Contents

1 Algebra Lineare

1.1 Spazi vettoriali

Definizione 1.1 *Un insieme K dotato di due operazioni dette somma e prodotto è detto campo se la somma è associativa, commutativa, ha un elemento neutro ed ammette l'opposto, e il prodotto è associativo, commutativo, ha un elemento neutro, ammette l'inverso ed è distributivo rispetto alla somma*

Definizione 1.2 *Uno spazio vettoriale su un campo K, i cui elementi sono detti* scalari, *è un insieme V, i cui elementi sono detti* vettori, *dotato di una operazione interna $+ : V \times V \to V$, detta* somma di vettori *e di una esterna $\cdot : K \times V \to V$, detta* prodotto di uno scalare per un vettore *e si indica $(V, +, \cdot)$, per cui valgono i seguenti assiomi:*

1. *$(V, +)$ è un gruppo abeliano, cioè valgono le proprietà:*
   - *a) Commutativa:* $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u} \quad \forall\, \mathbf{u}, \mathbf{v} \in V$
   - *b) Associativa:* $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w}) \quad \forall\, \mathbf{u}, \mathbf{v}, \mathbf{w} \in V$
   - *c) Elemento neutro:* $\exists\, \mathbf{0} \in V$ *per cui* $\mathbf{u} + \mathbf{0} = \mathbf{0} + \mathbf{u} = \mathbf{u} \quad \forall\, \mathbf{u} \in V$
   - *d) Elemento opposto:* $\exists - \mathbf{u} \in V$ *per cui* $\mathbf{u} + (-\mathbf{u}) = (-\mathbf{u}) + \mathbf{u} = \mathbf{0} \quad \forall\, \mathbf{u} \in V$
2. *per il prodotto scalare per vettore valgono le proprietà:*
   - *e) Distributiva rispetto alla somma di vettori:* $\lambda \cdot (\mathbf{u} + \mathbf{v}) = \lambda \cdot \mathbf{u} + \lambda \cdot \mathbf{v} \quad \forall\, \lambda \in K, \forall\, \mathbf{u}, \mathbf{v} \in V$
   - *f) Distributiva rispetto alla somma di scalari:* $(\lambda + \mu) \cdot \mathbf{u} = \lambda \cdot \mathbf{u} + \mu \cdot \mathbf{u} \quad \forall\, \lambda, \mu \in K, \forall\, \mathbf{u} \in V$
   - *g) Associativa rispetto al prodotto di scalari:* $\lambda \cdot (\mu \cdot \mathbf{u}) = (\lambda\mu) \cdot \mathbf{u} \quad \forall\, \lambda, \mu \in K, \forall\, \mathbf{u} \in V$
   - *h) Unitaria (elemento scalare neutro):* $\exists\, 1 \in K$ *per cui* $1 \cdot \mathbf{u} = \mathbf{u} \quad \forall\, \mathbf{u} \in V$

Esempio 1.1 *Prendendo come campo K l'insieme dei numeri reali $\mathbb{R}$ e come insieme V ancora $\mathbb{R}$, cone le usuali operazioni di somma e prodotto di numeri reali, si ottiene lo spazio vettoriale $\mathbb{R}$ e poiché il prodotto cartesiano di spazi vettoriali sullo stesso campo è ancora uno spazio vettoriale, anche $\mathbb{R}^n$ è uno spazio vettoriale per ogni intero n* $\diamondsuit$

Proprietà

- Il vettore nullo è unico
- Il vettore opposto di $\mathbf{u}$ è unico
- Semplificazione rispetto alla somma: $\mathbf{u} + \mathbf{w} = \mathbf{v} + \mathbf{w} \Rightarrow \mathbf{u} = \mathbf{v}$
- Annullamento del prodotto: $\lambda \cdot \mathbf{u} = \mathbf{0} \iff \lambda = 0$ oppure $\mathbf{u} = \mathbf{0}$
- $(-\lambda) \cdot \mathbf{u} = -(\lambda \cdot \mathbf{u}) = \lambda \cdot (-\mathbf{u})$

1.2 Sottospazi vettoriali

Sia V spazio vettoriale su K; un sottoinsieme $U \subseteq V$ è un *sottospazio vettoriale di* V se ha la struttura di spazio vettoriale su K rispetto alla restrizione delle operazioni

Teorema 1.1 *U è un sottospazio vettoriale di V se e solo se:*

1. $\mathbf{u} + \mathbf{v} \in U \quad \forall\, \mathbf{u}, \mathbf{v} \in U$
2. $\lambda \cdot \mathbf{u} \in U \quad \forall\, \lambda \in K, \forall\, \mathbf{u} \in U$

oppure:

$\lambda \cdot \mathbf{u} + \mu \cdot \mathbf{v} \in U \quad \forall\, \lambda, \mu \in K, \forall\, \mathbf{u}, \mathbf{v} \in U$

Proprietà

Se U e W sono sottospazi vettoriali di V lo è anche $U \cap W$, mentre in generale non lo è $U \cup W$

Esempio 1.2 *Dato lo spazio vettoriale $\mathbb{R}^n$ sono sottospazi vettoriali tutti gli insiemi $\mathbb{R}^i, i = 0, ..., n$*

$\mathbb{R}^0 = \{0\}$ ♢

1.3 Dipendenza lineare

Definizione 1.3

- *Dati* n *vettori* $\mathbf{v}_1, \cdots, \mathbf{v}_n \in V$ *spazio vettoriale su un campo* K *e* n *scalari* $\lambda_1, \cdots, \lambda_n \in K$ *il vettore* $\mathbf{u} = \lambda_1\mathbf{v}_1 + \cdots + \lambda_n\mathbf{v}_n$ *si dice* combinazione lineare dei vettori $\mathbf{v}_1, \cdots, \mathbf{v}_n$ secondo gli scalari $\lambda_1, \cdots, \lambda_n$
- n *vettori* $\mathbf{v}_1, \cdots, \mathbf{v}_n \in V$ *si dicono* linearmente indipendenti *se l'unica combinazione lineare che da il vettore nullo ha tutti i coefficienti nulli*
- n *vettori* $\mathbf{v}_1, \cdots, \mathbf{v}_n \in V$ *si dicono* linearmente dipendenti *se esiste una combinazione lineare che da il vettore nullo avente non tutti i coefficienti nulli*
- *Un insieme di* n *vettori* $\mathbf{v}_1, \cdots, \mathbf{v}_n \in V$ *linearmente indipendenti si dice* libero

Proprietà

$\mathbf{v}_1, \cdots, \mathbf{v}_n \in V$ sono linearmente dipendenti se e solo se uno di essi è combinazione lineare degli altri

Sia $A = \{\mathbf{a}_1, \cdots, \mathbf{a}_n\} \subseteq V$; si dice *sottospazio vettoriale generato da* A l'insieme di tutte le combinazioni lineari dei vettori di A e si indica con $\mathcal{L}(A)$; i vettori $\mathbf{a}_1, \cdots, \mathbf{a}_n$ sono detti *generatori di* $\mathcal{L}(A)$

Proprietà fondamentale

Sia $A = \{\mathbf{a}_1, \cdots, \mathbf{a}_n\}$ un insieme libero, allora ogni vettore $\mathbf{v} \in \mathcal{L}(A)$ si esprime in modo unico come combinazione lineare dei vettori di A

1.4 Base e dimensione

Un insieme $B = \{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$ libero di generatori di V è detto *base di* V

Proprietà

- Ogni vettore di V si scrive in modo unico come combinazione lineare dei vettori di una base
- Un insieme B di generatori di V è una base di V se ogni vettore di V si scrive in modo unico come combinazione lineare dei vettori di B

I coefficienti della combinazione lineare dei vettori di una base B che danno $\mathbf{v} \in V$ sono detti *componenti di* $\mathbf{v}$ *nella base* B
Se $\mathbf{v} = v_1\mathbf{b}_1 + \cdots + v_n\mathbf{b}_n$ allora $\mathbf{v} = (v_1, \cdots, v_n)$

Prodotto scalare di due vettori di V

Definizione 1.4 *Dati* $\mathbf{u}$ *e* $\mathbf{v} \in V$ *ad essi si associa uno scalare del campo* K *detto prodotto scalare e si indica con* $< \mathbf{u}, \mathbf{v} >$ *che soddisfa le seguenti condizioni:*

- *Simmetria:* $< \mathbf{u}, \mathbf{v} >=< \mathbf{v}, \mathbf{u} > \quad \forall\ \mathbf{u}, \mathbf{v} \in V$
- *Linearità:* $< a\mathbf{u} + b\mathbf{v}, \mathbf{w} >= a < \mathbf{u}, \mathbf{w} > + b < \mathbf{v}, \mathbf{w} > \quad \forall\ a, b \in K, \forall\ \mathbf{u}, \mathbf{v}, \mathbf{w} \in V$
- $< \mathbf{u}, \mathbf{u} >= 0 \iff \mathbf{u} = \mathbf{0}$

in $\mathbb{R}^n$ si ha anche:
Positività: $< \mathbf{u}, \mathbf{u} > \geq 0 \quad \forall\ \mathbf{u} \in V$
$\sqrt{< \mathbf{u}, \mathbf{u} >}$ è detto norma o modulo di $\mathbf{u}$
Il più comune prodotto scalare su $\mathbb{R}^n$ è definito come

$$< \mathbf{u}, \mathbf{v} >= u_1 v_1 + \cdots + u_n v_n$$

- Se $< \mathbf{u}, \mathbf{v} >$ è nullo i vettori $\mathbf{u}$ e $\mathbf{v}$ sono detti *ortogonali*
- Se i vettori di una base sono a due a due ortogonali la base è detta *ortogonale*; se i vettori hanno norma unitaria è detta *ortonormale*

Teorema 1.2 (Esistenza della base) *Dato un sistema di generatori $A = \{\mathbf{a}_1, \cdots, \mathbf{a}_m\}$, è sempre possibile estrarre da A una base di V*

Per ogni spazio vettoriale V esiste sempre una base e inoltre esistono infinite basi

Lemma 1.1 *Siano V uno spazio vettoriale su un campo K e $B = \{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$ una base di V, allora ogni insieme libero contiene al più n vettori*

Corollario 1.1 *Tutte le basi di un dato spazio vettoriale V hanno lo stesso numero di vettori*

Teorema 1.3 (Completamento della base) *Siano $B = \{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$ una base di V e $A = \{\mathbf{a}_1, \cdots, \mathbf{a}_k\}$ un insieme libero, quindi $k \leq n$, allora è sempre possibile aggiungere ad A $n - k$ vettori di B in modo da ottenere una nuova base di V*

Il numero di vettori di una base è detto *dimensione di V* e si indica con $dim\ V$ o $dim(V)$

Dato uno spazio vettoriale V con $dim\ V = n$, ogni sottospazio vettoriale di V di dimensione $n - 1$ è detto *iperpiano vettoriale* o semplicemente iperpiano

Un iperpiano vettoriale è individuato dai vettori $\mathbf{x} = (x_1, \cdots, x_n) \in V$ che soddisfano una equazione lineare $a_1x_1 + \cdots + a_nx_n = 0$ con $\mathbf{a} = (a_1, \cdots, a_n) \in V$

Proprietà

- $dim(\mathbb{R}^n) = n$
- Se $dim(V) = n$ allora $n + 1$ vettori di V sono linearmente dipendenti
- Se B è un insieme di n generatori di V con $dim(V) = n$ allora B è base di V
- Se B è un insieme libero di n vettori di V con $dim(V) = n$ allora B è base di V
- Se W è un sottospazio di V allora $dim(W) \leq dim(V)$

2 Programmazione lineare

2.1 Modelli matematici

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

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

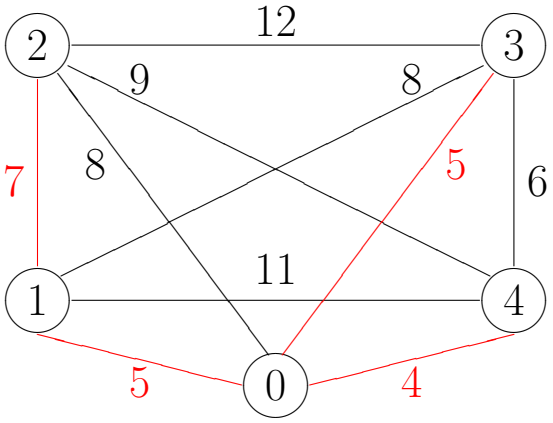
- Modelli su reti
  - Albero di costo minimo
  - Cammino minimo
  - Flusso massimo
  - Routing e scheduling
  - Localizzazione (*Location*)
  - PERT (*Project Evaluation and Review Techniques*)

