$$

quindi si ha:

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

Indicando con z_i il valore della funzione obiettivo nel vertice u_i, con z'_j il valore della funzione obiettivo nel punto v_j e con z^* il valore della funzione obiettivo nel punto x^* si ha per la linearità di z:

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

I valori z'_j sono tutti non positivi; infatti se esistesse un termine $z'_j > 0$ al crescere del coefficiente μ_j corrispondente anche z^* crescerebbe, pertanto si ha:

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

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

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

Osservazione 1.4.1

- *Il teorema precedente è detto fondamentale in quanto riduce la ricerca della soluzione ottimale ai soli vertici di S_a e costituisce il fondamento dell'algoritmo del simplesso.*
- *Anche se l'enunciato riportato è quello classico, si può osservare che in realtà si sottintende che il problema sia in forma canonica. Si consideri il seguente problema:*

$$\begin{array}{ll} max & z = x_1 + x_2 \\ s.t. & x_1 + x_2 \leq 1 \end{array}$$

Il problema ammette massimo, con $z^* = 1$ e $S_{ott} = \{(x_1, x_2) \in \mathbb{R}^2 | x_1 + x_2 = 1\}$ 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 $v_1, \ldots, v_n$, non necessariamente linearmente indipendenti e m vettori $a_1, \ldots, a_m$ appartenenti allo spazio generato da $v_1, \ldots, v_n$; i vettori $a_1, \ldots, a_m$ possono essere espressi come combinazione lineare dei vettori $v_1, \ldots, v_n$, cioè:

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

Se $\lambda_{ij} \neq 0$ è possibile sostituire il vettore a_i al vettore v_j nel senso che i vettori a_i e v_k, $k \neq j$ generano lo stesso spazio dei vettori $v_1, \ldots, v_n$, quindi è possibile esprimere i vettori v_j e a_h, $h \neq i$ come combinazione lineare dei vettori a_i e v_k, $k \neq j$; dall'espressione di a_i si ottiene:

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

e sostituendo si ha:

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

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

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

Usualmente i coefficienti si rappresentano con una tabella in cui ad ogni colonna corrisponde un vettore generatore e ad ogni riga un vettore espresso dalla combinazione

lineare. Si parte dalla tabella:

	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 sull'elemento λ_{ij}, cioè scambiando i vettori a_i e v_j, 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

L'algoritmo presentato da Dantzig nel 1947 è il più famoso algoritmo della programmazione matematica. Dato un problema lineare in forma canonica:

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

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

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

Le equazioni vincolari e la funzione obiettivo possono essere rappresentate dalla tabella:

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

L'intersezione dei semispazi $x_i \geq 0, i = 1, \ldots, n, u_j \geq 0, j = 1, \ldots, m$ 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 punto intersezione degli iperpiani ottenuti annullando le variabili che formano la base corrente. Tale punto è necessariamente un vertice.

L'ultima colonna rappresenta il valore delle variabili che non formano la base corrente e in particolare il coefficiente c_0 rappresenta il valore della funzione z in quel punto.

Se i coefficienti dell'ultima colonna, escluso c_0, sono tutti non negativi il punto risulta ammissibile in quanto sono verificati i vincoli $x \geq 0$, $u \geq 0$.

1.6.1 Ricerca della soluzione ottimale

L'algoritmo del simplesso cerca una soluzione ottimale spostandosi da un vertice ammissibile ad un altro vertice ammissibile adiacente, cioè formato dagli stessi iperpiani tranne uno, in modo che nel nuovo vertice la funzione z abbia un valore non peggiore. Per far questo 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	...	y_i	...	y_n	
y_{n+1}	α_{11}	...	α_{1i}	...	α_{1n}	β_1
...	...	...	...	...	...	
y_{n+j}	α_{j1}	...	α_{ji}	...	α_{jn}	β_j
...	...	...	...	...	...	
y_{n+m}	α_{m1}	...	α_{mi}	...	α_{mn}	β_m
z	γ_1	...	γ_i	...	γ_n	γ_0

Scelta della variabile uscente

Facendo cardine su α_{ji} il valore della funzione z risulta $\gamma_0 - \gamma_i \frac{\beta_j}{\alpha_{ji}}$. Poichè $-\frac{\beta_j}{\alpha_{ji}}$ è il nuovo valore di y_i che deve essere non negativo, 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 i nuovi valori delle variabili fuori base sono:

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

Le condizioni di ammissibilità $y'_i \geq 0, y'_{n+k} \geq 0, k \neq j$ forniscono:

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

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:

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

Questa relazione fornisce il criterio di scelta per la variabile entrante, una volta fissata la variabile uscente.

Osservazione 1.6.1

- *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; pertanto gli spigoli nella cui direzione la funzione z è crescente, per valori nella direzione crescente per y_i sono quelli con γ_i 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 che viene incontrato nella direzione crescente per y_i. Il punto di intersezione dell'iperpiano generatore

$y_{n+k} = 0, k = 1, \ldots, m$ 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

Se i coefficienti $\gamma_i, i = 1, \ldots, n$ sono tutti non positivi vuol dire che in tutta la regione ammissibile il valore della funzione z è non migliore di quello corrente.

Dall'ultima riga si ricava:

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

per cui per $y_i = 0, i = 1, \ldots, n$ si ha $z = \gamma_0$, mentre se qualche y_i è strettamente positiva (quindi anche per tutti i valori ammissibili) la funzione obiettivo assume un valore non superiore a γ_0.

Si può allora definire *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:

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

per cui se nella tabella in corrispondenza di un coefficiente $\gamma_i > 0$ si hanno coefficienti $\alpha_{ki} \geq 0, k = 1, \ldots, m$ vuol dire che le variabili y_{n+k} rimangono ammissibili al crescere della variabile y_i, per cui la quantità $\gamma_i y_i$ può crescere indefinitamente e la funzione z risulta superiormente illimitata.

Osservazione 1.7.1 (Regione ammissibile illimitata)

- *Indipendentemente dal valore di γ_i la condizione di avere una colonna con i coefficienti $\alpha_{ki} \geq 0, k = 1, \ldots, m$ vuol dire che la regione ammissibile ammette uno spigolo illimitato, cioè risulta essere un troncone.*

Esempio 1.7.1 *Risolvere con l'algoritmo del simplesso il seguente problema di programmazione lineare:*

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

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

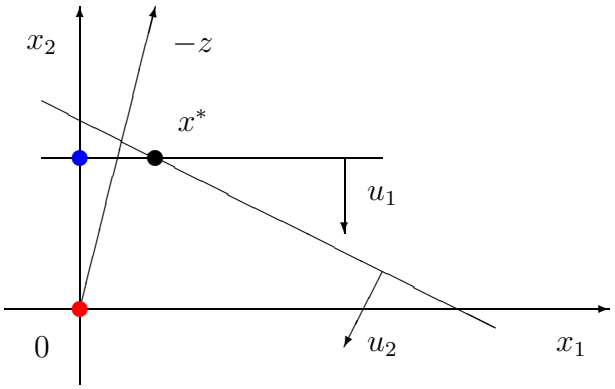
	x_1	x_2	
u_1	0	$\boxed{-1}$	2
u_2	-1	-2	5
$-z$	1	4	0

$x = (0,0), z = 0$

	x_1	u_1	
x_2	0	-1	2
u_2	$\boxed{-1}$	2	1
$-z$	1	-4	8

$x = (0,2), z = -8$

	u_2	u_1	
x_2	0	-1	3
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 ammette la forma normale la tabella iniziale associata all'origine è ammissibile, per cui è possibile applicare subito l'algoritmo del simplesso; nel caso in

cui ciò non sia vero la prima tabella non è ammissibile, pertanto è necessario determinare una tabella ammissibile, se esiste, cioè se la regione ammissibile è non vuota.

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

Poichè la tabella non è ammissibile esiste almeno un coefficiente $\beta_j < 0$; sulla riga corrispondente si cerca un coefficiente $\alpha_{ji} > 0$ e si determina così la variabile uscente y_i; per determinare la variabile entrante è necessario che il cardine sia discorde dal corrispondente termine noto, altrimenti la tabella successiva sarebbe certamente non ammissibile. Facendo cardine su un coefficiente $\alpha_{ki} < 0$ con $\beta_k > 0$ la variabile entrante è quella per cui si ha il minimo rapporto in valore assoluto $\frac{\beta_k}{\alpha_{ki}}$, in modo che tutti i termini noti non negativi restino non negativi. Facendo cardine su un coefficiente $\alpha_{ki} > 0$ con $\beta_k < 0$ la variabile entrante è quella per cui si ha il massimo rapporto in valore assoluto $\frac{\beta_k}{\alpha_{ki}}$, 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.

Osservazione 1.9.1

- *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 1.9.1 *Risolvere con l'algoritmo del simplesso il seguente problema di programmazione lineare:*

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

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

	x_1	x_2	
u_1	$\boxed{1}$	1	-1
u_2	-1	0	4
z	2	1	0

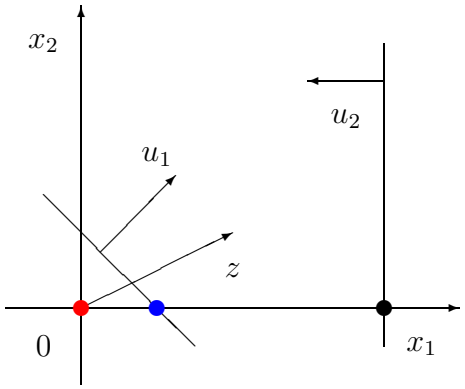
$x = (0, 0), z = 0$

	u_1	x_2	
x_1	1	-1	1
u_2	$\boxed{-1}$	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 ad un termine $\beta_j < 0$ corrispondono coefficienti $\alpha_{ji} \leq 0, i = 1, \ldots, n$ vuol dire che non esistono soluzioni ammissibili in quanto la condizione:

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

non può essere soddisfatta.

Osservazione 1.9.2 (Regione ammissibile limitata)

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

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

  si ricava:

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

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

1.10 Casi particolari

1.10.1 Soluzione degenere e ciclaggio

Se in una tabella un termine β_j è nullo vuol dire che il corrispondente iperpiano $y_{n+j} = 0$ pur non formando la base, passa ugualmente per il vertice rappresentato dalla tabella,

che risulta degenere essendo intersezione di più di n iperpiani. Se applicando l'algoritmo del simplesso la variabile y_{n+j} viene scelta come variabile entrante i valori delle variabili $y_{n+k}, k \neq j$ restano invariati e le variabili $y_i, i = 1, \ldots, n$ e y_{n+j} restano nulle. Questo vuol dire che le due tabelle rappresentano lo stesso vertice.

Talvolta può capitare che continuando ad applicare l'algoritmo del simplesso oltre a rimanere nello stesso vertice, dopo un certo numero di iterazioni si abbiano in base le stesse variabili e quindi che si presentino ciclicamente le stesse basi. Questo fenomeno detto ciclaggio può essere evitato utilizzando diversi metodi. Tra i più usati si possono ricordare la *regola di Bland* e il *metodo delle perturbazioni di Wolfe*.

1.10.2 Infinite soluzioni ottimali

Se in una tabella un termine γ_i è nullo vuol dire che facendo uscire dalla base e incrementando la corrispondente 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 (salvo nel caso di vertice degenere) in cui la funzione obiettivo assume lo stesso valore e quindi un nuovo vertice ottimale.

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

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

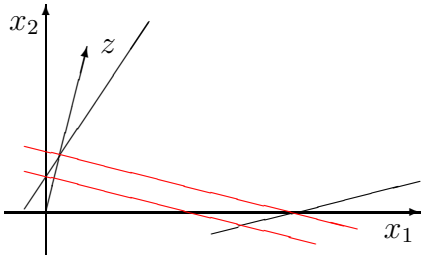
1.10.3 Problemi con vincoli di uguaglianza

Se il problema presenta vincoli di uguaglianza sono possibili differenti metodologie:

- ricondurlo alla forma canonica, introducendo due vincoli di disuguaglianza al posto ciascun vincolo di uguaglianza;
- 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 $1^T(Ax - b)$; quindi si determina una soluzione ottimale di valore nullo per il problema associato e successivamente si ritorna al problema dato. A questo punto le condizioni di ottimalità non devono considerare le variabili u relative ai vincoli di uguaglianza, nel senso che se sono in base il coefficiente nella riga della funzione obiettivo può avere anche valore positivo;
- 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; anche in questo caso le condizioni di ottimalità non devono considerare le variabili u relative ai vincoli di uguaglianza.

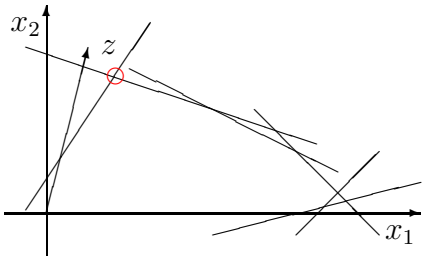
1.10.4 Scelta della variabile uscente

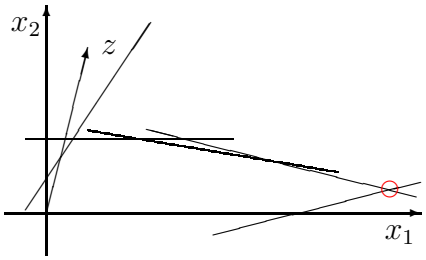
La scelta dell'elemento su cui "fare cardine" riveste un ruolo molto importante, ma non è possibile fissare regole generali, visto che la tabella rappresenta una situazione "locale"



Componente maggiore del gradiente o incremento maggiore della funzione obiettivo?

Dipende ...





Capitolo 2

Dualità

2.1 Problema duale

Sia dato il problema lineare:

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

È sempre possibile definire un problema ad esso associato della forma:

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

Il problema associato prende il nome di *problema duale* e il problema dato può essere indicato come *problema primale*.

