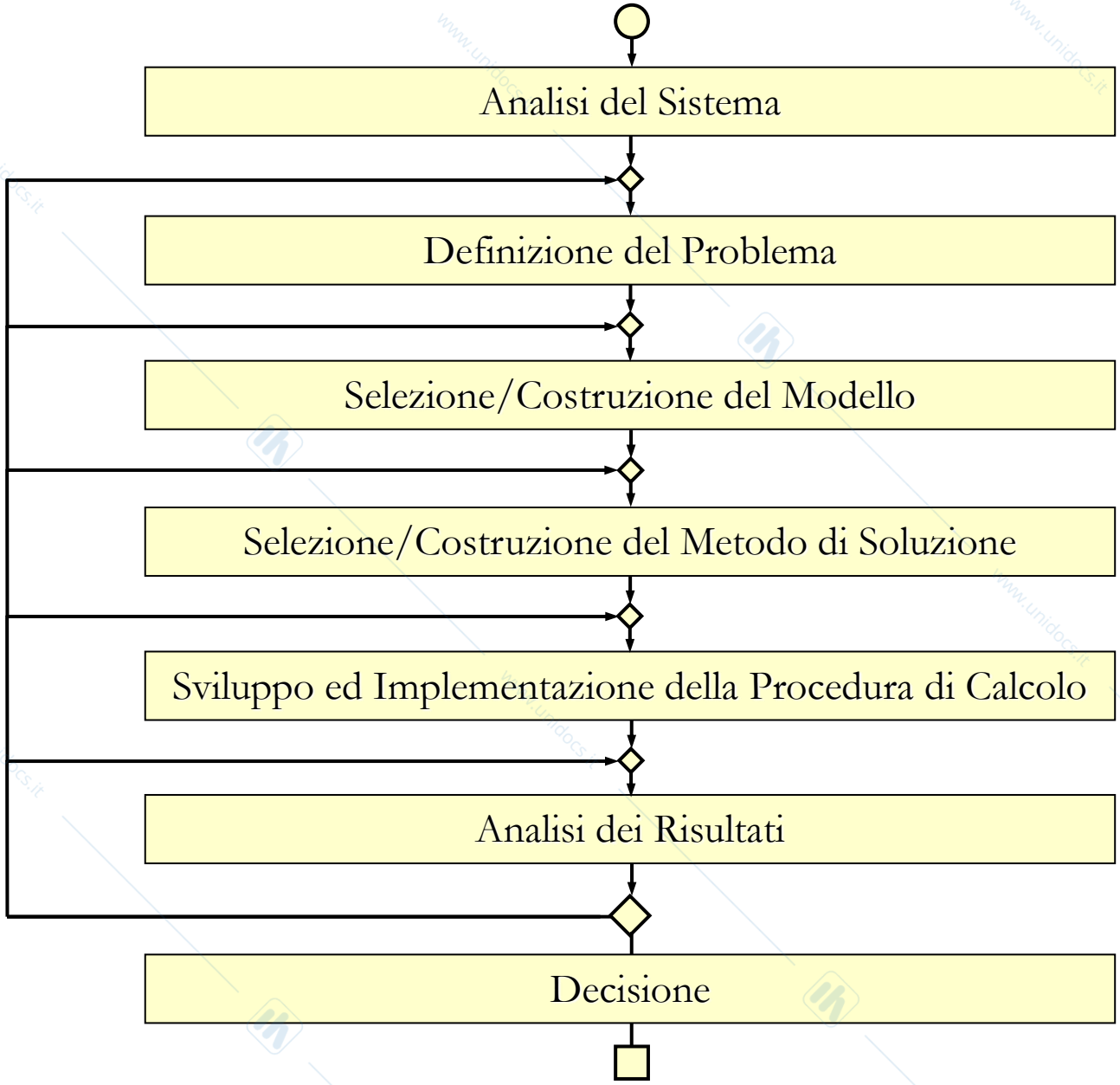
Problema decisionale

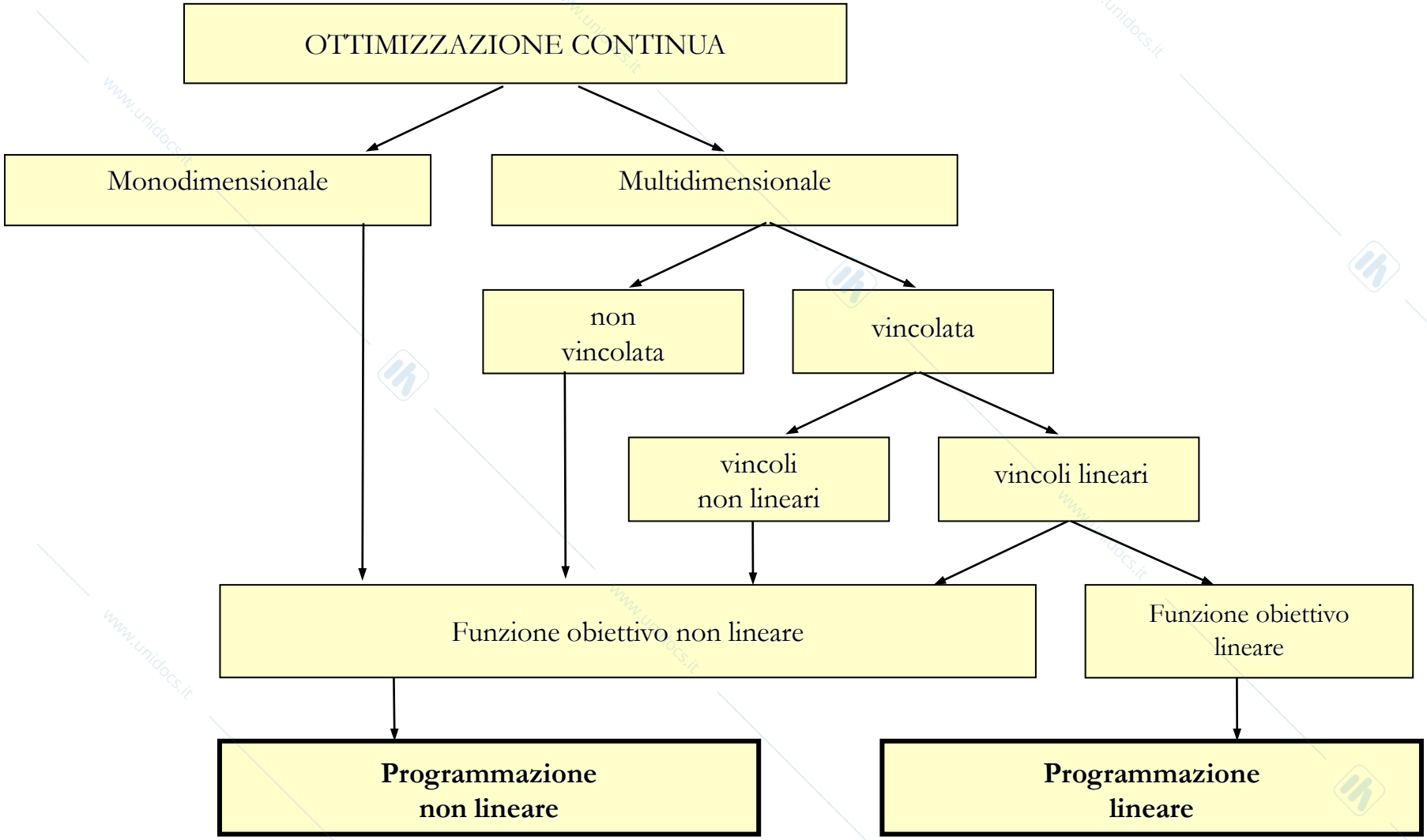
Un problema decisionale è un problema in cui bisogna effettuare una scelta fra diverse alternative, con determinati obiettivi, rispettando i vincoli del problema e la limitatezza delle risorse

La *Ricerca Operativa* consente di applicare metodi analitici per la soluzione di *problemi di decisione complessi* che si presentano in molteplici settori della vita socio-economica.

Schema di processo decisionale



Schema dei problemi di ottimizzazione continua



Ottimizzazione Monodimensionale

Modello matematico del problema monodimensionale

Punto di minimo di una funzione monodimensionale

$$x^* : f(x^*) \leq f(x)$$

per ogni x appartenente all'intervallo $[a, b]$

$$\text{Min } z = f(x) \quad \text{Funzione obiettivo}$$

$$\text{Vincoli} \quad x \geq a$$

$$x \leq b$$

Punto di ottimo di una funzione monodimensionale

e condizioni di ottimalità

Se la funzione $f(x)$ **è differenziabile** la condizione necessaria affinché un punto x^* sia un punto di minimo o massimo locale non vincolato è:

$$\left.\frac{df}{dx}\right|_{x=x^*} = 0$$

La condizione sufficiente affinché un punto x^* sia un punto di minimo (massimo) locale non vincolato è:

$$\left.\frac{d''f}{dx^2}\right|_{x=x^*} > 0 \quad (< 0)$$

Alcune considerazioni

- I dominio di ammissibilità potrebbe escludere il punto di ottimo assoluto
- La derivata potrebbe essere difficile da calcolare
- Potrebbe essere difficile trovare la soluzione x* dall'espressione della derivata nulla (pensiamo a una funzione complessa di ordine elevato)

Alcune considerazioni

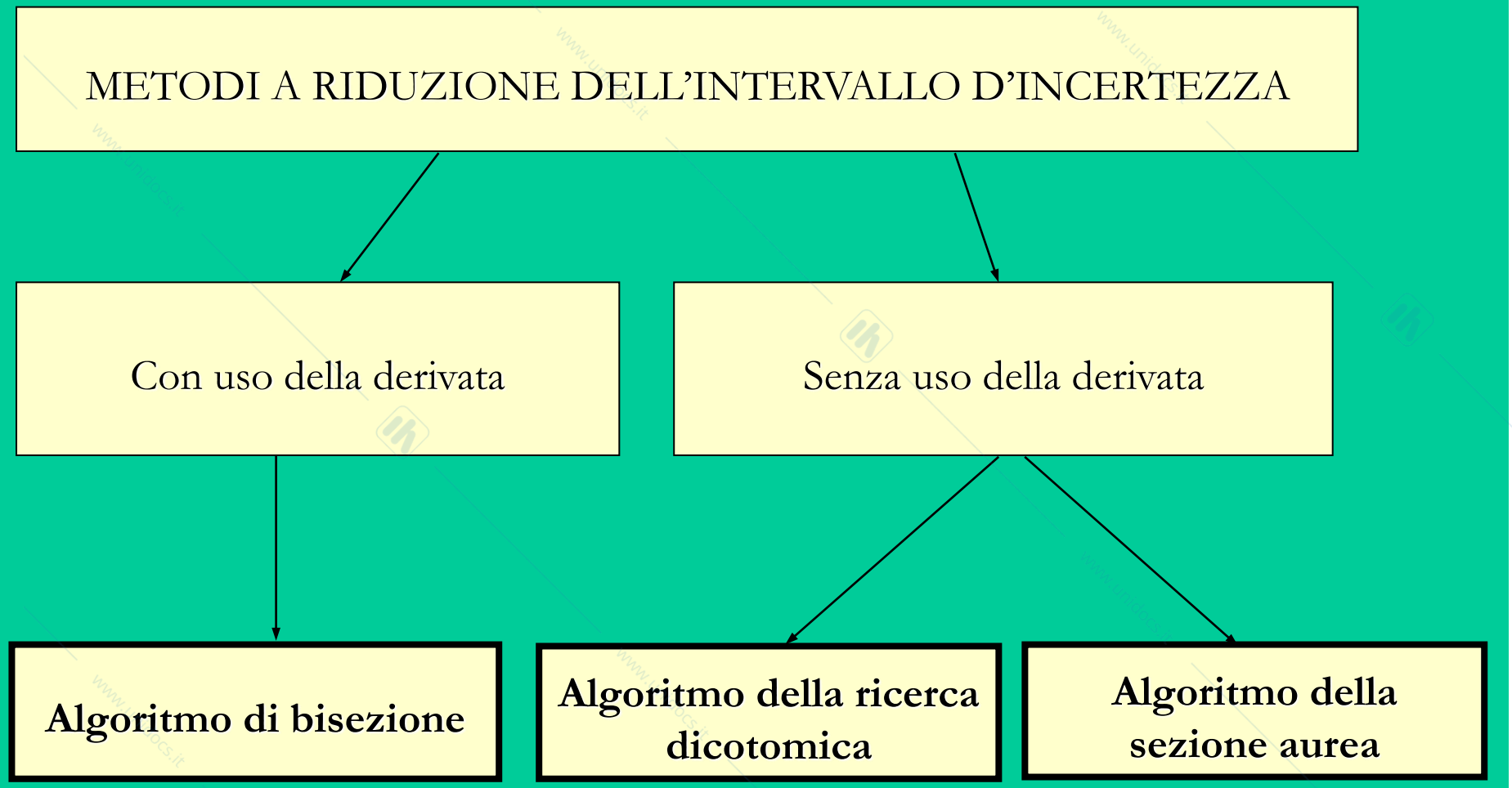
→ E' difficile determinare il punto di ottimo utilizzando solo le condizioni di esistenza del punto stesso.

→ Abbiamo bisogno di metodi di ottimizzazione monodimensionale

Metodi di ottimizzazione monodimensionale

- Metodi di riduzione dell'intervallo di incertezza
- Metodi di generazione di punti

- Metodi con uso della derivata
- Metodi senza uso della derivata



Riduzione dell'intervallo di incertezza con uso della derivata

Sia $f(x)$ una funzione pseudoconvessa sull'intervallo $[a,b]$ e c un punto interno all'intervallo $[a,b]$. Si può dimostrare che:

- se risulta $f'(c) > 0$ si ha $f(x) \geq f(c)$ $\forall x \in [c, b]$;
- se risulta $f'(c) < 0$ si ha $f(x) \geq f(c)$ $\forall x \in [a, c]$:
- se risulta $f'(c) = 0$ c è un punto di minimo globale.

Sia $[a_k,b_k]$ l'intervallo di incertezza sul punto di minimo all'iterazione k e sia c_k un punto interno, nel quale viene effettuato il calcolo della derivata della funzione obiettivo:

- se $f'(c_k) > 0$ $x^* \in [a_k,c_k]$. Si pone $a_{k+1} = a_k$, $b_{k+1} = c_k$;
- se $f'(c_k) < 0$ $x^* \in [c_k,b_k]$. Si pone $a_{k+1} = c_k$, $b_{k+1} = b_k$;
- se $f'(c_k) = 0$ $x^* = c_k$.

Riduzione dell'intervallo di incertezza senza uso della derivata

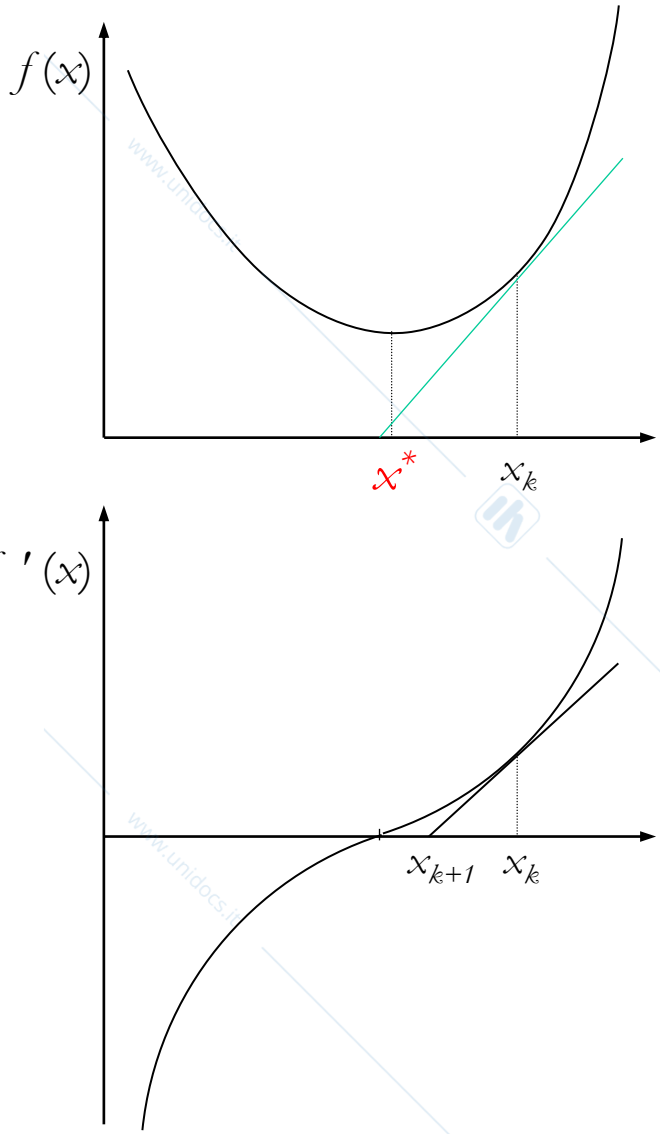
Sia $f(x)$ una funzione pseudoconvessa sull'intervallo $[a,b]$ e siano c, d due punti interni all'intervallo $[a,b]$ tali che $c < d$. Si può dimostrare che:

- se risulta $f(c) > f(d)$ si ha $f(x) > f(d)$ $\quad \forall\, x \in]\, a, c\, [$;
- se risulta $f(c) < f(d)$ si ha $f(x) > f(c)$ $\quad \forall\, x \in]\, d, b\, [$;
- se risulta $f(c) = f(d)$ si ha $f(x) \geq . f(c) = f(d)$ $\quad \forall\, x \in]\, a, c\, [\cup]\, d, b\, [$

Sia $[a_k,b_k]$ l'intervallo di incertezza sul punto di minimo all'iterazione k e siano c_k e d_k due punti interni, con $c_k < d_k$, nei quali viene effettuato il calcolo della funzione obiettivo:

- se $f(c_k) < f(d_k)$ $\quad x^* \in [a_k,d_k]$. $\quad$ Si pone $a_{k+1} = a_k$, $b_{k+1} = d_k$;
- se $f(c_k) > f(d_k)$ $\quad x^* \in [c_k,b_k]$. $\quad$ Si pone $a_{k+1} = c_k$, $b_{k+1} = b_k$.

Algoritmo della tangente (Newton)



- Si parte da un punto iniziale noto
- Si determina una successione di punti approssimando ad ogni iterazione la derivata seconda con la sua tangente.

Sia x_k il punto ottenuto all'iterazione k.

La tangente geometrica alla funzione $f'(x)$ incontra l'asse x in un punto x_{k+1}, con un angolo α.

La tangente trigonometrica di α, $f'(x_k)/(x_k - x_{k+1})$ può essere assunta come stima della derivata $f''(x_k)$.

Si può pertanto scrivere:

$f'(x_k)/(x_k - x_{k+1}) \cong f''(x_k)$, da cui

$f''(x_k)\cdot(x_k - x_{k+1}) - f'(x_k) \cong 0$

e quindi: $x_{k+1} = x_k - f'(x_k)/f''(x_k)$

La relazione viene applicata iterativamente.

Si parte da un punto noto x_0, finché risulta

$|x_k - x_{k+1}| < \varepsilon$, se ε è la precisione richiesta.

Algoritmo della tangente (Newton)

$$\frac{f'(x_k)}{(x_k - x_{k+1})} = f''(x_k)$$

$$f''(x_k)\cdot(x_k - x_{k+1}) - f'(x_k) = 0$$

$$\boldsymbol{x_{k+1} = x_k - f'(x_k) / f''(x_k)}$$

Convergenza su

$$| x_k - x_{k+1} | < \varepsilon$$

Condizioni di ottimalità di una funzione multidimensionale

Condizioni di ottimalità di funzioni differenziabili

Condizione necessaria affinché il punto $\boldsymbol{x}^*$

sia un punto di minimo (o di massimo), (locale o globale)

della funzione differenziabile $f(\boldsymbol{x})$ è che

$\boldsymbol{x}^*$ sia un punto di stazionarietà della funzione obiettivo,

cioè che sia

$$\nabla f(\boldsymbol{x}^*) = \mathbf{0}$$

Il gradiente di una funzione f(x)

Il gradiente di una funzione f(**x**) è un vettore le cui componenti sono le derivate parziali della funzione stessa e si indica con ∇f (**x**):

$$\nabla f\ (\boldsymbol{x}) = \partial f/\partial x_1\ \boldsymbol{e}_1 + \ldots + \partial f/\partial x_i\ \boldsymbol{e}_i + \partial f/\partial x_n\ \boldsymbol{e}_n$$

dove $\boldsymbol{e}_i$ è un vettore unitario che punta nella direzione positiva x_i

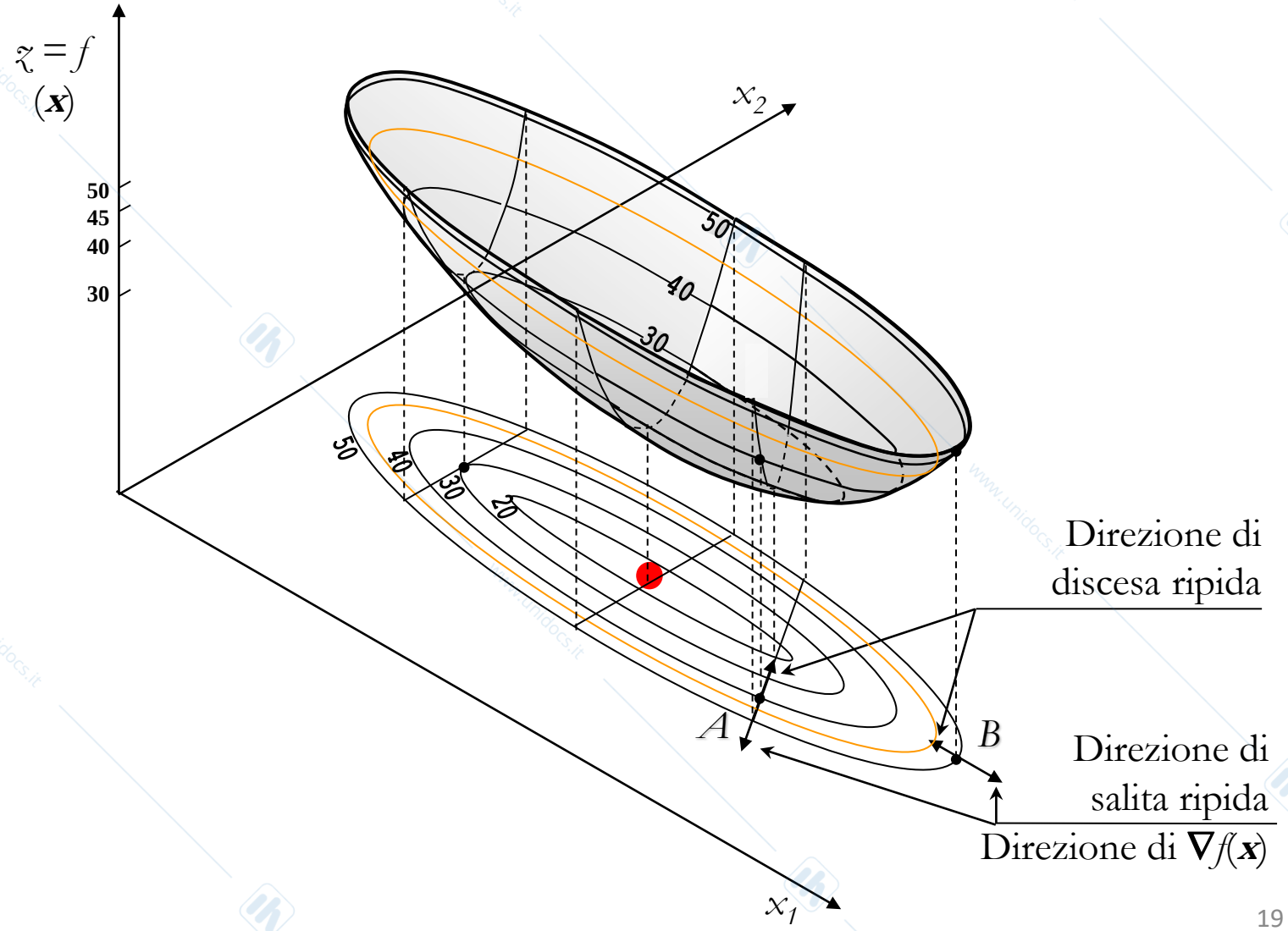
Il gradiente è un concetto di importanza fondamentale nella teoria dell'ottimizzazione. In ogni punto P di E^n, espresso dal vettore $\boldsymbol{x}^{\wedge}$, la componente di $\nabla \boldsymbol{f}$ nella direzione x_i (cioè la derivata parziale rispetto ad x_i) è il tasso di variazione di f rispetto ad uno spostamento da $\boldsymbol{x}^{\wedge}$ in quella direzione.