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

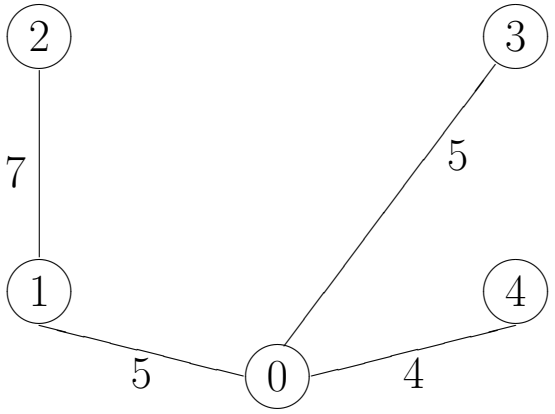
Esempio 2.1 (Connessione - Progettazione e gestione)

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

PROGETTAZIONE



GESTIONE



♢

2.2 Problema di programmazione matematica

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

Forma generale

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

oppure:

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

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

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

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

Riduzione alla forma generale

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

Soluzioni ammissibili

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

= insieme delle soluzioni ammissibili o insieme ammissibile o regione ammissibile

Può essere vuoto, limitato o illimitato

Soluzioni ottimali

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

= insieme delle soluzioni ottimali o insieme ottimale o regione ottimale

Può essere vuoto, limitato o illimitato

Soluzione di un problema di programmazione matematica

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

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

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

2.3 Problema di programmazione lineare

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

Forma canonica

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

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

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

Riduzione alla forma canonica

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

Soluzioni ammissibili

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

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

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

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

Un punto di S_a intersezione unica di più di n iperpiani generatori si dice vertice 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
I due vertici estremi di uno spigolo limitato si dicono adiacenti

Soluzioni ottimali

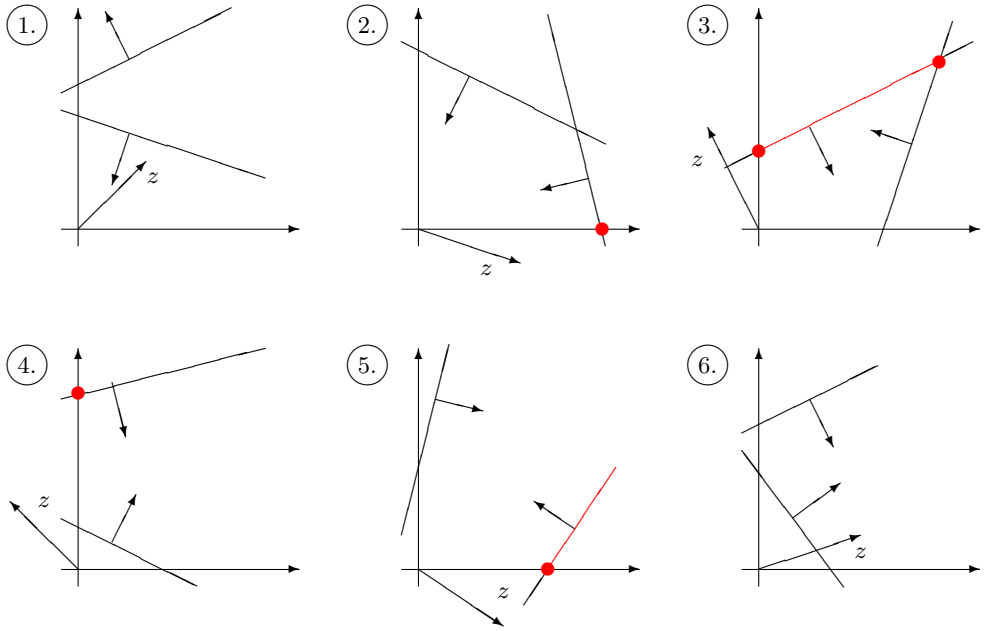
S_{ott} è insieme convesso ed è contenuto nella frontiera di S_a, se z non è costante, e può essere:

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

Se S_a è vuoto non esistono soluzioni ottimali

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

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



2.4 Teorema fondamentale

Teorema 2.1 (Teorema fondamentale della programmazione lineare)
Se un problema lineare ha soluzioni ottimali almeno una di esse è un vertice di S_a

Dimostrazione

Si possono distinguere due casi:

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

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

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

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

Posto $z_i = z(x_i)$ e con $z^* = z(x^*)$, 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 T può essere scritto come somma di un poliedro convesso K_1, avente gli stessi vertici di T, e un cono poliedrico K_0, avente gli spigoli paralleli agli spigoli illimitati di T:

$$T = K_1 + K_0$$

Siano $u_1, \ldots, u_p$ i vertici di K_1 e $v_1, \ldots, v_q$ i generatori di K_0
Sia x^* una soluzione ottimale del problema lineare che può essere scritta come:

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

A loro volta u e v possono essere scritti come:

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

quindi:

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

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

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

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

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

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

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

♣

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

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

  $z^* = 1$, $S_{ott} = \{(x_1, x_2) \in \mathbb{R}^2 | x_1 + x_2 = 1\}$ ma non esistono vertici
- Esiste un teorema, analogo ma meno importante, relativo alle soluzioni ammissibili:
  Se un problema lineare ha soluzioni ammissibili almeno una di esse è un vertice di S_a

Un problema lineare presenta uno dei seguenti casi mutualmente esclusivi:

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

1 e 3 si dicono casi di impossibilità

2.5 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 $v_1, \ldots, v_n$

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

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

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

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

e sostituendo si ha:

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

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

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

Rappresentazione tabellare:

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

Facendo cardine sull'elemento λ_{ij}, cioè scambiando i vettori a_i e v_j, si ha:

	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	$\lambda_{m1} - \lambda_{i1}\frac{\lambda_{mj}}{\lambda_{ij}}$	$\dots$	$\frac{\lambda_{mj}}{\lambda_{ij}}$	$\dots$	$\lambda_{mn} - \lambda_{in}\frac{\lambda_{mj}}{\lambda_{ij}}$

inverso del cardine

elemento diviso il cardine

elemento diviso il cardine cambiato di segno

elemento meno il prodotto degli elementi sulla riga del cardine colonna dell'elemento e sulla colonna del cardine riga dell'elemento diviso il cardine

2.6 Algoritmo del simplesso - Dantzig (1947)

Dato un problema lineare in forma canonica:

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

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

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

Rappresentazione tabellare:

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

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

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

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

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

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

2.6.1 Ricerca della soluzione ottimale

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

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

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

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

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

Scelta della variabile uscente

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

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

Scelta della variabile entrante

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

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

Per l'ammissibilità:

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

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:

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

- 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

2.6.2 Interpretazione geometrica

Scelta dell'iperpiano generatore uscente

I coefficienti della funzione obiettivo rappresentano le componenti del gradiente secondo la base corrente

z è crescente lungo gli spigoli con y_i crescente se γ_i è positivo

Scelta dell'iperpiano generatore entrante

Scegliendo $y_i = 0$ come iperpiano generatore uscente di base lo spostamento dal vertice corrente al vertice ammissibile adiacente lungo lo spigolo $y_h = 0, h \neq i$ si ottiene facendo entrare in base il primo iperpiano generatore non in base

Il punto di intersezione dell'iperpiano generatore $y_{n+k} = 0, k = 1, \ldots, m$ 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

2.7 Terminazione

2.7.1 Ottimalità

Dall'ultima riga si ricava:

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

Se $\gamma_i \leq 0, i = 1, \ldots, n$ allora in tutta la regione ammissibile il valore di z è non migliore di quello corrente
Si definisce *tabella ottimale* una tabella che ha l'ultima riga, tranne al più l'ultimo elemento, tutta non positiva e l'ultima colonna, tranne al più l'ultimo elemento, tutta non negativa

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

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

Esempio 2.2 *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:

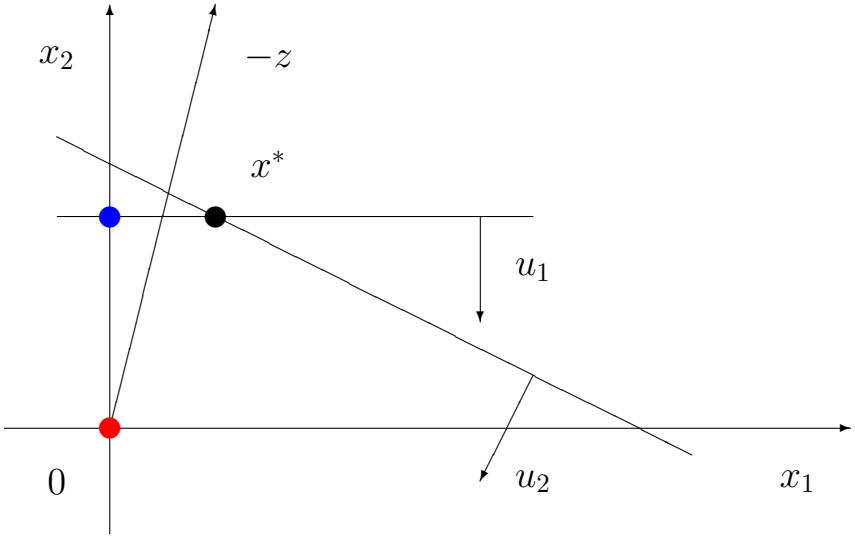
	x_1	x_2	
u_1	0	$\boxed{-1}$	2
u_2	-1	-2	5
$-z$	1	4	0

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

	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	2
x_1	-1	2	1
$-z$	-1	-2	9



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

2.8 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

2.9 Ricerca di una tabella ammissibile

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

In questo caso si applica un diverso criterio per determinare le variabili da scambiare
Per la non ammissibilità esiste almeno un $\beta_j < 0$
Si cerca un $\alpha_{ji} > 0$ e si determina così la variabile uscente y_i
Scegliendo come cardine $\alpha_{ki} < 0$ con $\beta_k > 0$ la variabile entrante è quella per cui si ha il minimo di $\left|\frac{\beta_k}{\alpha_{ki}}\right|$, in modo che tutti i termini noti non negativi restino non negativi
Facendo cardine $\alpha_{ki} > 0$ con $\beta_k < 0$ la variabile entrante è quella per cui si ha il massimo di $\left|\frac{\beta_k}{\alpha_{ki}}\right|$, in modo che tutti i termini noti negativi corrispondenti a coefficienti positivi nella colonna della variabile uscente diventino non negativi

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

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

Esempio 2.3 *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

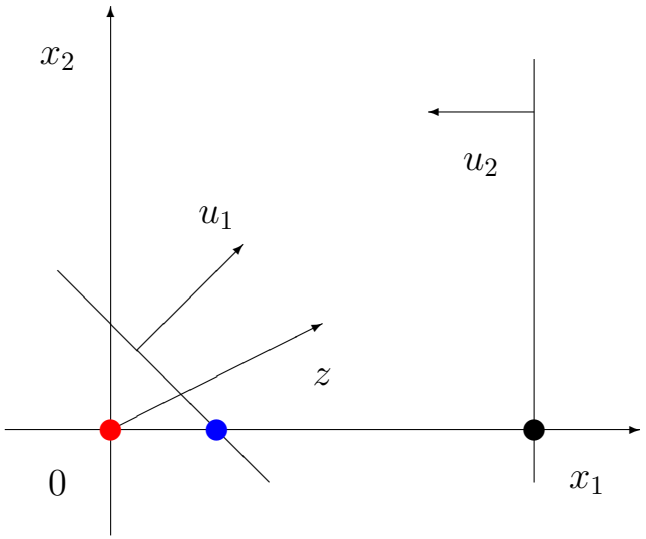
$x = (0,0), z = 0$

	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 a $\beta_j < 0$ corrispondono $\alpha_{ji} \leq 0, i = 1, \ldots, n$ allora non esistono soluzioni ammissibili perchè 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