Osservazione 2.1.1

- *I due problemi sono definiti in due spazi differenti. In particolare se A è una matrice $m \times n$ il vettore x ha n componenti, cioè il problema è in $\mathbb{R}^n$, mentre il vettore u ha m componenti, cioè il problema è 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:

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

il cui duale è:

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

che può essere riscritto come:

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

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:

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

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:

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

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:

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

le corrispondenti tabelle iniziali sono:

	x^T	
u	$-A$	b
z	c^T	0

	u^T	
x	A^T	$-c$
$-w$	$-b^T$	0

cioè le due tabelle sono, per la parte numerica, una la trasposta cambiata di segno dell'altra; applicando il metodo del simplesso alla prima tabella, si supponga di far cardine sull'elemento posto nella colonna i e nella riga j, cioè $-(A)_{ji}$; se nella seconda tabella si fa cardine sull'elemento posto nella colonna j e nella riga i, cioè $(A^T)_{ij}$, che coincide con il cardine scelto in precedenza, a meno del segno, le tabelle successive per i due problemi mantengono la relazione di essere una la trasposta cambiata di segno dell'altra, come si può verificare numericamente, e così di seguito fino alla tabella ottimale di uno dei due problemi. Questa tabella essendo ottimale ha l'ultima riga composta di termini non positivi, tranne al più l'ultimo elemento e l'ultima colonna composta di termini non negativi, tranne al più l'ultimo elemento, ma anche la sua trasposta cambiata di segno ha l'ultima riga composta di termini non positivi, tranne al più l'ultimo elemento e l'ultima colonna composta di termini non negativi, tranne al più l'ultimo elemento, cioè è anch'essa ottimale, per cui risulta dimostrato che se un problema ha soluzioni ottimali ne ha anche l'altro.
Dalla tabella ottimale del problema primale si ha che il valore della funzione obiettivo è:

$$max \quad z = \gamma_0$$

e di conseguenza dalla tabella ottimale del problema duale si ha:

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

cioè:

$$min \quad w = \gamma_0$$

per cui risulta dimostrato che $max \quad z = min \quad w$. ♣

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 rappresenterebbe 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. *II 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 $u^{*T} A x^*$ 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. ♣

Osservazione 2.2.1

- *Le relazioni del II teorema della dualità vengono chiamate anche condizioni di complementarietà. La prima esprime che se un vincolo è soddisfatto come disuguaglianza stretta la corrispondente variabile duale deve essere nulla, la seconda che se una variabile è non nulla il corrispondente vincolo duale deve essere soddisfatto come uguaglianza.*

- *Un vincolo di uno dei due problemi può essere soddisfatto come uguaglianza e la corrispondente variabile dell'altro problema può essere nulla.*

Osservazione 2.2.2

- *Dalla tabella ottimale di un problema, ricordando il I Teorema della dualità, si può leggere la soluzione del problema duale; infatti si ha:*

$$\begin{aligned}w^* &= z^*\\ u_i &= 0 \qquad \textit{se la variabile primale } u_i \textit{ è fuori base}\\ u_i &= -\gamma_i \qquad \textit{se la variabile primale } u_i \textit{ è in base}\end{aligned}$$

2.3 Interpretazione economica del problema duale

Si consideri il problema di produzione:

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

dove b rappresenta il vettore delle risorse, x il vettore dei beni prodotti, A la matrice tecnologica che esprime le unità di risorsa necessarie per unità di bene prodotto e c il vettore dei valori unitari dei beni prodotti, allora la funzione obiettivo esprime il valore complessivo dei beni prodotti, di cui si cerca il massimo e i vincoli indicano che per produrre i vari beni non si possono utilizzare più risorse di quelle disponibili e che le quantità di beni prodotti devono essere non negative.

Dimensionalmente si ha:

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

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

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

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

Alle variabili duali si assegna il significato di costo-ombra delle risorse, cioè di valore di una ulteriore unità di risorsa in quel processo produttivo; la funzione obiettivo rappresenta il costo-ombra globale delle risorse o valore-ombra del processo produttivo, di cui si cerca il minimo e i vincoli indicano che i costi-ombra unitari dei beni prodotti non possono essere inferiori ai valori corrispondenti e che i costi-ombra non possono essere negativi.

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, cioè con quella data matrice tecnologica e quelle date risorse.

Non si può affermare che le variabili duali rappresentano il valore unitario delle risorse perchè questo è un dato e non una variabile, inoltre non avrebbe senso che alcune risorse abbiano valore nullo.

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

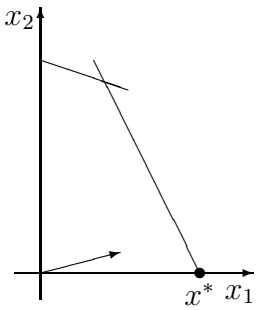
Se una risorsa non è completamente utilizzata non vi è nessun vantaggio ad averne una maggiore disponibilità, altrimenti il valore dei beni prodotti può essere incrementato disponendo di una ulteriore unità di risorsa (variabile duale non nulla) o può restare invariato

(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)

Una ulteriore interpretazione del problema duale del problema di produzione può essere la seguente. Si supponga che una ditta decida di voler acquistare tutte le risorse e debba conseguentemente determinare il vettore dei prezzi u da offrire per ogni unità di risorsa in modo da ottenere il miglior risultato possibile; la ditta cercherà di minimizzare la spesa di acquisto ($b^T u$), facendo in modo che il possessore delle risorse sia "interessato" a vendere le sue risorse, cioè stabilendo i prezzi in modo tale che il valore che pagherebbe per la quantità delle risorse necessarie a produrre una unità di bene sia non inferiore al suo valore unitario ($A^T u \geq c$) e fissando dei prezzi non negativi ($u \geq 0$).

Esempio 2.3.1 (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. Il problema può essere rappresentato come segue:

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

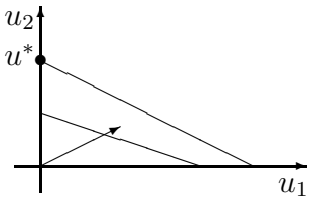


La soluzione è:

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

Il problema duale può essere rappresentato come segue:

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



e la soluzione è:

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

Si può osservare che $u_1^* = 0$ in accordo col fatto che la risorsa R_1 non è completamente utilizzata; si noti anche che la coppia di prezzi unitari delle risorse $u = (1, 0)$ porterebbe allo stesso valore di w, ma in questo caso si potrebbero vendere solo 9 unità di R_1, otte-

nendo 9, e utilizzare le risorse trattenute per produrre tre unità di B_1 con un valore 12, migliorando il risultato. ♢

Esempio 2.3.2 (Significato delle variabili duali) *Si consideri il caso in cui $u_i^* \neq 0$, il che equivale a $\sum_{j=1,\dots,n} a_{ij}x_j^* = b_i$.*
Incrementando di una unità la risorsa i, cioè ponendo $b_i^0 = b_i + 1$ si ha:

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

e quindi $z^{0*} = z^* + u_i^*$, cioè il valore dei beni prodotti aumenta di u_i^*. ♢

Esempio 2.3.3 (Significato dei vincoli duali) *Se $\sum_{j=1,\dots,n} a_{ij}x_j^* = b_i$ è l'unico vincolo soddisfatto per uguaglianza, si ha:*

$$u_k^* = 0, \quad k \neq 1$$

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

Esempio 2.3.4 (Paradosso dell'anellino) *Un artigiano stravagante fabbrica anellini di plastica con diamanti autentici. Egli 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. Volendo massimizzare il profitto si può risolvere il problema lineare seguente in cui x rappresenta il numero di anellini con diamante prodotti:*

max	$z = 99x$	*profitto*
$s.t.$	$x \leq 1$	*vincolo sugli anellini*
	$x \leq 2$	*vincolo sui diamanti*
	$x \geq 0$	

Risolvendo si ha:

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

	u_a	
x	-1	1
u_d	1	1
z	-99	99

La soluzione primale è data da $x^* = 1, z^* = 99$ (si produce un anellino con un profitto di 99 dollari); la soluzione duale è data da $u_a^* = 99, u_d^* = 0, w^* = 99$.

L'interpretazione della soluzione duale dice che 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

Se il problema dato non è in forma canonica, si può scrivere il corispondente duale, senza ricondursi alla forma canonica; le seguenti tabelle permettono di determinare il duale direttamente:

Problema primale di massimo	Problema duale di minimo
coefficienti della funzione obiettivo	termini noti dei vincoli
termini noti dei vincoli	coefficienti della funzione obiettivo
colonna j dei coefficienti	riga j dei coefficienti
riga i dei coefficienti	colonna i dei coefficienti
$x_j \geq 0$	$\sum_{i=1,\dots,m} a_{ij}u_i \geq c_j$
$x_j \leq 0$	$\sum_{i=1,\dots,m} a_{ij}u_i \leq c_j$
x_j libera	$\sum_{i=1,\dots,m} a_{ij}u_i = c_j$
$\sum_{j=1,\dots,n} a_{ij}x_j \leq b_i$	$u_i \geq 0$
$\sum_{j=1,\dots,n} a_{ij}x_j \geq b_i$	$u_i \leq 0$
$\sum_{j=1,\dots,n} a_{ij}x_j = b_i$	u_i libera

Problema primale di minimo	Problema duale di massimo
coefficienti della funzione obiettivo	termini noti dei vincoli
termini noti dei vincoli	coefficienti della funzione obiettivo
colonna j dei coefficienti	riga j dei coefficienti
riga i dei coefficienti	colonna i dei coefficienti
$x_j \geq 0$	$\sum_{i=1,\dots,m} a_{ij}u_i \leq c_j$
$x_j \leq 0$	$\sum_{i=1,\dots,m} a_{ij}u_i \geq c_j$
x_j libera	$\sum_{i=1,\dots,m} a_{ij}u_i = c_j$
$\sum_{j=1,\dots,n} a_{ij}x_j \leq b_i$	$u_i \leq 0$
$\sum_{j=1,\dots,n} a_{ij}x_j \geq b_i$	$u_i \geq 0$
$\sum_{j=1,\dots,n} a_{ij}x_j = b_i$	u_i libera

Capitolo 3

Programmazione lineare a numeri interi

3.1 Problemi lineari interi

Dato il problema lineare ordinario (PLO):

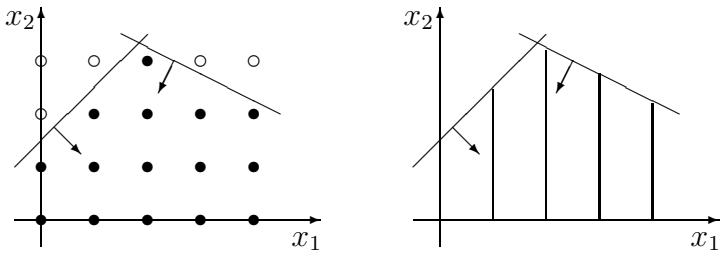
$$\begin{aligned} max \quad & z = c^T x \\ s.t. \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

aggiungendo la condizione di integrità:

$$x_i \in N, i \in I$$

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

Se la condizione di integrità vale per tutte le variabili il problema si dice puro, se vale per solo alcune variabili il problema si dice misto.



Proprietà generali

- La regione ammissibile del PLI, S'_a, è un sottoinsieme della regione ammissibile del PLO, in quanto è presente un vincolo in più, per cui si ha:

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

quindi se il PLO ha la regione ammissibile vuota ce l'ha anche il PLI.

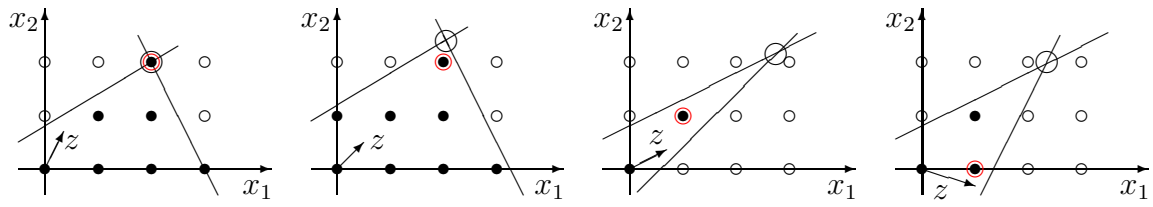
- La funzione obiettivo z' del PLI coincide con quella del PLO nei punti in cui entrambe sono definite, per cui essendo S'_a un sottoinsieme di S_a, si ha:

$$sup \quad z' \leq sup \quad z$$

quindi se il PLI ha la funzione obiettivo superiormente illimitata ce l'ha anche il PLO.

La soluzione di un PLI può risultare complessa poichè non esiste alcuna relazione con l'insieme delle soluzioni ottimali del PLO, pur essendo i problemi molto simili.

La figura seguente mostra che la soluzione ottimale del PLI può coincidere con quella del PLO, può essere ottenuta arrotondando quella del PLO, può essere il punto ammissibile a coordinate intere più vicino alla soluzione ottimale del PLO oppure le soluzioni ottimali del PLI e del PLO sono totalmente non correlate.



Si possono distinguere le seguenti classi di metodi risolutivi:
Metodi approssimati

- Arrotondamento o troncamento della soluzione del PLO
  Non garantisce nè l'ottimailità, nè l'ammissibilità.
- Punto ammissibilie 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

I metodi di questa classe si basano sull'idea di suddividere il problema assegnato in sotto-problemi più semplici da risolvere e utilizzare le soluzioni di questi per ricavare la soluzione del problema dato.

A tal fine si determina una limitazione superiore o inferiore della funzione obiettivo del problema, a seconda che si tratti di un problema di massimo o di minimo, quindi si suddivide il problema in più sottoproblemi per ognuno dei quali si determina una nuova limitazione; si prosegue quindi suddividendo ulteriormente un problema non ancora risolto fino a trovare una soluzione ammissibile del problema dato in cui la funzione obiettivo abbia un valore non peggiore delle limitazioni dei problemi ancora pendenti.

I metodi branch and bound si basano sui concetti di 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:

$$S_a(P) \subset S_a(P')$$
$$z(P')(x) \geq z(P)(x) \qquad x \in S_a(P)$$

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 $P_1, \ldots, P_n$ se $S_a(P_1), \ldots, S_a(P_n)$ costituiscono una partizione di $S_a(P)$, cioè:

$$S_a(P) = \bigcup_{i=1,\ldots,n} S_a(P_i)$$
$$S_a(P_i) \cap S_a(P_j) = \emptyset \qquad i \neq j$$

La funzione obiettivo z rimane la stessa per tutti i sottoproblemi, cioè:

$$z(P_i) = z(P) \qquad i = 1, \ldots, n$$

In generale la separazione si effettua considerando una variabile alla volta e separando P, a seconda dei valori interi ammissibili di quella variabile, in due o più sottoproblemi.