Il $\nabla \boldsymbol{f}$ (**x**) è la somma di tutte le componenti, cioè di tutte le variazioni. Pertanto in un certo punto P ($\boldsymbol{x}_i$) il gradiente $\nabla \boldsymbol{f}$ (**x**) punta nella direzione in cui il tasso di variazione di f è massimo e la sua lunghezza è pari al massimo tasso di variazione. Quindi $\nabla \boldsymbol{f}$ nel punto P è perpendicolare alla curva di livello passante per P.

La figura mostra una funzione $f(\boldsymbol{x})$ di due variabili, alcuni *contorni* della stessa, con le relative proiezioni sul piano (x_1,x_2), ed il vettore $\nabla \boldsymbol{f}$ nei punti A e B. Il vettore $\nabla \boldsymbol{f}$ è perpendicolare ai contorni che passano per i punti stessi. Il gradiente punta in un verso nella direzione di massima salita della funzione e nell'altro verso nella direzione di massima discesa della funzione.

Gradiente $\nabla f(\boldsymbol{x})$

Funzione $z = f(x_1, x_2)$, Curve di livello e Direzioni di discesa (salita) ripida



Determinazione del punto di ottimo multidimensionale

Per trovare il punto di ottimo di una funzione multidimensionale potremmo utilizzare il gradiente della funzione stessa ed imporre la sua nullità, così come potevamo fare con la derivata di una funzione monodimensionale.

Nel caso multidimensionale le difficoltà sono ancora maggiori.

Se imponiamo la nullità del gradiente generiamo infatti un sistema di equazioni alle derivate parziali che è tanto più complicato da risolvere quanto più è alto il grado della funzione.

Spostamento nello spazio delle variabili

Prima di spiegare l'algoritmo a generazione di una successione di punti in uno spazio multidimensionale è utile ricordare come si effettua uno spostamento nello spazio delle variabili

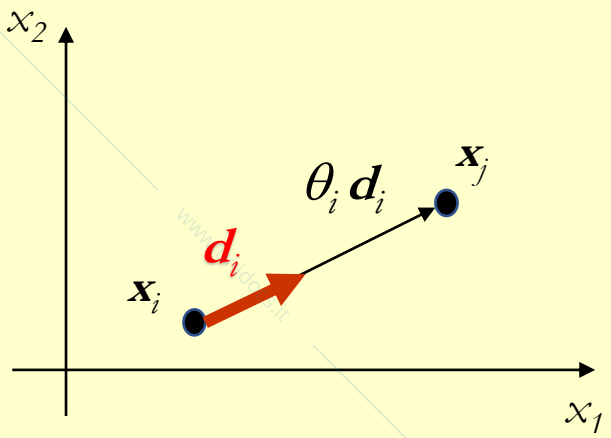
Uno spostamento nello spazio delle variabili dal punto $\boldsymbol{x}_i$ al punto $\boldsymbol{x}_j$

può essere espresso come somma di vettori,

aggiungendo al vettore $\boldsymbol{x}_i$ il vettore $\theta_i \, \boldsymbol{d}_i$

dove $\boldsymbol{d}_i$ è una direzione di spostamento e θ_i è la lunghezza dello spostamento

$$\boldsymbol{x}_j = \boldsymbol{x}_i + \theta_i \, \boldsymbol{d}_i$$



Generazione di una successione di punti

Punto iniziale x_0

Relazione ricorsiva $x_{k+1}=x_k+\theta_k\, d_k$

dove:

d_k vettore con n componenti → **direzione di spostamento**

θ_k scalare non negativo → **lunghezza dello spostamento**

$\{x_k\} \equiv \{x_0, x_1, x_2 \ldots\}$ → **successione di punti con miglioramento**

cioè $f(x_{k+1}) > f(x_k)$ se problema Max z

$f(x_{k+1}) < f(x_k)$ se problema Min z

$$x_1=x_0+\theta_0\, d_0$$
$$x_2=x_1+\theta_1\, d_1$$
$$x_3=x_2+\theta_2\, d_2$$
$$x_4=\ldots$$

Algoritmo di salita (discesa) ripida

L'algoritmo di salita (discesa) ripida si basa sulla relazione ricorsiva

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k \pm \theta_k \nabla f(\boldsymbol{x}_k)$$

La direzione di spostamento è quella del gradiente $\nabla f(\boldsymbol{x}_k)$ nel punto $\boldsymbol{x}_k$

θ_k scalare non negativo è la lunghezza dello spostamento nella direzione del gradiente

L'algoritmo genera una successione di punti da un punto iniziale $\boldsymbol{x}_0$

$$\{x_0, x_1, x_2 \ldots\}$$

con miglioramento della funzione obiettivo

$$\boldsymbol{f(x_{k+1}) > f(x_k)} \quad \textbf{se la f.o. è Max}$$

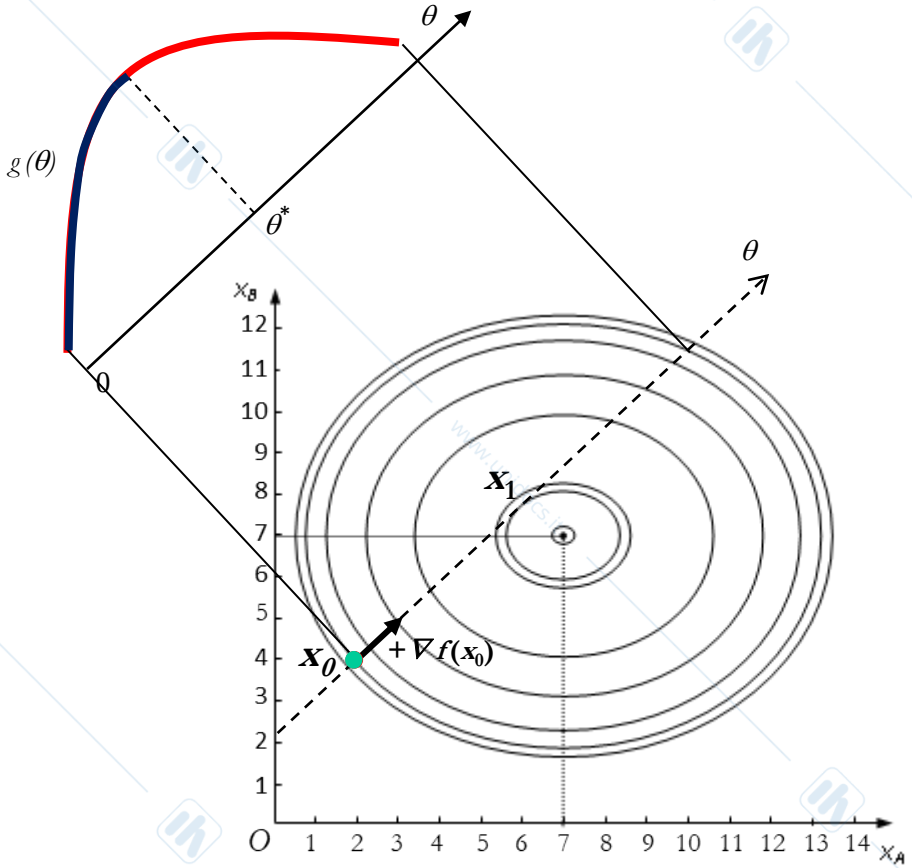
$$\boldsymbol{f(x_{k+1}) < f(x_k)} \quad \textbf{se la f.o. è Min}$$

Il criterio di convergenza è legato alla dimensione del gradiente con l'approssimazione desiderata: $\nabla f(\boldsymbol{x}^*) \leq \mathbf{e}$

Direzione di spostamento da x_o e funzione g(θ)

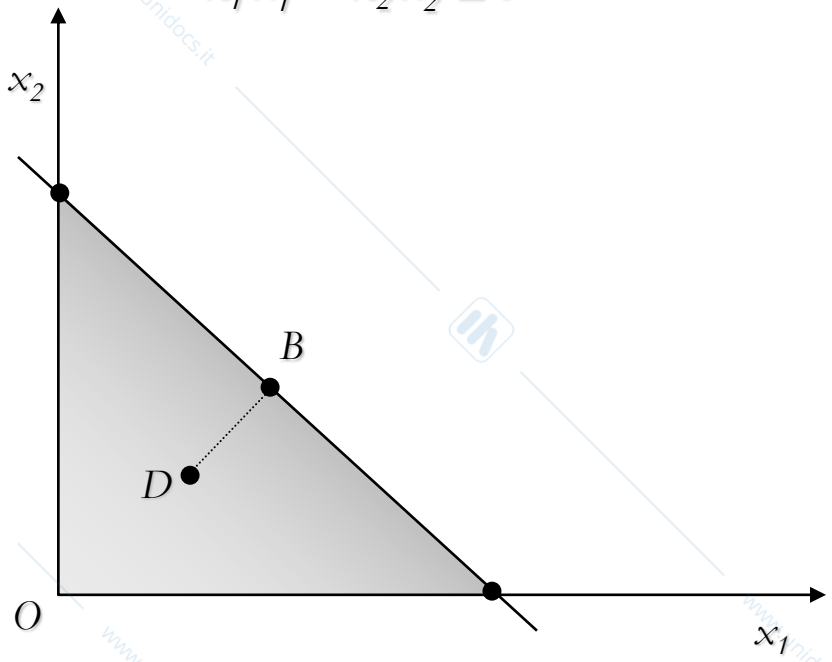
$$z = f(x_A, x_B) = 56x_A - 4x_A^2 + 84x_B - 6x_B^2 - 300$$

La slide mostra alcune curve di livello della funzione e la funzione $g(\theta_0)$ che si genera spostandosi da $\boldsymbol{x}_0$ lungo la direzione del gradiente in $\boldsymbol{x}_0$. La direzione del gradiente genera un asse coordinato θ. Se si effettua uno spostamento da $\mathbf{x_o}$ lungo quest'asse si percorre una curva $g(\theta)$ sulla funzione $z = f(x_A, x_B)$.



Vincolo di disuguaglianza ≤

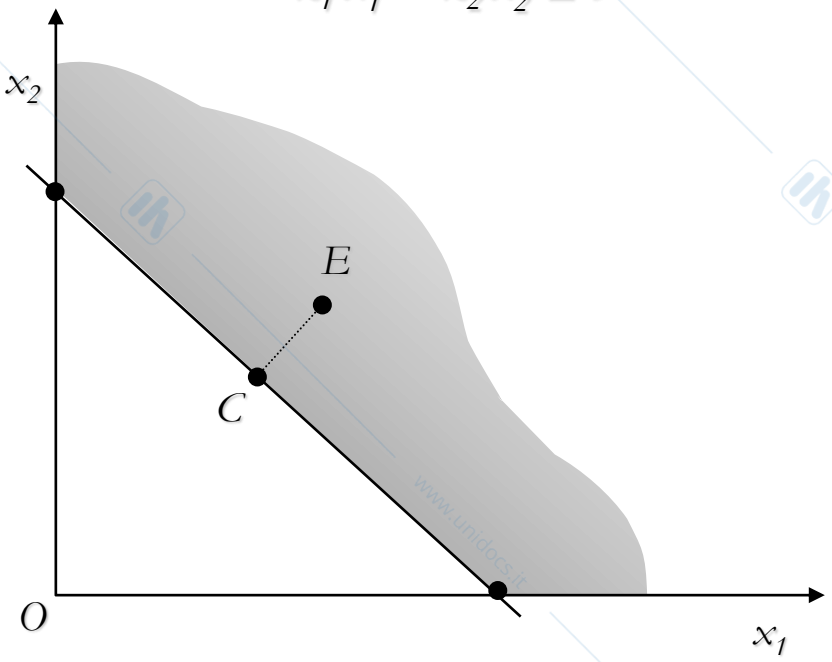
$$k_1 x_1 + k_2 x_2 \leq b$$



Nel punto B il vincolo è soddisfatto con il segno di =
Nel punto D il vincolo è soddisfatto con il segno di <
$k_1 x_1 - k_2 x_2 < b$

Vincolo di disuguaglianza ≥

$$k_1 x_1 + k_2 x_2 \geq b$$



Nel punto C il vincolo è soddisfatto con =
Nel punto E il vincolo è soddisfatto con >
$k_1 x_1 - k_2 x_2 > b$

Algoritmo a direzione ammissibile

I problemi di ottimizzazione lineare vincolata si risolvono con algoritmi a direzione ammissibile, che sono oggetto del corso di Ricerca Operativa 2.

Questi algoritmi sono basati sulla generazione di una successione di punti nel dominio ammissibile, a partire da un punto iniziale noto $\boldsymbol{x_0}$, attraverso la relazione ricorsiva già utilizzata nel caso non vincolato: $\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\theta_k \boldsymbol{d}_k$.

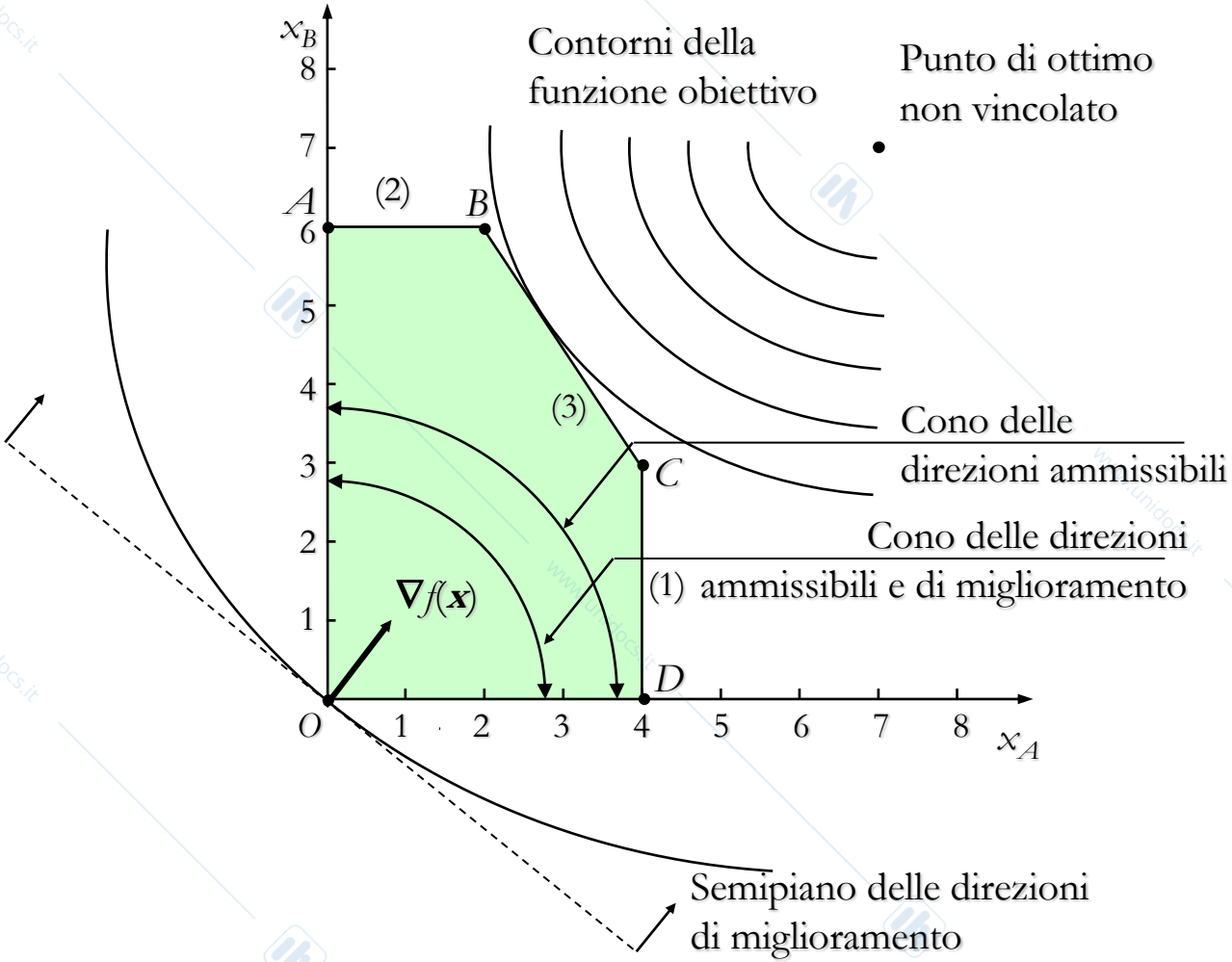
Poiché il problema è vincolato i punti della successione devono appartenere al dominio ammissibile e quindi le direzioni di spostamento devono essere di miglioramento (come già avveniva nel caso non vincolato) e ammissibili, cioè devono generare punti ancora appartenenti al dominio.

Gli algoritmi a direzione ammissibile sono basati quindi sulla individuazione di direzioni di spostamento ammissibili e di miglioramento.

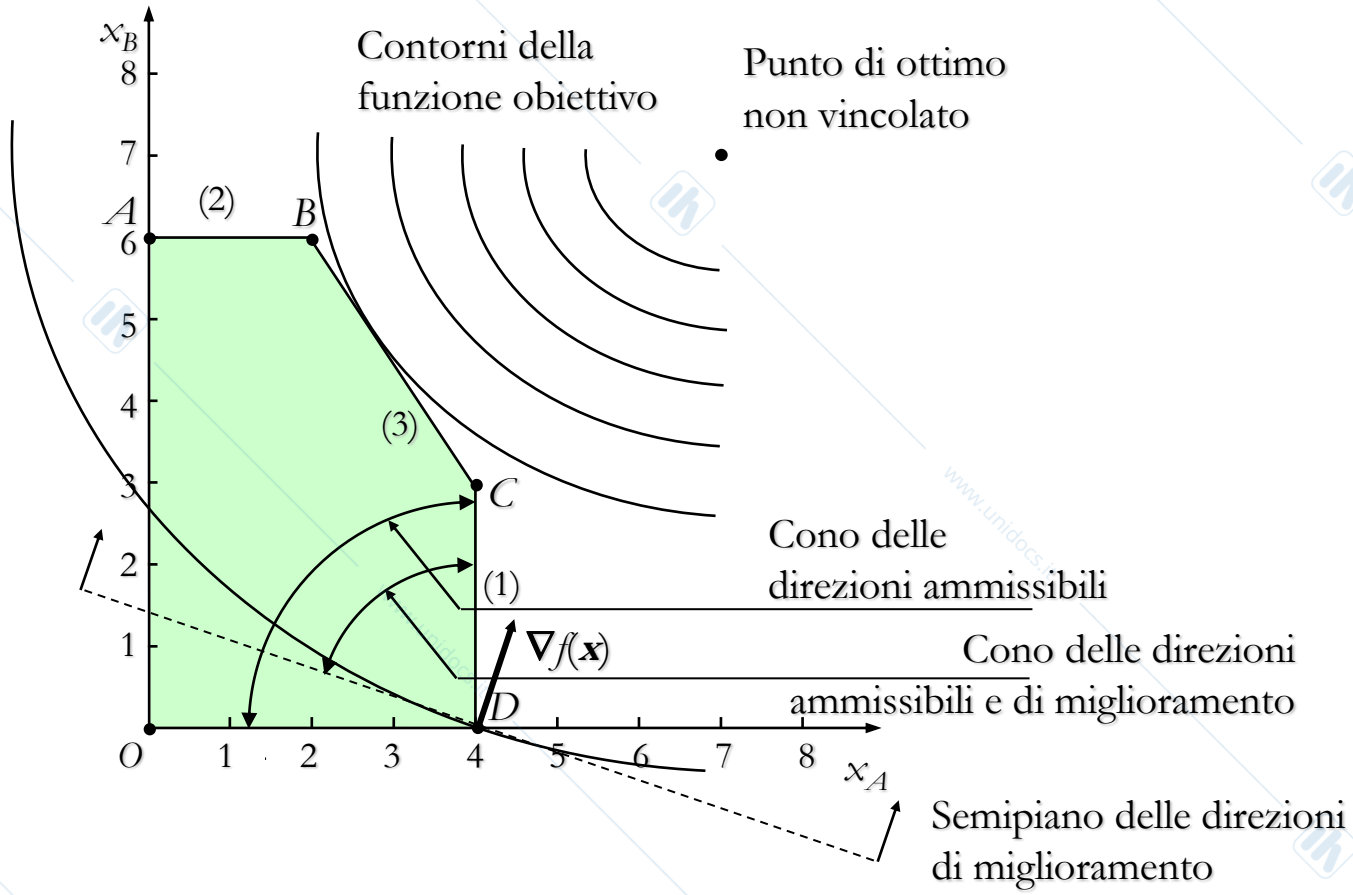
Nelle slide che seguono si illustrano graficamente i seguenti concetti:

- **cono delle direzioni ammissibili**
- **cono delle direzioni di miglioramento**
- **cono delle direzioni ammissibili e di miglioramento**
- Si illustra poi graficamente il **percorso di un algoritmo a direzione ammissibile** dal punto $\boldsymbol{x_0}$ iniziale al punto di ottimo $\boldsymbol{x^*}$**.**