- 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

2.10 Casi particolari

2.10.1 Soluzione degenere e ciclaggio

Se in una tabella esiste $\beta_j = 0$, il corrispondente iperpiano $y_{n+j} = 0$ pur non formando la base, passa ugualmente per il vertice rappresentato dalla tabella, che risulta degenere
Se y_{n+j} viene scelta come variabile entrante i valori delle variabili restano invariati e la nuova tabella rappresenta lo stesso vertice

Oltre a rimanere nello stesso vertice, possono presentarsi ciclicamente le stesse basi
Questo fenomeno detto ciclaggio può essere evitato utilizzando diversi metodi, ad esempio la *regola di Bland* e il *metodo delle perturbazioni di Wolfe*

2.10.2 Infinite soluzioni ottimali

Se $\gamma_i = 0$ incrementando la variabile y_i la funzione obiettivo non varia; pertanto se la tabella è ottimale scegliendo y_i come variabile uscente è possibile determinare un nuovo vertice ottimale (salvo nel caso di vertice degenere)
Per la linearità del problema vuol dire che tutti i punti dello spigolo sono ottimali
Se tutti gli $\alpha_{ji}, j = 1, \ldots, m$ sono non negativi allora esiste uno spigolo illimitato di punti ottimali

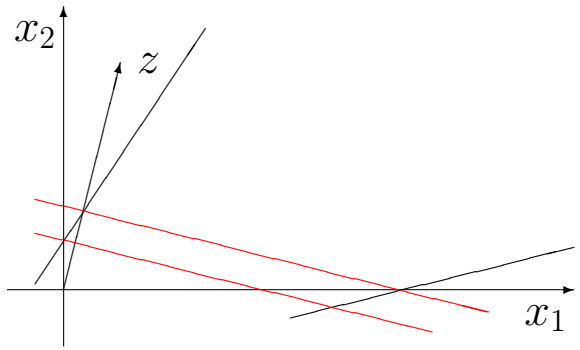
2.10.3 Problemi con vincoli di uguaglianza

Sono possibili differenti metodologie:

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

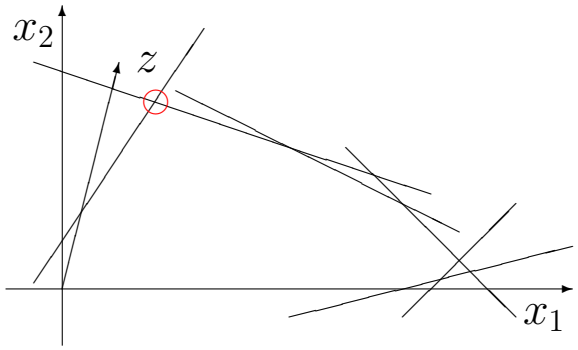
2.10.4 Scelta della variabile uscente

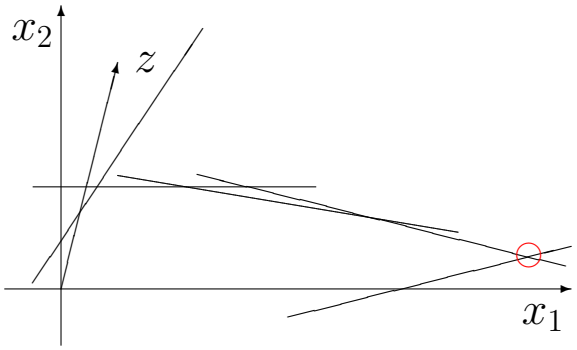
La scelta dell'elemento su cui "fare cardine" riveste un ruolo molto importante, ma non è possibile fissare regole generali, visto che la tabella rappresenta una situazione "locale"



Componente maggiore del gradiente o incremento maggiore della funzione obiettivo?

Dipende ...





3 Dualità

3.1 Problema duale

Dato il problema lineare (primale):

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

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

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

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

3.2 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:

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

   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:

$$Ax^0 \leq b,\ u^0 \geq 0 \ \Rightarrow\ (Ax^0)^T u^0 \leq b^T u^0$$
$$A^T u^0 \geq c,\ x^0 \geq 0 \ \Rightarrow\ (A^T u^0)^T x^0 \geq c^T x^0$$

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:

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

♣

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:

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

   le corrispondenti tabelle iniziali sono:

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

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

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

   Allora anche la sua trasposta cambiata di segno è ottimale

Dalla tabella ottimale del problema primale si ha:

$$max \quad z = \gamma_0$$

e dalla tabella ottimale del problema duale si ha:

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

cioè:

$$min \quad w = \gamma_0$$

per cui $max \quad z = min \quad w$ ♣

5. Se uno dei due problemi ha soluzione illimitata allora l'altro non ha soluzioni ammissibili
   Dimostrazione
   Se l'altro problema avesse una soluzione ammissibile il valore della funzione obiettivo in quel punto sarebbe un limitante per la funzione obiettivo del primo problema ♣
6. Se uno dei due problemi non ha soluzioni ammissibili allora l'altro o ha soluzione illimitata o non ha soluzioni ammissibili
   Dimostrazione
   Se l'altro problema avesse soluzioni ottimali le avrebbe anche il primo problema ♣

7. Il teorema della dualità

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

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

Dimostrazione

Se valgono le relazioni:

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

sommando membro a membro si ottiene:

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

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

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

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

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

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

Aggiungendo e togliendo la quantità $u^{*T} A x^*$ 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 ♣

- Le relazioni del II teorema della dualità vengono chiamate anche condizioni di complementarietà
- Se un vincolo di uno dei due problemi non è soddisfatto come uguaglianza allora la corrispondente variabile dell'altro problema deve essere nulla
- Se una variabile di uno dei due problemi è non nulla allora il corrispondente vincolo dell'altro problema deve essere soddisfatto come uguaglianza
- La tabella ottimale di un problema fornisce la soluzione del problema duale:

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

3.3 Interpretazione economica del problema duale

Problema di produzione

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

b	vettore delle risorse
x	vettore dei beni prodotti
A	matrice tecnologica (unità di risorsa per unità di bene prodotto)
c	vettore dei valori unitari dei beni prodotti
z	valore dei beni prodotti
vincoli	non si possono utilizzare più risorse di quelle disponibili
	quantità di beni prodotti non negative

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

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

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

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

Problema duale

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

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

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

Se una risorsa non viene completamente utilizzata il costo-ombra è nullo, altrimenti può essere positivo o nullo
Un bene non viene prodotto se il costo-ombra è superiore al suo valore, altrimenti può essere prodotto o meno

Se una risorsa non è completamente utilizzata non vi è nessun vantaggio ad averne una maggiore disponibilità, altrimenti il valore dei beni prodotti può essere incrementato disponendo di una ulteriore unità di risorsa (variabile duale non nulla) o può restare invariato (variabile duale nulla)
Se il costo-ombra delle risorse necessarie a produrre un bene è superiore al suo valore è svantaggioso produrlo, altrimenti può essere svantaggioso produrlo (variabile primale nulla) o può non esserlo (variabile primale non nulla)

Ulteriore interpretazione

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

La ditta cerca di minimizzare la spesa di acquisto

$$min \quad b^T u$$

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

$$A^T u \geq c$$

I prezzi devono essere non negativi

$$u \geq 0$$

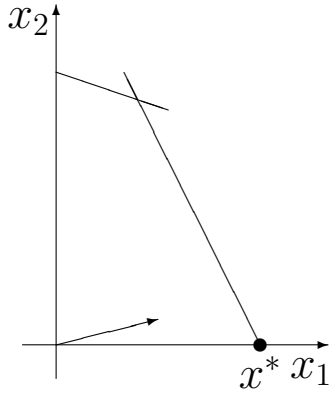
Esempio 3.1 (Modello di produzione lineare)

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

Problema primale

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

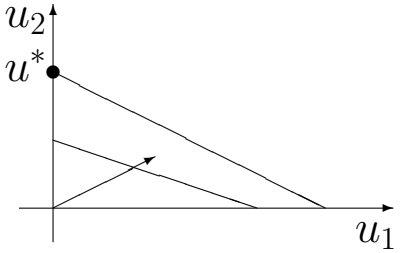
$x^* = (3, 0)$; $z^* = 12$



Problema duale

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

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



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

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

Esempio 3.2 (Significato delle variabili duali)
Sia $u_i^* \neq 0$, *cioè* $\sum_{j=1,\ldots,n} a_{ij}x_j^* = b_i$
Se $b_i^0 = b_i + 1$ *si ha:*

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

e quindi $z^{0*} = z^* + u_i^*$ ♢

Esempio 3.3 (Significato dei vincoli duali)
Sia $\sum_{j=1,\ldots,n} a_{ij}x_j^* = b_i$, $u_i^* \neq 0$ *e* $u_k^* = 0, k \neq i$
I vincoli duali si riducono a $a_{ij}u_i^* \geq c_j$ *da cui si ricava:*

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

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

Esempio 3.4 (Paradosso dell'anellino) *Un artigiano stravagante fabbrica anellini di plastica con diamanti autentici*
Dispone di un anellino del costo di 1 dollaro e di due diamanti del costo di 900 dollari
Un anellino con diamante viene venduto per 1000 dollari

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

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

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

$x^* = 1, z^* = 99$
si produce un anellino con un profitto di 99 dollari

$u_a^* = 99, u_d^* = 0, w^* = 99$
l'artigiano è disposto a pagare fino a 99 dollari in più per un altro anellino, ma non è disposto a pagare alcunchè per un altro diamante ♢

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

Problema primale di massimo	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
$x_j \geq 0$	$\sum_{i=1,\ldots,m} a_{ij}u_i \geq c_j$
$x_j \leq 0$	$\sum_{i=1,\ldots,m} a_{ij}u_i \leq c_j$
x_j libera	$\sum_{i=1,\ldots,m} a_{ij}u_i = c_j$
$\sum_{j=1,\ldots,n} a_{ij}x_j \leq b_i$	$u_i \geq 0$
$\sum_{j=1,\ldots,n} a_{ij}x_j \geq b_i$	$u_i \leq 0$
$\sum_{j=1,\ldots,n} a_{ij}x_j = b_i$	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
$x_j \geq 0$	$\sum_{i=1,\ldots,m} a_{ij}u_i \leq c_j$
$x_j \leq 0$	$\sum_{i=1,\ldots,m} a_{ij}u_i \geq c_j$
x_j libera	$\sum_{i=1,\ldots,m} a_{ij}u_i = c_j$
$\sum_{j=1,\ldots,n} a_{ij}x_j \leq b_i$	$u_i \leq 0$
$\sum_{j=1,\ldots,n} a_{ij}x_j \geq b_i$	$u_i \geq 0$
$\sum_{j=1,\ldots,n} a_{ij}x_j = b_i$	u_i libera

4 Programmazione lineare a numeri interi

4.1 Problemi lineari interi

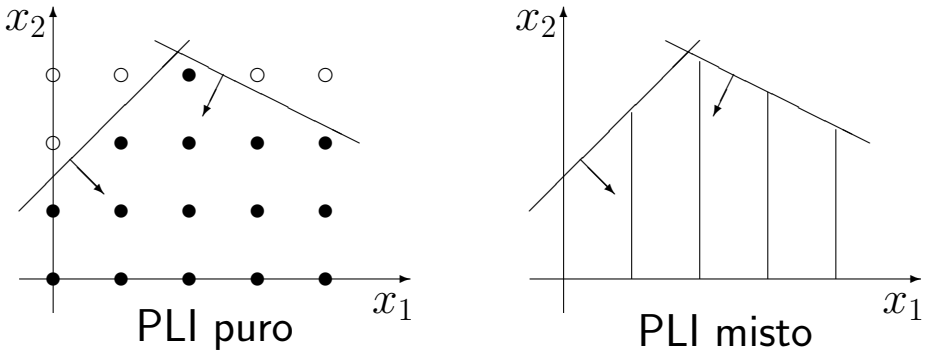
Dato il problema lineare ordinario (PLO):

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

aggiungendo la condizione di integrità:

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

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



Proprietà generali

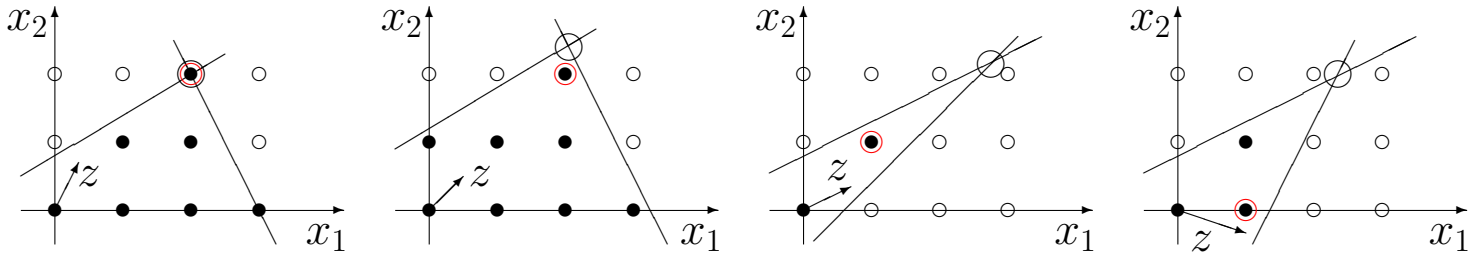
- $S'_a \subseteq S_a$ per cui

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

-

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

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



Classi di metodi risolutivi:

Metodi approssimati

- Arrotondamento o troncamento della soluzione del PLO
  Non garantisce nè l'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)

4.2 Metodi branch and bound

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

I concetti fondamentali sono rilassamento, separazione e eliminazione

Notazioni

Dato un problema P qualsiasi siano:

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

4.2.1 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

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

4.2.2 Separazione

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

- $S_a(P) = \bigcup_{i=1,\ldots,n} S_a(P_i)$
  $S_a(P_i) \cap S_a(P_j) = \varnothing \quad i \neq j$
- $z(P_i) = z(P) \quad i = 1, \ldots, n$

• È opportuno non avere una partizione troppo fine per non dover risolvere troppi sottoproblemi

Proprietà della separazione

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

4.2.3 Eliminazione

Un problema P si dice eliminato se un suo rilassamento P' verifica una delle tre condizioni seguenti:
- $E1 \quad S_a(P') = \varnothing$
- $E2 \quad z^*(P') \leq z^* \quad$ dove z^* è il valore della soluzione ottimale corrente
- $E3 \quad x^*(P') \in S_a(P)$

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

Proprietà dell'eliminazione

- $E1 \Rightarrow S_a(P) = \varnothing$
- $E2 \Rightarrow z^*(P) \leq z^*$
- $E3$ e $z(P')(x^*(P')) = z(P)(x^*(P')) \Rightarrow x^*(P') = x^*(P)$

4.2.4 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 $S_a(P') = \varnothing$ allora $S_a(P) = \varnothing$, eliminare il problema P per il criterio $E1$ e tornare a b);

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

f) se $x^*(P') \in S_a(P)$ e $z(P')(x^*(P')) = z(P)(x^*(P'))$ allora $x^*(P') = x^*(P)$, eliminare P per il criterio $E3$ e tornare a b);
se $z^*(P) > z^*$ allora porre $(x^*(P), z^*(P))$ nuova soluzione del problema, porre $z^* = z^*(P)$ e tornare a b);

g) se P non è stato eliminato decidere se abbandonare il problema;
se si abbandona separare P, aggiornare la lista dei problemi da risolvere e tornare a b);
altrimenti considerare un nuovo rilassamento P' e tornare a c);

h) se è stata trovata una soluzione $(x^*(P), z^*(P))$ questa è la soluzione ottimale;
altrimenti il problema non ammette soluzioni;

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

Gestione della lista

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

5 Modelli lineari a numeri interi

5.1 Problema dello zaino

n oggetti di peso p_i e valore c_i

Si ha a disposizione uno zaino di capienza assegnata P

Massimizzare il valore trasportato

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

5.2 Problema del trasporto

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

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

Condizione di capacità produttiva

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

5.3 Problema del magazzino

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

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

Minimizzare il costo di produzione

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

dove $y_0 = H_0$

Condizione di disponibilità

$$\sum_{j=1,\dots,i} d_j \leq H_0 + \sum_{j=1,\dots,i} C_j, \quad i = 1, ..., n$$

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

5.4 Problema dell'assegnazione

n macchine a cui assegnare un operaio tra n

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

Minimizzare i costi di assegnazione

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

5.5 Problema del commesso viaggiatore

n città da visitare senza mai ripassare da alcuna città

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

Minimizzare il costo (circuito hamiltoniano di costo minimo)

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

6 Applicazioni del metodo Branch and Bound

6.1 Problema dello zaino (Problema del massimo)

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

$peso\ massimo\ trasportabile = 30$

Rapporti valore/peso degli oggetti:

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

Ordine decrescente:

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

$peso\ massimo\ trasportabile = 30$

Problema lineare intero a variabili 0-1:

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

Bound di Dantzig

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

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

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

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

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

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

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

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

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

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

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

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

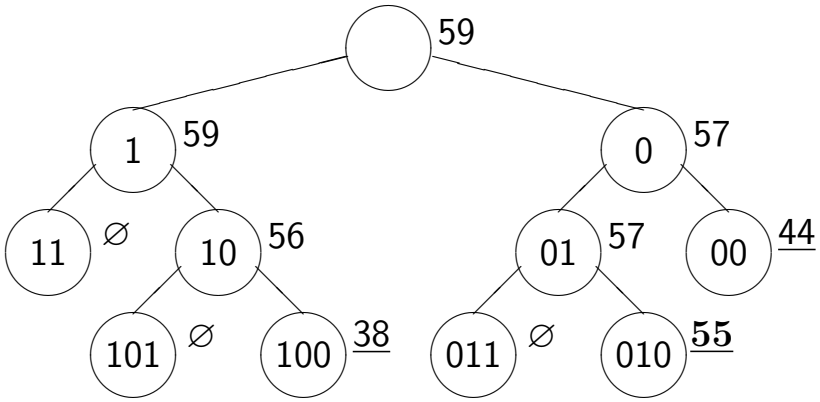
Portare C e B e non portare A $(x_1 = 1, x_2 = 0, x_3 = 1)$; portare C e non portare A e B $(x_1 = 1, x_2 = 0, x_3 = 0)$

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

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

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

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

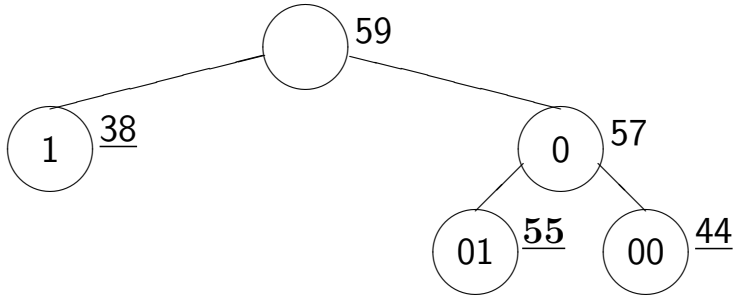


Miglioramento del calcolo della limitazione

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

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

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



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

- Si consideri il seguente problema dello zaino:

$oggetto$	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:

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

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

6.2 Problema dell'assegnazione (problema di minimo)

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

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

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

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

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

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

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

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

Assegnare l'operaio D alla macchina b; non assegnarlo

- $(not\ D - b)$

$$L = 51 + 4 = 55$$

- $(D - b)$

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

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

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

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

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

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

Assegnare l'operaio D alla macchina a; non assegnarlo

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

$$L = 55 + 3 = 58$$

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

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

$$L = 55 + 2 = 57$$

Si riprende il problema $(D-b)$

Assegnare l'operaio B alla macchina a; non assegnarlo

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

$$L = 56 + 2 = 58$$

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

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

$$L = 56$$

Si riprende il problema $B-a$

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

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

$$L = 56 + 1 = 57$$

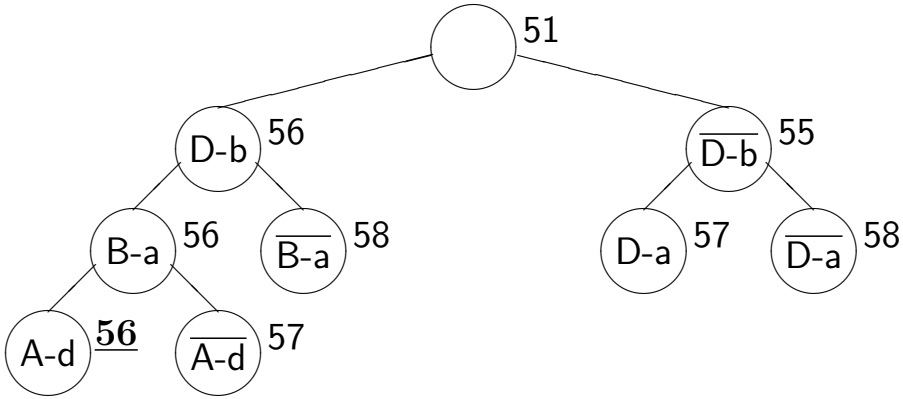
- $(D-b)(B-a)(A-d)$

	c
C	0

$$L = 56$$

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

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


51
D-b
56
$\overline{\text{D-b}}$
55
B-a
56
$\overline{\text{B-a}}$
58
D-a
57
$\overline{\text{D-a}}$
58
A-d
56
$\overline{\text{A-d}}$
57

6.3 Problema con variabili intere qualsiasi (PLI misto)

Si risolve con l'algoritmo del simplesso il rilassamento ottenuto eliminando il vincolo di integrità, determinando una prima limitazione
Se la variabile intera x_i assume il valore non intero x_i^* si separa il problema aggiungendo i vincoli $x_i \leq \lfloor x_i^* \rfloor$ e $x_i \geq \lceil x_i^* \rceil$
Si risolvono con l'algoritmo del simplesso o con l'algoritmo duale i sottoproblemi ottenuti, determinando le corrispondenti limitazioni, fino ad ottenere una soluzione intera per un sottoproblema; il sottoproblema è eliminato (condizione E3) e si possono eliminare anche tutti i sottoproblemi pendenti che hanno una limitazione peggiore (condizione E2)
Si prosegue finchè la lista dei problemi è vuota; la soluzione trovata è ottimale

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

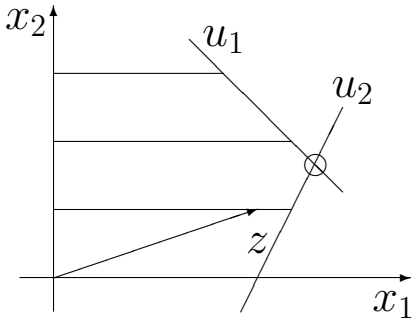
Esempio 6.1 (PLI misto) *Sia dato il seguente PLI misto:*

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

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

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

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



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

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

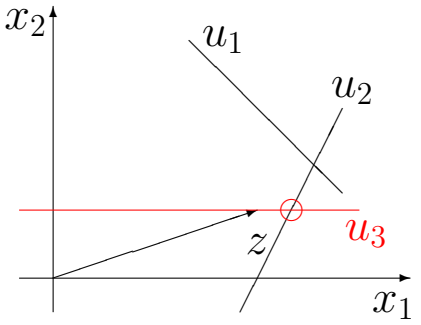
Dalla tabella finale si ricava $x_2 = \frac{1}{3}u_2 - \frac{2}{3}u_1 + \frac{5}{3}$*:*

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

Risolvendo $P1$ *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	$-\frac{1}{3}$	$\boxed{\frac{2}{3}}$	$-\frac{2}{3}$
z	$-\frac{2}{3}$	$-\frac{5}{3}$	$\frac{79}{6}$

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

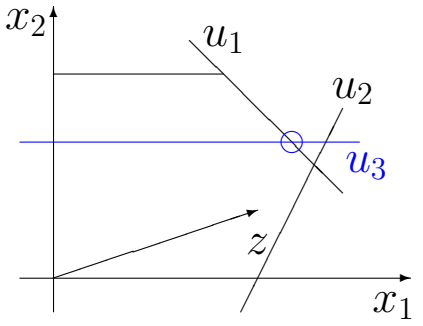


La soluzione $x^* = (\frac{7}{2}, 1)$*,* $z^* = \frac{23}{2}$ *è ammissibile*

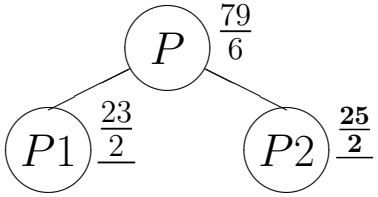
Risolvendo $P2$ si ottiene:

	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	$\boxed{\frac{1}{3}}$	$-\frac{2}{3}$	$-\frac{1}{3}$
z	$-\frac{2}{3}$	$-\frac{5}{3}$	$\frac{79}{6}$

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



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



$\diamondsuit$

6.4 Rilassamenti

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

6.4.1 Rilassamento continuo

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

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

6.4.2 Eliminazione di vincoli

Problema ben strutturato

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

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

6.4.3 Rilassamento lagrangiano

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

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

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

6.4.4 Rilassamento surrogato

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

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

6.5 Risoluzione per ispezione

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

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

Non esistono regole generali

Esempio 6.2 (Rilassamento lagrangiano di un problema dello zaino)

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

Applicando il rilassamento lagrangiano:

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

cioè:

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

che risolto per ispezione fornisce la soluzione ottimale:

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

♢

Esempio 6.3 (Rilassamento surrogato di un problema a variabili 0-1)

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

Applicando il rilassamento surrogato:

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

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

7 Complementi di programmazione lineare

7.1 Variazione dei coefficienti iniziali

Tabella iniziale:

	x^T	
u	A	b
z	c^T	0

Tabella finale:

	ξ^T	
ν	α	β
z	χ^T	χ_0

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

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

x' variabili in base nella tabella iniziale e successivamente uscite

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

Tabella finale partizionata

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

Tabella iniziale con la partizione indotta

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

Utilizzando la matrice A_{11} come cardine matriciale:

	u'^T	x''^T	
x'	$(A_{11})^{-1}$	$-(A_{11})^{-1}A_{12}$	$-(A_{11})^{-1}b_1$
u''	$A_{21}(A_{11})^{-1}$	$A_{22-}A_{21}(A_{11})^{-1}A_{12}$	$b_{2-}A_{21}(A_{11})^{-1}b_1$
z	$c_1^T(A_{11})^{-1}$	$c_2^T - c_1^T(A_{11})^{-1}A_{12}$	$-c_1^T(A_{11})^{-1}b_1$

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

7.1.1 Applicazioni

- **Stabilità della soluzione ottimale**
  Verificare se al variare di qualche coefficiente la soluzione data resta ottimale ed eventualmente come cambiano le coordinate della soluzione ottimale e il valore ottimale
- **Aggiunta di un vincolo**
  Un ulteriore vincolo, ad esempio nel metodo degli iperpiani secanti, può essere scritto in funzione delle variabili in base nella tabella iniziale:

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

  e in funzione delle variabili in base nella tabella finale:

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

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

7.2 Algoritmi greedy

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

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

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

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

7.2.1 Algoritmo greedy per il problema dello zaino

Si ordinano gli oggetti in ordine decrescente di rapporto valore/peso
Si prende un oggetto se può essere preso

L'algoritmo richiede $O(n)$ operazioni, dove n è il numero di oggetti
L'approssimazione è discreta se i pesi degli oggetti non sono molto differenti tra loro e sono piccoli rispetto al peso massimo trasportabile

Esempio 7.1 (Caso peggiore)

Oggetto	A	B
Valore	2	M
Peso	1	M

Peso massimo trasportabile $= M$

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

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

♢

Esempio 7.2 (Soluzione approssimata con l'algortimo greedy)

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

Peso massimo trasportabile $= 20$

Applicando l'algoritmo Branch and Bound:

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

per cui $z^* \leq 220$

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

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

Si può verificare che la soluzione trovata è esatta ♢

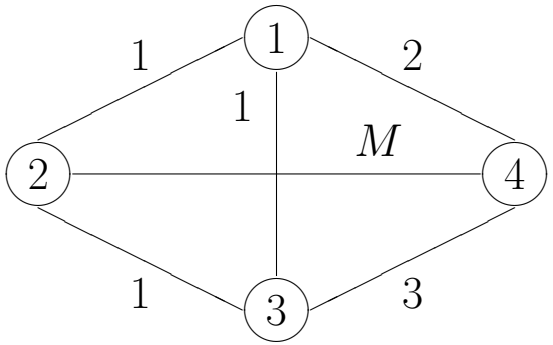
7.2.2 Algoritmo greedy per il problema del commesso viaggiatore

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

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

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

Esempio 7.3 (Caso peggiore)



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

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

♢

8 Scheduling

Ordinamento di una sequenza di lavori

Miglioramento della pianificazione della produzione o dell'utilizzo di una risorsa (macchina)

Formalizzazione

$N = \{1, ..., n\}$	lavori
$t_1, ..., t_n$	tempi di esecuzione
$\alpha_1, ..., \alpha_n$	costi per unità di tempo
σ_0	ordine iniziale dei lavori

8.1 Modello di base

- Una sola sequenza di lavori
- non è possibile iniziare, interrompere e riprendere un lavoro
- il costo dipende linearmente dal tempo (disponibilità immediata)

Dato un ordinamento il costo è:

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

dove $P(\sigma, i)$ sono i lavori che precedono i secondo σ

Esempio 8.1 (Scheduling) *L'officina di un'azienda deve riparare 4 macchinari:*

$macchinario$	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 è:

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

Riparando i macchinari per tempi di riparazione crescenti si ha l'ordine A - C - B - D e il costo:

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

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

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

♢

Ordinamento per indici di urgenza $u_i = \dfrac{\alpha_i}{t_i}, i \in N$ decrescenti (Smith, 1956)

Esempio 8.2 (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:*

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

Soluzione ottimale ♢

Dati due ordinamenti σ^1 e σ^2 che differiscono per due lavori consecutivi i e j (in σ^1 i precede j e in σ^2 i segue j):

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

σ^2 è più conveniente se $u_j > u_i$

8.2 Modelli evoluti

- *Sequenza*
  - Più macchine (identiche o differenti)
  - Più macchine per ogni lavoro con ordine prestabilito o meno
- *Lavori*
  - Eseguibili solo dopo un certo istante (*release time*)
  - Da completare inderogabilmente prima di un certo instante (*deadline*)
  - Penale sui ritardi (*due date*)
  - Precedenze assegnate
- *Costi*
  - Massimo ritardo sui lavori
  - Somma dei ritardi
  - Costo per anticipo

9 Teoria delle reti

9.1 Grafi

Intuitivamente un grafo è un insieme finito di punti (*nodi* o *vertici*) ed un insieme di frecce (*archi*) che uniscono coppie di punti
Il verso della freccia assegna un *orientamento* all'arco

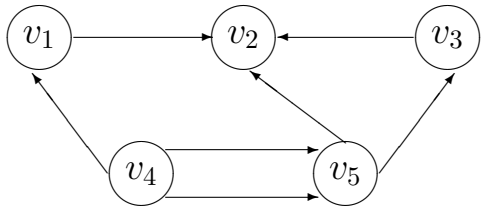
Se non si tiene conto dell'orientamento degli archi il grafo è detto *multigrafo* o *non orientato* e gli archi sono detti *spigoli*

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

Definizione 9.1 *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 9.1 (Elementi di un grafo)

$N = \{v_1, v_2, v_3, v_4, v_5\}$
$A = \{a_{12}, a_{32}, a_{41}, a_{45}, a_{45}, a_{52}, a_{53}\}$
$p = 2; n = 5; m = 7$



♢

Definizione 9.2

- *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*.*

9.2 Reti

Definizione 9.3 *Una rete è un grafo* $G(N, A)$ *nel quale ad ogni arco* $a_{ij} \in A$ *si associano tre valori interi* l_{ij}, u_{ij}, c_{ij} *detti rispettivamente* capacità minima, capacità massima *e* costo unitario *dell'arco e ad ogni nodo* v_i *si associa un valore intero* b_i*; se* $b_i > 0$ *il nodo* v_i *è detto* sorgente *e* b_i *è detta* disponibilità*, se* $b_i < 0$ *il nodo* v_i *è detto* pozzo *e* b_i *è detta* domanda*, se* $b_i = 0$ *il nodo* v_i *è detto* di attraversamento

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

9.2.1 Flusso su una rete

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

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

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

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

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

9.2.2 Problema del flusso di costo minimo

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

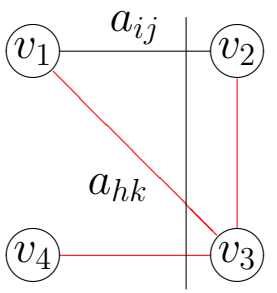
In questa forma possono essere rappresentati tutti i problemi di reti

9.3 Minimo Spanning Tree

Trovare uno spanning tree di costo minimo
Se gli archi hanno tutti lo stesso costo la soluzione è un qualsiasi spanning tree

Teorema 9.1 *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

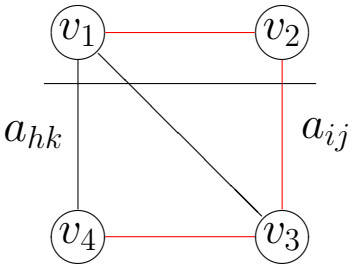
♣

Teorema 9.2 *Dato un grafo $G(N, A)$ sia C un ciclo del grafo G e sia $a_{ij} = (v_i, v_j)$ un arco di peso massimo appartenente al ciclo C, allora esiste un albero di peso minimo non contenente a_{ij}*

Dimostrazione

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

Assurdo



♣

Algoritmo di Prim (1957)

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

Algoritmo

a) $A' = \varnothing; N' = \{v_1\}$;

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 9.1

Algoritmo di Kruskal (1956)

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

Algoritmo

a) $A' = \varnothing$;

b) determinare l'arco $a_{ij} = (v_i, v_j)$ di peso minimo tale che $a_{ij} \in A/A'$ e non forma cicli;

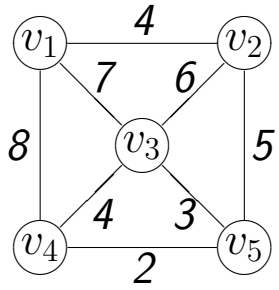
c) $A' = A' \cup \{a_{ij}\}$;

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 9.2

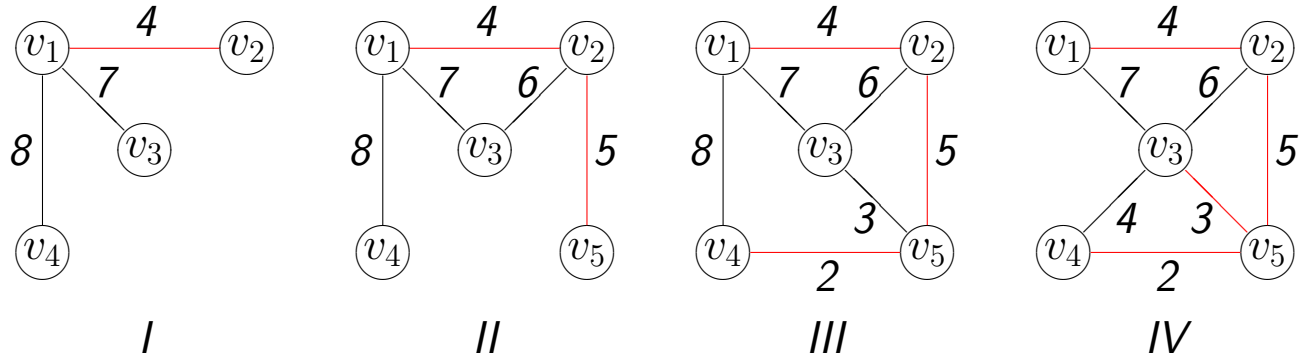
- Esiste una variante dell'algoritmo di Kruskal che elimina l'arco di peso massimo che non fa perdere la connessione

Esempio 9.2 (Minimo Spanning Tree)