Osservazione 3.2.1

- *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:

- $E1 \quad S_a(P') = \emptyset$
- $E2 \quad z^*(P') \leq z^*$ dove z^* è la soluzione
- $E3 \quad X^*(P') \in S_a(P)$

Osservazione 3.2.2

- *Spesso nei metodi branch and bound si utilizza una stima $z_s(P') \geq z^*(P')$, per cui 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 quindi richieda troppo tempo per risolvere il sottoproblema.*
- *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

- Se si verifica $E1$ vuol dire che $S_a(P) = \emptyset$.
- Se si verifica $E2$ vuol dire che $z^*(P) = z^*$.
- Se si verifica $E3$ e $z(P')(x^*(P')) = z(P)(x^*(P'))$ vuol dire che $x^*(P') = x^*(P)$, cioè P è stato risolto.

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 $z^*(P') = z^*$ (oppure $z_s(P') = z^*$) allora $z^*(P) = z^*$, eliminare il problema P per il criterio $E2$ e tornare a b);

f) se $x^*(P\prime) \in S_a(P)$ 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 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;

Osservazione 3.2.3

- *Quello presentato è l'algoritmo base, nel senso che di volta in volta è necessario precisare i criteri di rilassamento, di valutazione e di separazione.*

Gestione della lista

La gestione della lista dei problemi da risolvere può essere fatta secondo diverse strategie, perseguendo obiettivi diversi.

Volendo semplificare la gestione della memoria si può utilizzare la strategia depth-first o LIFO, cioè si risolve l'ultimo problema inserito nella lista.

La conoscenza delle limitazioni $z_s(P')$ permette di utilizzare la strategia highest-first (si risolve il problema che ha la limitazione migliore); in questo modo si può trovare una

soluzione del problema che permetta di eliminare, per il criterio E2, alcuni dei problemi presenti nella lista.

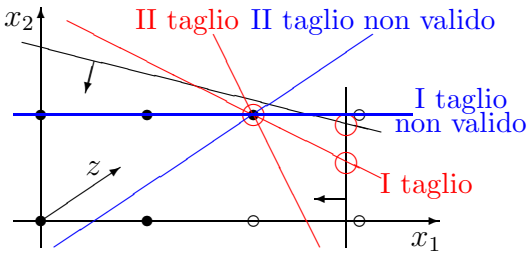
La lista può essere gestita in modo casuale, o pseudocasuale nei casi in cui le informazioni risultino particolarmente imprecise.

Osservazione 3.2.4

- *Al fine di determinare più facilmente con quale dei problemi presenti nella lista proseguire, si può utilizzare un albero in cui ogni foglia rappresenta un sottoproblema con associata la sua limitazione, superiore o inferiore a seconda che si tratti di un problema di massimo o di minimo.*

3.3 Metodi cutting plane

I metodi di questa classe si basano sull'idea di determinare una soluzione del problema ottenuto eliminando i vincoli di integrità; 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 finchè non si determina una soluzione ottimale intera, cioè gli iperpiani introdotti non "creano" un vertice corrispondente alla soluzione.



La figura precedente esemplifica la risoluzione grafica di un PLI con un metodo cutting plane; in particolare sono evidenziati due possibili tagli non validi: il primo non elimina la soluzione ottimale trovata e il secondo elimina alcune soluzioni intere ammissibili. D'altra parte 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	$i = 1, \ldots, n$	variabili in base
y_{n+j}	$j = 1, \ldots, m$	variabili fuori base

3.3.1 Algoritmo elementare (Dantzig, 1959)

La soluzione ottimale del problema lineare rilassato è caratterizzata come intersezione degli iperpiani generatori $y_i = 0, i = 1, \ldots, n$; 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:

$$y_1 + \ldots, y_n \geq 1$$

costituisce un iperpiano secante valido in quanto non è soddisfatto dalla soluzione ottimale ottenuta e non esclude nessuna altra soluzione intera. Questo primo metodo per generare gli iperpiani secanti, pur rivelandosi efficiente, non garantisce la convergenza, che è verificata per l'algoritmo di Gomory.

3.3.2 Algoritmo di Gomory (1958)

L'algoritmo di Gomory richiede che la tabella iniziale sia a coefficienti interi, ciò è sempre possibile, eventualmente approssimando i coefficienti.

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

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

e a partire da questo si genera il vincolo seguente:

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

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

Teorema 3.3.1 *Il vincolo* $y'_{n+k} = f_{k1} y_1 + \ldots + f_{ki} y_i + \ldots + f_{kn} y_n - f_k \geq 0$ *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 $y_i = 0, i = 1, \ldots, n$ si ha:

$$y'_{n+k} = -f_k$$

che non verifica il vincolo essendo negativo.

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

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

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

Dovendo essere y_{n+k} intera per ipotesi, risulta intera anche y'_{n+k}.
Allora essendo:

$$y'_{n+k} + f_k = f_{k1}y_1 + \ldots + f_{ki}y_i + \ldots + f_{kn}y_n = 0$$

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

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

♣

Osservazione 3.3.1

- *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.*

Capitolo 4

Modelli lineari a numeri interi

4.1 Problema dello zaino

Si hanno n oggetti ciascuno dei quali ha peso p_i e valore c_i; per trasportarli si ha a disposizione uno zaino di capienza assegnata P; si vuol sapere quali oggetti bisogna trasportare per massimizzare il valore trasportato.

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

4.2 Problema del trasporto

Un'industria ha n centri di produzione ciascuno dei quali ha una capacità produttiva di p_i unità e m centri di distribuzione ciascuno dei quali richiede r_j unità; il costo unitario di trasporto dal centro di produzione i al centro di distribuzione j è c_{ij}; si vuol sapere come si devono rifornire i centri di distribuzione per minimizzare i costi di trasporto.

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

dove deve essere verificata la condizione di fattibilità:

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

detta condizione di capacità produttiva.

4.3 Problema del magazzino

Un'industria vuole pianificare la produzione per n periodi consecutivi; per ciascun periodo $i = 1, ..., n$ sono noti la domanda d_i, la capacità produttiva massima C_i e il costo di produzione unitario c_i; l'industria può utilizzare un magazzino di capacità H_i, dove H_0 è la disponibilità iniziale, e per ogni periodo il costo di magazzinaggio unitario è h_i; si vuole determinare, per ogni periodo, la quantità da produrre, x_i, e la quantità da immagazzinare, y_i, per minimizzare i costi di produzione.

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

dove $y_0 = H_0$ e devono essere verificate le condizioni di fattibilità:

$$\sum_{j=1,...,i} d_j \leq \sum_{j=1,...,i} C_j, i = 1, ..., n$$

detta condizione di capacità produttiva e

$$d_i \leq C_i + H_{i-1}, i = 1, ..., n$$

detta condizione di disponibilità per il periodo i.

4.4 Problema dell'assegnazione

Ci sono n macchine a ciascuna delle quali bisogna assegnare un operaio scelto tra n; l'attitudine e l'esperienza di ogni operaio a lavorare con ciascuna macchina determinano il costo di assegnazione c_{ij} dell'operaio i alla macchina j; si vuol sapere a quale macchina bisogna assegnare ciascun operaio per minimizzare i costi.

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

4.5 Problema del commesso viaggiatore

Un commesso viaggiatore deve visitare n città compiendo un giro senza mai ripassare da alcuna città; ogni tragitto ha un costo c_{ij}, non necessariamente uguale al tragitto inverso

(asimmetria); si vuole determinare il percorso di costo minimo che tocca tutte le città senza sottogiri (circuito hamiltoniano di costo minimo).

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

Capitolo 5

Applicazioni del metodo Branch and Bound

5.1 Problema dello zaino (Problema del massimo)

Sia dato il seguente problema dello zaino:

oggetto	A	B	C	D
peso	22	18	13	7
valore	43	32	26	12

$peso\ massimo\ trasportabile = 30$

Determinare quali oggetti bisogna trasportare per massimizzare il valore trasportato rispettando il peso massimo trasportabile.

Calcolando i rapporti valore/peso di ciascun oggetto si ha:

oggetto	A	B	C	D
valore/peso	$1,955$	$1,777$	$2,000$	$1,714$

Riordinando gli oggetti in ordine decrescente di rapporto valore/peso e associando a ciascuno una variabile booleana 0-1 si ha:

oggetto	C	A	B	D
peso	13	22	18	7
valore	26	43	32	12
variabile associata	x_1	x_2	x_3	x_4

$peso\ massimo\ trasportabile = 30$

Il problema può essere riscritto come problema lineare intero a variabili 0-1 nella forma:

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

Il valore trasportabile è non superiore alla parte intera del valore ottenuto prendendo gli oggetti secondo l'ordine dato, senza superare il peso massimo trasportabile e una frazione (eventualmente nulla), tale da riempire esattamente lo zaino, dell'oggetto per cui il peso massimo trasportabile viene superato (*oggetto critico*); in questo modo si ottiene la limitazione L (bound di Dantzig).

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

Partendo dall'oggetto C avente il miglior rapporto valore/peso, si generano due sottoproblemi (1) e (0) corrispondenti a portare l'oggetto $C(x_1 = 1)$ e non portare l'oggetto $C(x_1 = 0)$.

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

Essendo più conveniente l'ipotesi di portare l'oggetto C, si generano due sottoproblemi (11) e (10) corrispondenti a portare gli oggetti C e $A(x_1 = 1, x_2 = 1)$ e portare l'oggetto C e non portare l'oggetto $A(x_1 = 1, x_2 = 0)$.

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

Essendo più conveniente l'ipotesi di non portare l'oggetto C, si generano due sottoproblemi (01) e (00) corrispondenti a non portare l'oggetto C e portare l'oggetto $A(x_1 = 0, x_2 = 1)$ e non portare gli oggetti C e $A(x_1 = 0, x_2 = 0)$.

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

Poichè non vi sono altri oggetti la soluzione del problema (00) è ammissibile con valore uguale alla limitazione.

Essendo più conveniente l'ipotesi di non portare l'oggetto C e portare l'oggetto A, si generano due sottoproblemi (011) e (010) corrispondenti a non portare l'oggetto C e portare gli oggetti A e B ($x_1 = 0, x_2 = 1, x_3 = 1$) e non portare gli oggetti C e B e portare

l'oggetto $A(x_1 = 0, x_2 = 1, x_3 = 0)$.

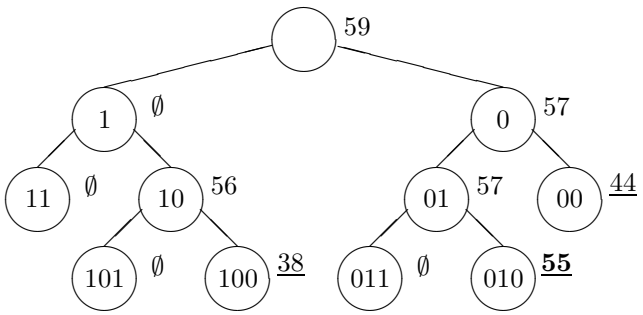
$$\begin{array}{lll} (011) \quad max & 43 + 32 + 12x_4 & \\ \quad s.t. & 7x_4 \leq -10 & S_a = \emptyset \\ (010) \quad max & 43 + 12x_4 & \\ \quad s.t. & 7x_4 \leq 8 & L(010) = \lfloor 43 + 12 \rfloor = \underline{55} \end{array}$$

Poichè non vi sono altri oggetti la soluzione del problema (010) è ammissibile con valore uguale alla limitazione.

Essendo più conveniente l'ipotesi di portare l'oggetto C e non portare l'oggetto A, si generano due sottoproblemi (101) e (100) corrispondenti a portare gli oggetti C e B e non portare l'oggetto $A(x_1 = 1, x_2 = 0, x_3 = 1)$ e portare l'oggetto C e non portare gli oggetti A e $B(x_1 = 1, x_2 = 0, x_3 = 0)$.

$$\begin{array}{lll} (101) \quad max & 26 + 32 + 12x_4 & \\ \quad s.t. & 7x_4 \leq -1 & S_a = \emptyset \\ (100) \quad max & 26 + 12x_4 & \\ \quad s.t. & 7x_4 \leq 17 & L(100) = \lfloor 26 + 12 \rfloor = \underline{38} \end{array}$$

Poichè non vi sono altri oggetti 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; quindi si portano gli oggetti A e D con valore complessivo 55 e peso complessivo 29.



Miglioramento del calcolo della limitazione

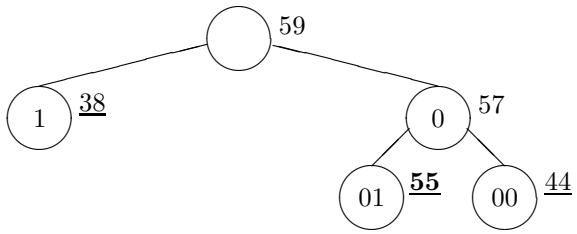
Nel calcolo delle limitazioni dei sottoproblemi è possibile migliorare il risultato tenendo conto del fatto che non potranno essere trasportati gli oggetti che superano il peso residuo trasportabile, cioè la differenza tra il peso massimo trasportabile e il peso complessivo degli oggetti per i quali si è fatta l'ipotesi di portarli.

Operando in questo modo la soluzione del problema precedente risulta modificata come

segue:

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

La soluzione del problema (01) è ottimale, avendo valore migliore di tutte le limitazioni correnti e coincide con la soluzione precedente.



Osservazione 5.1.1

- *Nel caso vi siano diverse soluzioni ottimali corrispondenti a differenti combinazioni di oggetti aventi tutte valore massimo ma peso differente, il metodo non le distingue in base al peso ma le considera equivalenti avendo tutte peso ammissibile.*

- *Si consideri il seguente problema dello zaino:*

oggetto	*A*	*B*	*C*	*D*	*E*
valore	20	15	5	3	3
peso	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:

$$L = \lfloor 20 + (\tfrac{4}{5})15 \rfloor = 32$$
$$L(1) = \lfloor 20 + 5 + (\tfrac{1}{2})3 \rfloor = 26$$
$$L(0) = \lfloor 15 + 5 + (\tfrac{1}{2})3 \rfloor = 21$$
$$L(11) = S_a = \emptyset$$
$$L(10) = \lfloor 20 + 5 + (\tfrac{1}{2})3 \rfloor = 26$$
$$L(101) = \lfloor 20 + 5 \rfloor = \underline{25}$$
$$L(100) = \lfloor 20 + 3 + 3 \rfloor = \mathbf{\underline{26}}$$

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)

Sia data la matrice dei costi di assegnazione:

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

determinare a quale macchina assegnare ciascun operaio per minimizzare il costo globale.