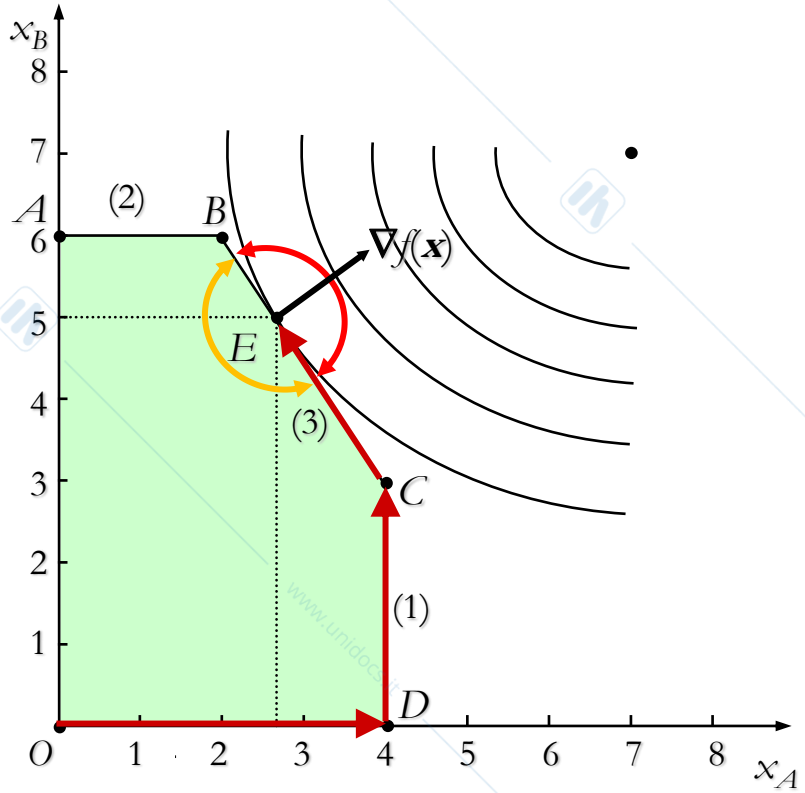
Direzioni ammissibili e di miglioramento nel vertice O



Direzioni ammissibili e di miglioramento nel vertice D

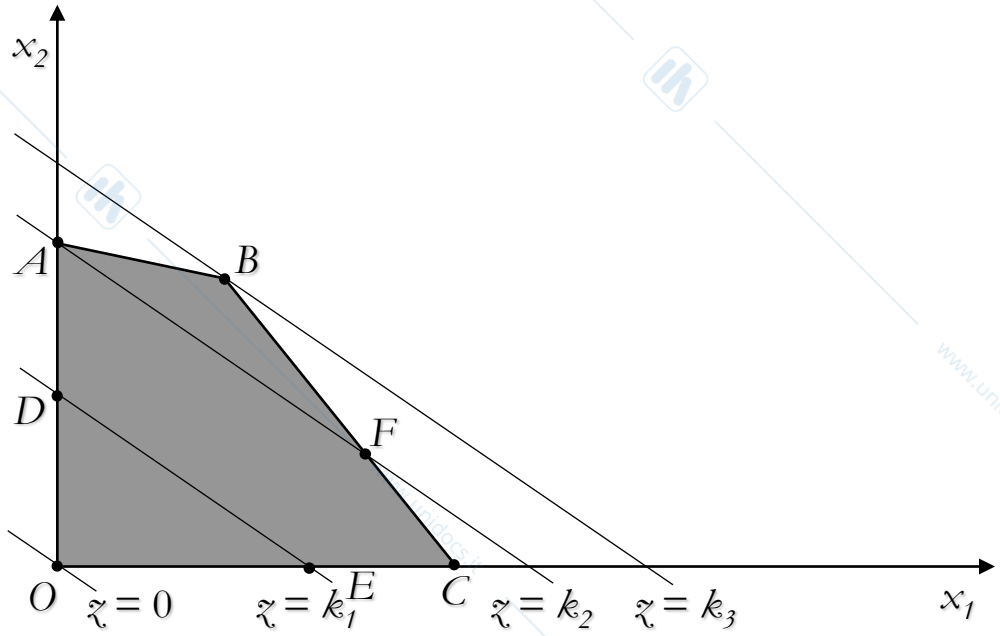


Percorso di un algoritmo a direzione ammissibile



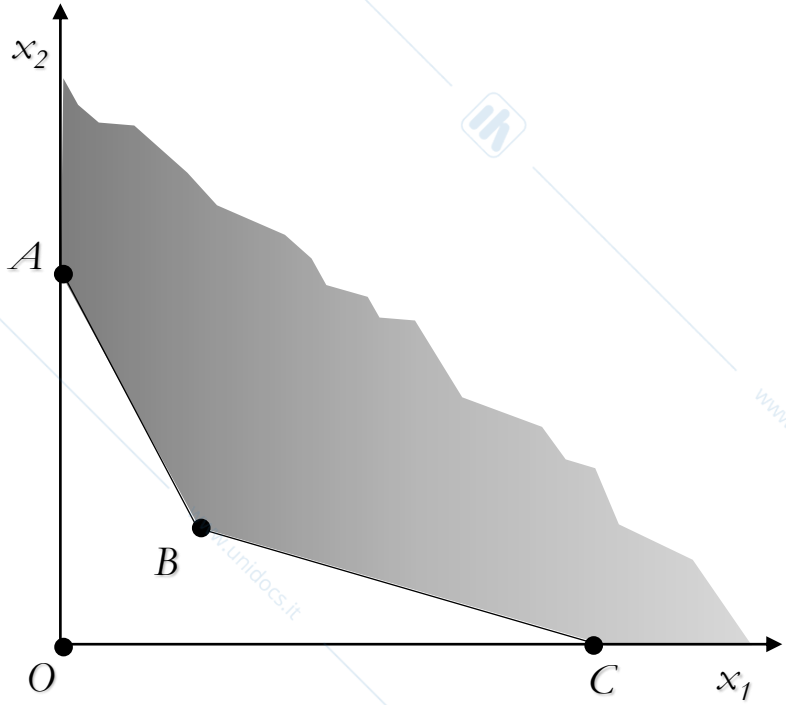
Funzione obiettivo z e dominio di ammissibilità nello spazio bidimensionale x_1,x_2

Il punto di ottimo di un problema di programmazione lineare è sulla frontiera del dominio ammissibile ed in particolare è un vertice del dominio.

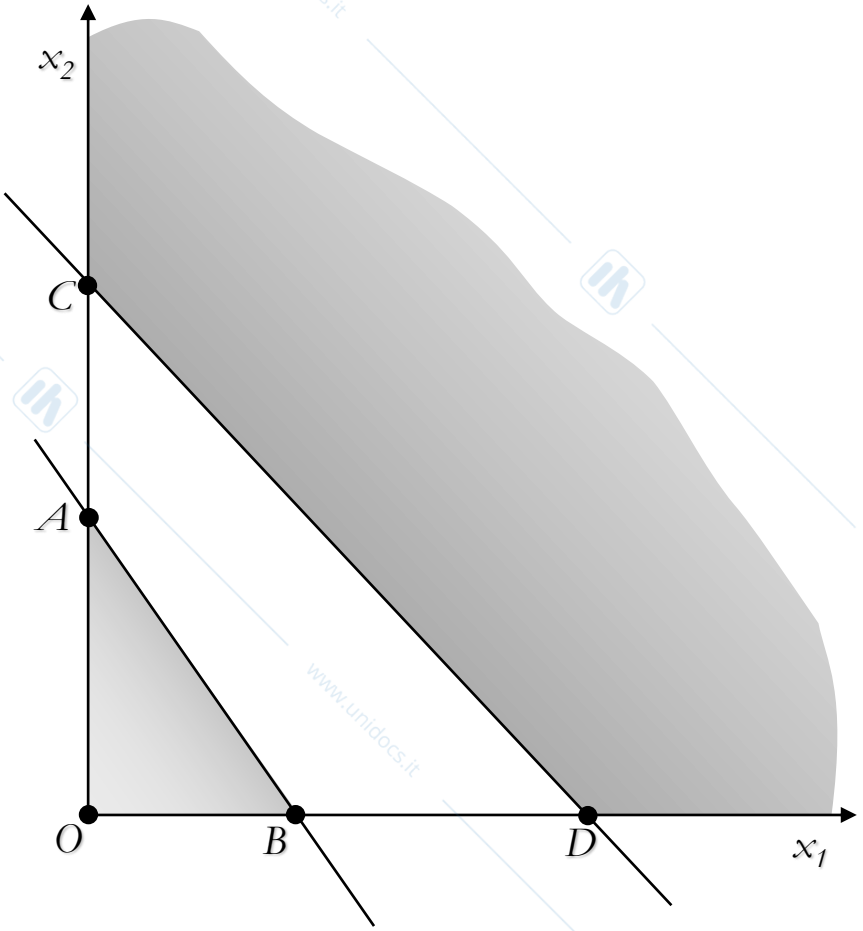


Formulazione di un modello di Programmazione Lineare

Dominio aperto



Dominio inconsistente



Ottimizzazione lineare

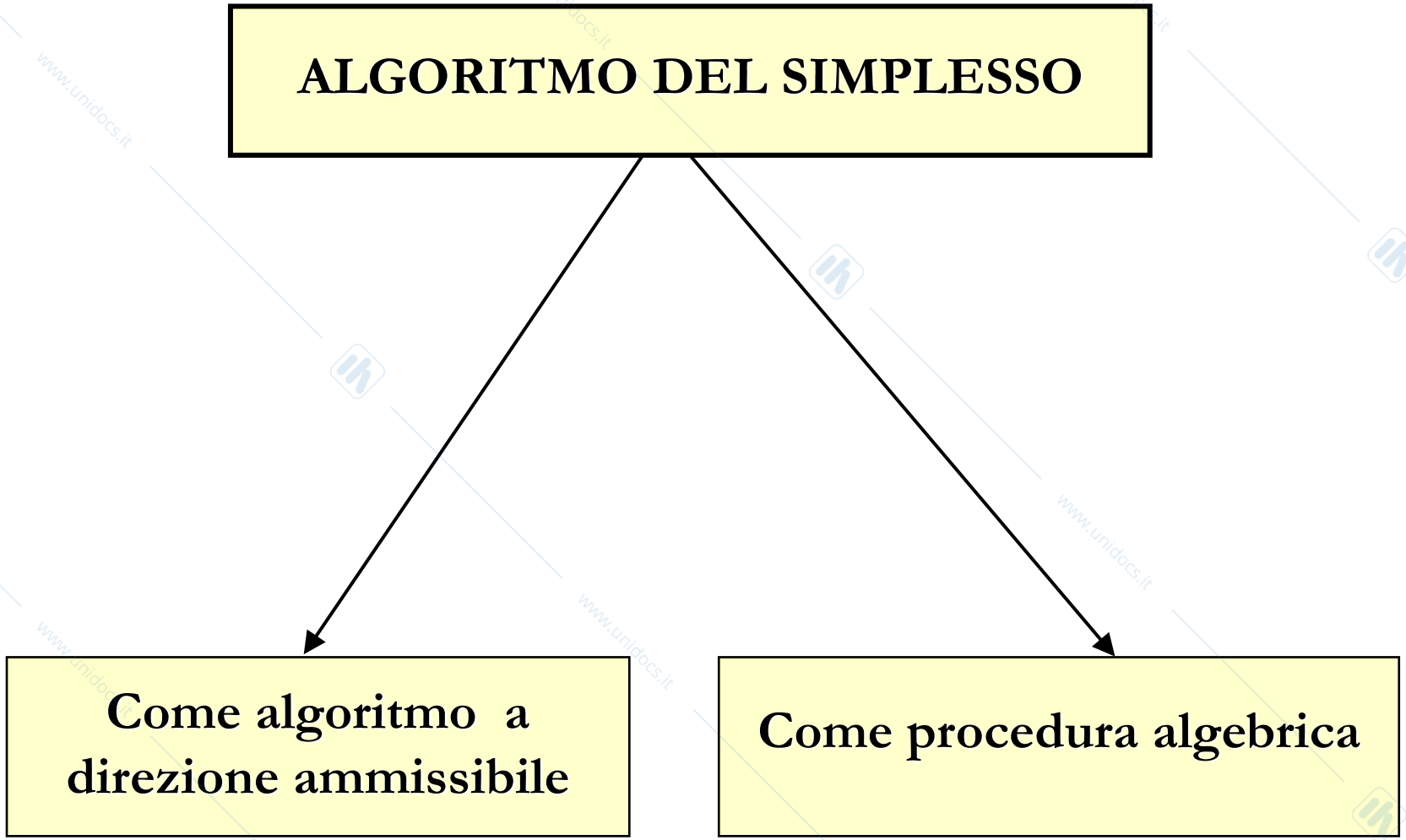
Algoritmo del Simplesso

Algoritmo del Simplesso

Per la soluzione dei problemi di programmazione lineare si utilizza comunemente il noto algoritmo del simplesso, proposto da G.B. Dantzig alla fine degli anni ‘40.

In questa lezione l’algoritmo viene presentato inizialmente come algoritmo a direzione ammissibile, specializzato per il caso lineare, e poi come algoritmo basato su una procedura di tipo algebrico, basata sulla trasformazione dei vincoli (di disuguaglianza e uguaglianza) in un sistema di equazioni, volta a generare particolari soluzioni del sistema di equazioni.

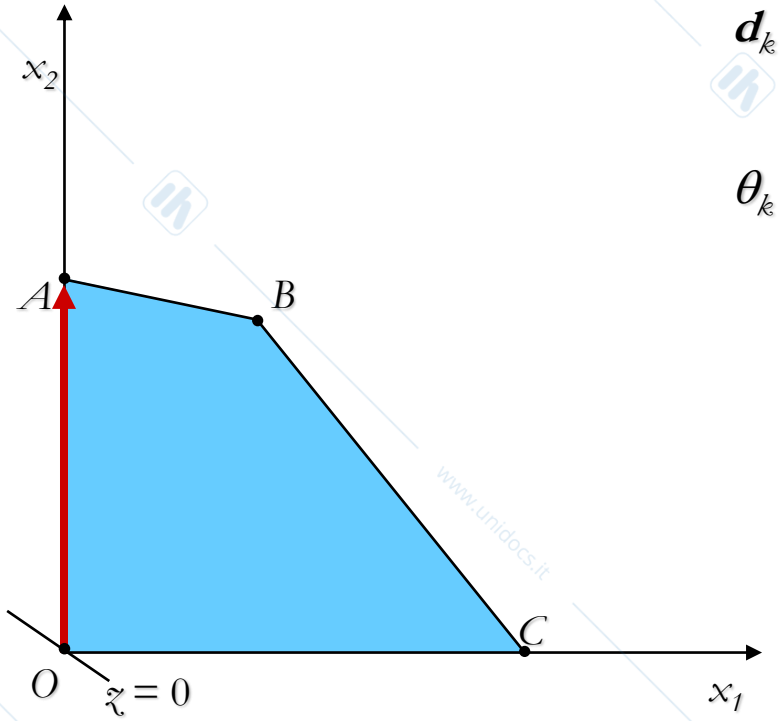
L’algoritmo viene descritto prima per il caso in cui tutti i vincoli sono del tipo ≤ e poi per il caso in cui sono presenti anche vincoli del tipo = e/o ≥.



Algoritmo del Simplesso come algoritmo a direzione ammissibile

Algoritmo del Simplesso come algoritmo a direzione ammissibile

Gli algoritmi a direzione ammissibile generano una successione di punti attraverso la relazione: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \theta_k \boldsymbol{d}_k$



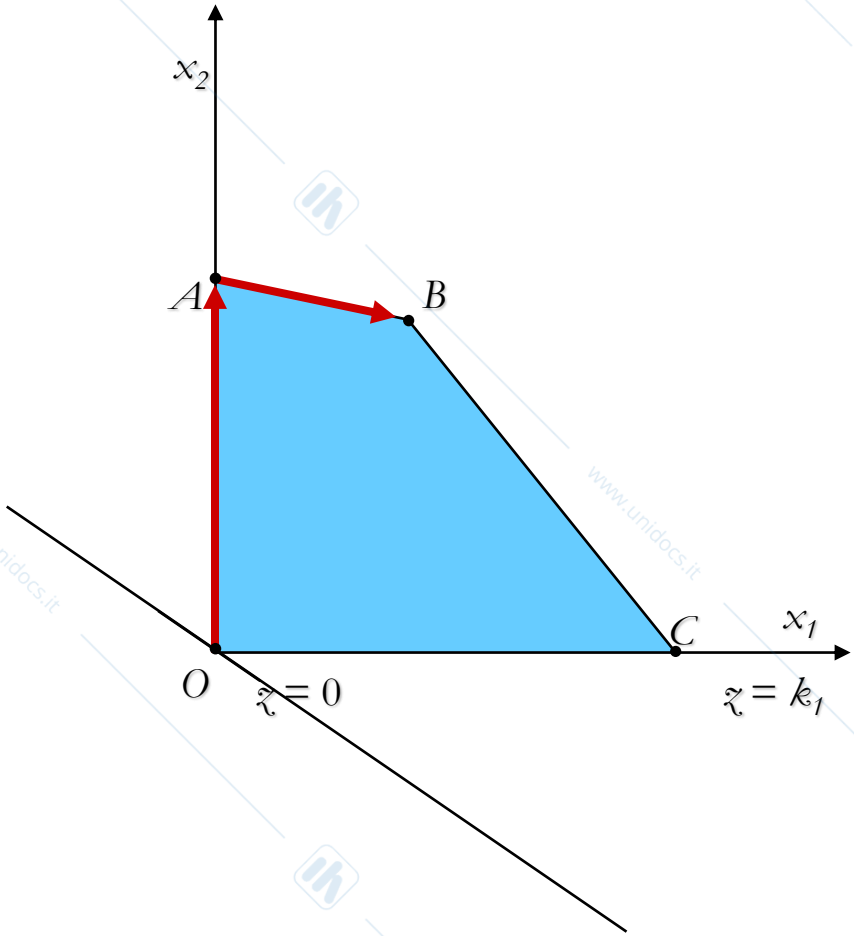
$\boldsymbol{d}_k$ **direzione di spostamento ammissibile e di miglioramento**

θ_k **lunghezza dello spostamento**

A partire dal vertice O la direzione di spostamento lungo lo spigolo OA è una direzione ammissibile di miglioramento per la funzione obiettivo z (da z = 0 a z = k_1).

Algoritmo del Simplesso come algoritmo a direzione ammissibile

Gli algoritmi a direzione ammissibile generano una successione di punti attraverso la relazione: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \theta_k \boldsymbol{d}_k$



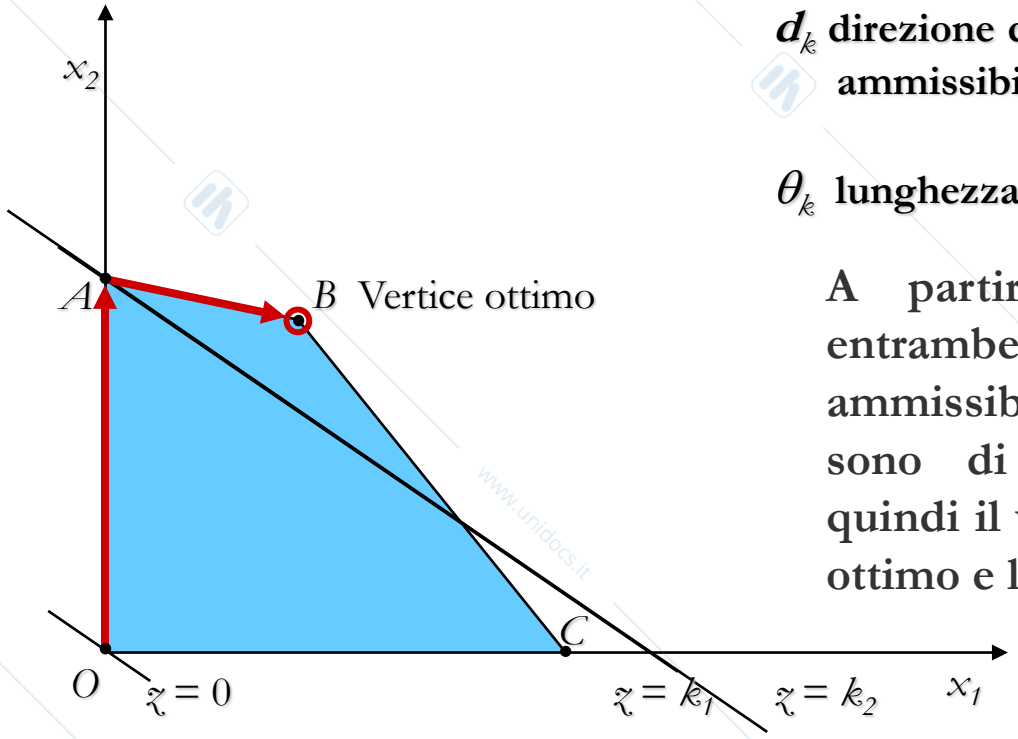
$\boldsymbol{d}_k$ **direzione di spostamento ammissibile e di miglioramento**

θ_k **lunghezza dello spostamento**

A partire dal vertice A la direzione di spostamento lungo lo spigolo AO è di peggioramento, mentre quella lungo AB è di miglioramento (si passa da $z = k_1$ a $z = k_2$).

Algoritmo del Simplesso come algoritmo a direzione ammissibile

Gli algoritmi a direzione ammissibile generano una successione di punti attraverso la relazione: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \theta_k \boldsymbol{d}_k$



$\boldsymbol{d}_k$ **direzione di spostamento ammissibile e di miglioramento**

θ_k **lunghezza dello spostamento**

A partire dal vertice B entrambe le direzioni ammissibili di spostamento sono di peggioramento e quindi il vertice B è il vertice ottimo e l'algoritmo termina.

Step dell'Algoritmo del Simplesso

Soluzione iniziale

Si deve disporre di una soluzione iniziale x_0 (corrispondente al vertice origine degli assi del problema originario o, se necessario, a un altro vertice, da determinarsi con una procedura opportuna).

Scelta della direzione di spostamento

Per generare soluzioni corrispondenti ai vertici del dominio, le direzioni di spostamento devono essere quelle degli spigoli del dominio. Quindi in ogni vertice $\mathbf{x_k}$ bisogna esaminare le direzioni di spostamento degli spigoli incidenti in ciascun vertice.

Queste direzioni sono tutte ammissibili (perché gli spigoli costituiscono la frontiera del dominio).

Di esse una o più possono essere di miglioramento, altre sono di peggioramento.

Si può scegliere una qualunque delle direzioni ammissibili di miglioramento, ma tradizionalmente si sceglie quella corrispondente al più alto valore di derivata direzionale della funzione obiettivo (positivo per un problema di *max*, negativo per un problema di *min*).

Step dell'Algoritmo del Simplesso

Lunghezza dello spostamento

La lunghezza dello spostamento è quella massima consentita lungo lo spigolo fino a raggiungere il vertice adiacente, senza uscire dal dominio di ammissibilità.

Determinazione di una nuova soluzione ammissibile

Viene effettuata con una procedura algebrica basata sulla trasformazione del sistema di vincoli del modello in un sistema di equazioni e sulla generazione di particolari soluzioni di questo sistema (soluzioni basiche ammissibili) corrispondenti ai vertici del dominio di ammissibilità, come verrà illustrato nel seguito.