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

Algoritmo di Prim



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

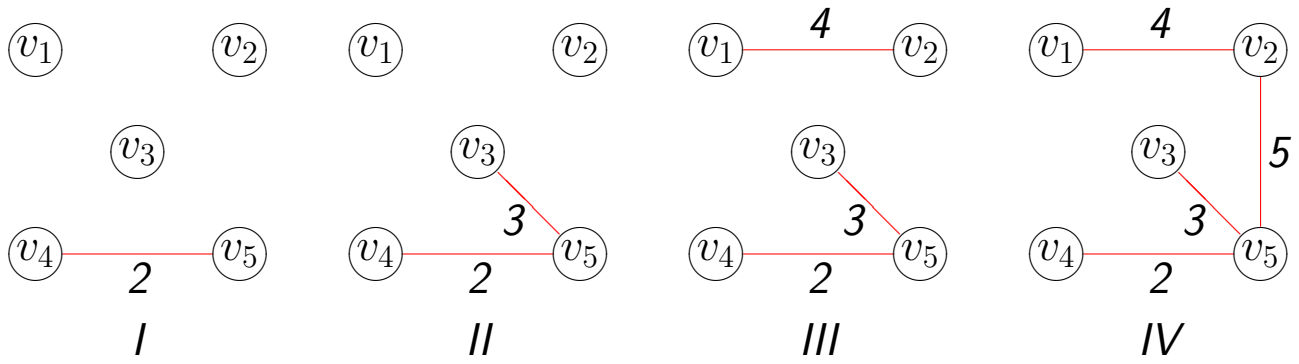
I $A' = \{a_{12}\}; N' = \{v_1, v_2\}$

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 $A' = \varnothing$

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* $(card(A') = card(N) - 1)$ ♢

9.4 Cammino minimo

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

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

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

9.4.1 SPP da un nodo v_s a tutti gli altri nodi

Ad ogni nodo v_i si assegnano un valore o etichetta $d(i)$ che indica il costo del cammino corrente da v_s a v_i e un puntatore $pred(i)$ che indica il predecessore di v_i nel cammino corrente
Si parte da un valore temporaneo per $d(i)$ che può essere modificato ad ogni iterazione fino ad ottenere il valore esatto

Esistono due classi di algoritmi:

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

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

Algoritmo di Dijkstra (1959)

Algoritmo

a) $d(s) = 0$;

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

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

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

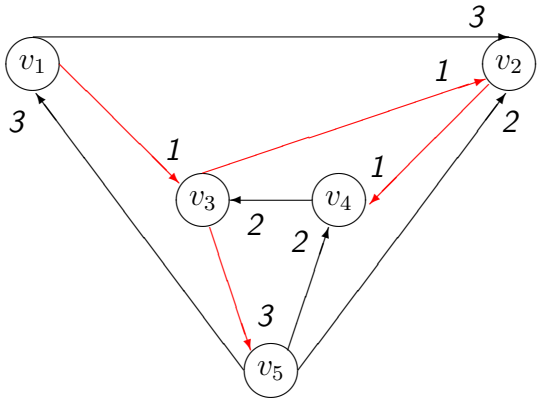
Variante di Bellman (1958)

Si esaminano gli archi secondo un ordinamento prefissato

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

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

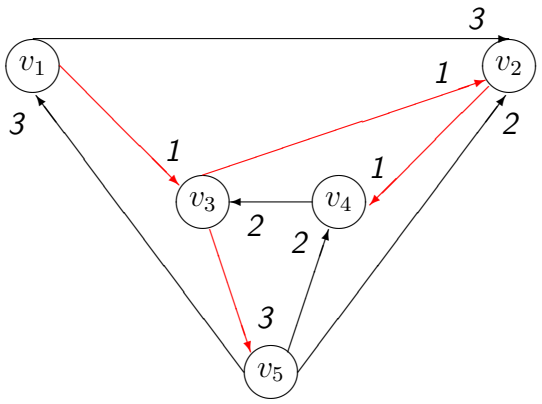
Esempio 9.3 (Grafo con archi di peso non negativo)



Algoritmo di Dijkstra

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

STOP



Algoritmo di Bellmann

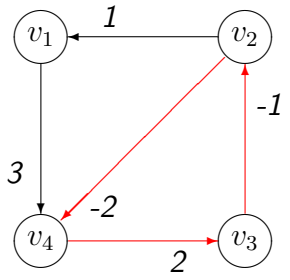
Ordinamento per indici crescenti

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

STOP

$\diamondsuit$

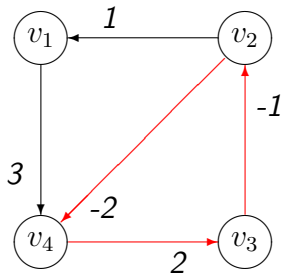
Esempio 9.4 (Grafo con ciclo negativo)



Algoritmo di Dijkstra

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

STOP (non identificato ciclo negativo)



Algoritmo di Bellmann

Ordinamento per indici crescenti

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

STOP (identificato ciclo negativo) ♢

9.4.2 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)

9.5 Flusso massimo

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

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

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

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

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

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

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

Problema di programmazione lineare a numeri interi:

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

Algoritmi di programmazione lineare o algoritmi specifici

Algoritmi di cammino aumentante

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

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

Algoritmi di preflusso

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

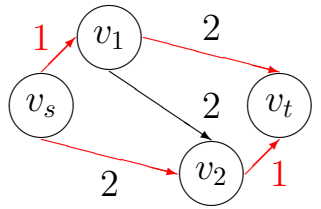
- Algoritmo di preflusso (Karzanov, 1974 - Goldberg e Tarjan, 1986)
- Variante (Goldberg e Tarjan, 1986)
- Algoritmo di scaling dell'eccesso (Ahuja e Orlin, 1991)

Algoritmo del contrassegno

a) inizializzare x con un flusso ammissibile (se le capacità minime sono tutte nulle il flusso nullo è ammissibile);

b) $V = \{v_s\}$ (V è l'insieme dei nodi contrassegnati);

c) se esistono due nodi adiacenti $v_i \in V$ e $v_j \notin V$ contrassegnare v_j
con $+i$ se $j \in \omega^+(i)$ e $x_{ij} < u_{ij}$ oppure
con $-i$ se $j \in \omega^-(i)$ e $x_{ji} > l_{ji}$ e andare a d);
altrimenti STOP (flusso massimo);



d) se $v_t \in V$ esiste un cammino non orientato C da v_s a v_t sul quale si può variare il flusso di Δ andare a e);
altrimenti tornare a c);

e) costruire un nuovo flusso x ponendo:

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

tornare a b);

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

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

Eliminando i termini di segno opposto si ha:

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

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

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

Definizione 9.4 *Data una rete di trasporto $G(N, A)$ e il taglio generato da un insieme $U \subset N$, con $v_s \in U$ e $v_t \notin U$ si chiama capacità del taglio la quantità* $\sum_{v_i \in U, v_j \notin U} u_{ij} - \sum_{v_i \in U, v_j \notin U} l_{ji}$

Teorema 9.3 (Teorema di Ford e Fulkerson (max flow - min cut, 1956)) *Data una rete di trasporto $G(N, A)$ si ha:*

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

Problema del flusso compatibile

Definizione 9.5 *Si definisce distanza di un flusso x dalle capacità l e u la quantità:*

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

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

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

Algoritmo di Herz (1967)

a) inizializzare x in modo che siano soddisfatti i vincoli di conservazione del flusso ma non (necessariamente) quelli di capacità (flusso nullo);

b) se esiste $x_{hk} < l_{hk}$ contrassegnare v_k con $+h$ e andare a c);

b') se esiste $x_{kh} > u_{kh}$ contrassegnare v_k con $-h$;

c) se si contrassegna v_t si contrassegna anche v_s con $+t$;

c') se si contrassegna v_s si contrassegna anche v_t con $-s$;

d) se esistono due nodi adiacenti v_i contrassegnato e v_j non contrassegnato, si contrassegna v_j con $+i$ se $j \in \omega^+(i)$ e $x_{ij} < u_{ij}$ oppure con $-i$ se $j \in \omega - (i)$ e $x_{ji} > l_{ji}$ e andare a e);
altrimenti STOP (non esiste un flusso ammissibile);

e) se si contrassegna v_h esiste un ciclo non orientato C sul quale si può variare il flusso di Δ e andare a f);
altrimenti tornare a c);

f) costruire un nuovo flusso x ponendo:

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

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

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

Teorema 9.4 (Teorema di Hoffman, 1960)

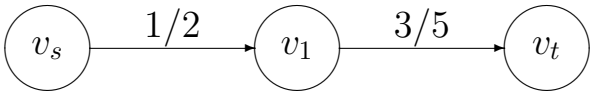
Data una rete di trasporto $G(N, A)$ esiste un flusso ammissibile x se e solo se preso comunque $U \subset N$,con $v_s, v_t \in U$ oppure $v_s, v_t \notin U$ si ha:

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

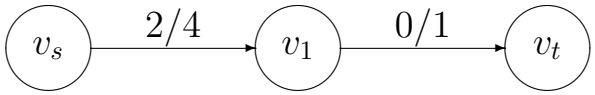
L'insieme dei nodi non contrassegnati al termine dell'algoritmo quando non esiste il flusso ammissibile non verifica la condizione del teorema

Esempio 9.5 *Si considerino le seguenti due situazioni di flusso impossibile:*

a)



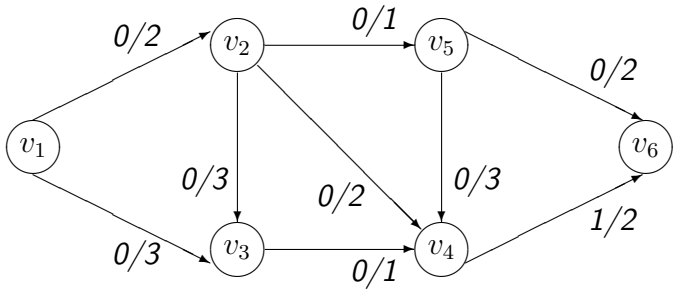
b)



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

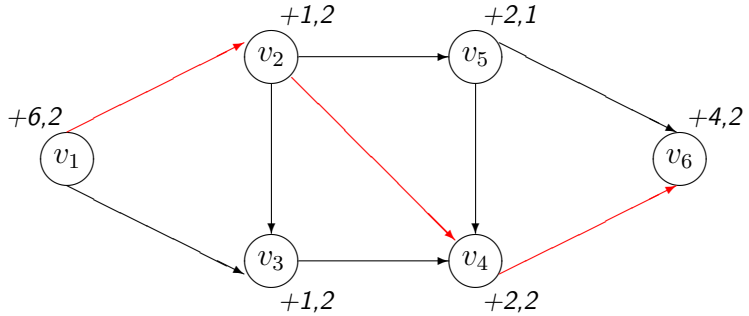
Nel caso b) dopo aver determinato il flusso 1 su entrambi gli archi, si contrassegna v_1 e l'insieme $\{v_s, v_t\}$ non verifica la condizione del teorema ♢

Esempio 9.6 (Flusso massimo)



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

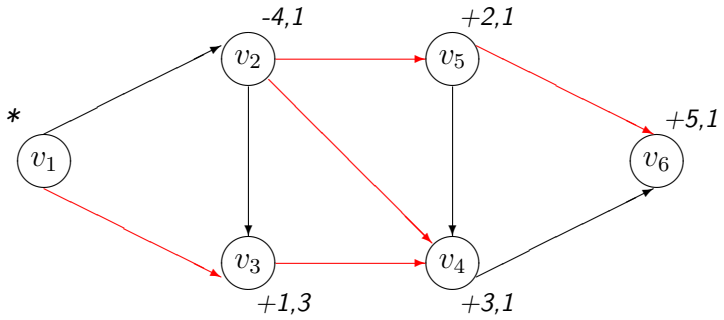
Algoritmo di Herz dall'arco a_{46}



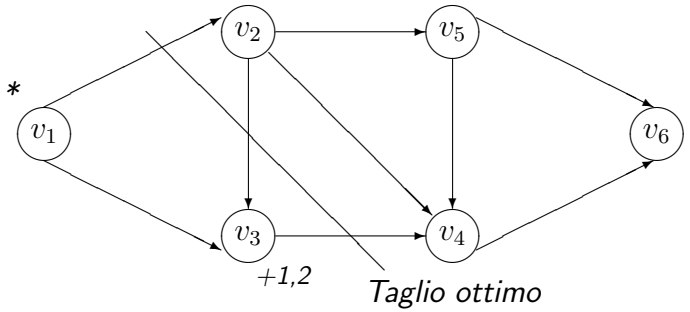
Ciclo: $v_4 - v_6 - v_1 - v_2 - v_4$; $\Delta = 2$