Poichè ogni operaio deve essere assegnato ad una macchina si può sottrarre ad ogni riga il minimo della riga stessa, ottenendo:

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

Poichè ad ogni macchina deve essere assegnato un operaio si può sottrarre ad ogni colonna il minimo della colonna stessa; la somma dei minimi sottratti costituisce il costo fisso della matrice e la limitazione L.

Per effettuare la separazione si suppone di effettuare una assegnazione o di non effettuarla, scegliendo una assegnazione di costo nullo e tra queste quella che ha il massimo costo di alternativa; riducendo la matrice e indicando come apice i costi delle alternative

per le assegnazioni di costo nullo si ha:

	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

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

A partire dall'assegnazione $D-b$ che ha il massimo costo di alternativa si considerano due sottoproblemi $(D - b)$ e $(not\ D - b)$ corrispondenti ad assegnare l'operaio D alla macchina b e a non assegnarlo.

- $(not\ D - b)$
  La limitazione si ottiene sommando alla limitazione precedente il costo dell'alternativa.
  $$L = 51 + 4 = 55$$

- $(D - b)$
  La limitazione si ottiene sommando alla limitazione precedente il costo fisso della matrice.
  La nuova matrice dei costi, non contenente la riga di D e la colonna di b, è:

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

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

Essendo più conveniente l'ipotesi di non assegnare l'operaio D alla macchina b si attribuisce costo M all'assegnazione $D - b$; la nuova matrice ridotta è:

	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

A partire dall'assegnazione $D-a$ che ha il massimo costo di alternativa si considerano due sottoproblemi $(D - a)$ e $(not\ D - a)$ corrispondenti ad assegnare l'operaio D alla macchina a e a non assegnarlo.

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

$$L = 55 + 3 = 58$$

- $(not\ D-b)(D-a)$
  La nuova matrice dei costi, non contenente la riga di D e la colonna di a, è:

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

$$L = 55 + 2 = 57$$

Essendo più conveniente l'ipotesi di assegnare l'operaio D alla macchina b si riprende il problema $(D-b)$ e a partire dall'assegnazione $B-a$ che ha il massimo costo di alternativa si considerano due sottoproblemi $(B-a)$ e $(not\ B-a)$ corrispondenti ad assegnare l'operaio B alla macchina a e a non assegnarlo.

- $(D-b)(not\ B-a)$

$$L = 56 + 2 = 58$$

- $(D-b)(B-a)$ La nuova matrice dei costi, non contenente la riga di B e la colonna di a, è:

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

$$L = 56$$

Essendo più conveniente l'ipotesi di assegnare l'operaio B alla macchina a a partire dall'assegnazione $A-d$ (o dall'assegnazione $C-c$) che ha il massimo costo di alternativa si considerano due sottoproblemi $(A-d)$ e $(not\ A-d)$ corrispondenti ad assegnare l'operaio A alla macchina d e a non assegnarlo.

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

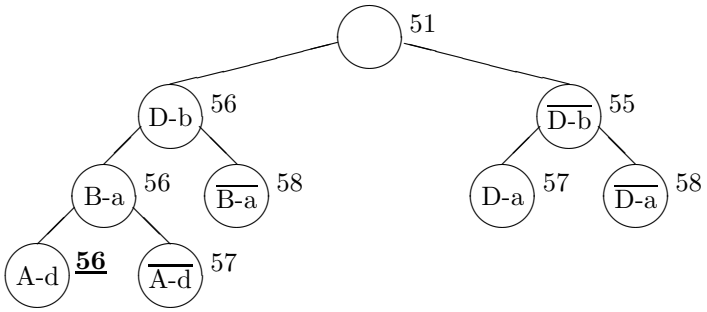
$$L = 56 + 1 = 57$$

- $(D-b)(B-a)(A-d)$ La nuova matrice dei costi, non contenente la riga di A e la colonna di d, è:

	c
C	0

$$L = 56$$

La soluzione del problema $(D-b)(B-a)(A-d)$ è ammissibile, in quanto ad ogni macchina è stato assegnato un operaio e ottimale poichè ha la miglior limitazione. 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 può essere utilizzato nella risoluzione di un generico PLI, anche se le variabili non sono legate ai valori 0-1. In particolare è utile se non tutte le variabili devono essere intere (problema lineare misto) e quindi non si può usare l'algoritmo di Gomory.

Innanzitutto si considera il rilassamento ottenuto eliminando per le variabili il vincolo di integrità e si risolve il problema con l'algoritmo del simplesso, determinando una prima limitazione superiore.

Se la variabile intera x_i assume il valore non intero x_i^* si procede ad una separazione in due sottoproblemi aggiungendo al problema dato i vincoli $x_i \leq \lfloor x_i^* \rfloor$ e $x_i \geq \lceil x_i^* \rceil$ dove $\lfloor x_i^* \rfloor$ e $\lceil x_i^* \rceil$ rappresentano rispettivamente la parte intera inferiore e superiore di x_i^*.

Quindi si risolvono con l'algoritmo del simplesso o con l'algoritmo duale i sottoproblemi ottenuti, determinando una limitazione superiore per ciascun sottoproblema.

Se la soluzione di un sottoproblema è intera essa costituisce una soluzione ammissibile del problema dato quindi il sottoproblema è eliminato (condizione E3) e si possono eliminare anche tutti i sottoproblemi pendenti che hanno una limitazione peggiore (condizione E2).

Se la lista dei problemi è vuota la soluzione trovata è quella ottimale, altrimenti si prosegue, finchè la lista dei problemi è vuota.

Osservazione 5.3.1

- *I vincoli aggiunti eliminano di volta in volta soluzioni ammissibili per il problema rilassato, ma non per il PLI assegnato, quindi la procedura è corretta.*

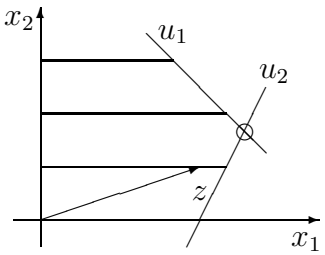
Esempio 5.3.1 (PLI misto) *Sia dato il seguente PLI misto:*

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

	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	$\frac{1}{2}$	$-\frac{1}{2}^*$	$\frac{5}{2}$
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 ottenuta $x^* = (\frac{23}{6}, \frac{5}{3})$, $z^* = \frac{79}{6}$ *non è ammissibile poichè* x_2 *non è intero.*
Il valore $\frac{79}{6}$ *costituisce la limitazione superiore e si separa il problema* P *nei due sotto-problemi:*

$$P1 = \{P \cup x_2 \leq \lfloor \tfrac{5}{3} \rfloor\} = \{P \cup x_2 \leq 1\}$$
$$P2 = \{P \cup x_2 \geq \lceil \tfrac{5}{3} \rceil\} = \{P \cup x_2 \geq 2\}$$

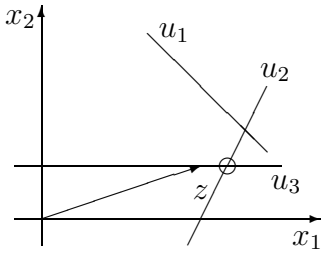
Dalla tabella finale si ricava $x_2 = \frac{1}{3}u_2 - \frac{2}{3}u_1 + \frac{5}{3}$, *per cui si ha:*

$$P1 = \{P \cup -\tfrac{1}{3}u_2 + \tfrac{2}{3}u_1 - \tfrac{2}{3} \geq 0\}$$
$$P2 = \{P \cup \tfrac{1}{3}u_2 - \tfrac{2}{3}u_1 - \tfrac{1}{3} \geq 0\}$$

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	$-\frac{2}{3}$	$-\frac{5}{3}$	$\frac{79}{6}$

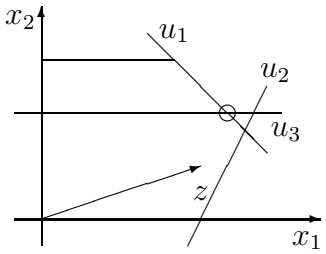
	u_2	u_3	
x_2	0	-1	1
x_1	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{7}{2}$
u_1	$\frac{1}{2}$	$\frac{3}{2}$	1
z	$-\frac{3}{2}$	$-\frac{5}{2}$	$\frac{23}{2}$



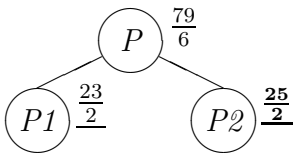
La soluzione ottenuta $x^* = (\frac{7}{2}, 1)$, $z^* = \frac{23}{2}$ *è ammissibile. Risolvendo P2 con l'algoritmo duale si ottiene:*

	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 ottenuta $x^* = (\frac{7}{2}, 2)$, $z^* = \frac{25}{2}$ *è ammissibile ed essendo il valore di* z^* *maggiore del valore ottenuto per P1 costituisce la soluzione ottimale del problema* P.



$\diamondsuit$

5.4 Rilassamenti

5.4.1 Rilassamento continuo

Si eliminano i vincoli di integrità sulle variabili.

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

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

5.4.2 Eliminazione di vincoli

Si eliminano alcuni vincoli del problema dato al fine di ottenere un problema ben strutturato, cioè più facilmente risolubile.

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

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

5.4.3 Rilassamento lagrangiano

Si inseriscono i vincoli nella funzione obiettivo, eventualmente con un peso non negativo li, in modo che le soluzioni non ammissibili per il problema dato risultino penalizzate per

il problema rilassato e riducendo le dimensioni del problema.

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

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

5.4.4 Rilassamento surrogato

Si sostituisce ai vincoli la somma degli stessi, eventualmente con un peso non negativo p_i, in modo da ridurre le dimensioni del problema.

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

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

Osservazione 5.4.1

- *I rilassamenti precedenti possono essere applicati parzialmente, cioè solo ad alcune variabili o ad alcuni vincoli, o essere combinati tra loro.*
- *La soluzione del problema rilassato costituisce in generale una limitazione (superiore o inferiore) della soluzione ottimale ma può anche essere la soluzione ottimale se è verificata la condizione E3 del rilassamento.*

5.5 Risoluzione per ispezione

I problemi ottenuti utilizzando il rilassamento largangiano o il rilassamento surrogato risultano in generale strutturalmente molto semplici, per cui si può utilizzare una metodologia risolutiva particolarmente efficiente detta ispezione.

L'ispezione consiste nel considerare caratteristiche peculiari del problema rilassato che permettono di valutare rapidamente la soluzione ottimale, o almeno una buona limitazione della stessa. Evidentemente non esistono regole generali ma la metodologia fornisce spesso buoni risultati.

Esempio 5.5.1 (Rilassamento lagrangiano di un problema dello zaino) *Sia dato il seguente problema dello zaino:*

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

Applicando il rilassamento lagrangiano si ottiene il seguente problema:

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

che si può riscrivere come:

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

che risolto per ispezione fornisce la soluzione ottimale:

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

♢

Esempio 5.5.2 (Rilassamento surrogato di un problema a variabili 0-1) *Sia dato il seguente problema a variabili 0-1:*

$$\begin{array}{lll} max & \sum_{j=1,\ldots,n} c_j x_j & \\ s.t. & \sum_{j=1,\ldots,n} a_j x_j \leq b_i & i = 1,\ldots,m \\ & x_j \in \{0,1\} & j = 1,\ldots,n \end{array}$$

Applicando il rilassamento surrogato si ottiene il seguente problema:

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

che costituisce 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* $\sum_{i=1,\dots,m} \pi_i b_i$. ♢

Capitolo 6

Complementi di programmazione lineare

6.1 Variazione dei coefficienti iniziali

Sia data la seguente tabella iniziale relativa a un programma lineare:

	x^T	
u	A	b
z	c^T	0

e la seguente tabella finale:

	ξ^T	
ν	α	β
z	χ^T	χ_0

Dividendo per chiarezza le variabili in base ξ tra le variabili u' (fuori base nella tabella iniziale e successivamente entrate in base) e x'' (in base nella tabella iniziale e rimaste in base) e analogamente le variabili fuori base ν tra le variabili x'' (in base nella tabella iniziale e successivamente uscite) e u'' (fuori base nella tabella iniziale e rimaste fuori base), la tabella finale diventa:

	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

Partizionando nello stesso modo la tabella iniziale si ha:

	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 ed eseguendo le operazioni del metodo del cardine trattando matrici e vettori come se fossero scalari si ottiene:

	u'^T	x''^T	
$x\prime$	$(A_{11})^{-1}$	$-(A_{11})^{-1}A_{12}$	$-(A_{11})^{-1}b_1$
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	$c_1^T(A_{11})^{-1}$	$c_2^T - c_1^T(A_{11})^{-1}A_{12}$	$-c_1^T(A_{11})^{-1}b_1$

Sostituendo nella tabella ottenuta i valori dati dalla tabella iniziale si ottengono delle identità, quindi le relazioni sono verificate e la tabella ottenuta è effettivamente quella finale espressa in funzione dei coefficienti iniziali.

6.1.1 Applicazioni

- **Stabilità della soluzione ottimale**
  Data la tabella finale in funzione dei coefficienti iniziali si può facilmente verificare se al variare di qualche coefficiente la soluzione data resta ottimale, cioè se i coefficienti della funzione obiettivo restano non positivi ed eventualmente come cambiano le coordinate della soluzione ottimale e il valore ottimale.

- **Aggiunta di un vincolo**
  Nel caso in cui si debba aggiungere un ulteriore vincolo, ad esempio nel metodo degli iperpiani secanti, questo può essere scritto in funzione delle variabili inizialmente in base in forma partizionata come:

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

  e in funzione delle variabili in base nella tabella finale in forma partizionata come:

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

  Il vincolo è ora espresso in funzione delle variabili in base nella tabella corrente per cui può essere aggiunto alla tabella per proseguire con il simplesso normalmente.

6.2 Algoritmi greedy

Gli algoritmi greedy (ghiotti) costituiscono una classe di algoritmi approssimati caratterizzati da un livello di approssimazione spesso piuttosto basso ma anche da una elevata velocità di esecuzione.