Step dell'Algoritmo del Simplesso

Test di ottimalità sulla soluzione corrente

Il test richiede l'analisi dei valori di derivata direzionale della funzione obiettivo lungo gli spigoli incidenti nel vertice stesso, che costituiscono le direzioni ammissibili di spostamento:

Se esistono valori di derivata direzionale positivi per un problema di *max*, o negativi per un problema di *min*, allora esistono direzioni ammissibili di miglioramento. Si può determinare quindi una nuova soluzione a partire dal vertice corrente lungo una direzione di spigolo da scegliere opportunamente.

Se i valori di derivata direzionale lungo gli spigoli incidenti nel vertice corrente sono tutti non positivi (per un problema di *max*) o tutti non negativi (per un problema di *min*), allora nessuna delle direzioni ammissibili è una direzione di miglioramento: è stata raggiunta la soluzione ottima e l'algoritmo termina.

Algoritmo del Simplesso come procedura algebrica

Trasformazione di un sistema di vincoli in un sistema di equazioni

L'elemento fondamentale dell'algoritmo del Simplesso è la trasformazione del sistema di vincoli (disequazioni e/o equazioni) in un sistema di equazioni.

Questa operazione si compie attraverso l'aggiunta di variabili di compensazione (variabili *slack* in inglese):

- In ciascun vincolo di disuguaglianza ≤ si aggiunge una variabile slack (non negativa), con segno +.

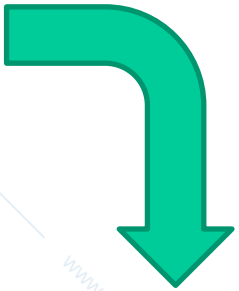
- In ciascun vincolo di disuguaglianza ≥ si aggiunge una variabile slack (non negativa) con segno -, quindi si sottrae una variabile slack.

- I vincoli di uguaglianza restano inalterati.

Nella slide successiva si riporta la trasformazione di un sistema di *m* disequazioni e *n* variabili in un sistema di *m* equazioni in (*n + m)* variabili (le *n* variabili originarie + le *m* variabili slack).

Trasformazione di un sistema di disequazioni in un sistema di equazioni con l'aggiunta delle variabili slack

$$
\begin{array}{lllllllll}
a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & \leq & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & \geq & b_2 \\
\dots & & \dots & & \dots & & \dots & & \dots \\
a_{i1}x_1 & + & a_{i2}x_2 & + & \dots & + & a_{in}x_n & \leq & b_i \\
\dots & & \dots & & \dots & & \dots & & \dots \\
a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & \geq & b_m
\end{array}
$$



$$
\begin{array}{lllllllllll}
a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & + y_1 & & = & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & & - y_2 & = & b_2 \\
\dots & & \dots & & \dots & & \dots & & & & \dots \\
a_{i1}x_1 & + & a_{i2}x_2 & + & \dots & + & a_{in}x_n & & + y_i & = & b_i \\
\dots & & \dots & & \dots & & \dots & & & & \dots \\
a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & & - y_m & = & b_m
\end{array}
$$

Trasformazione di un sistema di disequazioni in un sistema di equazioni con l'aggiunta delle variabili slack

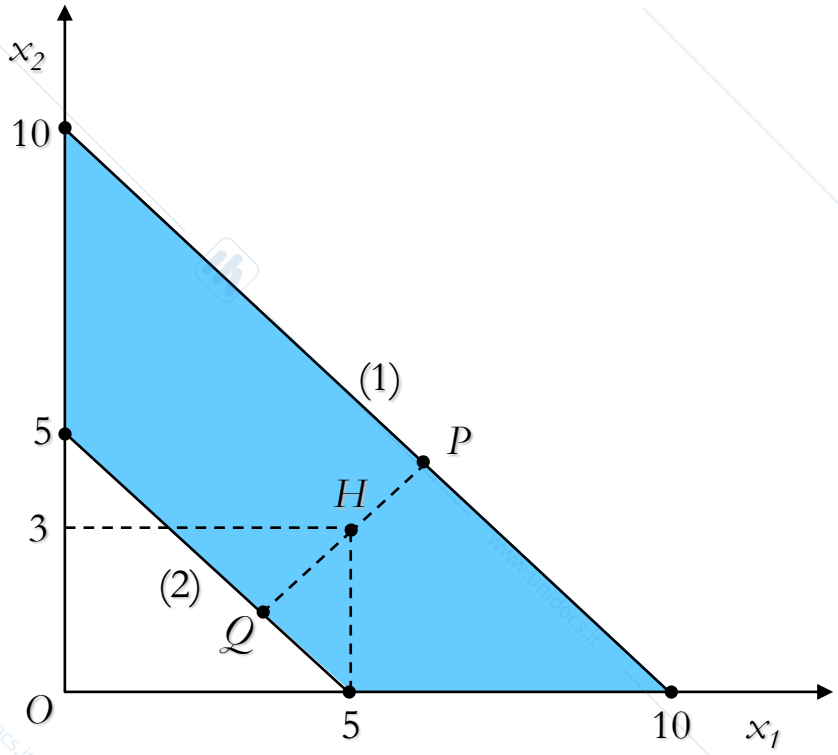
- Nella slide successiva si riporta un semplice esempio in 2 variabili volto a:
- illustrare graficamente il significato delle variabili slack
- mostrare che l'aggiunta delle variabili slack non muta le soluzioni del sistema di disequazioni.

Variabili slack

Sistema di disequazioni in due variabili e sua trasformazione in un sistema di equazioni.

$$(1)\quad x_1 + x_2 \le 10$$
$$(2)\quad x_1 + x_2 \ge 5$$

$$(1)\quad x_1 + x_2 + y_1 = 10$$
$$(2)\quad x_1 + x_2 - y_2 = 5$$



Vincolo ≤

Nel punto P il vincolo 1 è soddisfatto con il segno = e quindi $y_1 = 0$

Nel punto H, $\boldsymbol{x} = (x_1, x_2) = (5, 3)$ il vincolo 1 è soddisfatto con il segno < e quindi $y_1 = 10 - x_1 - x_2 = 2 > 0$

Vincolo ≥

Nel punto Q il vincolo 2 è soddisfatto con il segno = e quindi $y_2 = 0$

Nel punto H, $\boldsymbol{x} = (x_1, x_2) = (5, 3)$ il vincolo 2 è soddisfatto con il segno > e quindi $y_2 = x_1 + x_2 - 5 = 3 > 0$

Soluzioni del sistema di disequazioni/equazioni

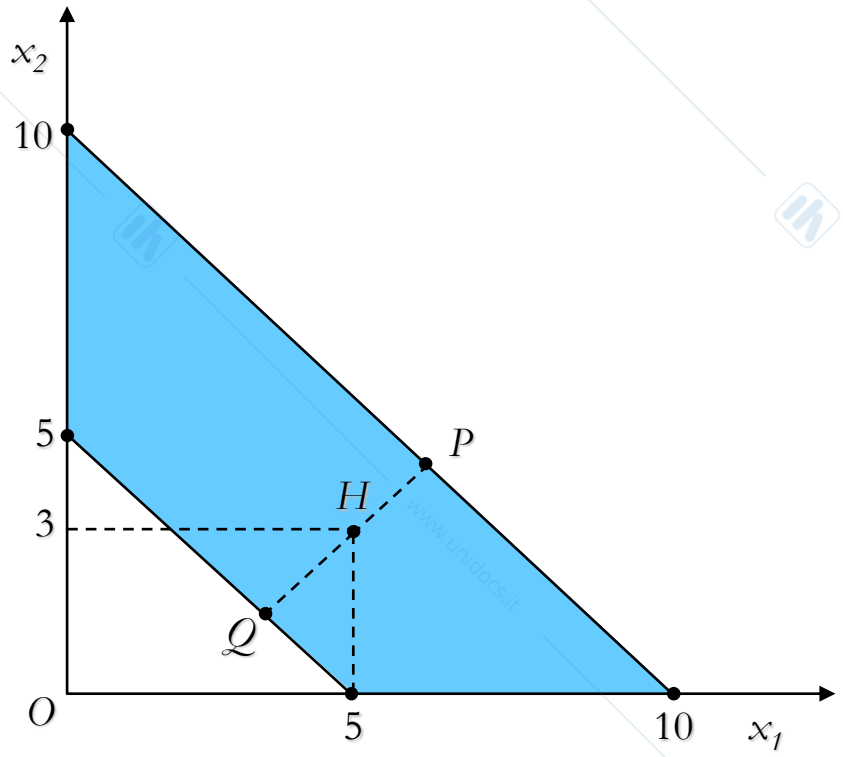
I punti del dominio corrispondono a soluzioni del sistema di disequazioni, nello spazio x_1, x_2

Le soluzioni del sistema di equazioni sono nello spazio ampliato x_1, x_2, y_1, y_2.

Per esempio, il punto H:

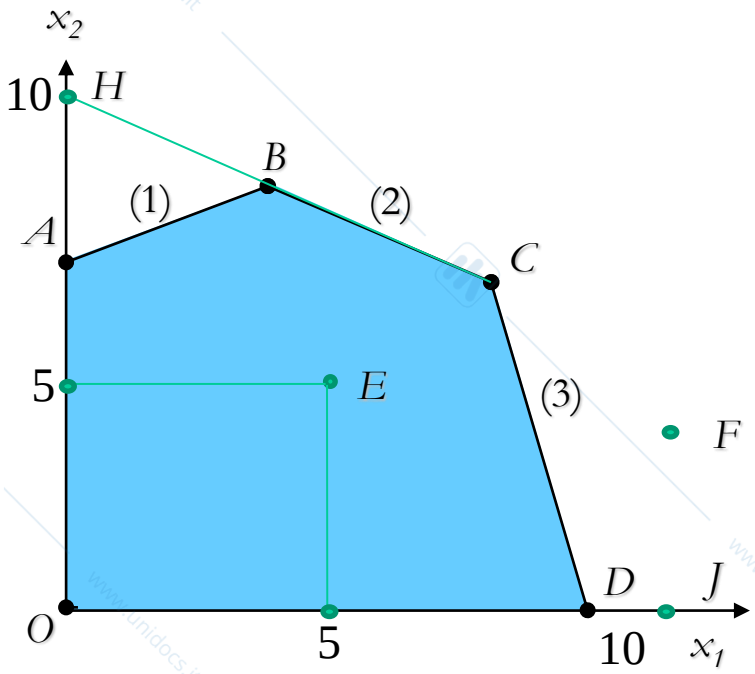
nello spazio 2dimensionale (x_1, x_2)
è **[x]** = (x_1, x_2) = (5, 3)

nello spazio 4dimensionale (x_1, x_2, y_1, y_2)
è [**x:y**] = (x_1, x_2, y_1, y_2) = (5, 3, 2, 3) =



Quindi c'è una corrispondenza biunivoca tra
le soluzioni del sistema di disequazioni e le soluzioni del sistema di equazioni

Soluzioni di un sistema di equazioni con m equazioni ed n variabili



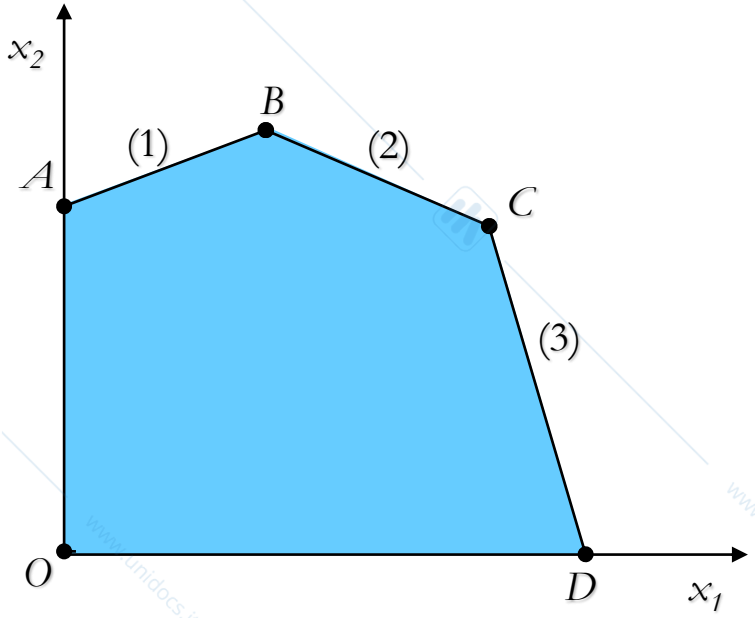
Punto F $x_1>0, x_2>0, y_1>0, y_2>0, y_3<0$

Punto J $x_1>0, x_2=0, y_1>0, y_2>0, y_3<0$

Punto E $x_1>0, x_2>0, y_1>0, y_2>0, y_3>0$

Punto H $x_1=0, x_2>0, y_1<0, y_2=0, y_3>0$

Vertici del dominio e valori delle variabili



	x_1	x_2	y_1	y_2	y_3
Vertice O	$= 0$	$= 0$	> 0	> 0	> 0
Vertice A	$= 0$	> 0	$= 0$	> 0	> 0
Vertice B	> 0	> 0	$= 0$	$= 0$	> 0
Vertice C	> 0	> 0	> 0	$= 0$	$= 0$
Vertice D	> 0	$= 0$	> 0	> 0	$= 0$

I vertici del dominio di ammissibilità corrispondono a particolari soluzioni (*n-m* valori nulli, *m* valori non negativi) che si definiscono soluzioni basiche ammissibili

Soluzioni di un sistema di equazioni con m equazioni ed n variabili

Le soluzioni del sistema di equazioni così ottenuto possono essere classificate nel seguente modo:

- **Soluzioni generiche [*n* valori qualunque (positivi, negativi, nulli)];**
  - **Come ad esempio i punti F e J**
- **Soluzioni ammissibili (*n* valori non negativi), sicuramente appartenenti al dominio ammissibile;**
  - **Come ad esempio il punto E**
- **Soluzioni basiche (*n-m* valori nulli, *m* valori qualunque);**
  - **Come ad esempio il punto H**
- **Soluzioni basiche ammissibili (*n-m* valori nulli, *m* valori non negativi), sicuramente appartenenti al dominio ammissibile;**
  - **Tutti e soli i vertici del dominio**

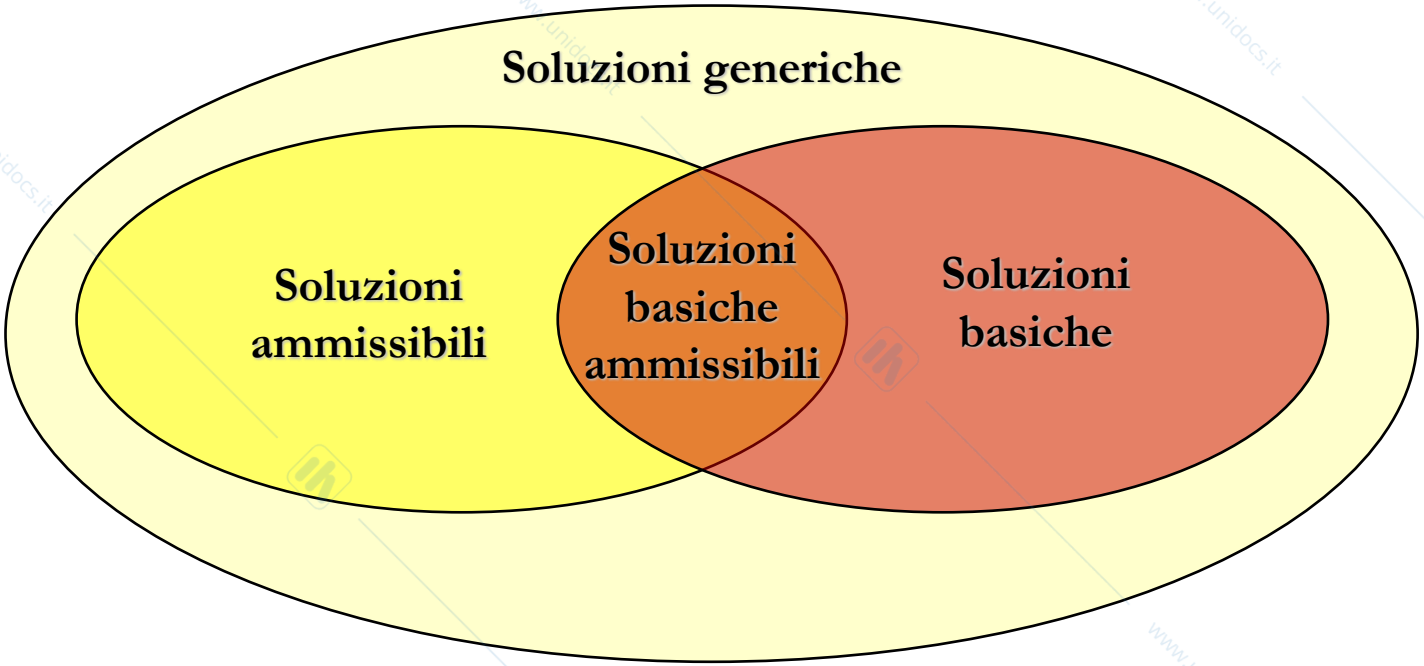
Le variabili nulle di una soluzione basica ammissibile saranno indicate nel seguito come variabili non basiche. Le variabili non negative (≥ 0) saranno indicate nel seguito come variabili basiche.

Soluzioni di un sistema di equazioni con m equazioni ed n variabili

Soluzioni generiche:	**n valori qualunque**
Soluzioni ammissibili	**n valori non negativi**
Soluzioni basiche:	**n-m valori nulli, m valori qualunque**
Soluzioni basiche ammissibili:	**n-m valori nulli, m valori non negativi**

La slide successiva mostra questi insiemi di soluzioni e la loro relazione

Soluzioni di un sistema di equazioni con m equazioni ed n variabili



Soluzioni generiche:	**n valori qualunque**
Soluzioni ammissibili	**n valori non negativi**
Soluzioni basiche:	**n-m valori nulli, m valori qualunque**
Soluzioni basiche ammissibili:	**n-m valori nulli, m valori non negativi**

Algoritmo del simplesso come generatore di soluzione basiche ammissibili

Si è mostrato in precedenza che il punto di ottimo di un problema di Programmazione Lineare corrisponde a un vertice del dominio ammissibile.

Le slide precedenti hanno mostrato che i vertici del dominio di ammissibilità corrispondono a soluzioni basiche ammissibili.

Dunque, è necessario costruire un algoritmo che generi soluzioni basiche ammissibili verificando per ciascuna di esse il soddisfacimento del test di ottimalità.

A tale scopo è necessario sviluppare una procedura algebrica.

Procedura algebrica dell'Algoritmo del Simplesso

Si distingueranno nel seguito due casi:

1. **tutti i vincoli sono del tipo ≤,**
2. **almeno un vincolo è del tipo ≥ o =.**

L'algoritmo sarà descritto inizialmente per il caso 1.

Nel caso 2 l'algoritmo è sostanzialmente lo stesso. Esso differisce dal caso 1 solo perché, come si vedrà, la prima soluzione basica ammissibile non è immediatamente disponibile e quindi va opportunamente determinata.

Ottimizzazione lineare

Algoritmo del Simplesso

Algoritmo del Simplesso con vincoli del tipo ≤

Algoritmo del simplesso con vincoli del tipo ≤

Nelle slide successive si descrive inizialmente l'algoritmo del simplesso nel caso in cui il modello presenti solo vincoli del tipo ≤.

In questo caso siamo sicuri che l'origine degli assi appartiene al dominio di ammissibilità e quindi la soluzione basica ammissibile iniziale è quella associata al vertice origine.

Questa condizione facilita inoltre la determinazione della soluzione iniziale dell'algoritmo. Infatti, se i vincoli sono tutti del tipo ≤ si aggiungono solo variabili *slack* con il segno positivo, e quindi la matrice del sistema di equazioni ottenuto dalla trasformazione contiene sicuramente una matrice identità.

Il sistema di equazioni è dunque in forma canonica. Avere un sistema in forma canonica consente di determinare immediatamente una prima soluzione basica ammissibile, come descritto nelle slide successive.

Sistemi in forma canonica e soluzioni basiche ammissibili

Si ricordi che un sistema di equazioni si dice in forma canonica quando nella matrice $\boldsymbol{A}$ dei coefficienti a_{ij} delle equazioni è individuabile una matrice identità, cioè una matrice in cui tutti gli elementi della diagonale principale sono pari ad 1 e tutti gli altri sono pari a 0.

A partire da un sistema in questa forma è molto facile determinare una soluzione basica ammissibile. Infatti se si pongono a zero le variabili rispetto alle quali il sistema non è in forma canonica ("variabili non basiche") da ciascuna riga è possibile ricavare immediatamente il valore di una delle variabili rispetto alle quali il sistema è in forma canonica ("variabili basiche").

Quindi in ogni equazione è presente una variabile basica (quella corrispondente alla colonna nella quale c'è il termine 1 della forma canonica).

Soluzioni basiche ammissibili e sistemi in forma canonica

Il sistema di equazioni relativo ai vincoli del dominio rappresentato nella figura precedente, con 2 vincoli di ≤ e 4 variabili (x_1,x_2, y_1, y_2), è in forma canonica rispetto alle variabili *slack* y_1 ed y_2.

$$a_{11}x_1 + a_{12}x_2 + y_1 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + y_2 = b_2$$

Pertanto annullando x_1 e x_2, variabili rispetto alle quali il sistema non è in forma canonica, dalla prima equazione si ricava immediatamente $y_1 = b_1$, dalla seconda $y_2 = b_2$. La soluzione basica ammissibile determinata è quindi $x_1 = x_2 = 0$, $y_1 = b_1$, $y_2 = b_2$ (vertice O in figura).

Soluzioni basiche ammissibili e sistemi in forma canonica

Si è visto che per determinare una s.b.a. si deve mettere il sistema in forma canonica.

Si deve costruire quindi un algoritmo che generi sistemi in forma canonica.

La procedura algebrica dell'algoritmo, come si è già detto e come si vedrà in dettaglio nel seguito, a partire da una prima forma canonica, e quindi da una prima soluzione basica ammissibile, genera altre forme canoniche e quindi altre soluzioni basiche ammissibili. Ogni soluzione basica ammissibile viene sottoposta a un test di ottimalità, per decidere se si è raggiunta la soluzione ottima oppure se è necessario fare un'altra iterazione per determinare una s.b.a. migliore.

Nel seguito si svolge un esercizio numerico per illustrare i passi dell'algoritmo.

Test di ottimalità sulla prima soluzione basica ammissibile

Bisogna effettuare il test di ottimalità per stabilire se è possibile trovare una soluzione migliore, associata ad un altro vertice del dominio.

A tale scopo si analizzano **le derivate direzionali della funzione z sugli spigoli disponibili nell'origine degli assi,** cioè gli spigoli corrispondenti agli assi coordinati. Il problema in esame è a massimizzare e quindi ci interessano i valori di derivata direzionale positivi, cioè i coefficienti positivi della z.

Sugli assi coordinati **le derivate direzionali di z nel punto O coincidono con le derivate parziali,** rappresentate dai coefficienti della funzione stessa, e sono quindi già disponibili.