Flusso ammissibile

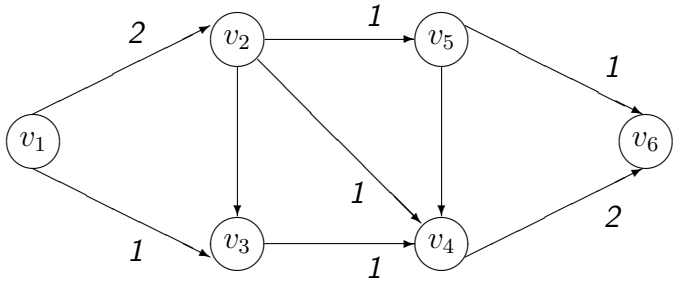
Algoritmo di Ford e Fulkerson



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



Flusso massimo



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

Relazione di dualità tra flusso massimo e taglio minimo

Dato il problema di flusso massimo:

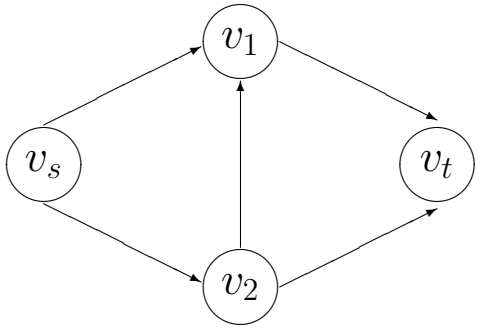
$$
\begin{aligned}
max \;\; & z = F \\
s.t. \;\; & \sum_{j \in \omega^+(s)} x_{sj} - F = 0 \\
& \sum_{j \in \omega^+(i)} x_{ij} - \sum_{j \in \omega^-(i)} x_{ji} = 0 && v_i \in N \setminus \{v_s, v_t\} \\
& - \sum_{j \in \omega^-(t)} x_{jt} + F = 0 \\
& l_{ij} \leq x_{ij} \leq u_{ij}; x_{ij} \; intero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

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

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

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

Esempio 9.7 (Max flow - min cut)



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

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

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

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

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

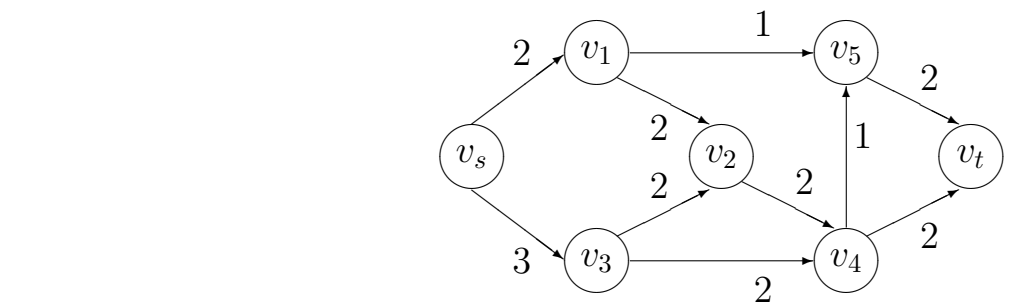
9.6 Algoritmo del minimo cammino aumentante (Edmonds e Karp, 1972)

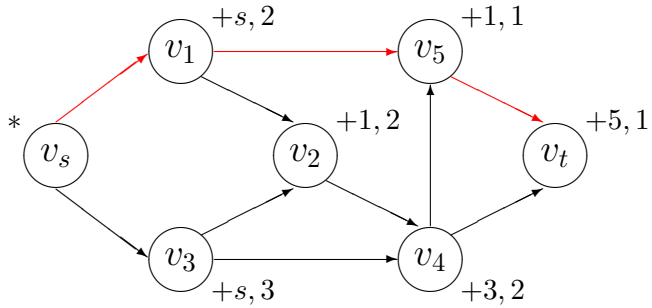
Si analizzano i nodi etichettati in ampiezza (ordine di etichettatura)

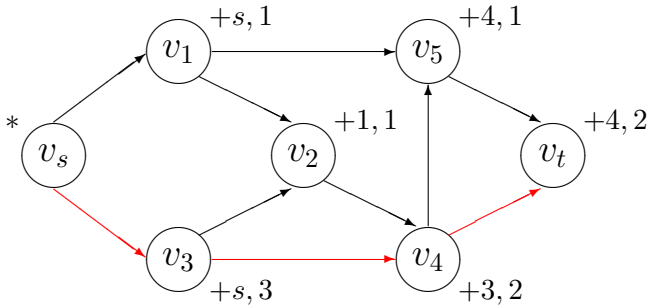
E' sufficiente modificare il passo c) nel modo seguente:

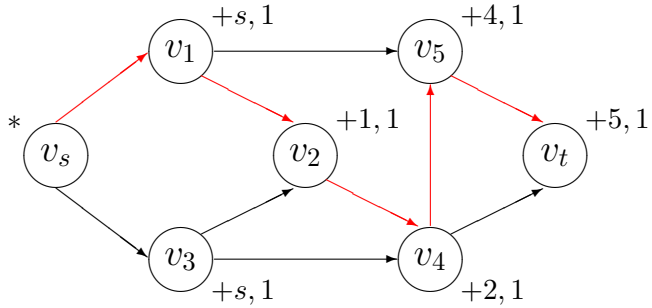
c') se $V \neq \varnothing$ detto v_i il primo nodo di V secondo l'ordine di etichettatura, contrassegnare tutti i nodi adiacenti $v_j \notin V$ con "+i" se $j \in w^+(i)$ e $x_{ij} < u_{ij}$ oppure con "-i" se $j \in w^-(i)$ e $x_{ij} > l_{ij}$; eliminare v_i da V e andare a d);

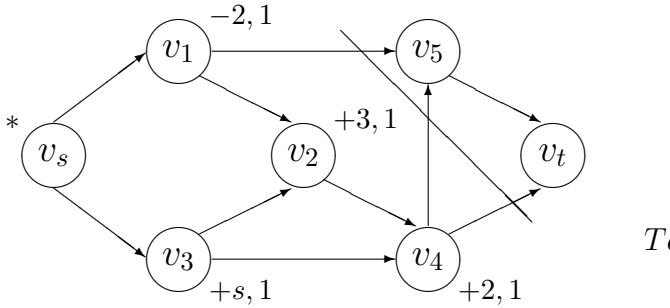
Esempio 9.8 (Algoritmo di Edmonds e Karp)



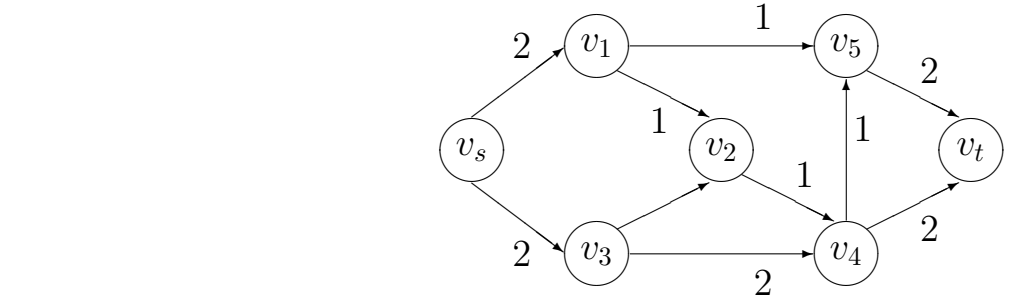






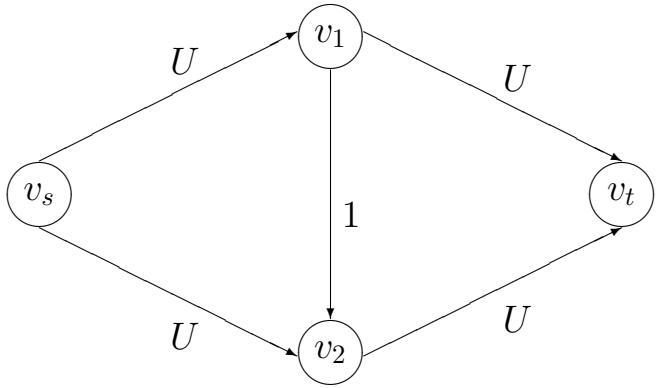


Taglio ottimo



$\diamondsuit$

Esempio 9.9 (Confronto di complessità)



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

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

9.7 Complessità computazionale degli algoritmi di flusso massimo

Si consideri una rete con n nodi e m archi:

Algoritmo di Ford e Fulkerson

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

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

Algoritmo di Edmonds e Karp

- Analisi degli archi per ogni aggiornamento $= \mathsf{O}(m)$
- Aggiornamento del flusso $= \mathsf{O}(n)$ - Ogni cammino contiene al più $n-1$ archi
- Numero dei cammini minimi $= \mathsf{O}(nm)$ - Per ogni lunghezza ci sono al più $\mathsf{O}(m)$ 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 $\mathsf{O}((n+m)nm)$ e quindi l'algoritmo ha complessità $\mathsf{O}(nm^2)$

Reti sparse ($m = \mathsf{O}(n)$) e dense ($m = \mathsf{O}(n^2)$):

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

- Se $U = \mathsf{O}(n)$ 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

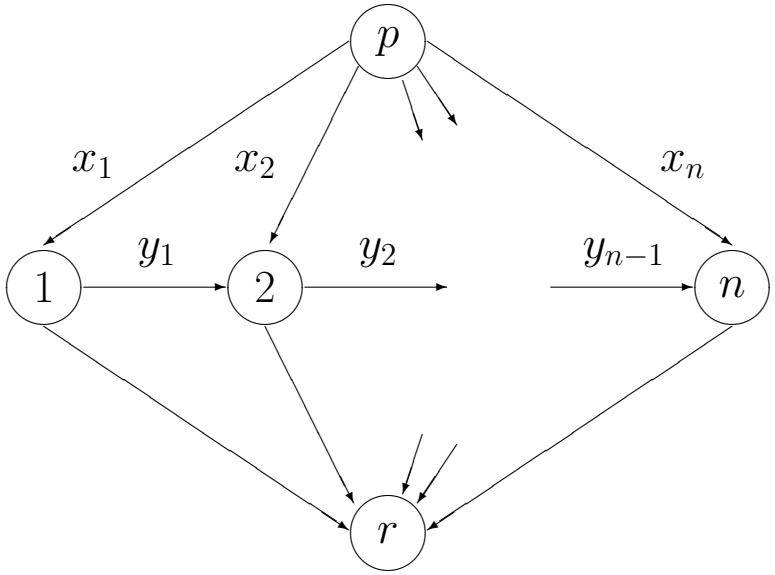
9.8 Problema di produzione e magazzino

Problema di produzione e magazzino

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

- d_i domanda
- c_i costo di produzione unitario
- h_i costo di immagazzinamento unitario
- C_i capacità produttiva massima
- H_i capacità del magazzino

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



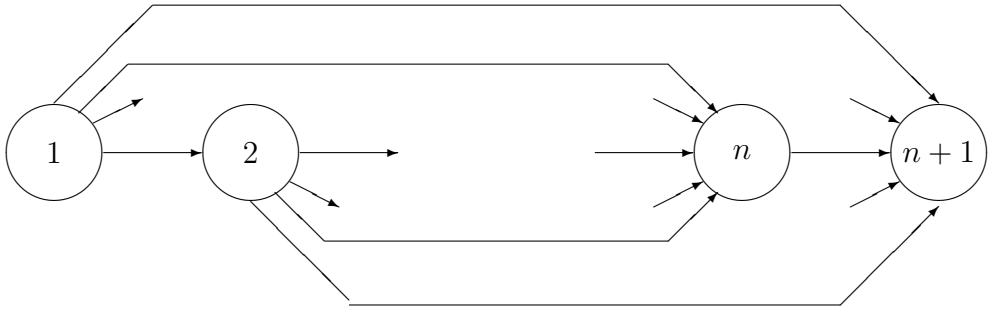
I vincoli di bilanciamento nel nodo i sono i vincoli dinamici del problema del magazzino:

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

Problema di flusso di costo minimo da p a r

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

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



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

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

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

9.9 Problema di routing e scheduling

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

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

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

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

La complessità del problema può diventare molto elevata

9.10 Problema di progettazione

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

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

Esempio 9.10 (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					

♢

Modelli

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

9.10.1 CPM (Roy, 1960)

- **Dati**
  - all'attività i si associa il nodo v_i del grafo
  - l'arco (v_i, v_j) indica che l'attività i precede l'attività j $(i \prec j)$
  - all'arco (v_i, v_j) si associa il costo d_i corrispondente alla durata dell'attività i
- **Ipotesi**
  - si aggiungono un'attività iniziale α e un'attività finale ω, entrambe di durata nulla; il nodo v_α viene collegato tramite archi uscenti ai nodi corrispondenti alle attività che non devono essere precedute da altre e il nodo v_ω viene collegato tramite archi entranti ai nodi corrispondenti alle attività che non sono seguite da altre
  - il grafo è aciclico per costruzione (un'attività non può iniziare dopo il suo termine)

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

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

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

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

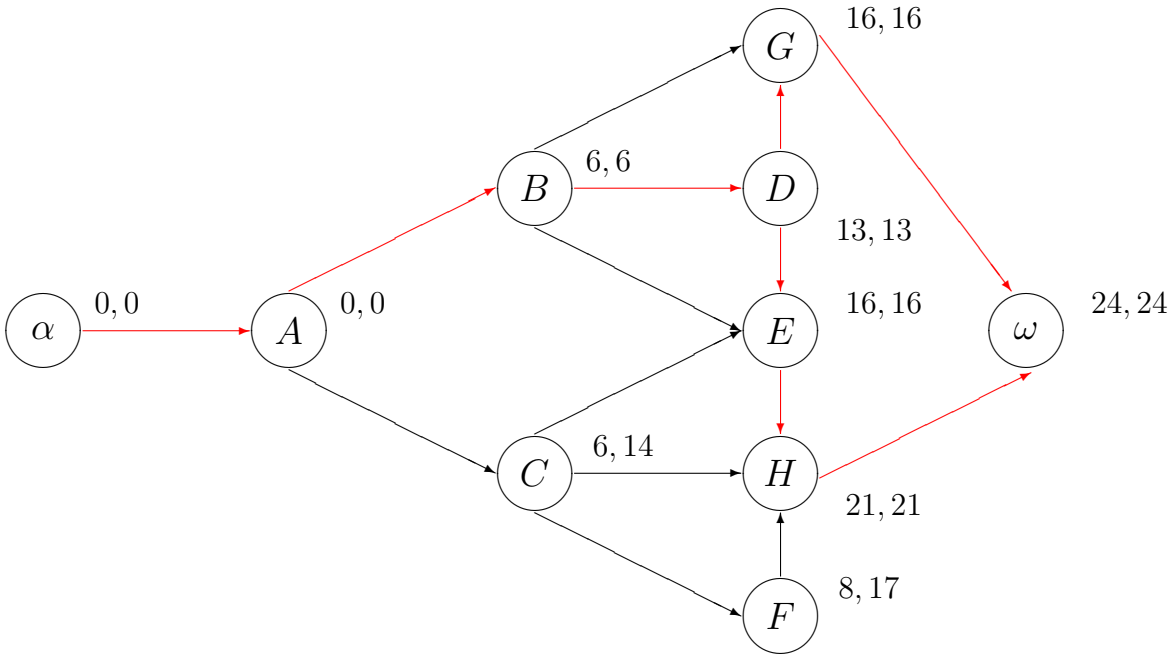
- **Cammino critico**
  - si determinano i ritardi massimi $m_i = T_i - t_i$, si identificano le attività critiche e i cammini critici

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

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

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

Esempio 9.11 (Cammino critico) *Riferendosi all'Esempio 9.10 si ottengono i seguenti risultati:*



Tempo minimo: 24

Attività critiche: A, B, D, E, G, H

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

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

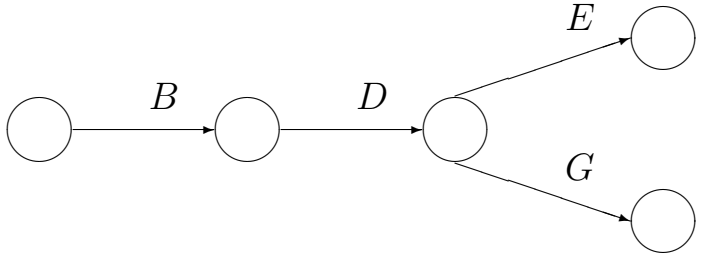
9.10.2 PERT (Malcom, Roseboom, Clark, Fazar, 1959)

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

Esempio 9.12 (PERT) *Riferendosi all'Esempio 9.10 si può osservare che:*

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

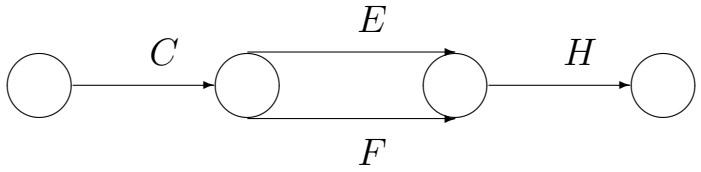
cioè:



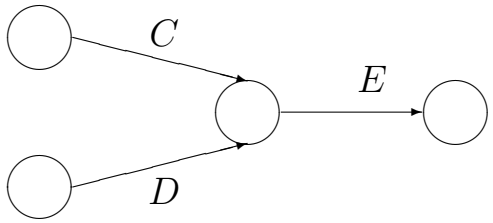
Inoltre si ha:

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

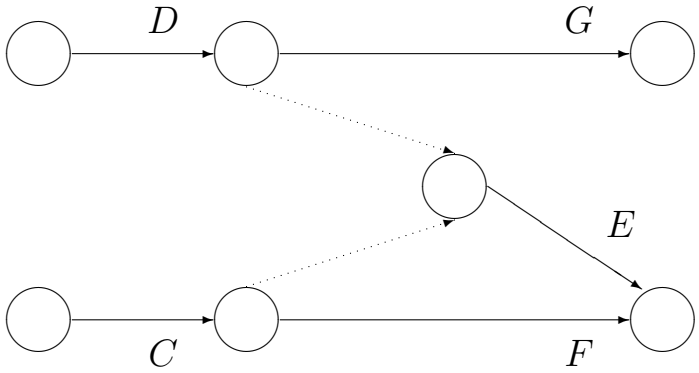
cioè:



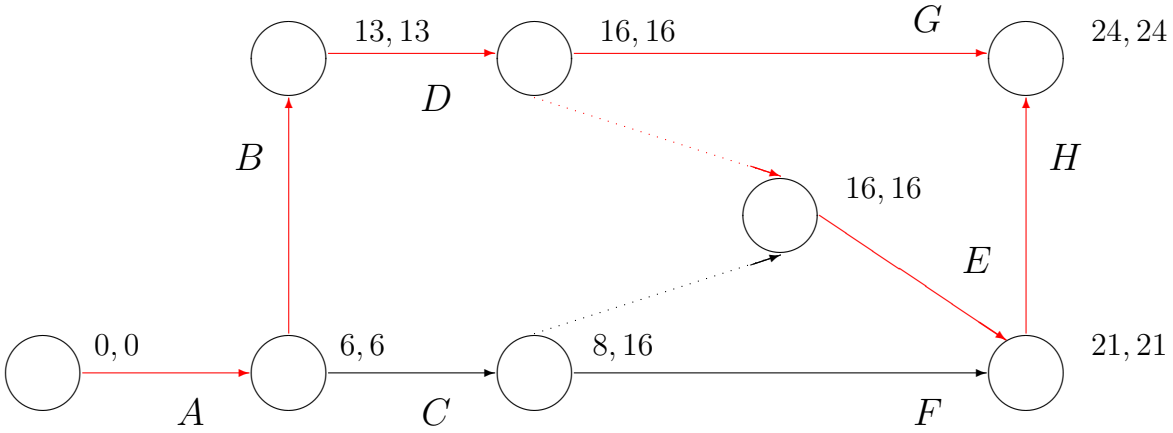
Le relazioni $C \prec E$ e $D \prec E$ non possono essere unificate come:



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



A questo punto si può ottenere il grafo finale:

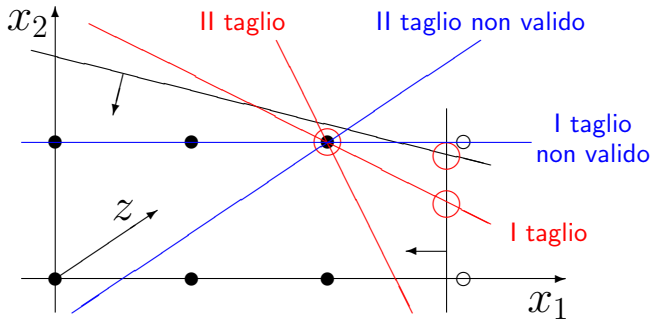


Il tempo minimo, le attività critiche e i cammini critici sono gli stessi ♢

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

10 Metodi cutting plane

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



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

I metodi si differenziano per il modo in cui si generano gli iperpiani secanti
È importante che il criterio di generazione degli iperpiani secanti permetta di trovare la soluzione ottimale in un numero finito di passi

Notazioni

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

10.0.1 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
Questo metodo è efficiente ma non garantisce la convergenza

10.0.2 Algoritmo di Gomory (1958)

La tabella iniziale deve essere a coefficienti interi
Se nella tabella ottimale del problema rilassato un termine β_k non è intero, si considera il vincolo:

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

e si genera il vincolo:

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

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 10.1 *Il vincolo* $\tilde{y}_{n+k} = f_{k1}y_1 + \ldots + f_{ki}y_i + \ldots + f_{kn}y_n - f_k \geq 0$ *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:

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

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:

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

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

$$\boxed{-\alpha_{ki} = q_{ki} + f_{ki} \Rightarrow -q_{ki} = \alpha_{ki} + f_{ki}}$$
$$\boxed{\beta_k = q_k + f_k \Rightarrow q_k = \beta_k - f_k}$$

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

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

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

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

♣

- 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

Esempio 10.1 (Algoritmo di Gomory) *Sia dato il seguente PLI:*

$$
\begin{aligned}
P: \quad max \quad & x_1 + 2x_2 \\
s.t. \quad & 2x_1 - x_2 \leq 3 \\
& x_1 + 3x_2 \leq 4 \\
& x_1, x_2 \geq 0 \text{ e interi}
\end{aligned}
$$

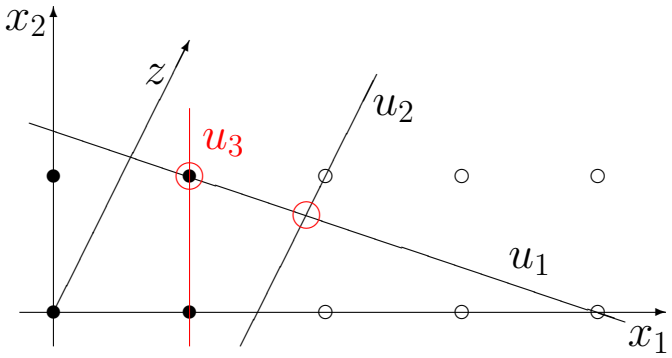
	x_1	x_2	
u_1	$\boxed{-2}$	1	3
u_2	-1	-3	4
z	1	2	0

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

	u_1	u_2	
x_1	$-\frac{3}{7}$	$-\frac{1}{7}$	$\frac{13}{7}$
x_2	$\frac{1}{7}$	$-\frac{2}{7}$	$\frac{5}{7}$
z	$-\frac{1}{7}$	$-\frac{5}{7}$	$\frac{23}{7}$

La tabella è ottimale, ma la soluzione non è intera

Dalla prima riga si ottiene $u_3 = \frac{3}{7}u_1 + \frac{3}{7}u_2 - \frac{6}{7} = -x_1 + 1$



	u_1	u_2	
x_1	$-\frac{3}{7}$	$-\frac{1}{7}$	$\frac{13}{7}$
x_2	$\frac{1}{7}$	$-\frac{2}{7}$	$\frac{5}{7}$
u_3	$\boxed{\frac{3}{7}}$	$\frac{1}{7}$	$-\frac{6}{7}$
z	$-\frac{1}{7}$	$-\frac{5}{7}$	$\frac{23}{7}$

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

La tabella è ottimale e la soluzione $x^* = (1, 1), z^* = 3$ *è intera*

$\diamondsuit$