Questi algoritmi si basano sull'analisi delle variabili decisionali, secondo un semplice test di ottimalità locale, per cui una variabile decisionale viene posta al miglior valore ammissibile in quel momento. In questo modo si ottiene una soluzione ammissibile che trascura qualsiasi visione globale del problema e risente fortemente dell'ordine con cui vengono analizzate le variabili decisionali.

Gli algoritmi greedy trovano applicazione soprattutto nei problemi decisionali a variabili 0-1, ma possono essere utilizzati in molti casi.

6.2.1 Algoritmo greedy per il problema dello zaino

L'algoritmo consiste nell'ordinare gli oggetti in ordine decrescente di rapporto valore/peso e quindi di prendere un oggetto se può essere preso, cioè se la somma del suo peso e di quello degli oggetti già presi non supera il peso massimo trasportabile.

L'algoritmo richiede solo $O(n)$ operazioni, dove n è il numero di oggetti, e fornisce una discreta approssimazione nel caso in cui i pesi degli oggetti non siano molto differenti tra loro e siano piccoli rispetto al peso massimo trasportabile.

Nel caso peggiore il rapporto tra il valore approssimato fornito dall'algoritmo greedy e il valore ottimale del problema (problema di massimo) tende a zero.

Esempio 6.2.1 (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 il valore approssimato $z_s = 2$*, mentre la soluzione ottimale è prendere il secondo oggetto e scartare il primo, con valore ottimale* $z^* = M$*, per cui si ha:*

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

♢

Esempio 6.2.2 (Soluzione approssimata con l'algortimo greedy) *Si consideri il seguente problema dello zaino:*

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 si inizia con la stima:

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

per cui si ha $z^* \leq 220$.
L'algoritmo greedy prende gli oggetti A, B, C, G fornendo il valore approssimato $z_s = 219$ *con un errore inferiore al 1%:*

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

Questo esempio dimostra l'utilità dell'algoritmo greedy nei casi in cui la velocità di calcolo e una buona approssimazione sono più importanti della soluzione esatta.
Inoltre si può verificare che la soluzione trovata è esatta. ♢

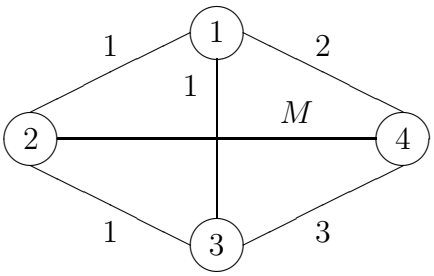
6.2.2 Algoritmo greedy per il problema del commesso viaggiatore

L'algoritmo consiste nello scegliere per ogni città di andare nella città più vicina, in termini di costi, dove non si è ancora stati (nearest unvisited) iniziando dalla città 1.

L'algoritmo richiede solo $O(n^2)$ operazioni, dove n è il numero di città, e fornisce una discreta approssimazione nel caso in cui le città non siano collocate disordinatamente rispetto alla città 1 e i costi rispettino le disuguaglianze triangolari.

Nel caso peggiore il rapporto tra il valore ottimale del problema e il valore approssimato fornito dall'algoritmo greedy (problema di minimo) tende a zero.

Esempio 6.2.3 (Caso peggiore) *Si consideri il seguente problema del commesso viaggiatore:*



Partendo dalla città 1*, l'algoritmo greedy determina il percorso:*

$$1 - 3 - 2 - 4 - 1$$

fornendo il valore approssimato $z_s = 4 + M$*, mentre la soluzione ottimale è:*

$$1 - 2 - 3 - 4 - 1$$

con valore ottimale $z^* = 7$*, per cui si ha:*

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

$\diamondsuit$

Capitolo 7

Teoria delle reti

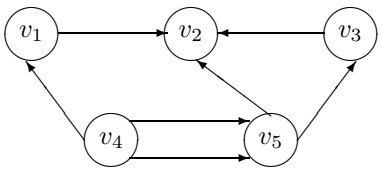
7.1 Grafi

Intuitivamente un grafo può essere definito come un insieme finito di punti ed un insieme di frecce che uniscono coppie di punti.

I punti si chiamano *nodi* o *vertici* del grafo e le frecce si chiamano *archi* del grafo; il verso della freccia assegna un *orientamento* all'arco corrispondente; se non si tiene conto dell'orientamento degli archi il grafo è detto *multigrafo* o *non orientato* e gli archi sono detti *spigoli*. Tra due nodi possono esistere più archi orientati nello stesso verso; il numero massimo di archi orientati nello stesso verso presenti tra due nodi si indica con p e il grafo è detto *p-grafo*. Il numero di nodi è detto *ordine* del grafo e si indica con n; il numero di archi si indica con m.

Definizione 7.1.1 *Un grafo è una coppia $G(N, A)$, dove $N = \{v_1, \ldots, v_n\}$ 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 7.1.1 (Elementi di un grafo) *Si consideri il grafo $G(N, A)$ dove:*
$N = \{v_1, v_2, v_3, v_4, v_5\}$
$A = \{a_{12}, a_{32}, a_{41}, a_{45}, a_{45}, a_{52}, a_{53}\}$
$p = 2; n = 5; m = 7$



♢

Definizione 7.1.2

- *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 7.2.1 *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.

Osservazione 7.2.1

- 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 $x : A \to \mathbb{Z}$ che ad ogni arco a_{ij} associa un valore intero x_{ij} detto *flusso sull'arco*, in modo che siano soddisfatti i *vincoli di capacità degli archi*, cioè il flusso su ogni arco sia compreso tra la capacità minima e la capacità massima dell'arco e i *vincoli di bilanciamento nei nodi*, cioè la differenza tra il flusso uscente e il flusso entrante in ciascun nodo sia uguale alla disponibilità o alla domanda. Al flusso sull'arco a_{ij} si associa un costo dato dal prodotto tra il costo unitario c_{ij} e il flusso x_{ij}; la somma dei costi di tutti gli archi è detta *costo associato al flusso* x. I vincoli e il costo possono essere espressi nella forma seguente:

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

Osservazione 7.2.2

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

7.2.2 Problema del flusso di costo minimo

Il problema consiste nel determinare un flusso x su una rete $G(N, A)$, minimizzando il costo associato; formalmente può essere espresso come segue:

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

In questa forma possono essere rappresentati tutti i problemi di reti.

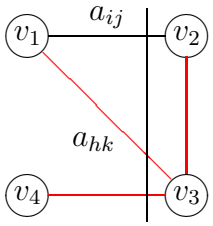
7.3 Minimo Spanning Tree

Data una rete $G(N, A)$ si deve trovare uno spanning tree di costo minimo, cioè i cui archi hanno globalmente costo minimo. Se gli archi hanno tutti lo stesso costo la soluzione è un qualsiasi spanning tree.

Come premessa si considerino due teoremi.

Teorema 7.3.1 *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}*

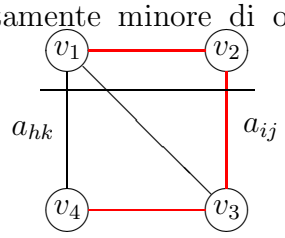
Dimostrazione
Per assurdo sia T' un albero non contenente a_{ij} e di peso strettamente minore di ogni albero contenente a_{ij} e sia a_{hk} l'arco di T' appartenente al taglio generato da N' e facente parte con a_{ij} di un ciclo. Sostituendo a_{ij} ad a_{hk} si ottiene un albero di peso non superiore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



♣

Teorema 7.3.2 *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)

L'algoritmo costruisce uno spanning tree $T(N, A')$ del grafo $G(N, A)$ tramite uno spanning tree parziale $T'(N', A')$ al quale ad ogni iterazione viene aggiunto un arco all'insieme A' e i relativi estremi all'insieme 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 7.3.1.

Algoritmo di Kruskal (1956)

L'algoritmo costruisce uno spanning tree $T(N, A')$ del grafo $G(N, A)$ aggiungendo ad ogni iterazione all'insieme A' l'arco di peso minimo che non genera cicli, fino a che $card(A') = card(N) - 1$.
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) $A' = A' \cup \{a_{ij}\}$;

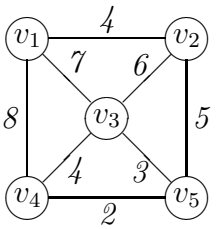
d) se card$(A') < card(N) - 1$ tornare a b);
altrimenti STOP

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 7.3.2.

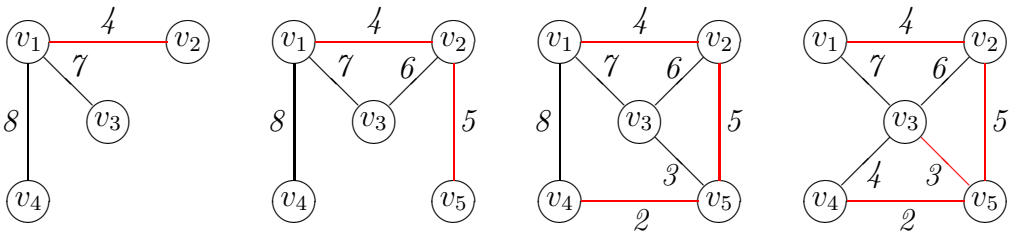
Osservazione 7.3.1

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

Esempio 7.3.1 (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 $A' = \emptyset; N' = \{v_1\}$

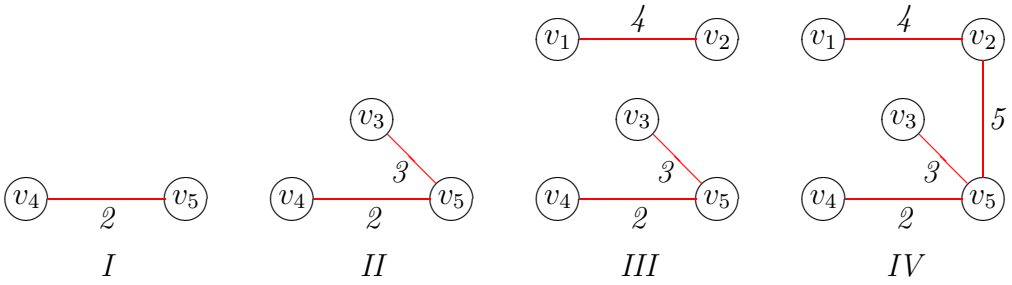
I $A' = \{a_{12}\}; N' = \{v_1, v_2\}$

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 $A' = \{a_{12}, a_{25}, a_{45}, a_{35}\}; N' = \{v_1, v_2, v_5, v_4, v_3\}$; *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* $(card(A') = card(N) - 1)$ ♢

7.4 Cammino minimo

Data una rete $G(N, A)$ si deve 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.

Pur essendo un problema semplice, servono algoritmi efficienti perchè è un sottoproblema di molti altri problemi, ad esempio reti di trasporto, reti di comunicazione, percorsi veicolari, progettazione di reti.

Per memorizzare i cammini di costo minimo si utilizza un puntatore che per ogni nodo indica il predecessore; infatti il predecessore è unico mentre il successore può non esserlo.

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)

Richiede che nessun arco abbia costo negativo per evitare il caso dei cicli di costo negativo.

Algoritmo

a) $d(s) = 0$;

b) per $i = 1, ..., n$ $d(i) = c_{si}$ e $pred(i) = s$;

c) sia $d(h) = min\{d(i)|d(i)$ 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)

Le etichette vengono aggiornate analizzando gli archi in ordine casuale.

Algoritmo

a) $d(s) = 0$;

b) per $i = 1, ..., n$ $d(i) = M$ e $pred(i) = s$ (M maggiorante delle etichette esatte);

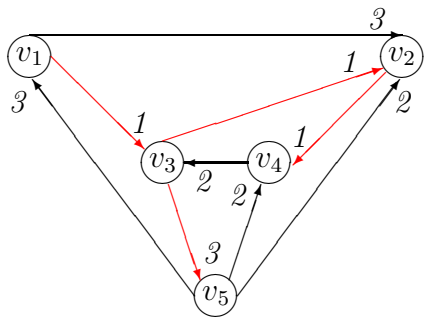
c) se esiste un arco a_{hi} con $d(i) > d(h) + c_{hi}$ porre $d(i) = d(h) + c_{hi}$ e $pred(i) = h$ e ripetere c);
altrimenti STOP (le etichette sono tutte esatte);

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)

Se si esaminano gli archi secondo un ordinamento prefissato l'algoritmo termina dopo aver esaminato gli archi al più $n-1$ volte, inoltre determina l'esistenza di cicli di costo negativo. Se l'algoritmo modifica qualche etichetta per tutte le $n-1$ iterazioni si esegue una ulteriore iterazione e se l'etichetta $d(i)$ viene modificata vuol dire che esiste un cammino da v_s a v_i composto da almeno n archi che quindi contiene un ciclo che ha necessariamente costo negativo.

Esempio 7.4.1 (Grafo con archi di peso non negativo) *Determinare lo SPP dal nodo v_1 a tutti gli altri nodi per il seguente grafo:*



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

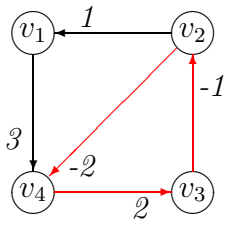
Algoritmo di Bellmann

Ordinamento forward star: $u = (2, 3, 4, 2, 5, 3, 1, 2, 4)$; $p = (1, 3, 4, 6, 7, 10)$

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 7.4.2 (Grafo con ciclo negativo) *Determinare lo SPP dal nodo* v_1 *a tutti gli altri nodi per il seguente grafo:*



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 forward star: $u = (4, 1, 4, 2, 3)$; $p = (1, 2, 4, 5, 6)$

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

Sia 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).

Si deve 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 $F(v_s, v_t)$ e definito come segue:

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

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

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

Questo problema è utile anche per valutare l'affidabilità di una rete in quanto si può prevedere il suo comportamento in caso di indisponibilità di qualche arco.

Il problema può essere scritto in forma di problema di programmazione lineare a numeri interi:

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

Questo problema può essere risolto con i metodi della programmazione lineare oppure con uno specifico algoritmo.