Test di ottimalità sulla prima soluzione basica ammissibile

In questo caso si ha che:

$c_1 = 11$, derivata parziale della funzione z rispetto ad x_1,
è la derivata direzionale della funzione stessa lungo l'asse x_1.

$c_2 = 10$, derivata parziale della funzione z rispetto ad x_2,
è la derivata direzionale della funzione stessa lungo l'asse x_2.

In questo caso dunque i coefficienti della f.o. sono entrambi positivi e quindi su entrambi gli spigoli è possibile effettuare uno spostamento con miglioramento, cioè uno spostamento che ci porta in un vertice al quale corrisponde una soluzione basica ammissibile migliore.

Dunque la soluzione in esame non è la soluzione ottima.

Determinazione della variabile entrante

In generale si sceglie la variabile a cui corrisponde il c_j migliore, (cioè più elevato in modulo):

- servono i coefficienti $c_j > 0$ per un problema di massimizzazione

  e i coefficienti $c_j < 0$, per un problema di minimizzazione

In genere si sceglie la variabile

$x_s : c_s = \max_j \{c_j, c_j > 0, \forall j \text{ non basico}\}$

per un problema a massimizzare;

e la variabile

$x_s: c_s = \min_j \{c_j, c_j < 0, \forall j \text{ non basico}\}$

per un problema a minimizzare.

Variabile uscente

Per determinare la variabile uscente bisogna ricordare che la struttura di una s.b.a. è bloccata (cioè si hanno n-m variabili nulle ed m variabili non negative) e quindi, se una variabile che era nulla diventa > 0 (si dice che entra nella soluzione) ci deve essere una variabile, che era > 0, che si annulla (si dice che esce dalla soluzione).

Riepilogando:

• una variabile nulla (non basica) della precedente s.b.a. diventa >0 (basica) e si definisce variabile entrante

• una variabile > 0 (basica) della precedente s.b.a. diventa nulla (non basica) e si definisce variabile uscente

Dunque la nuova soluzione basica ammissibile differisce dalla precedente per una sola variabile. E questo garantisce che ci spostiamo in un vertice adiacente del dominio.

Determinazione della variabile uscente

Se x_1 entra nella soluzione il suo valore diventa > 0.

Vediamo cosa succede all'aumentare del valore di x_1, nelle singole equazioni, per le variabili che ora sono > 0, cioè $y_1 = 20, y_2 = 6.5, y_3 = 36, y_4 = 16$

(1)	$-5x_1 + 4x_2 + y_1 = 20$	Dalla 1° equazione ricaviamo $\boldsymbol{y_1 = 20 + 5x_1 - 4x_2}$
(2)	$x_2 + y_2 = 6.5$	Dalla 2° equazione ricaviamo $\boldsymbol{y_2 = 6.5 - x_2}$
(3)	$3x_1 + 4x_2 + y_3 = 36$	Dalla 3° equazione ricaviamo $\boldsymbol{y_3 = 36 - 3x_1 - 4x_2}$
(4)	$2x_1 x_2 + y_4 = 16$	Dalla 4° equazione ricaviamo $\boldsymbol{y_4 = 16 - 2x_1 - x_2}$

All'aumentare di x_1 (x_2 resta 0 perché ci stiamo spostando sull'asse x_1)
i valori delle y_i cambiano, come si illustra nella slide successiva.

Nella 1a equazione x_1 può crescere indefinitamente senza far diminuire y_1

Nella 2a equazione x_1 è assente e quindi y_2 non cambia qualunque sia il valore assunto da x_1.

Nella 3a equazione se $x_1 = 36/3 = 12$ allora $y_3 = 0$

Nella 4a equazione se $x_1 = 16/2 = 8$ allora $y_4 = 0$

Determinazione della variabile uscente

Dunque da ogni equazione ricaviamo un valore limite per x_1, in corrispondenza del quale in ogni equazione c'è una variabile che si annulla e oltre il quale quella stessa variabile diventa negativa. In questo caso leggiamo che ci sono 2 valori limite che la variabile x_1 può assumere:

- Se x_1 arriva ad assumere il valore $36/3 = 12$, y_3 diventa 0 nella 3a equazione.

  Se x_1 andasse oltre il valore 12 y_3 diventerebbe negativa.

- Se x_1 arriva ad assumere il valore $16/2 = 8$, y_4 diventa 0 nella 4a equazione.

  Se x_1 andasse oltre il valore 8 y_4 diventerebbe negativa

Poiché nessuna delle y_i può diventare negativa, la variabile x_1 può arrivare ad un valore che garantisca questa condizione, quindi è necessario scegliere il minimo dei valori limite determinati. In questo modo una delle variabili che prima erano > 0 si annulla (è lei la variabile uscente) e le altre restano > 0. Cioè tutte restano non negative e la nuova soluzione è ammissibile.

Bisogna quindi scegliere il minimo dei rapporti (36/3 e 16/2) prima determinati.

In questo modo la variabile uscente è y_4.

$\text{Min}_i\ b_i/a_{i1}\ (a_{i1}>0) = \text{Min}\ \{36/3, 16/2\} = 16/2 \Rightarrow$ Variabile uscente y_4

Determinazione della nuova s.b.a.

Ricordiamo che la prima soluzione basica ammissibile è la seguente:

$$x_1 = 0,\ x_2 = 0,\ y_1 = 20,\ y_2 = 6.5,\ y_3 = 36,\ y_4 = 16$$

e che essa è stata determinata a partire da un sistema in forma canonica rispetto alle variabili y_1, y_2, y_3, y_4 , che sono le variabili > 0 nella soluzione stessa

Bisogna quindi trovare una soluzione migliore di questa, nella quale la variabile x_1 che è nulla diventi >0 e la variabile y_4 che è >0 diventi nulla. Cioè bisogna trovare una soluzione basica ammissibile nella quale:

$$x_1 > 0,\ x_2 = 0\ ,\ y_1 > 0,\ y_2 > 0,\ y_3 > 0,\ y_4 = 0$$

che sarà la soluzione corrispondente al vertice E.

Determinazione della nuova s.b.a.

Ricordiamo che una soluzione basica ammissibile può essere facilmente trovata a partire da un sistema in forma canonica. **Quindi dobbiamo costruire un sistema in forma canonica che ci permetta di trovare questa nuova soluzione.**

A tale scopo effettuiamo **una trasformazione in forma equivalente del nostro sistema di equazioni (che è in forma canonica) in un'altra forma canonica**, nella quale le variabili della forma canonica siano proprio le variabili che vogliamo siano >0 nella nuova soluzione.

Dobbiamo quindi costruire **un sistema in forma canonica nel quale le colonne della matrice identità corrispondano alle colonne delle variabili che vogliamo siano >0.**

Nella slide successiva si riportano le operazioni algebriche che realizzano questa trasformazione, definite **operazioni di *pivoting*** perché si svolgono attorno ad un elemento perno (pivot). Si sviluppa poi l'esercizio numerico.

Trasformazione del sistema (*pivoting*)

- Se r è la riga pivot, ed s è la colonna pivot, bisogna dividere tutti gli elementi della riga *pivot* r ($\boldsymbol{e}_r$) per a_{rs}, ottenendo una nuova configurazione $\boldsymbol{e}_r' = \boldsymbol{e}_r / a_{rs}$, che presenterà quindi sicuramente un 1 nell'elemento perno di posto rs ($a_{rs}' = 1$);

- Bisogna poi trasformare ciascuna riga i del sistema in modo da indurre la presenza di uno 0 nell'elemento di posto is (deve essere $a_{is} = 0$, per ogni $i = 1, \ldots, m$);
- L'operazione da compiere è la seguente:
- a ciascuna riga i si sottrae la riga *pivot* trasformata ($\boldsymbol{e}_r'$) moltiplicata per l'elemento della riga i che è nella colonna *pivot* s (a_{is}) : $\boldsymbol{e}_i' = \boldsymbol{e}_i - a_{is}\, \boldsymbol{e}_r'$.
- ***La riga della funzione obiettivo partecipa alle operazioni di pivoting. Si può dimostrare infatti che i coefficienti della funzione obiettivo trasformata sono le nuove derivate direzionali nel vertice corrispondente alla soluzione che si sta determinando.***

Colonna pivot, riga pivot ed elemento perno

La determinazione della variabile entrante e della variabile uscente consente di individuare una colonna pivot, una riga pivot e l'elemento perno corrispondente all'incrocio delle due.

	x_1	x_2	y_1	y_2	y_3	y_4	$-z$	b
y_1	-5	4	1	0	0	0	0	20
y_2	0	1	0	1	0	0	0	6.5
y_3	3	4	0	0	1	0	0	36
y_4	2	1	0	0	0	1	0	16
$-z$	11	10	0	0	0	0	1	0

$x_1 = 0, x_2 = 0 \Rightarrow y_1 = 20, y_2 = 6.5, y_3 = 36, y_4 = 16, z = 0$

Test di ottimalità: $c_{jnb} = \{11, 10\} \Rightarrow$ Variabile entrante x_1

$\text{Min}_i\ b_i/a_{i1}\ (a_{i1}>0) = \text{Min}\ \{36/3, 16/2\} = 16/2 \Rightarrow$ Variabile uscente y_4

Test di ottimalità sulla nuova soluzione basica ammissibile

La nuova s.b.a. è la seguente:

$y_3 = 0\ , y_4 = 0, y_1 = 28.8, y_2 = 1.7, x_1 = 5.6, x_2 = 4.8, z = 109.6$

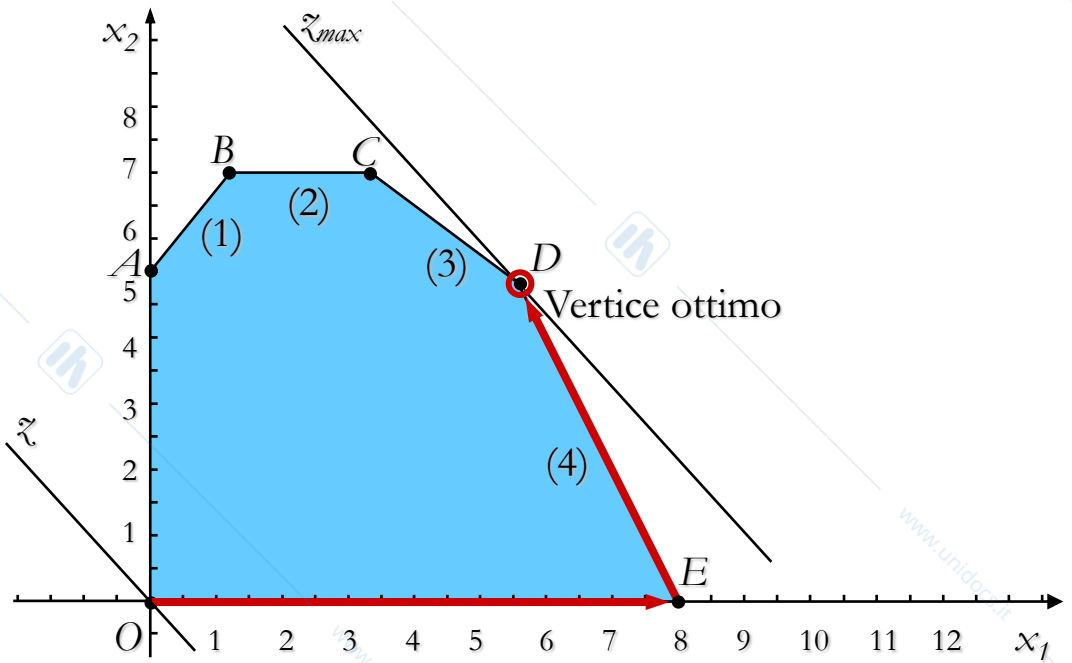
Bisogna sottoporre la nuova s.b.a. al test di ottimalità.
Il test di ottimalità è soddisfatto se si verifica che:

$\forall\ j$ corrispondente ad una variabile x_j non basica:

- $c_j \leq 0$ per un problema di massimizzazione
- $c_j \geq 0$, per un problema di minimizzazione

Test di ottimalità: $c'_{jnb} = \{ -1.8, -2.8\}$ $\Rightarrow$ **Soluzione ottima**

Percorso dell'algoritmo del Simplesso dal vertice O alla soluzione ottima in D



	x_1	x_2	y_1	y_2	y_3	y_4	z	
1ª s.b.a.	0	0	20	6.5	36	16	0	Vertice O
2ª s.b.a.	8	0	60	6.5	12	0	88	Vertice E
3ª s.b.a.	5.6	4.8	28.8	1.7	0	0	109.6	Vertice D

Soluzioni basiche ammissibili degeneri

Soluzioni basiche ammissibili degeneri

Una s.b.a. di un problema $m \times n$ (m vincoli ed n variabili) si definisce degenere quando una (o più) delle m variabili basiche assume valore nullo. Le variabili basiche nulle si definiscono variabili degeneri.

La soluzione presenta quindi $m' < m$ variabili strettamente positive ed $n - m'$ ($> n - m$) variabili nulle.

In un vertice si ha una s.b.a. degenere quando ci sono uno o più vincoli ridondanti passanti per quel vertice.

Se l'algoritmo passa per quel vertice il minimo dei rapporti b_i/a_{is} si ha in corrispondenza di più righe. Quindi, se entra in base la variabile x_s, più variabili basiche si annulleranno contemporaneamente e **pertanto più variabili basiche diventano candidate ad essere variabile uscente**.

Poichè una sola variabile basica deve uscire dalla base, le altre resteranno nella base e assumeranno valore nullo, come mostrato nell'esercizio numerico successivo.

Soluzioni basiche ammissibili degeneri

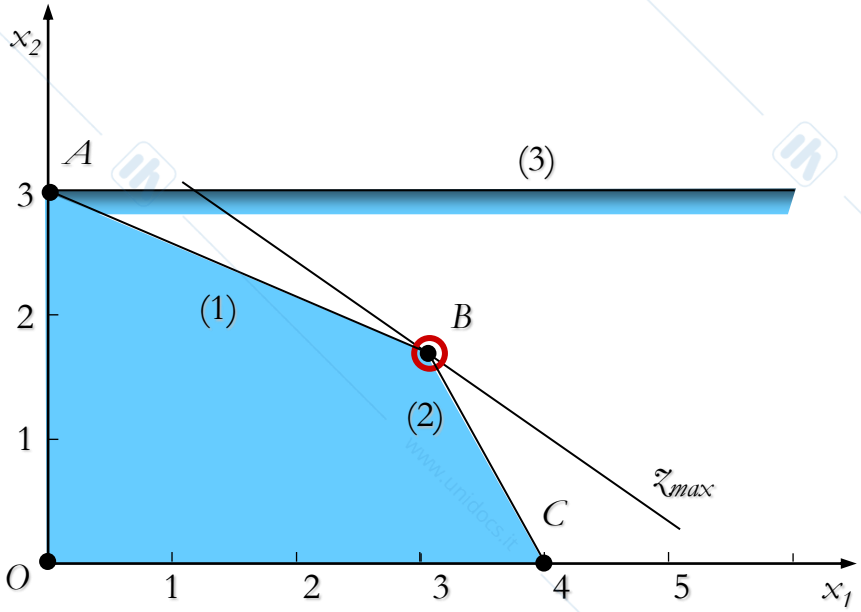
In uno step successivo l'operazione di pivoting potrebbe portare ad un cambiamento di s.b.a. non collegato ad uno spostamento da un vertice ad un altro. In generale in uno step ancora successivo l'algoritmo genera una nuova soluzione basica ammissibile non degenere e l'algoritmo prosegue quindi nelle sue iterazioni.

In casi molto particolari c'è la possibilità che dopo un certo numero di s.b.a. "degeneri", tutte corrispondenti allo stesso vertice, si passi una seconda volta per una soluzione degenere già generata in precedenza.

Si incorre in questo caso nel cosiddetto fenomeno della "circolazione", che consiste nella determinazione ciclica della stessa soluzione basica ammissibile degenere, senza convergere alla soluzione ottima. Per superare questo problema si possono utilizzare diverse tecniche "anticircolazione".

Esempio numerico:
Soluzione basica ammissibile degenere

$$\begin{array}{lllllll} \text{Max} & z = & 4x_1 & + & 5x_2 & & \\ \text{s.a} & (1) & 3x_1 & + & 7x_2 & \leq & 21 \\ & (2) & 8x_1 & + & 4x_2 & \leq & 32 \\ & (3) & & & x_2 & \leq & 3 \end{array}$$



$$\begin{array}{llllll} (1) & 3x_1 + 7x_2 + y_1 & & & = & 21 \\ (2) & 8x_1 + 4x_2 & + y_2 & & = & 32 \\ (3) & & x_2 & + y_3 & = & 3 \end{array}$$

Tabella 1. 1ª soluzione basica ammissibile

	x_1	x_2	y_1	y_2	y_3	$-z$	$\boldsymbol{b}$
y_1	3	7	1	0	0	0	21
y_2	8	4	0	1	0	0	32
y_3	0	1	0	0	1	0	3
$-z$	4	5	0	0	0	1	0

$x_1 = 0, x_2 = 0 \Rightarrow y_1 = 21, y_2 = 32, y_3 = 3, z = 0$ (Vertice O)

Test di ottimalità: $c_{jnb} = \{4, 5\} \Rightarrow$ Variabile entrante x_2

$\text{Min}_i\ b_i/a_{i2}\ (a_{i2}>0) = \text{Min}\ \{21/7, 32/4, 3/1\} = 3/1$

Si noti che il minimo dei rapporti si ha in corrispondenza di 2 righe, quindi potrebbe uscire la variabile y_1 o la variabile y_3 . Facciamo uscire la variabile uscente y_3. La y_1 si annulla comunque, ma resta basica, e quindi è la variabile degenere della prossima soluzione.

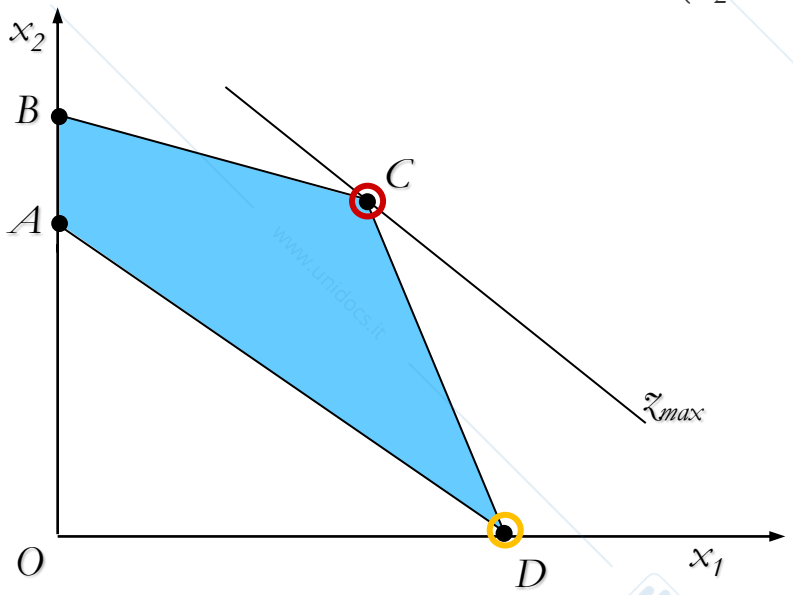
Soluzione basica ammissibile degenere su un asse coordinato

È possibile che una s.b.a. degenere non sia associata alla presenza di un vincolo ridondante in senso stretto.

Il vertice D in figura corrisponde a una soluzione basica ammissibile degenere.
Infatti si ha:

$$x_1>0,\ x_2=0,\ y_1>0,\ y_2=0,\ y_3=0.$$

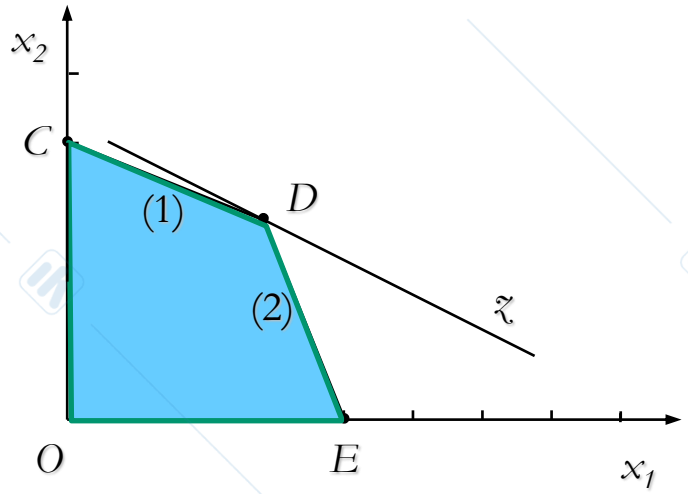
In questo caso il vincolo ridondante è l'asse delle ascisse ($x_2 \geq 0$) .