L'algoritmo più comune è dovuto a Ford e Fulkerson (1956) ed è detto del contrassegno perchè contrassegna i nodi in modo da costruire una sequenza di cammini dalla sorgente v_s al pozzo v_t sui quali incrementare il flusso fino a raggiungere il massimo.

L'algoritmo del contrassegno appartiene alla classe degli *algoritmi di cammino aumentante* di cui fanno parte anche l'algoritmo del minimo cammino aumentante (Edmonds e Karp, 1972) e l'algoritmo delle reti stratificate (Dinic, 1970 - Ahuja e Orlin, 1991).

Esiste un'altra classe di algoritmi più flessibili, gli *algoritmi di preflusso*, che consentono di aumentare il flusso non su un cammino da v_s a v_t, ma su un singolo arco con una operazione di invio (*push*), cui appartengono l'algoritmo di preflusso (Karzanov, 1974 - Goldberg e Tarjan, 1986) e la sua variazione (Goldberg e Tarjan, 1986) e l'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, corrispondenti ai flussi sugli archi aventi entrambi gli estremi in U, si ha:

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

D'altra parte preso $V = \{\text{nodi contrassegnati al termine dell'algoritmo}\}$ si ha:

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

dove l'ultima eguaglianza deriva dalla situazione finale dell'algoritmo.

Definizione 7.5.1 *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* $\sum_{v_i \in U, v_j \notin U} u_{ij} - \sum_{v_i \in U, v_j \notin U} l_{ji}$.

Un importante risultato è costituito dal seguente teorema.

Teorema 7.5.1 (Teorema di Ford e Fulkerson (max flow - min cut, 1956)) *Data una rete di trasporto $G(N, A)$ si ha:*

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

Osservazione 7.5.1 *Il problema del flusso massimo e del taglio minimo per una data rete sono legati da una relazione di dualità, quindi il Teorema di Ford e Fulkerson non è altro che un caso particolare del I Teorema della dualità.*

Un problema strettamente legato a quello del flusso massimo è quello del *flusso compatibile* necessario per inizializzare il flusso degli algoritmi di cammino aumentante. Anche in questo caso esistono numerosi algoritmi, tra i quali il più comune è quello di Herz (1967).

Definizione 7.5.2 *Si definisce distanza di un flusso x dalle capacità l e u la quantità:*

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

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

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

Algoritmo di Herz

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:

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

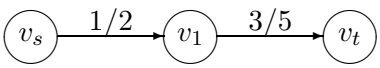
g) se $d(x) = 0$ STOP (il flusso x è ammissibile)
altrimenti tornare a b);

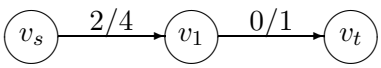
L'algoritmo converge poichè ad ogni iterazione la distanza $d(x)$ si riduce.
Vale inoltre il seguente:

Teorema 7.5.2 (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:

$$\sum_{v_i \in U} \sum_{j \in \omega^+(i)} l_{ij} \leq \sum_{v_i \in U} \sum_{j \in \omega^-(i)} u_{ji}$$

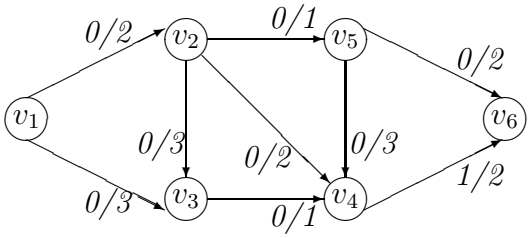
Esempio 7.5.1 *Si considerino le seguenti due situazioni di flusso impossibile:*

a) 

b) 

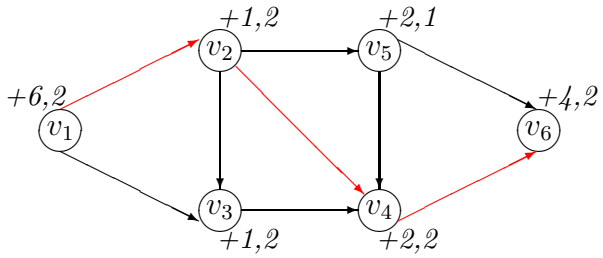
Nel caso a) l'insieme $U = \{v_1\}$ e nel caso b) l'insieme $U = \{v_s, v_t\}$ non verificano le ipotesi del teorema. ♢

Esempio 7.5.2 (Flusso massimo) *Sia data la seguente rete in cui sono indicate le capacità minime e massime di ogni arco:*

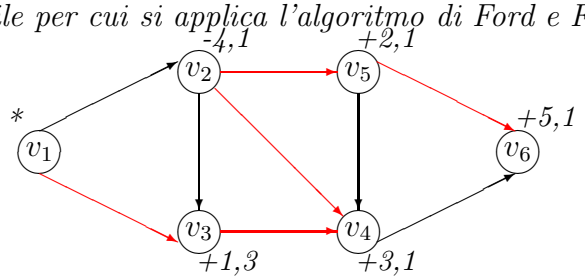


- *si esaminano nodi e archi secondo l'ordine crescente degli indici, sia nella scelta del nodo contrassegnato sia nella scelta dei nodi da contrassegnare;*
- *si contrassegnano tutti i nodi possibili per non dover riesaminare successivamente lo stesso nodo;*
- *al contrassegno si aggiunge un termine che indica la quantità massima di cui si può incrementare il flusso lungo il cammino fino a quel nodo.*

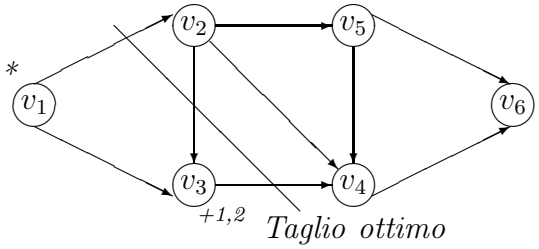
Il flusso nullo non è ammissibile per cui si applica l'algoritmo di Herz selezionando l'arco a_{46}.



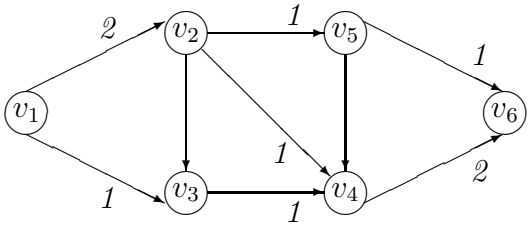
Il ciclo è dato da: $v_4 - v_6 - v_1 - v_2 - v_4$ *con incremento di 2 unità.*
Il flusso è ammissibile per cui si applica l'algoritmo di Ford e Fulkerson.



Il cammino aumentante è dato da: $v_1 - v_3 - v_4 - v_2 - v_5 - v_6$ *con incremento di 1 unità.*



Poichè non è possibile contrassegnare il pozzo v_6 *non esistono cammini aumentanti e il flusso massimo è quello rappresentato in basso.*



Utilizzando i nodi contrassegnati e non contrassegnati si determina il taglio minimo $(v_1, v_2), (v_2, v_3), (v_3, v_4)$, *rappresentato in alto, che ha capacità di 3 unità.* ♢

7.5.1 Relazione di dualità tra flusso massimo e taglio minimo

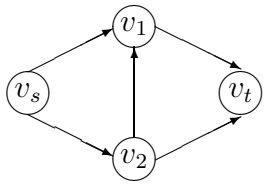
Il problema del flusso massimo e del taglio minimo per una data rete sono legati da una relazione di dualità. Dato un problema di flusso massimo, trascurando il vincolo di integrità del flusso, si può scriverne il duale nella forma:

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

Osservazione 7.5.2

- *Le variabili 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.*
- *L'ultimo vincolo di uguaglianza assicura $y_t = 1$ e $y_s = 0$, cioè $v_s \in U, v_t \notin U$.*

Esempio 7.5.3 (Max flow - min cut) *Sia data la seguente rete:*



La tabella a lato rappresenta in maniera compatta il problema lineare di flusso massimo (sulle righe) e il suo duale (sulle colonne); non compaiono i vincoli di non negatività delle variabili α e di non positività delle variabili β.

	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. ♢

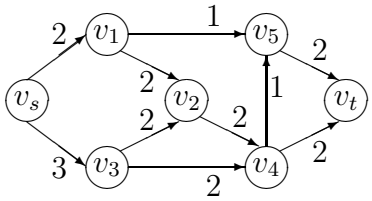
Si può concludere che il Teorema di Ford e Fulkerson non è altro che un caso particolare del I Teorema della dualità.

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

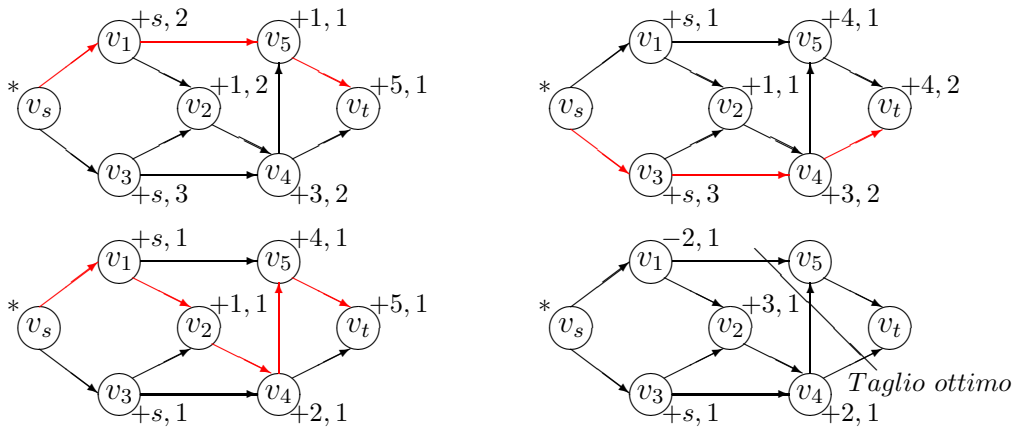
Se nell'algoritmo di Ford e Fulkerson si analizzano i nodi etichettati nell'ordine in cui sono stati etichettati (ricerca in ampiezza) il cammino aumentante che si determina è composto dal minimo numero di archi. Formalmente è sufficiente modificare il passo c) nel modo seguente:

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 $x_{ji} > l_{ji}$; eliminare v_i da V e andare a d);

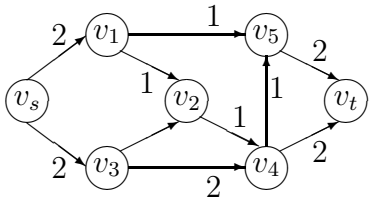
Esempio 7.5.4 (Algoritmo di Edmonds e Karp) *Sia data la seguente rete in cui sono indicate le capacità massime (le capacità minime sono tutte nulle):*



Etichettando i nodi con una visita in ampiezza si ottiene:

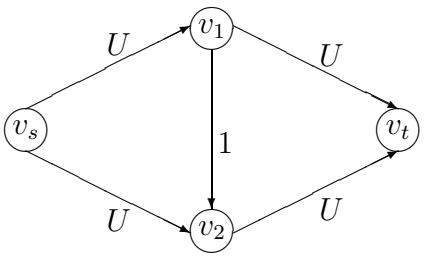


Il taglio ottimo comprende gli archi a_{15}, a_{45}, a_{4t},con capacità 4. I flussi sugli archi sono riportati di seguito:



♢

Esempio 7.5.5 (Confronto di complessità) *Sia data la seguente rete in cui sono indicate le capacità massime:*



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. ♢

7.5.3 Complessità computazionale degli algoritmi di flusso massimo

Valutando la complessità computazionale sul comportamento nel caso peggiore e considerando una rete con n nodi e m archi si ha:

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 $n-1$ 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).

Per evidenziare le caratteristiche di complessità consideriamo i due casi di 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)

Osservazione 7.5.3

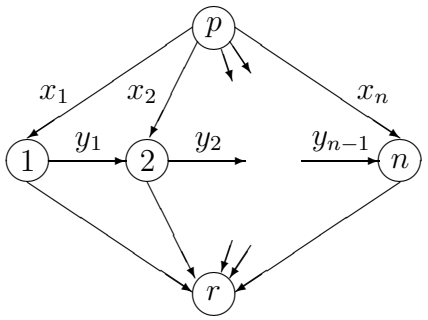
- *Se U = O(n) i due algoritmi sono equivalenti per le reti sparse.*
- *Il Teorema della decomposizione del flusso:* Dati due flussi ammissibili, relativi alla stessa rete, è sempre possibile passare dall'uno all'altro decomponendo il flusso in al più $n+m$ cammini e cicli di cui al più m cicli *fornisce 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 da indicazioni sul come determinare i cammini e i cicli.*

7.6 Problema di produzione e magazzino

Una azienda produce per n periodi consecutivi; per ciascun periodo $i = 1, ..., n$ sono noti la domanda d_i, il costo di produzione unitario c_i e il costo di immagazzinamento unitario h_i e si vogliono conoscere la quantità da produrre x_i e la quantità da immagazzinare y_i. Inoltre sono note la capacità produttiva massima C_i e la capacità del magazzino H_i. Si

costruisce la rete sottostante e si pone:

$c_{pi} = c_i \qquad i = 1, ..., n$
$c_{ir} = 0 \qquad i = 1, ..., n$
$c_{i,i+1} = h_i \qquad i = 1, ..., n-1$
$l_{ij} = 0 \qquad a_{ij} \in A$
$u_{pi} = C_i \qquad i = 1, ..., n$
$u_{ir} = d_i \qquad i = 1, ..., n$
$u_{i,i+1} = H_i \qquad i = 1, ..., n-1$
$b_p = \sum_{i=1,...,n} d_i$
$b_r = -\sum_{i=1,...,n} d_i$
$b_i = 0 \qquad i = 1, ..., n$



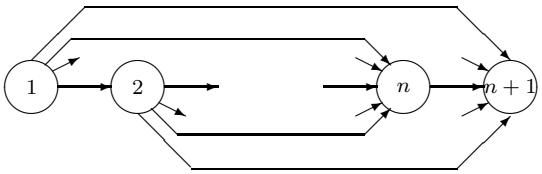
I vincoli di bilanciamento nel nodo i sono i vincoli dinamici del problema del magazzino:

$$x_i + y_{i-1} = d_i + y_i \qquad i = 1, ..., n$$

ed esprimono che, per ciascun periodo, la somma tra la quantità prodotta e la quantità presente in magazzino all'inizio del periodo è uguale alla somma tra la quantità richiesta e la quantità presente in magazzino alla fine del periodo. Il risultato è un 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; infatti, non essendoci vincoli, per ciascun periodo si sceglie la strategia migliore tra attingere al magazzino o attingere alla produzione. Questa proprietà permette di risolvere il problema anche nel caso in cui per ogni periodo $i = 1, ..., n$ esista un costo fisso di produzione F_i, cioè un costo indipendente dalla quantità prodotta che deve essere pagato solo se in quel periodo la produzione è non nulla, nel qual caso la funzione obiettivo non è più lineare. A tal fine si costruisce la seguente rete con $n+1$ nodi in cui ogni nodo è collegato ai successivi e il costo dell'arco a_{ij} è dato dalla somma del costo per produrre nel periodo i per tutti i periodi successivi fino al periodo $j-1$ (costo fisso e costo complessivo della produzione) e del costo di immagazzinare la produzione di ciascun periodo fino al periodo di vendita, cioè:

$$c_{ij} = F_i + \sum_{k=i,...,j-1} c_i d_k + \sum_{h=i,...,j-2} \sum_{k=h+1,...,j-1} h_h d_k \qquad \begin{array}{l} i = 1, ..., n \\ j = i+1, ..., n+1 \end{array}$$



Si ottiene un problema di cammino minimo da 1 a $n + 1$; gli archi facenti parte della soluzione indicano in quali periodi bisogna produrre e per quanti periodi.