Algoritmo del Simplesso con vincoli del tipo = o ≥

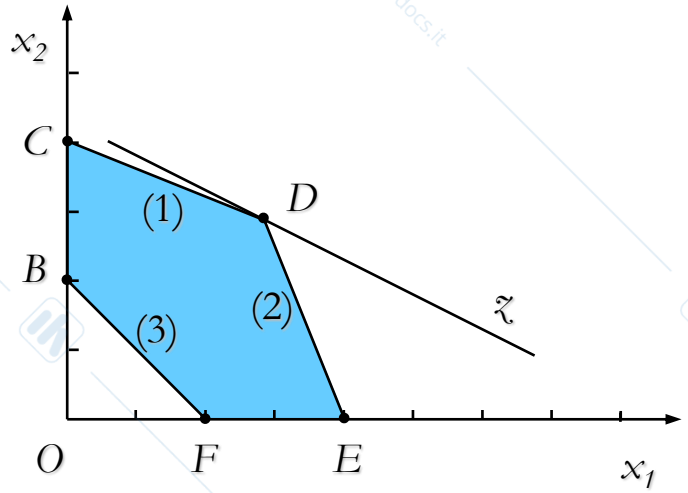
Modello con vincoli del tipo ≤

$$
\begin{array}{lrcrcl}
\text{Max} & z = & x_1 & + 2x_2 & & \\
\text{s.a} & & 2x_1 & + 5x_2 & \leq & 100 \\
& & 5x_1 & + 2x_2 & \leq & 100 \\
& & & & & \\
& & x_1, x_2 & & \geq & 0
\end{array}
$$



Modello con vincoli del tipo ≥

$$
\begin{array}{lrcrcl}
\text{Max} & z = & x_1 & + 2x_2 & & \\
\text{s.a} & & 2x_1 & + 5x_2 & \leq & 100 \\
 & & 5x_1 & + 2x_2 & \leq & 100 \\
 & & x_1 & + x_2 & \geq & 10 \\
 & & & x_1, x_2 & \geq & 0
\end{array}
$$

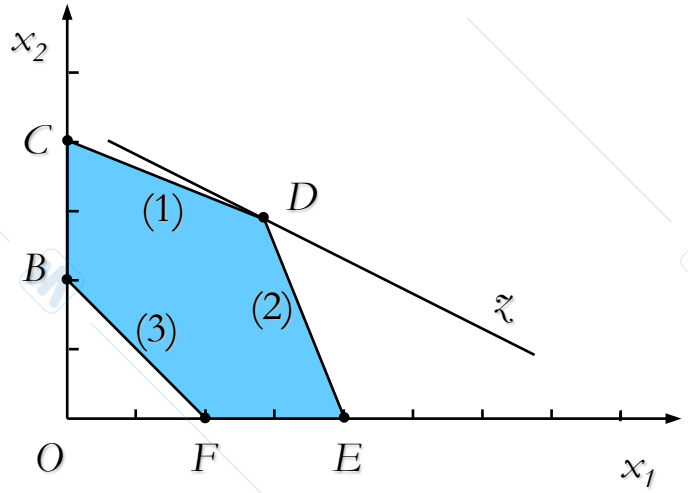


Se il modello presenta almeno un vincolo del tipo ≥,

- l'origine degli assi non appartiene più necessariamente al dominio ammissibile,
- la matrice del sistema di equazioni risultante dall'aggiunta delle variabili slack non contiene una matrice identità e quindi il sistema non è in forma canonica,
- non è possibile pertanto individuare immediatamente una prima soluzione basica ammissibile per l'innesco dell'algoritmo del simplesso
- **ciascun vincolo del tipo ≥ viene trasformato in un vincolo di uguaglianza inserendo una variabile *slack* con segno negativo.**

Modello con vincoli del tipo ≥

$$
\begin{aligned}
\text{Max} \quad z &= x_1 + 2x_2 \\
\text{s.a} \quad 2x_1 &+ 5x_2 \le 100 \\
5x_1 &+ 2x_2 \le 100 \\
x_1 &+ x_2 \ge 10 \\
x_1, x_2 &\ge 0
\end{aligned}
$$



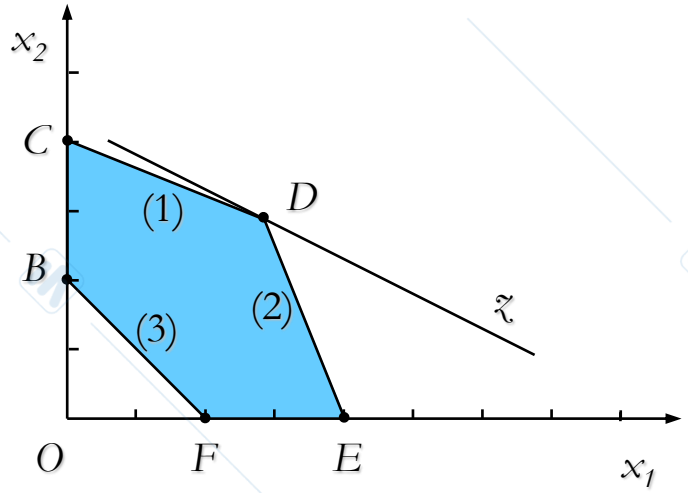
$$
\begin{aligned}
2x_1 + 5x_2 + y_1 &= 100 \\
5x_1 + 2x_2 + y_2 &= 100 \\
x_1 + x_2 - y_3 &= 10
\end{aligned}
$$

Variabile artificiale in un vincolo di ≥

In questi casi, in ciascun vincolo *i* del tipo ≥, oltre alla variabile *slack* con segno negativo, si aggiunge anche una variabile "artificiale" h_i con coefficiente unitario e segno positivo, che ripristina una colonna della matrice identità.

Modello con vincoli del tipo ≥

$$
\begin{array}{lrcrcrcl}
\text{Max} & z = & & x_1 & + 2x_2 & & & \\
\text{s.a} & & 2x_1 & + 5x_2 & \le & 100 & \\
& & 5x_1 & + 2x_2 & \le & 100 & \\
& & x_1 & + x_2 & \ge & 10 & \\
& & & x_1, x_2 & \ge & 0 &
\end{array}
$$



$$
\begin{array}{rcrcrcrcl}
2x_1 & + 5x_2 & + y_1 & & & & & = & 100 \\
5x_1 & + 2x_2 & & + y_2 & & & & = & 100 \\
x_1 & + x_2 & & & - y_3 & + h_1 & & = & 10
\end{array}
$$

Variabile artificiale in un vincolo di =

I vincoli del tipo = non hanno bisogno di trasformazione e quindi non richiedono variabili *slack*.

Pertanto se il modello presenta almeno un vincolo del tipo =, la matrice del sistema di equazioni risultante dalla trasformazione non contiene una matrice identità e quindi il sistema non è in forma canonica. Non è possibile pertanto individuare immediatamente una prima soluzione basica ammissibile per l'innesco dell'algoritmo del simplesso.

In ciascun vincolo *i* del tipo = si aggiunge quindi solo una variabile "artificiale" h_i, , con coefficiente unitario e segno positivo, che genera una colonna della matrice identità.

Prima soluzione basica ammissibile

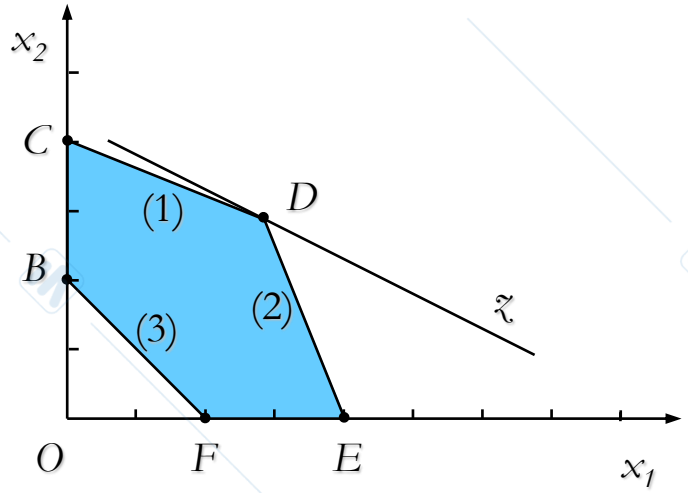
L'aggiunta delle variabili artificiali quindi, in presenza di vincoli di = o ≥, **consente di generare** comunque, all'interno della matrice dei coefficienti del sistema di equazioni, **una matrice identità costituita dalle colonne delle variabili *slack* con segno positivo e/o delle variabili artificiali.**

L'aggiunta di una o più variabili artificiali consente di determinare la s.b.a. necessaria per innescare l'algoritmo del simplesso.

Le variabili originarie e le variabili slack con segno negativo sono variabili non basiche, mentre le variabili slack con segno positivo e le variabili artificiali sono variabili basiche.

Modello con vincoli del tipo ≥

$$
\begin{array}{lrcrcrcl}
\text{Max} & z = & & x_1 & + 2x_2 & & & \\
\text{s.a} & & 2x_1 & + 5x_2 & \leq & 100 & \\
& & 5x_1 & + 2x_2 & \leq & 100 & \\
& & x_1 & + x_2 & \geq & 10 & \\
& & & x_1, x_2 & \geq & 0 &
\end{array}
$$



$$
\begin{array}{rcrcrcrcl}
2x_1 & + 5x_2 & + y_1 & & & & & = & 100 \\
5x_1 & + 2x_2 & & + y_2 & & & & = & 100 \\
x_1 & + x_2 & & & - y_3 & + h_1 & & = & 10
\end{array}
$$

Prima soluzione basica ammissibile

$x_1 = x_2 = y_3 = 0 \Rightarrow y_1 = 100, y_2 = 100, h_1 = 10$

A quale vertice corrisponde?

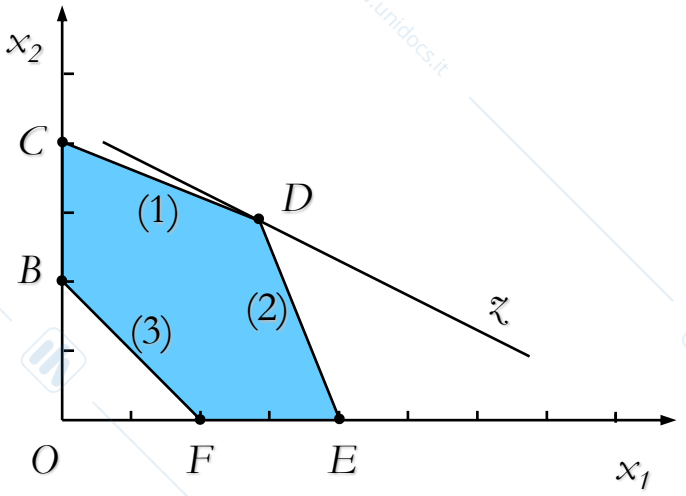
Dominio di ammissibilità esteso

Si noti che l'aggiunta delle variabili artificiali ha ampliato lo spazio delle variabili e ha definito un nuovo dominio di ammissibilità. La soluzione basica ammissibile individuata corrisponde ad un vertice di questo dominio di ammissibilità esteso.

Nelle slide seguenti si propone una rappresentazione grafica del dominio di ammissibilità esteso, nel caso semplice di un problema con 2 variabili e 3 vincoli, con una sola variabile artificiale.

Rappresentazione grafica del dominio esteso

$$
\begin{array}{lrcrcl}
\text{Max} & z = & x_1 & + 2x_2 & & \\
\text{s.a} & & 2x_1 & + 5x_2 & \leq & 100 \\
& & 5x_1 & + 2x_2 & \leq & 100 \\
& & x_1 & + x_2 & \geq & 10 \\
& & & x_1, x_2 & \geq & 0
\end{array}
$$

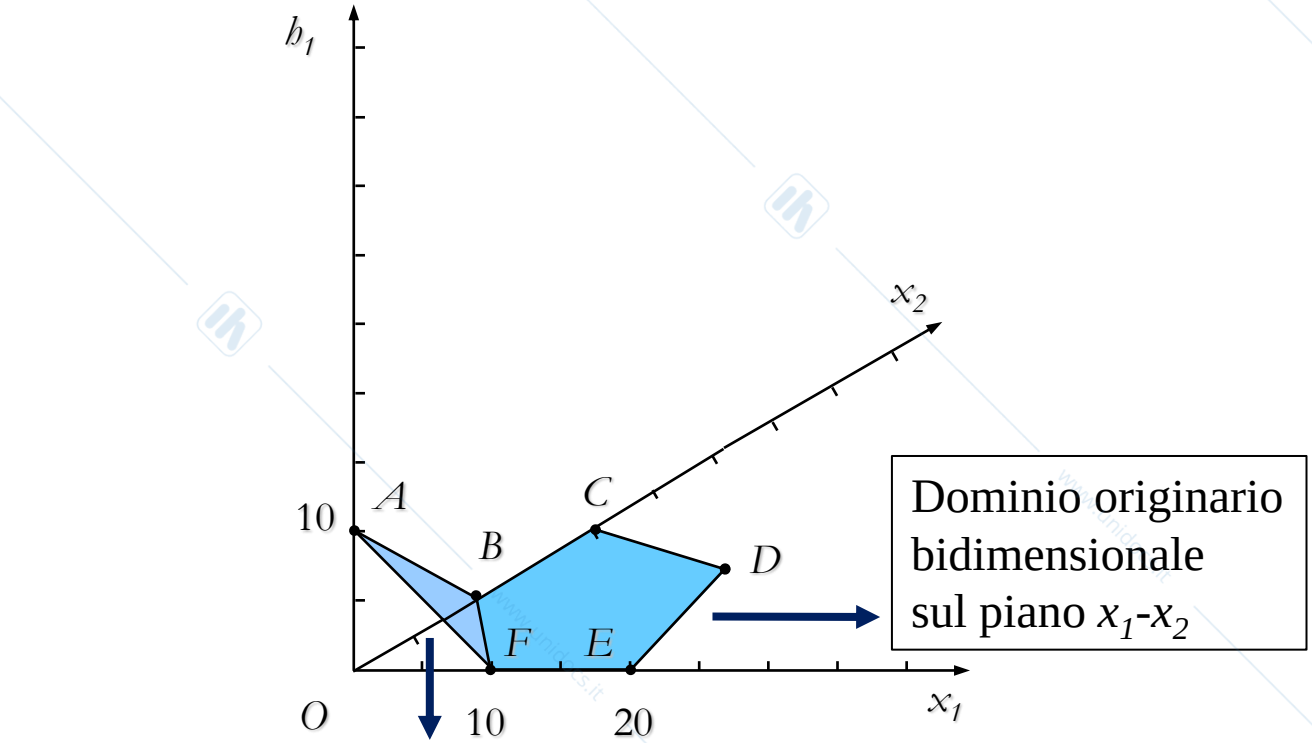


$$
\begin{array}{rcrcrcl}
2x_1 & + 5x_2 & + y_1 & & & = & 100 \\
5x_1 & + 2x_2 & & + y_2 & & = & 100 \\
x_1 & + x_2 & & & - y_3 + h_1 & = & 10
\end{array}
$$

$$
\begin{array}{rcrcrcl}
2x_1 & + 5x_2 & & & & \leq & 100 \\
5x_1 & + 2x_2 & & & & \leq & 100 \\
x_1 & + x_2 & & & + h_1 & \geq & 10
\end{array}
$$

Per realizzare la rappresentazione grafica del dominio esteso si può riscrivere il sistema di equazioni (con la variabile artificiale) come sistema di disequazioni, eliminando le variabili *slack*. Si ottiene in questo modo un sistema di 3 disequazioni in 3 variabili che genera il dominio rappresentato nella slide seguenti.

Dominio esteso

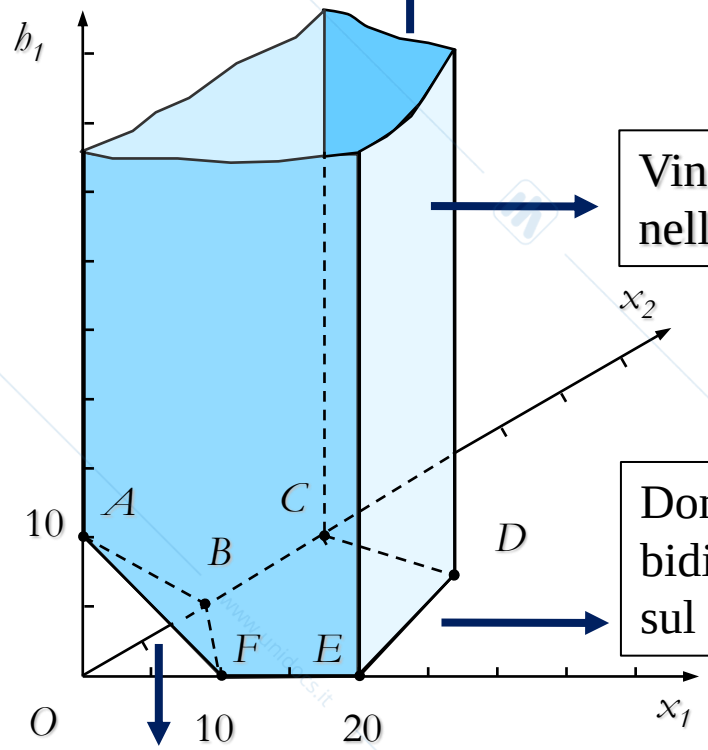


Dominio originario bidimensionale sul piano x_1-x_2

Vincolo $x_1 + x_2 + h_1 \geq 10$

Dominio esteso

Vincolo $2x_1 + 5x_2 \leq 100$ nello spazio x_1, x_2, h_1

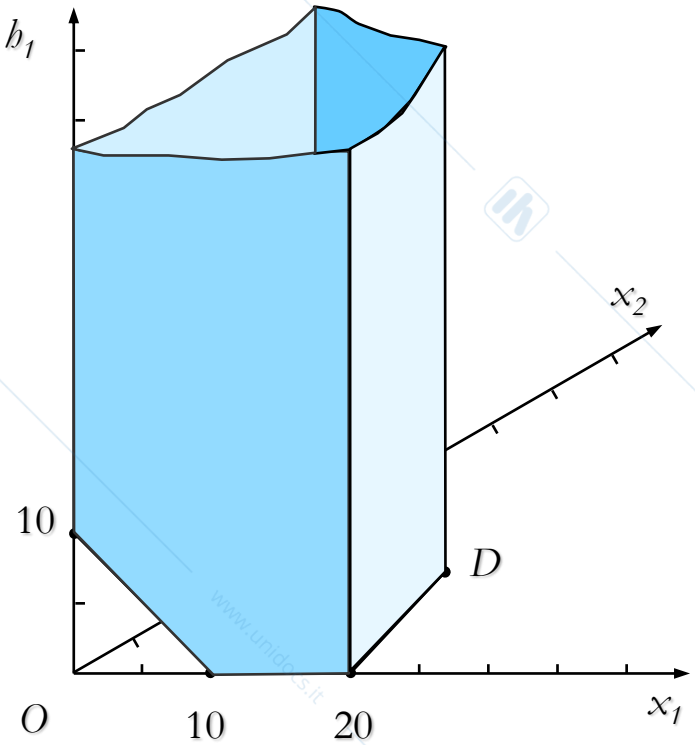


Vincolo $5x_1 + 2x_2 \leq 100$ nello spazio x_1, x_2, h_1

Dominio originario bidimensionale sul piano x_1, x_2

Vincolo $x_1 + x_2 + h_1 \geq 10$

Dominio esteso



Il metodo del Big M

Il metodo del *Big M* è basato sulla modifica della funzione obiettivo *z* del problema originario. Si aggiungono ad essa le variabili artificiali h_j con segno opportuno, negativo se la funzione obiettivo *z* è a massimizzare, positivo se z è a minimizzare.

In questo modo si favorisce l'annullamento delle variabili artificiali e quindi la loro uscita dalla base. Per accelerare questo processo le variabili h_j vengono moltiplicate per un coefficiente di modulo elevato, che si indica con *M*, da cui il nome di metodo del *Big M*.

L'algoritmo segue un percorso che parte da un vertice del dominio esteso e arriva a un vertice del dominio originario, proseguendo poi fino alla soluzione ottima del dominio originario.

Nel seguito si riportano due esempi numerici.

Algoritmo del Simplesso in 2 fasi

Il metodo del Simplesso in due fasi opera diversamente dal metodo del *Big M*.

Non trasforma la funzione obiettivo originaria, ma costruisce una nuova funzione obiettivo ausiliaria *w*, definita come somma delle variabili artificiali:

$w = \sum_{i=1,k} h_i$, se k è il numero di variabili artificiali presenti nel sistema.

Le variabili artificiali sono non negative, e quindi anche la loro somma sarà non negativa.

Il minimo della funzione *w* è dunque pari a 0 e corrisponde all'annullamento di tutte le variabili artificiali.

Quindi minimizzare questa funzione obiettivo *w* significa portare a 0 le variabili artificiali.

L'algoritmo opera in due fasi.

- **Nella prima fase si minimizza la nuova funzione obiettivo *w* aggiunta al sistema di equazioni.** Partendo dalla soluzione basica ammissibile costruita con le variabili artificiali si cerca di determinare una soluzione basica ammissibile nella quale le variabili artificiali siano pari a 0, cioè una soluzione basica ammissibile appartenente anche al dominio originario.
- **Nella seconda fase,** a partire da questa s.b.a., restando nel dominio originario, **e con la funzione obiettivo originaria, si determina la soluzione ottima del problema**.

Algoritmo del Simplesso in 2 fasi

La prima fase termina quando il test di ottimalità è soddisfatto ($d_j' \geq 0$, $\forall$ *j* non basico).

Si analizza a questo punto il valore di w^* (si ricordi che $w^* = \sum_{i=1,m} h_i^*$).

Si possono verificare 2 casi: **1. $w^* > 0$,** **2. $w^* = 0$.**

- **1. Se $w^* > 0$**, vuol dire che almeno una variabile artificiale è ancora > 0.
- Quindi non è stato possibile eliminare tutte le variabili artificiali dalla base. Non esiste alcuna soluzione del problema ampliato con $h_i = 0$, $\forall i \in \{1, \ldots, m\}$, dunque il problema originario è inconsistente e l'algoritmo termina.

- **2. Se $w^* = 0$**, si è trovata una soluzione con $\boldsymbol{h_i = 0, \forall\, i \in \{1, \ldots, m\}}$. Tutte le variabili artificiali sono state annullate e quindi si è ottenuta una soluzione basica ammissibile che appartiene al dominio originario. Si possono verificare due sottocasi:
- **2.a. Tutte le variabili artificiali (nulle) sono non basiche**. Allora si elimina la funzione obiettivo artificiale *w* e le colonne corrispondenti alle variabili artificiali e si ottiene una tabella del problema originario, già in forma canonica rispetto ad un insieme di variabili basiche non artificiali. Si passa quindi alla II fase dell'algoritmo nella quale, a partire dalla soluzione basica ammissibile determinata, si opera nel dominio originario con la funzione obiettivo originaria, per determinare la soluzione ottima.
- **2.b. <u>Non</u> tutte le variabili artificiali (nulle) sono non basiche**. C'è almeno una variabile artificiale $h_k = 0$, ma in base, in una riga *i*-esima: la soluzione quindi è degenere rispetto a questa variabile artificiale. **La variabile artificiale potrebbe rientrare in base nelle successive iterazioni, cambiando valore e diventando positiva.** Bisogna quindi operare per evitare problemi nella 2° fase dell'algoritmo, come descritto nella slide successiva.

Algoritmo del Simplesso in 2 fasi

Nel caso 2b bisogna evitare che la variabile artificiale possa rientrare in base nelle successive iterazioni, cambiando valore e diventando positiva. A tale scopo è necessario operare una trasformazione del sistema volta a configurare una s.b.a. che non contenga variabili artificiali basiche. Si possono verificare due situazioni.

- Se nella riga *i*, in cui la variabile h_k è basica degenere, esiste un elemento $a_{ij} \neq 0$ in corrispondenza di una variabile x_j non basica si effettua una operazione di *pivoting* in corrispondenza di questo elemento di posto *ij*.

Poiché $b_i = 0$, l'operazione è possibile anche se $a_{ij} < 0$, e non determina variazioni della funzione obiettivo w, perché x_j entra in base con valore 0. In questo modo si fa uscire h_k dalla base.

Se necessario si ripete l'operazione per tutte le variabili h_k in base a valore 0 e ci si riporta in questo modo al caso (2a) già descritto, nel quale tutte le variabili artificiali sono nulle e non basiche.

- Se invece tutti gli elementi a_{i1}, …, a_{in} della riga *i*-esima sono nulli si eliminano le colonne delle variabili artificiali non basiche e si ottiene una tabella equivalente a quella originale, ma con una riga nulla. Questo implica che la matrice $\boldsymbol{A}$ non è di rango m, e quindi la riga i-esima (nella quale è in base h_k), linearmente dipendente dalle altre, può essere eliminata insieme alla colonna di h_k, riportandosi in questo modo al caso (2a).

Nel seguito si riportano un esempio numerico, relativo al caso 2a.

Struttura algebrica del sistema di equazioni

Nell'algoritmo del Simplesso Standard fin qui descritto, il sistema iniziale $\boldsymbol{Ax} = \boldsymbol{b}$ di dimensioni mxn (m equazioni in n variabili), è comprensivo delle variabili originarie e delle variabili ausiliarie (*slack* e/o *artificiali)*.

Per le considerazioni svolte sul significato e sul ruolo delle variabili *slack* e delle variabili artificiali, il sistema si presenta inizialmente in forma canonica rispetto alle variabili ausiliarie (*slack* e/o *artificiali)* e nel corso dell'algoritmo viene trasformato, con la procedura algebrica descritta, in forme canoniche equivalenti, al fine di generare soluzioni basiche ammissibili sempre migliori, fino alla soluzione basica ammissibile ottima.

Nelle slide seguenti si descrive la struttura algebrica dell'algoritmo del simplesso standard.

Algebra del Simplesso

Matrice di base
di una soluzione basica ammissibile

Per ogni soluzione basica ammissibile del sistema di equazioni

si definisce *matrice di base*

la matrice *B* costituita dalle colonne della tabella iniziale

relative alle variabili della forma canonica (basiche) di quella soluzione

Esempi di matrice di base

Tabella iniziale (Vertice origine)

	X1	x2	y1	y2	y3	−z	B
y1	3	6	1	0	0	0	57
y2	3	15	0	1	0	0	135
y3	12	3	0	0	1	0	84
−z	90	60	0	0	0	1	0

Vertice D

	x1	x2	y1	y2	y3	−z	b
y1	0	21/4	1	0	-1/4	0	36
y2	0	57/4	0	1	-1/4	0	114
x1	1	¼	0	0	1/12	0	7
−z	0	37.5	0	0	-7.5	1	-630

Variabili della forma canonica (basiche) di questa soluzione: *y1, y2, x1*

Colonne della tabella iniziale relative a queste variabili

y1	y2	x1
1	0	3
0	1	3
0	0	12

Matrice di base B della soluzione del vertice D

1	0	3
0	1	3
0	0	12

Esempi di matrice di base

Tabella iniziale (Vertice origine)

	X1	x2	y1	y2	y3	−z	B
y1	3	6	1	0	0	0	57
y2	3	15	0	1	0	0	135
y3	12	3	0	0	1	0	84
−z	90	60	0	0	0	1	0

Vertice C

	x1	x2	y1	y2	y3	−z	b
x2	0	1	0.190	0	-0.047	0	6.86
y2	0	0	-2.71	1	0.429	0	16.3
x1	1	0	-0.048	0	0.095	0	5.29
−z	0	0	-7.125	0	-5.737	1	-887.25

Variabili della forma canonica (basiche) di questa soluzione: *x2, y2, x1*

Colonne della tabella iniziale relative a queste variabili

x2	y2	x1
6	0	3
15	1	3
3	0	12

Matrice di base B della soluzione del vertice C

6	0	3
15	1	3
3	0	12

Operazioni con le tabelle del Simplesso

Matrice di base B della soluzione del vertice D

1	0	3
0	1	3
0	0	12

Inversa della matrice di base B della soluzione del vertice D

1	0	-1/4
0	1	-1/4
0	0	1/12

Pre-moltiplichiamo la matrice della tabella iniziale per l'inversa della matrice di base della sba del vertice D

1	0	-1/4
0	1	-1/4
0	0	1/12

×

X1	*x2*	*y1*	*y2*	*y3*	*b*
3	6	1	0	0	57
3	15	0	1	0	135
12	3	0	0	1	84

=

x1	*x2*	*y1*	*y2*	*y3*	*b*
0	21/4	1	0	-1/4	36
0	57/4	0	1	-1/4	114
1	¼	0	0	1/12	7

	x1	*X2*	*y1*	*y2*	*y3*	*b*
y1	0	21/4	1	0	-1/4	36
y2	0	57/4	0	1	-1/4	114
x1	1	¼	0	0	1/12	7

Operazioni con le tabelle del Simplesso

Matrice di base B della soluzione del vertice C

6	0	3
15	1	3
3	0	12

Inversa della matrice di base B della soluzione del vertice C

0.190	0	-0.047
-2.71	1	0.429
-0.048	0	0.095

Pre-moltiplichiamo la matrice della tabella iniziale per l'inversa della matrice di base della sba del vertice C

0.190	0	-0.047
-2.71	1	0.429
-0.048	0	0.095

×

x1	*x2*	*y1*	*y2*	*y3*	*b*
3	6	1	0	0	57
3	15	0	1	0	135
12	3	0	0	1	84

=

x1	*x2*	*y1*	*y2*	*y3*	*b*
0	1	0.190	0	-0.047	6.86
0	0	-2.71	1	0.429	16.3
1	0	-0.048	0	0.095	5.29

	x1	*x2*	*y1*	*y2*	*y3*	*b*
x2	0	1	0.190	0	-0.047	6.86
y2	0	0	-2.71	1	0.429	16.3
x1	1	0	-0.048	0	0.095	5.29

Dunque . . .

- La premoltiplicazione del sistema iniziale per l'inversa della matrice di base di una qualunque s.b.a. genera la tabella (la soluzione) di quella s.b.a.

- Poiché il sistema iniziale è in forma canonica l'inversa della matrice di base di una s.b.a. si trova nella tabella di quella s.b.a., nelle colonne della matrice identità della tabella iniziale.

Struttura algebrica del sistema di equazioni

Nell'algoritmo del Simplesso Standard fin qui descritto, il sistema iniziale $\boldsymbol{Ax} = \boldsymbol{b}$ di dimensioni mxn (m equazioni in n variabili), è comprensivo delle variabili originarie e delle variabili ausiliarie (*slack* e/o *artificiali)*.

Per le considerazioni svolte sul significato e sul ruolo delle variabili *slack* e delle variabili artificiali, il sistema si presenta inizialmente in forma canonica rispetto alle variabili ausiliarie (*slack* e/o *artificiali)* e nel corso dell'algoritmo viene trasformato, con la procedura algebrica descritta, in forme canoniche equivalenti, al fine di generare soluzioni basiche ammissibili sempre migliori, fino alla soluzione basica ammissibile ottima.

Nelle slide seguenti si descrive la struttura algebrica dell'algoritmo del simplesso standard.

Struttura algebrica del sistema di equazioni

Si consideri il sistema di equazioni $\boldsymbol{Ax} = \boldsymbol{b}$, ottenuto dalla trasformazione del sistema di disequazioni e/o equazioni, di dimensioni m x n (m equazioni in n variabili) con $m < n$, dove $\boldsymbol{x}$ è il vettore delle n variabili, $\boldsymbol{A}$ è la matrice dei coefficienti delle variabili nelle m equazioni, $\boldsymbol{b}$ è il vettore degli m termini noti. Sia $\boldsymbol{x}_b$ il vettore delle variabili basiche in un certo stadio k dell'algoritmo e sia $\boldsymbol{x}_{nb}$ il vettore delle restanti variabili (non basiche).

Si definisce *matrice di base* la matrice B costituita dalle colonne della matrice A iniziale relative alle variabili di x_b. Sia $\boldsymbol{NB}$ la matrice delle restanti colonne di $\boldsymbol{A}$. Il vettore $\boldsymbol{x}$ e la matrice $\boldsymbol{A}$ possono dunque essere scritti come:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{nb} \\ \dots \\ \boldsymbol{x}_b \end{bmatrix} \qquad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{NB} \,|\, \boldsymbol{B} \end{bmatrix}$$

Il sistema $\boldsymbol{Ax} = \boldsymbol{b}$ si potrà scrivere pertanto nel modo seguente:

$$\begin{bmatrix} \boldsymbol{NB} \,|\, \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{nb} \\ \dots \\ \boldsymbol{x}_b \end{bmatrix} = \boldsymbol{b}, \qquad \boldsymbol{NB}\boldsymbol{X}_{\boldsymbol{NB}} + \boldsymbol{B}\boldsymbol{X}_{\boldsymbol{B}} = \boldsymbol{b}$$

Questo sistema di equazioni iniziale può essere messo in forma canonica rispetto a un certo insieme di variabili basiche, effettuando una sola operazione di tipo algebrico per determinare una soluzione basica ammissibile.

Struttura Algebrica dell'Algoritmo

Infatti, se si premoltiplica il sistema iniziale $\boldsymbol{Ax = b}$, scritto come $\boldsymbol{NBx_{nb} + Bx_b = b}$, per $\boldsymbol{B^{-1}}$, matrice inversa di $\boldsymbol{B}$ allo stadio k , si ottiene:

$\boldsymbol{B^{-1}}[\boldsymbol{NBx_{nb}} + \boldsymbol{Bx_b}] = \boldsymbol{B^{-1}b}$ e quindi: $\boldsymbol{B^{-1}NBx_{nb}} + \boldsymbol{B^{-1}Bx_b} = \boldsymbol{B^{-1}b}$

Ponendo $\boldsymbol{B^{-1}NB = C}$ ed essendo $\boldsymbol{B^{-1}B = I}$, si ottiene un'espressione del sistema in forma canonica rispetto alle variabili del vettore $\boldsymbol{x_b}$, come inizialmente affermato: $\boldsymbol{Cx_{nb} + Ix_b = B^{-1}b}$

Se nel sistema così ottenuto si pone $\boldsymbol{x_{nb}} = \boldsymbol{0}$ (variabili non basiche) si ricava il vettore $\boldsymbol{x_b}$ delle variabili basiche: $\boldsymbol{x_b = B^{-1}b}$.

Pertanto, **è possibile mettere un sistema $\boldsymbol{Ax = b}$ in forma canonica rispetto a un prefissato insieme di variabili basiche e determinare una soluzione basica ammissibile, premoltiplicando il sistema stesso per la matrice $\boldsymbol{B^{-1}}$, inversa della matrice di base $\boldsymbol{B}$,** costituita dalle colonne iniziali di $\boldsymbol{A}$ relative alle variabili rispetto alle quali si vuole il sistema in forma canonica.

Se le colonne di $\boldsymbol{B}$ sono le ultime m colonne della matrice iniziale $\boldsymbol{A}$ (o, che è lo stesso, le colonne di $\boldsymbol{A}$ vengono riordinate in modo che le colonne di $\boldsymbol{B}$ siano le ultime m), la matrice $\boldsymbol{I}$ della forma canonica si troverà nella stessa posizione, cioè nelle ultime m colonne. Se invece le colonne di $\boldsymbol{B}$ sono "distribuite" in $\boldsymbol{A}$, la matrice $\boldsymbol{I}$ sarà analogamente distribuita.

Come si è detto la matrice $\boldsymbol{B}$ prende il nome di "matrice di base". Con il termine "base" si suole anche indicare l'insieme di variabili basiche le cui colonne in $\boldsymbol{A}$ costituiscono $\boldsymbol{B}$. La matrice $\boldsymbol{B^{-1}}$ si indica con il termine "inversa della matrice di base" o, sinteticamente, "inversa di base".

Struttura Algebrica dell'Algoritmo

Si possono svolgere due semplici considerazioni di tipo algebrico per mostrare che la matrice $\boldsymbol{B}^{-1}$ è presente in una particolare posizione della tabella del simplesso ottenuta con l'operazione di pre-moltiplicazione sopradescritta.

1. Il sistema $\boldsymbol{Ax} = \boldsymbol{b}$ è in forma canonica sin dalla prima tabella (per l'aggiunta delle variabili *slack* e/o delle variabili artificiali). Il sistema quindi può essere scritto nella forma $[\boldsymbol{D} \mid \boldsymbol{I}]\ \boldsymbol{x} = \boldsymbol{b}$ (riordinando le colonne è sempre possibile ottenere una configurazione di questo tipo).

2. Ogni tavola del simplesso contiene un sistema in forma canonica, necessario per identificare una soluzione basica ammissibile. Sulla base delle considerazioni svolte in precedenza essa quindi può essere pensata come ottenuta da una trasformazione del sistema iniziale attraverso la pre-moltiplicazione per la matrice $\boldsymbol{B}^{-1}$, inversa della matrice di base $\boldsymbol{B}$ relativa all'insieme di variabili basiche di quella tabella, come mostrato in precedenza.

Sulla base di queste due considerazioni, se si pre-moltiplica il sistema iniziale per $\boldsymbol{B}^{-1}$ (al fine di ottenere il sistema in forma canonica rispetto all'insieme di variabili la cui matrice di base è $\boldsymbol{B}$) si ottiene:

$$\boldsymbol{B}^{-1}\ [\boldsymbol{D} \mid \boldsymbol{I}]\ \boldsymbol{x} = \boldsymbol{B}^{-1}\boldsymbol{b} \qquad \text{e quindi:} \qquad [\boldsymbol{B}^{-1}\boldsymbol{D} \mid \boldsymbol{B}^{-1}\boldsymbol{I}]\ \boldsymbol{x} = \boldsymbol{B}^{-1}\boldsymbol{b}$$

Ponendo $\boldsymbol{B}^{-1}\boldsymbol{D} = \boldsymbol{E}$ ed essendo $\boldsymbol{B}^{-1}\boldsymbol{I} = \boldsymbol{B}^{-1}$, si ottiene:

$$[\boldsymbol{E} \mid \boldsymbol{B}^{-1}]\ \boldsymbol{x} = \boldsymbol{B}^{-1}\boldsymbol{b}$$

Quindi ogni tabella del Simplesso ottenuta con l'operazione di *pivoting* contiene la matrice $\boldsymbol{B}^{-1}$, inversa della matrice di base associata alla soluzione relativa a quella tabella. Essa si trova nelle colonne in cui era presente la matrice identità iniziale. La sua posizione dunque è identificabile in relazione alla configurazione della tavola iniziale.

Analisi di stabilità post-ottimale

Analisi post-ottimale

Nei modelli di ottimizzazione continua presentati si è fatta l'ipotesi che i dati del problema siano di tipo deterministico, cioè definiti senza incertezza. In realtà essi possono presentare un certo grado di aleatorietà, tipico del funzionamento dei sistemi industriali e territoriali. La disponibilità di una risorsa può essere condizionata dal sistema di distribuzione delle merci. La domanda di un bene sul mercato può variare per la presenza di un prodotto concorrente. Un coefficiente di costo o di profitto può variare in relazione all'andamento del sistema monetario internazionale.

I parametri utilizzati nel modello sono riconducibili quindi ad una distribuzione di frequenza di cui si assume il valore medio. La soluzione del modello, determinata sulla base dei dati disponibili, contiene quindi elementi di incertezza.

E' necessario allora sviluppare un'*analisi post-ottimale*, basata sulla determinazione degli effetti che le possibili variazioni dei parametri del modello hanno sulla soluzione stessa. Tradizionalmente l'analisi post-ottimale si divide in analisi di stabilità ed analisi parametrica.

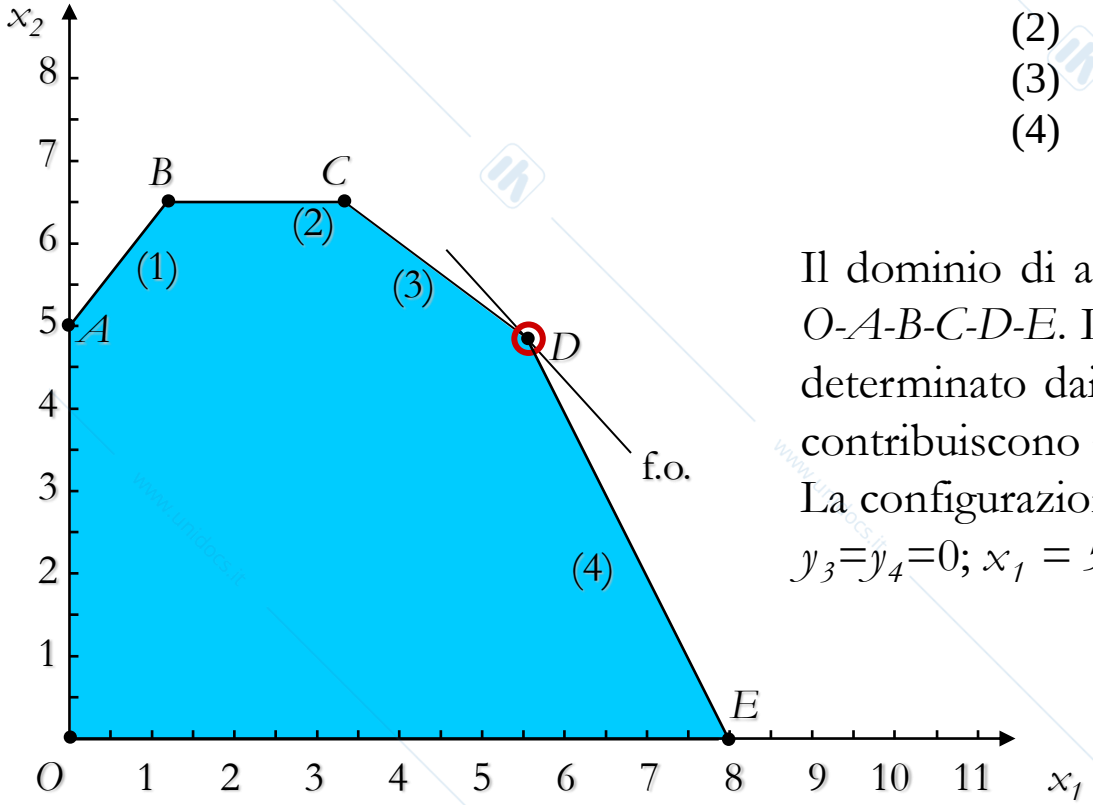
L'analisi di stabilità determina i campi di variazione dei termini noti b_i e dei coefficienti c_j della funzione obiettivo all'interno dei quali rimane inalterata la configurazione della soluzione basica ammissibile ottima.

L'analisi parametrica studia l'andamento della soluzione (valori ottimi delle variabili e della funzione obiettivo) al variare dei parametri oggetto di studio. Per approfondimenti su questo argomento si rinvia ai manuali indicati nei riferimenti bibliografici.

Analisi post-ottimale

Si prenda in considerazione il seguente problema con due variabili e quattro vincoli:

$$
\begin{array}{llrcrcrcl}
 & \text{Max} & z & = & 11x_1 & + & 10x_2 & & \\
\text{s.a} & & (1) & - & 5x_1 & + & 4x_2 & \leq & 20 \\
 & & (2) & & & & x_2 & \leq & 6.5 \\
 & & (3) & & 3x_1 & + & 4x_2 & \leq & 36 \\
 & & (4) & & 2x_1 & + & x_2 & \leq & 16
\end{array}
$$



Il dominio di ammissibilità è il poliedro di vertici *O-A-B-C-D-E*. La soluzione ottima è nel vertice *D*, determinato dai vincoli 3 e 4. I vincoli 1 e 2 non contribuiscono alla formazione dell'ottimo.
La configurazione della s.b.a. ottima è la seguente:
$y_3=y_4=0$; $x_1 = 5.6$, $x_2 = 4.8$, $y_1 = 28.8$ 0, $y_2 = 1.7$.

Analisi di stabilità rispetto all'incremento del termine noto b_i di un vincolo che partecipa alla formazione dell'ottimo

Il vincolo 3 contribuisce alla formazione del vertice ottimo.

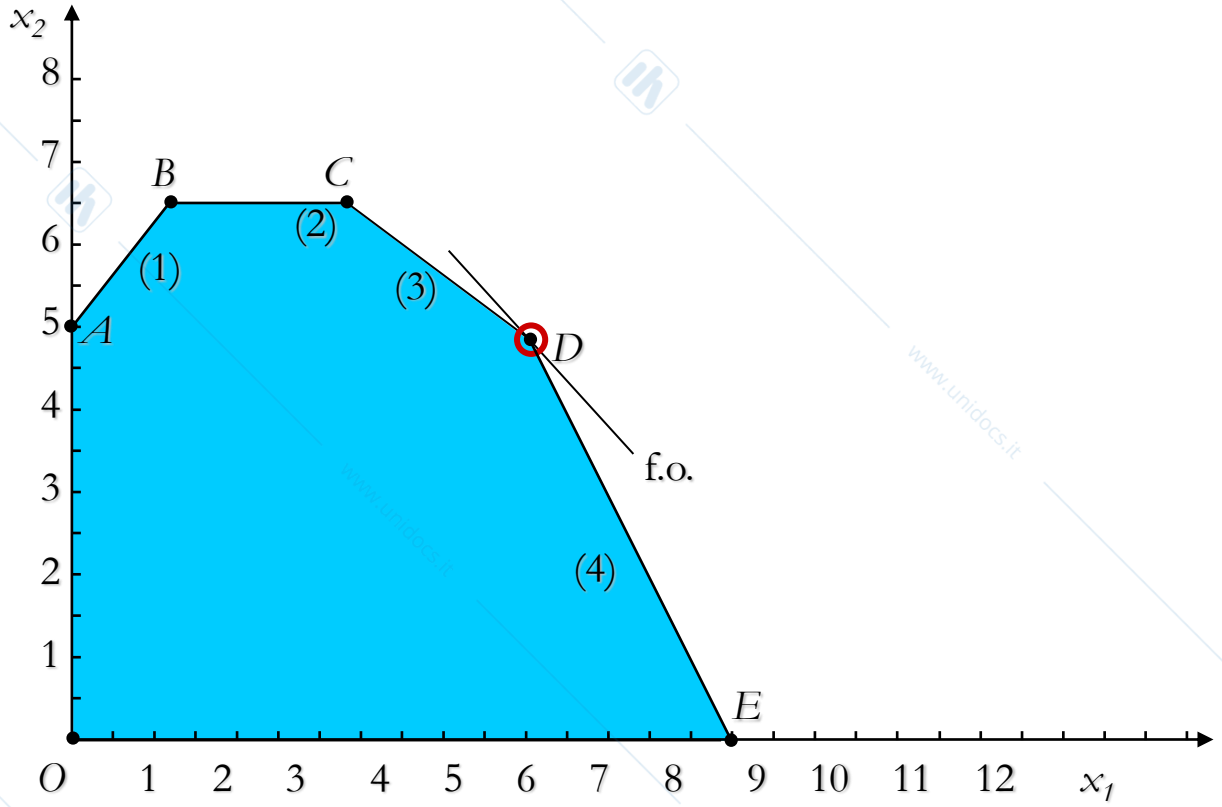
Si supponga che il suo termine noto b_3 subisca un incremento.

Come si può vedere nelle slide successive lo spigolo corrispondente (*C-D*) si sposta verso l'alto, per esempio fino alla posizione *C'-D'*, il dominio diventa *O-A-B-C'-D'-E* e il vertice ottimo *D* si sposta nel punto *D'*, in corrispondenza del quale non muta la configurazione della s.b.a. ottima ($y_3=y_4=0$; $x_1 > 0$, $x_2 > 0$, $y_1 > 0$, $y_2 > 0$). Il termine noto b_3 può aumentare ancora fino a determinare una posizione limite del vincolo in corrispondenza della quale il dominio diventa *O-A-B-D^-E*. Il vincolo 3 diventa ridondante ed il vertice ottimo si porta nella posizione $D^$.