7.7 Problema di routing e scheduling

Un problema di routing è una variante del problema del trasporto in cui non si considerano percorsi punto-punto, ma itinerari reali, durante i quali possono essere effettuate operazioni di carico e scarico. In generale si suppone che esista un'unica origine o deposito (*depot*) ma più destinazioni. Ovviamente è possibile che un unico viaggio non consenta di soddisfare tutte le richieste delle destinazioni. Se si suppone che le operazioni presso le destinazioni debbano rispettare vincoli temporali (*finestre*) allora il problema prende nome di problema di routing e scheduling. La complessità del problema può diventare molto elevata.

Capitolo 8

Problema del flusso di costo minimo

8.1 Formulazione del problema

E' il problema più generale tra i problemi su reti, in quanto ogni problema può essere ricondotto a questo; può essere scritto in forma di problema di programmazione lineare a numeri interi:

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

Osservazione 8.1.1

- *I dati devono verificare la condizione* $\sum_{i \in N} b_i = 0$*, cioè la somma delle disponibilità deve essere uguale alla somma delle domande.*

Esistono numerosi algoritmi per risolvere questo problema il più generale dei quali è il simplesso su reti; l'algoritmo *Out of Kilter* è interessante perchè si basa sulle condizioni di ottimalità date dal II Teorema della dualità.

Trascurando il vincolo di integrità (P4) il problema duale può essere scritto nella forma:

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

Osservazione 8.1.2

- *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.*

Le condizioni di ottimalità del II Teorema della dualità possono essere scritte come:

$$\begin{aligned}
&\left(-\sum_{j\in\omega^+(i)} x_{ij} + \sum_{j\in\omega^-(i)} x_{ji} + b_i\right) y_i = 0 && \forall\ v_i \in N && (C1)\\
&(x_{ij} - l_{ij})\, s_{ij} = 0 && \forall\ a_{ij} \in A && (C2)\\
&(-x_{ij} + u_{ij})\, t_{ij} = 0 && \forall\ a_{ij} \in A && (C3)\\
&(y_i - y_j + s_{ij} - t_{ij} - c_{ij})\, x_{ij} = 0 && \forall\ a_{ij} \in A && (C4)
\end{aligned}$$

Osservazione 8.1.3

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

L'osservazione precedente permette di semplificare il problema riducendolo all'analisi di 3 casi (per ogni arco) che coinvolgono i valori ammissibili delle variabili primali:

$$\begin{array}{llllll}
Caso\ 1 & x_{ij} = l_{ij} & \Rightarrow t_{ij} = 0 & \Rightarrow y_i - y_j + s_{ij} = c_{ij} & \Rightarrow c^*_{ij} \geq 0\\
Caso\ 2 & l_{ij} < x_{ij} < u_{ij} & \Rightarrow s_{ij} = t_{ij} = 0 & \Rightarrow y_i - y_j = c_{ij} & \Rightarrow c^*_{ij} = 0\\
Caso\ 3 & x_{ij} = u_{ij} & \Rightarrow s_{ij} = 0 & \Rightarrow y_i - y_j + t_{ij} = c_{ij} & \Rightarrow c^*_{ij} \leq 0
\end{array}$$

Risolvere il problema di flusso di costo minimo equivale a risolvere il seguente sistema:

$$\begin{cases}
\displaystyle\sum_{j\in\omega^+(i)} x_{ij} - \sum_{j\in\omega^-(i)} x_{ji} = b_i & \forall\ v_i \in N & (S1)\\
l_{ij} \leq x_{ij} \leq u_{ij} & \forall\ a_{ij} \in A & (S2)\\
c^*_{ij} \geq 0 & \text{se } x_{ij} = l_{ij} & (S3)\\
c^*_{ij} = 0 & \text{se } l_{ij} < x_{ij} < u_{ij} & (S4)\\
c^*_{ij} \leq 0 & \text{se } x_{ij} = u_{ij} & (S5)
\end{cases}$$

8.2 Problema di circolazione

Un particolare problema di flusso si ha quando i nodi sono tutti di attraversamento, cioè mancano sorgenti e pozzi, cioè $b_i = 0, \forall\ i \in N$.

Ogni problema di flusso può essere ricondotto ad un problema di circolazione equivalente, modificando la rete nel modo seguente:

- Ricordando che $\sum_{i \in N} b_i = 0$, 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 $b_{s^*} = b_{t^*} = 0$, ottenendo un problema di circolazione.

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:

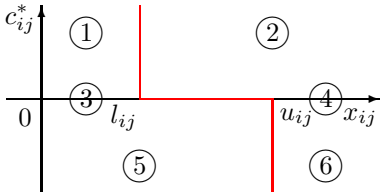
1)	$c^*_{ij} > 0, x_{ij} < l_{ij}$	3)	$c^*_{ij} = 0, x_{ij} < l_{ij}$	5)	$c^*_{ij} < 0, x_{ij} < u_{ij}$
2)	$c^*_{ij} > 0, x_{ij} > l_{ij}$	4)	$c^*_{ij} = 0, x_{ij} > u_{ij}$	6)	$c^*_{ij} < 0, x_{ij} > u_{ij}$

Osservazione 8.3.1

- *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.*

Definizione 8.3.1 *Se un arco soddisfa le condizioni di ottimalità (S2-S5) è detto* ammissibile (In Kilter) *altrimenti è detto* non ammissibile (Out of Kilter).

A ciscun arco si può associare un diagramma di ammissibilità (Kilter diagram); a seconda dei valori di x_{ij} e di c^*_{ij} si determina la posizione dell'arco nel diagramma (i segmenti spessi rappresentano i valori ammissibili).



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 oppure determinare l'impossibilità del problema quando non è possibile modificare i potenziali. I due passi di modifica sono i seguenti:

Modifica del flusso

Per modificare il flusso si considera un arco non ammissibile (v_i, v_j), ad esempio quello avente violazione massima, e 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. A questo punto si etichettano i nodi fino ad etichettare l'altro nodo dell'arco (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 dell'arco (v_i, v_j) si modifica il flusso sul ciclo C trovato della quantità $\Delta = min\ \{\Delta_{ij}\}$. Per modificare il flusso si assegna a C l'orientamento indotto dall'arco (v_i, v_j) e si assegnano i nuovi flussi ponendo:

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

Osservazione 8.3.2

- *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 startegie di etichettatura.*

Modifica dei potenziali

Se non si etichetta l'altro nodo dell'arco (v_i, v_j) si cerca di modificare i potenziali; utilizzando l'etichettatura corrente si determina la quantità $c^0 = min\ \{c_1, -c_2\}$, dove:

$$c_1 = min\ \{c^*_{hk} > 0 \text{ s.t. } v_h \text{ è etichettato e } v_k \text{ non è etichettato con } x_{hk} \leq u_{hk}\}$$
$$c_2 = max\ \{c^*_{hk} < 0 \text{ s.t. } v_h \text{ non è etichettato e } v_k \text{ è etichettato con } x_{hk} \geq l_{hk}\}$$

I nuovi potenziali vengono assegnati ponendo:

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

Osservazione 8.3.3

- *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);

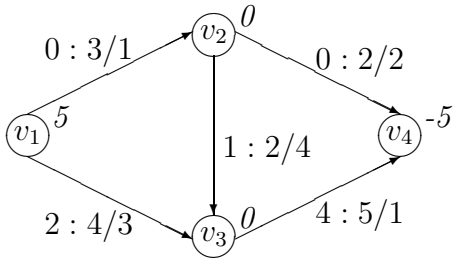
d) se si determina un ciclo modificare il flusso e tornare a b);
altrimenti determinare c^0 e andare a e);

e) se $c^0 < +\infty$ modificare i potenziali e tornare a c)
altrimenti STOP (non ci sono soluzioni ammissibili);

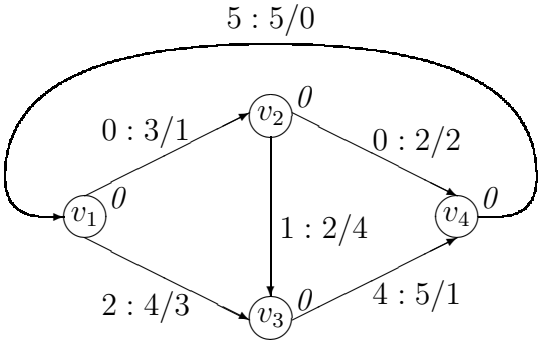
Osservazione 8.3.4

- *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 8.3.1 (Algoritmo Out of Kilter) *Sia data la seguente rete in cui sono indicate le capacità minime e massime e i costi di ogni arco (la notazione è* l:u/c*) e le domande e disponibilità di ogni nodo:*

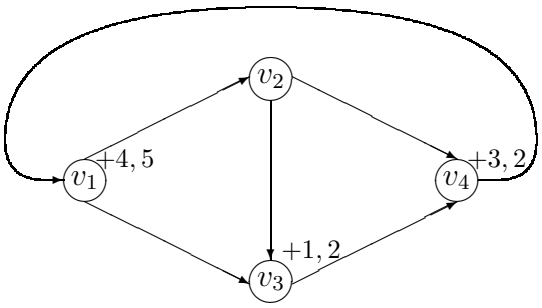


Il problema di circolazione associato è:



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

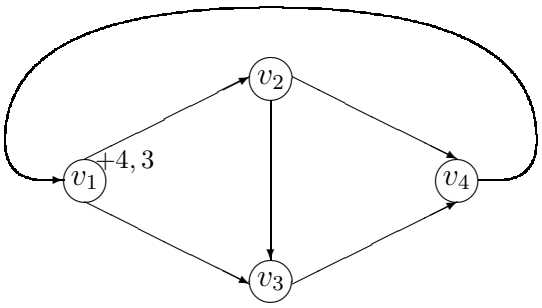
Scegliendo l'arco (v_4, v_1) con violazione massima, si ottiene la seguente etichettatura:



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

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

Scegliendo ancora l'arco (v_4, v_1) con violazione massima, si ottiene la seguente etichettatura:



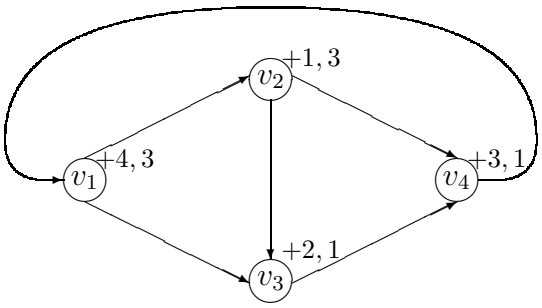
Non essendo stato determinato un ciclo si modificano i potenziali:

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

ottenendo:

potenziali	$y_1 = 1$	$y_2 = 0$	$y_3 = 0$	$y_4 = 0$	
costi ridotti	$c^*_{12} = 0$	$c^*_{13} = 2$	$c^*_{23} = 4$	$c^*_{24} = 2$	$c^*_{34} = 1$	$c^*_{41} = 1$

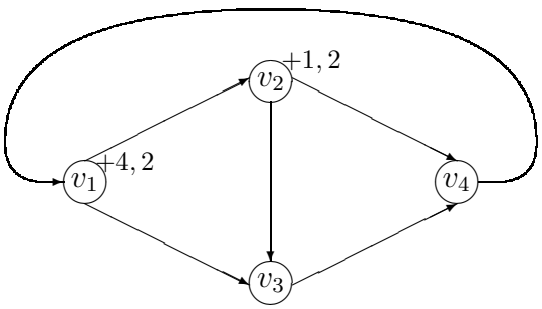
L'arco (v_4, v_1) rimane non ammissibile e si prosegue ad etichettare, ottenendo:



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

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 l'arco (v_4, v_1) con violazione massima, si ottiene la seguente etichettatura:



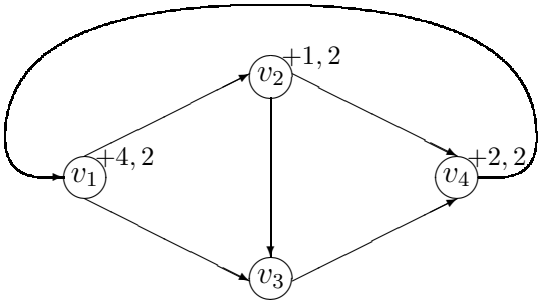
Non essendo stato determinato un ciclo si modificano i potenziali:

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

ottenendo:

potenziali	$y_1 = 3$	$y_2 = 2$	$y_3 = 0$	$y_4 = 0$		
costi ridotti	$c^*_{12} = 0$	$c^*_{13} = 0$	$c^*_{23} = 2$	$c^*_{24} = 0$	$c^*_{34} = 1$	$c^*_{41} = 3$

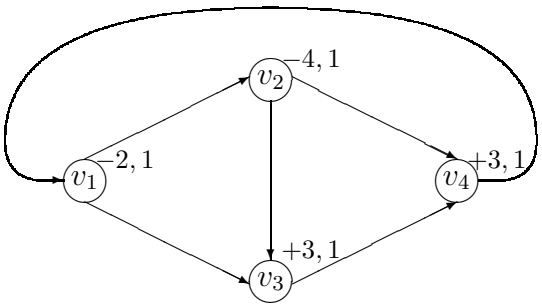
L'arco (v_4, v_1) rimane non ammissibile e si prosegue ad etichettare, ottenendo:



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

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	$c^*_{12} = 0$	$c^*_{13} = 2$	$c^*_{23} = 2$	$c^*_{24} = 0$	$c^*_{34} = 1$	$c^*_{41} = 3$
Kilter number	$k_{12} = 0$	$k_{13} = 0$	$k_{23} = 0$	$k_{24} = 0$	$k_{34} = 1$	$k_{41} = 0$

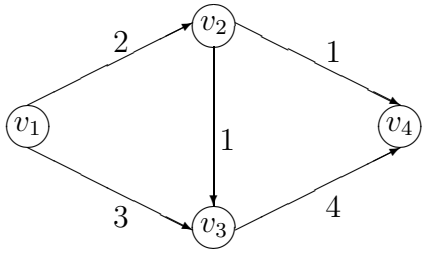
Scegliendo l'arco (v_3, v_4) con violazione massima, si ottiene la seguente etichettatura:



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

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	$c^*_{12} = 0$	$c^*_{13} = 2$	$c^*_{23} = 2$	$c^*_{24} = 0$	$c^*_{34} = 1$	$c^*_{41} = 3$
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 $2 \cdot 1 + 3 \cdot 3 + 1 \cdot 4 + 1 \cdot 2 + 4 \cdot 1 = 21$. $\diamondsuit$

Capitolo 9

Scheduling

L'ordinamento di una sequenza di lavori può essere analizzata al fine di minimizzare i costi dovuti ai tempi di esecuzione. La pianificazione della produzione o l'utilizzo di una risorsa (macchina) possono risultare notevolmente migliorati grazie ad una schedulazione ottimale.

La situazione può essere formalizzata nel seguente modo:

E' dato un insieme $N = \{1, ..., n\}$ di lavori che devono essere eseguiti sequenzialmente, per ciascuno dei quali sono noti i tempi di esecuzione $t_1, ..., t_n$ e i costi per unità di tempo $\alpha_1, ..., \alpha_n$; è inoltre assegnato un ordine iniziale dei lavori σ_0.

9.1 Modello di base

Il modello più semplice di problema di scheduling o sequenziamento suppone che esista una sola sequenza preassegnata di lavori, che questi debbano essere eseguiti ininterrottamente, cioè non è possibile iniziare un lavoro e poi interromperlo per riprenderlo successivamente, infine si suppone che il costo dipende linearmente dal tempo in cui il lavoro viene completato, ipotizzando che il prodotto e il relativo profitto siano immediatamente disponibili non appena il lavoro termina.

Questa situazione è tipica di un servizio fornito ad uno sportello dove ogni utente attende il suo turno e non appena è stato servito ha conseguito il suo scopo. Sulla base di questo criterio, assegnato un ordinamento σ ogni lavoro $i \in N$ attende l'esecuzione dei lavori che lo precedono $P(\sigma, i)$ e ogni unità di tempo viene valutata al costo α_i, cioè $\alpha_i \sum_{j \in P(\sigma,i)} t_j$. Globalmente il costo C_σ associato all'ordinamento σ è:

$$C_\sigma = \sum_{i \in N} \alpha_i \sum_{j \in P(\sigma,i)} t_j$$

Esempio 9.1.1 (Scheduling) *L'officina di un'azienda deve riparare 4 macchinari per i quali è noto il tempo necessario per la riparazione e il costo unitario derivante dal non*

utilizzo del macchinario:

macchinario	A	B	C	D
tempo di riparazione	1	3	2	4
costo di fermo	2	1	3	9

Se i macchinari vengono riparati nell'ordine in cui sono stati consegnati all'officina il costo della riparazione è:

$$1 * 2 + (1 + 3) * 1 + (1 + 3 + 2) * 3 + (1 + 3 + 2 + 4) * 9 = 2 + 4 + 18 + 90 = 114$$

Poichè i macchinari appartengono tutti alla stessa azienda è possible cercare di ridurre i costi utilizzando un diverso ordine di intervento. Riparando i macchinari per tempi di riparazione crescenti si ha l'ordine A - C - B - D a cui corrisponde il costo di riparazione:

$$1 * 2 + (1 + 2) * 3 + (1 + 2 + 3) * 1 + (1 + 2 + 3 + 4) * 9 = 2 + 9 + 6 + 90 = 107$$

Ordinando i macchinari per costi di fermo decrescenti $(D - C - A - B)$ *il costo di riparazione si riduce a:*

$$4 * 9 + (4 + 2) * 3 + (4 + 2 + 1) * 2 + (4 + 2 + 1 + 3) * 1 = 36 + 18 + 14 + 10 = 78$$

Un approccio meno empirico può migliorare il risultato? ♢

La risposta alla domanda finale dell'Esempio 9.1.1 è affermativa.

Nel 1956 Smith propone una soluzione basata sull'indice di urgenza dato dal rapporto tra costo di fermo e tempo di riparazione, cioè $u_i = \dfrac{\alpha_i}{t_i}, i \in N$; una volta ottenuti gli indici di urgenza è sufficiente ordinare i lavori secondo indici decrescenti.

Esempio 9.1.2 (Scheduling - Seconda parte) *Nel caso precedente gli indici di urgenza sono rispettivamente* $2, \frac{1}{3}, \frac{3}{2}, \frac{9}{4}$ *e l'ordine per indici decrescenti è* $D - A - C - B$ *a cui corrisponde il costo di riparazione:*

$$4 * 9 + (4 + 1) * 2 + (4 + 1 + 2) * 3 + (4 + 1 + 2 + 3) * 1 = 36 + 10 + 21 + 10 = 77$$

che è il miglior risultato ottenibile (soluzione ottimale). ♢

Per comprendere il metodo di Smith si possono considerare due ordinamenti σ^1 e σ^2 che differiscono solo per la posizione di due lavori consecutivi i e j, in particolare in σ^1 i precede j e in σ^2 i segue j. Applicando il criterio di costo suddetto la differenza di costo tra i due ordinamenti è data dalla variazione di costo dei lavori i e j:

$$\begin{aligned} C_{\sigma^1} - C_{\sigma^2} &= \alpha_i \sum_{k \in P(\sigma^1, i)} t_k + \alpha_j \sum_{k \in P(\sigma^1, j)} t_k - \alpha_j \sum_{k \in P(\sigma^2, j)} t_k - \alpha_i \sum_{k \in P(\sigma^2, i)} t_k \\ &= \alpha_j t_i - \alpha_i t_j = t_i t_j \left(\frac{\alpha_j}{t_j} - \frac{\alpha_i}{t_i} \right) \\ &= t_i t_j (u_j - u_i) \end{aligned}$$

La variazione è positiva, cioè l'ordinamento σ^2 è più conveniente se $u_j > u_i$ e quindi se il lavoro j ha un indice di urgenza maggiore del lavoro i deve precederlo.

9.2 Modelli evoluti

Ulteriori modelli sono stati definiti, variando i parametri del modello base.

- *Sequenza*
  Possono essere disponibili più macchine invece di una; le macchine possono essere identiche oppure differenti; un lavoro deve essere processato da più macchine e l'ordine può essere prestabilito o meno.

- *Lavori*
  Un lavoro può essere eseguito solo dopo un certo istante (*release time*) oppure essere completato prima di un certo instante inderogabilmente (*deadline*) o paga una penale sul ritardo (*due date*); i lavori devono rispettare le precedenze assegnate.

- *Costi*
  Il costo può dipendere dal massimo ritardo sui lavori o dalla somma dei ritardi; si può assegnare un costo anche all'anticipo rispetto ad un dato istante.

Capitolo 10

Pianificazione di attività

10.1 Problema di progettazione

Dato un progetto composto da n attività, per le quali sono note le durate, $d_i, i = 1, ..., n$, e le relazioni di precedenza, si vogliono determinare la durata minima del progetto e per ogni attività l'istante $t_i, i = 1, ..., n$ in cui può cominciare, in quanto sono terminate tutte le attività precedenti, e l'istante $T_i, i = 1, ..., n$ entro cui deve cominciare per non ritardare la realizzazione del progetto.

Esempio 10.1.1 (Descrizione di un progetto) *Sia dato il progetto rappresentato dal seguente schema:*

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

$\diamondsuit$

La teoria dei grafi permette di modellizzare e risolvere il problema. Esistono due tipi di modelli, che permettono di ottenere le stesse informazioni, ma si differenziano in quanto il primo, noto come CPM (Critical Path Method), fa corrispondere alle attività i nodi del grafo, mentre il secondo, noto come PERT (Project Evaluation and Review Technique), fa corrispondere alle attività gli archi del grafo e può richiedere di inserire ulteriori archi (*attività fittizie*) per rappresentare correttamente le precedenze.

10.2 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; in questo modo il nodo v_α è l'unico senza predecessori e il nodo v_ω è l'unico senza successori.
  - il grafo è aciclico per costruzione, in quanto 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*, cioè il ritardo che l'attività i può avere senza che il progetto sia ritardato;
  - le attività per cui $T_i = t_i$ sono dette *attività critiche*, in quanto qualsiasi loro ritardo si ripercuote sulla conclusione del progetto;
  - 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 rappresenta il costo del cammino più lungo da v_α a v_i;
    essendo il grafo aciclico si può calcolarlo con la seguente formula ricorsiva:

$$t_\alpha = 0$$
$$t_i = max\ \{t_j + d_j; v_j \in \omega^-(i)\}$$

  - $T_\omega - T_i$ rappresenta il costo del cammino più lungo da v_i a v_ω;
    essendo il grafo aciclico si può calcolarlo con la seguente formula ricorsiva:

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

- **Cammino critico**

– a questo punto si determinano i ritardi massimi $m_i = T_i - t_i$, si identificano le attività critiche e i cammini critici.

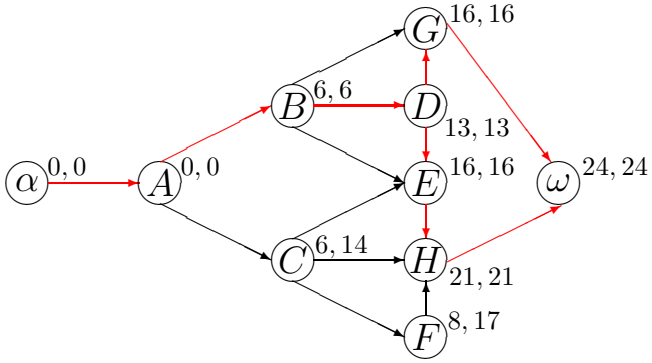
Osservazione 10.2.1

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

$$t_i = max\ \{t_j + d_j; j < i\}$$
$$T_i = min\ \{T_j - d_i; j > i\}$$

- *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 10.2.1 (Cammino critico) *Riferendosi all'Esempio 10.1.1 si ottengono i seguenti risultati:*



Il tempo minimo per completare il progetto è 24, le attività critiche sono A, B, D, E, G, H che danno origine ai cammini critici $A - B - D - G$ e $A - B - D - E - H$.
Si noti che anche i cammini $A-B-G$ e $A-B-E-H$ passano solo per nodi corrispondenti ad attività critiche. ♢

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

Il secondo modello è sostanzialmente simile al precedente, salvo che in questo caso le attività corrispondono agli archi del grafo, mentre i nodi corrispondono agli eventi di fine delle attività associate agli archi entranti e di inizio delle attività associate agli archi uscenti; anche in questo caso si determinano per ogni nodo due valori, t_i e T_i che rappresentano gli istanti tra i quali deve verificarsi quell'evento.

- **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.

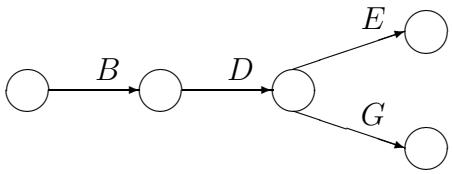
Osservazione 10.3.1

- *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 10.3.1 (PERT) *Riferendosi all'Esempio 10.1.1 si può osservare che:*

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

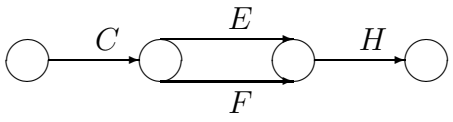
cioè:



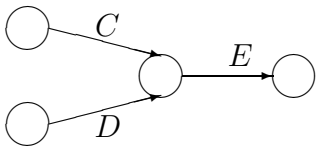
Inoltre si ha:

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

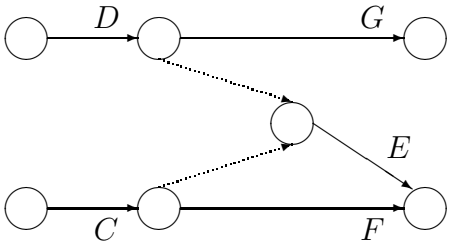
cioè:



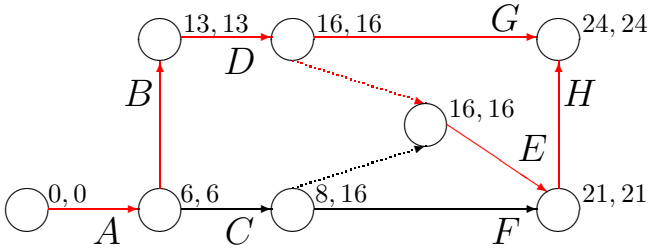
Le relazioni $C \prec E$ e $D \prec E$ non possono essere unificate come:



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. ♢

Osservazione 10.3.2

- *Il modello PERT è in generale più complesso da realizzare, soprattutto per la difficoltà di esprimere le relazioni di precedenza, con un numero limitato di attività fittizie; d'altra parte risulta più semplice tenere conto di variazioni nella durata delle attività, in quanto è sufficiente modificare la lunghezza di un solo arco.*

- *Il modello CPM è più vantaggioso se è necessario introdurre vincoli non standard, in quanto è sufficiente aggiungere 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*.*

Indice