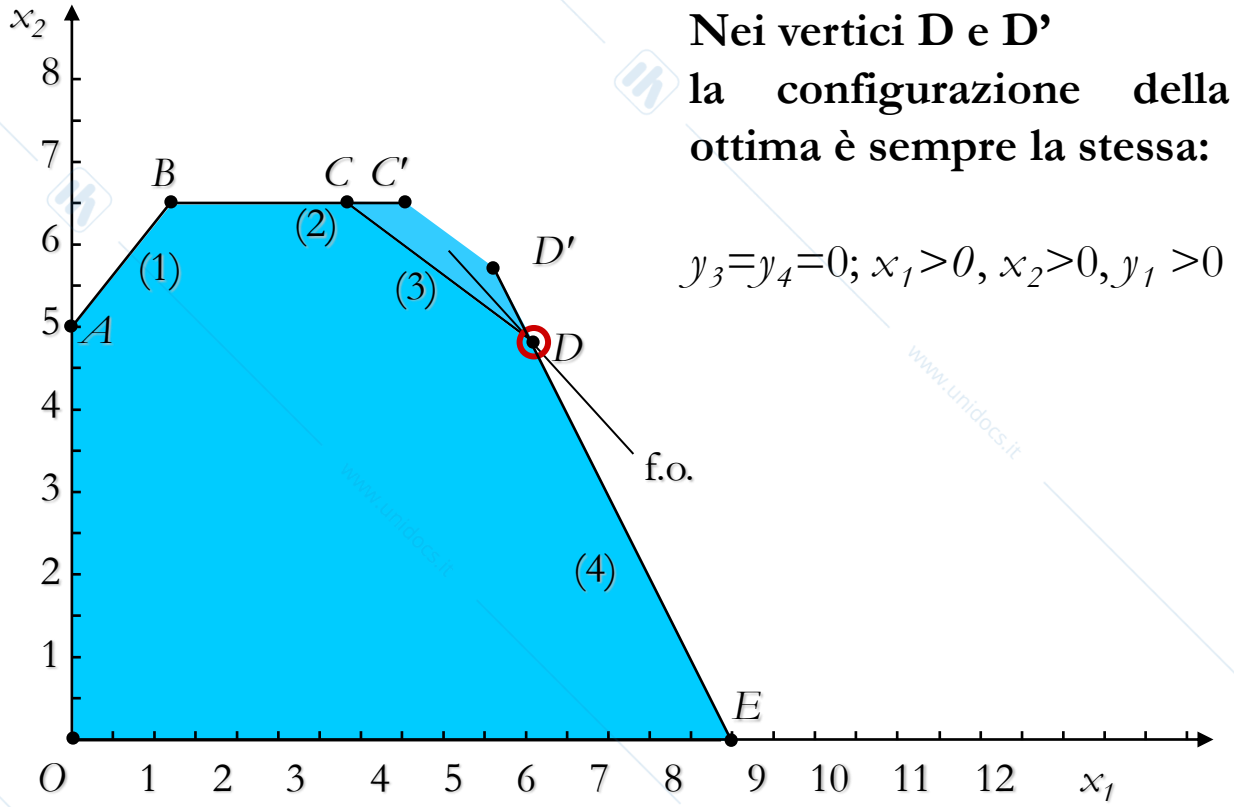
In questo vertice la configurazione della s.b.a. ottima resta immutata, anche se la soluzione è degenere, infatti è: $y_3 = y_4 = 0$, $x_1 > 0$, $x_2 > 0$, $y_1 > 0$, $y_2 = 0$ (basica). **L'incremento limite del termine noto b_3, con il quale il punto di ottimo diventa $D^$ e non cambia la configurazione della s.b.a. ottima, si indica con Δb_3^+.**

Se il termine noto aumentasse ulteriormente il punto di ottimo resterebbe in $D^$, ma la configurazione della s.b.a. ottima muterebbe, perché la variabile y_3, nulla e non basica in $D^$, diventerebbe positiva e quindi basica, mentre y_2, nulla e basica in $D^$, resterebbe nulla ma diventerebbe non basica.

Analisi di stabilità
rispetto all'incremento del termine noto b_i
di un vincolo che partecipa alla formazione dell'ottimo



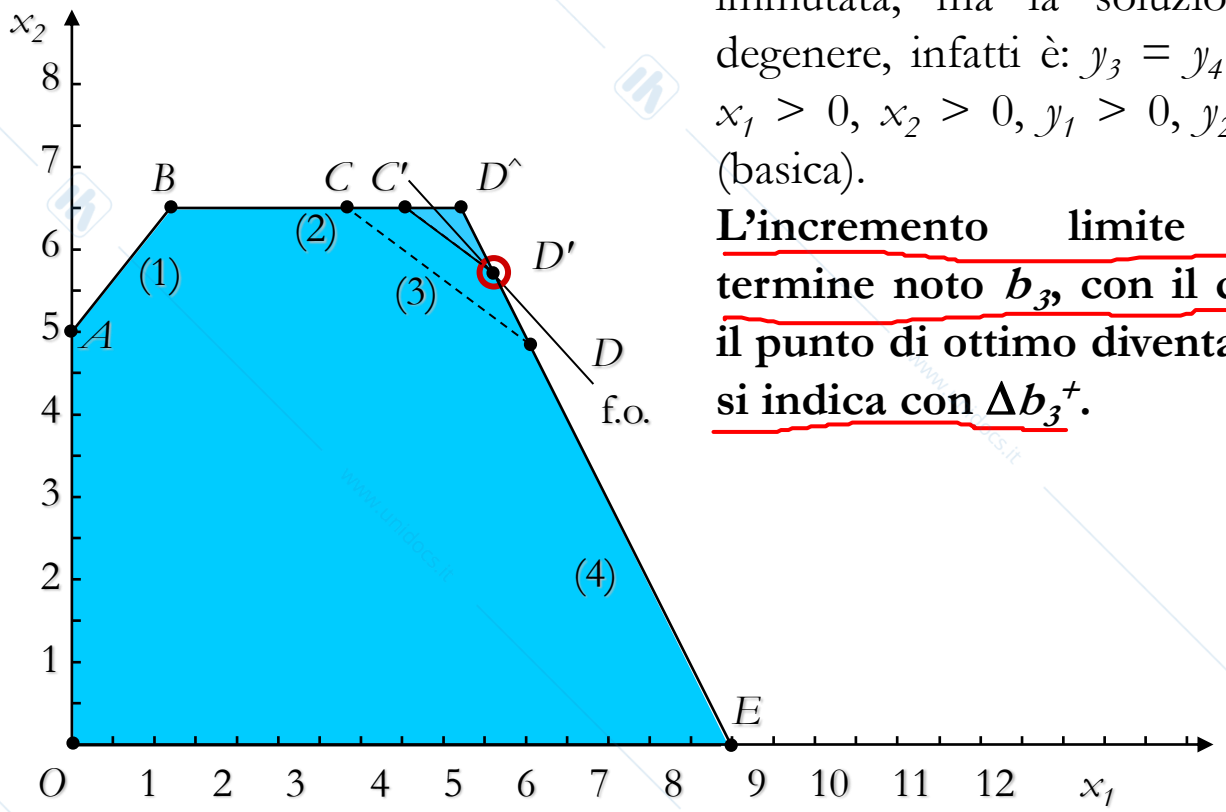
Analisi di stabilità rispetto all'incremento del termine noto b_i
di un vincolo che partecipa alla formazione dell'ottimo



Nei vertici D e D' la configurazione della s.b.a. ottima è sempre la stessa:

$y_3=y_4=0;\ x_1>0,\ x_2>0,\ y_1>0,\ y_2>0.$

Analisi di stabilità rispetto all'incremento del termine noto b_i
di un vincolo che partecipa alla formazione dell'ottimo



Nel vertice $D\hat{}$ la configurazione della s.b.a. ottima resta immutata, ma la soluzione è degenere, infatti è: $y_3 = y_4 = 0$, $x_1 > 0$, $x_2 > 0$, $y_1 > 0$, $y_2 = 0$ (basica).

L'incremento limite del termine noto b_3, con il quale il punto di ottimo diventa $D\hat{}$, si indica con Δb_3^+.

Analisi di stabilità
rispetto al decremento del termine noto b_i
di un vincolo che partecipa alla formazione dell'ottimo

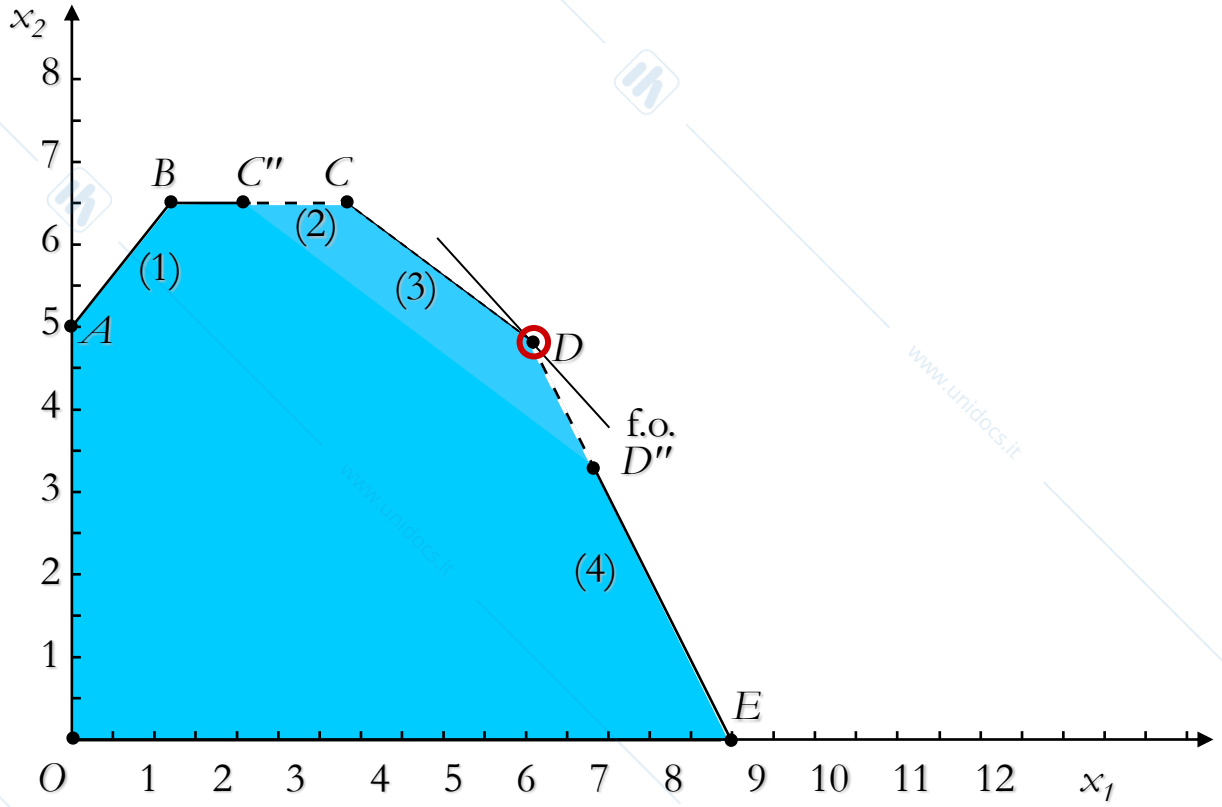
Si supponga ora che il termine noto b_3 subisca un decremento.

Come si può vedere nelle slide successive, lo spigolo *C-D* si sposta verso il basso, per esempio fino alla posizione *C''-D''*, ed il vertice ottimo *D* si sposta nel punto *D''*, in corrispondenza del quale non muta la configurazione della s.b.a. ottima. Il termine noto b_3 può diminuire fino ad una posizione del vincolo *B-D'''*, in corrispondenza della quale il vincolo 2 diventa ridondante ed il vertice ottimo si sposta nel punto *D'''*, nel quale ancora non muta la s.b.a. ottima.

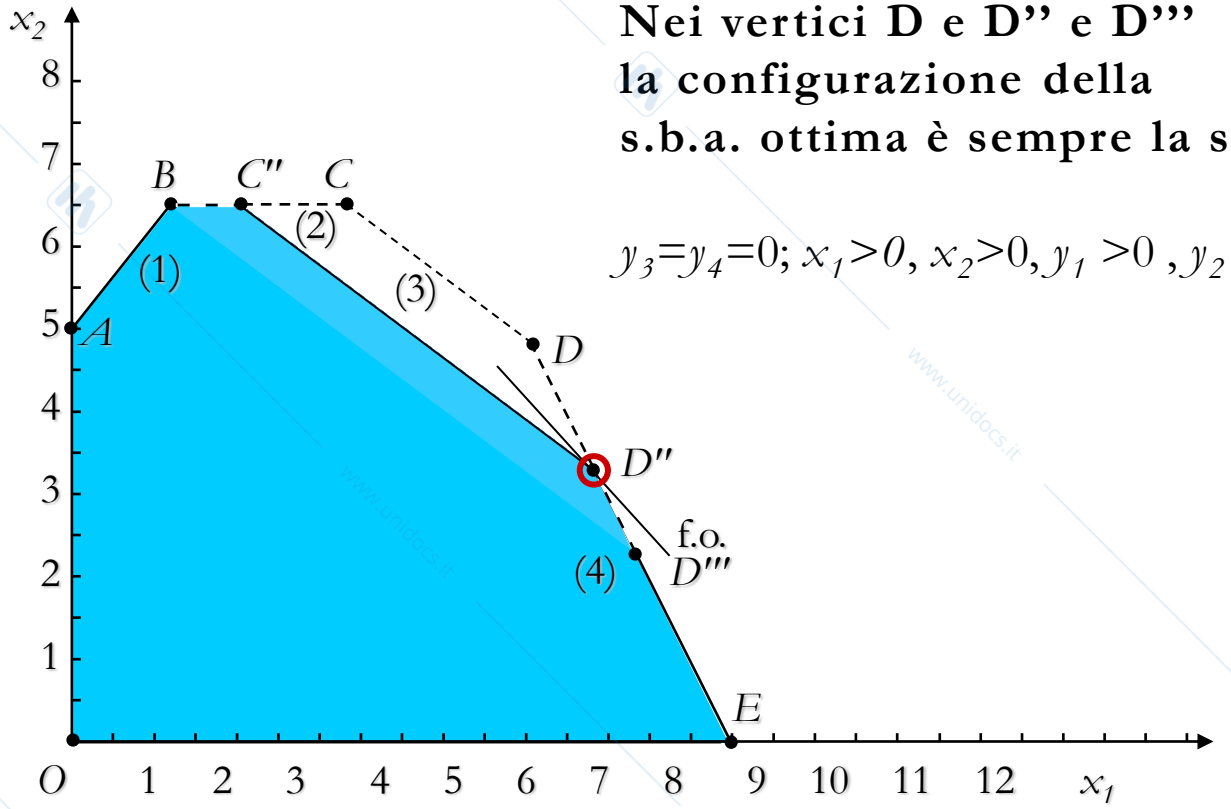
Il termine noto b_3 può diminuire ancora fino alla posizione limite del vincolo *C'''-E*, in corrispondenza della quale diventa ridondante il vincolo 4 e il vertice ottimo si sposta in *E*. La configurazione della s.b.a. ottima resta immutata, ma la soluzione è degenere, infatti sarà: $y_3 = y_4 = 0$, $x_1 > 0$, $x_2 = 0$ (basica degenere), $y_1 > 0$, $y_2 > 0$. **Il decremento limite del termine noto, corrispondente alla posizione del vincolo *C'''-E*, si indica con Δb_3^-.**

Se il termine noto b_3 diminuisse ulteriormente il punto di ottimo si sposterebbe lungo l'asse *x*, sul segmento *EO*, verso l'origine O. La configurazione della s.b.a. ottima muterebbe di nuovo, perché la variabile y_4, nulla e non basica, diventerebbe positiva e quindi basica, mentre x_2, nulla e basica in *E*, resterebbe nulla, ma diventerebbe non basica.

Analisi di stabilità
rispetto al decremento del termine noto b_i
di un vincolo che partecipa alla formazione dell'ottimo



Analisi di stabilità
rispetto al decremento del termine noto b_i
di un vincolo che partecipa alla formazione dell'ottimo



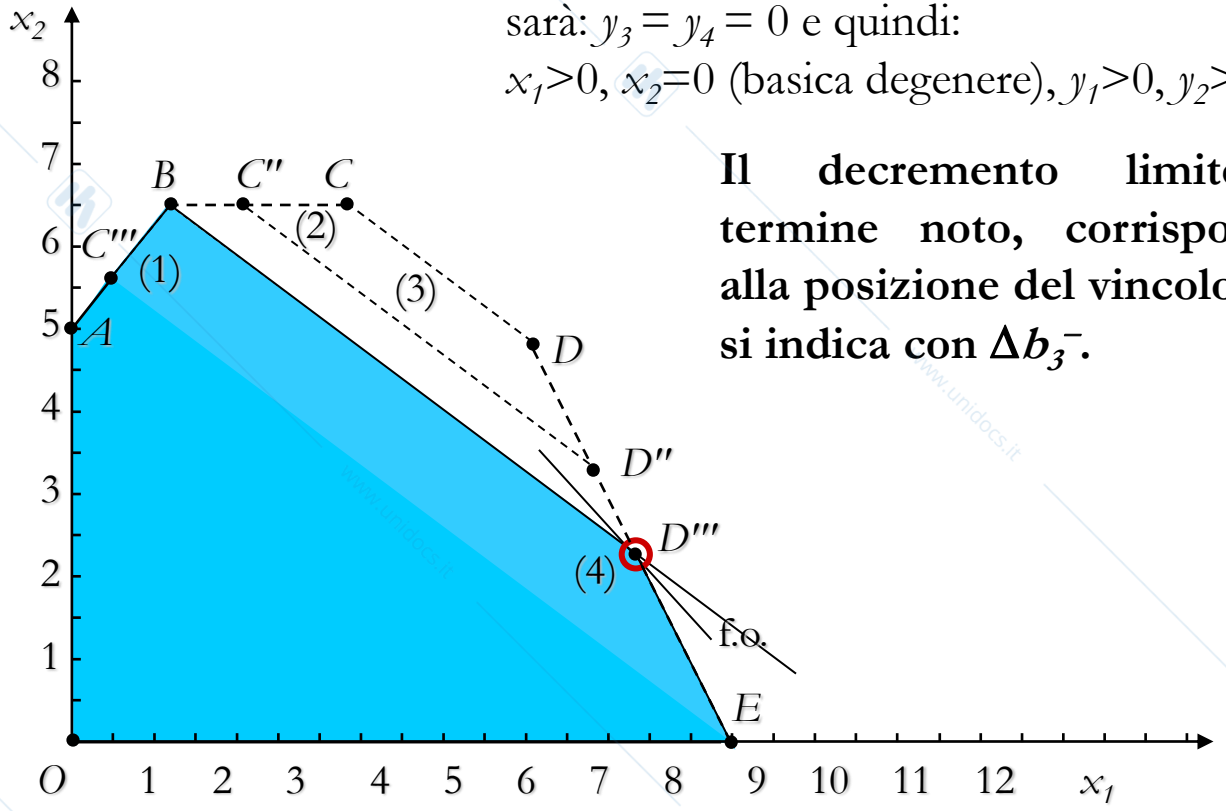
Nei vertici D e D'' e D''' la configurazione della s.b.a. ottima è sempre la stessa:

$y_3=y_4=0;\ x_1>0,\ x_2>0,\ y_1>0,\ y_2>0.$

Analisi di stabilità
rispetto al decremento del termine noto b_i
di un vincolo che partecipa alla formazione dell'ottimo

In E la configurazione della s.b.a. ottima resta immutata, ma la soluzione è degenere, infatti sarà: $y_3 = y_4 = 0$ e quindi:
$x_1>0$, $x_2=0$ (basica degenere), $y_1>0$, $y_2>0$.

Il decremento limite del termine noto, corrispondente alla posizione del vincolo C'''-E, si indica con Δb_3^-.



Analisi di stabilità rispetto al termine noto b_i
di un vincolo che non partecipa alla formazione dell'ottimo

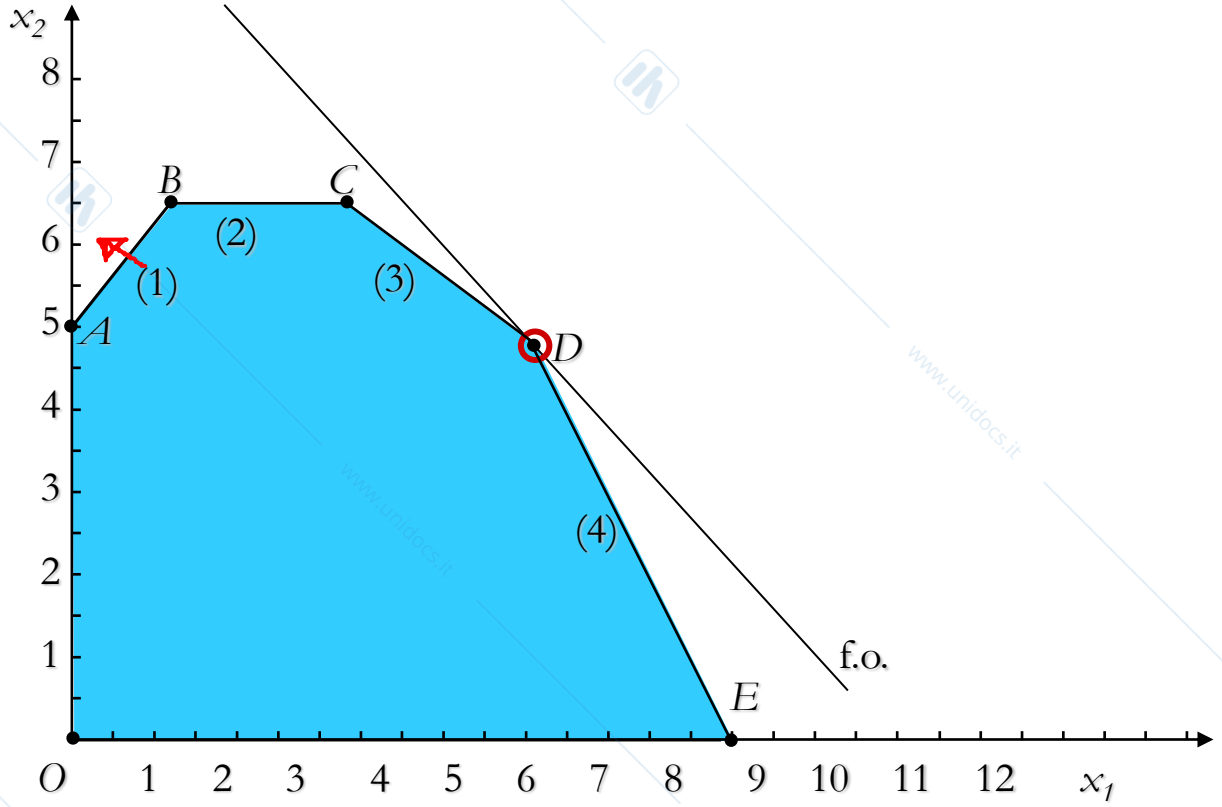
Si prenda ora in considerazione il vincolo 1, che non contribuisce alla formazione del vertice ottimo.

Come si può vedere nelle slide successive, **se il suo termine noto b_1 subisce un incremento,** lo spigolo corrispondente A-B si sposta verso l'alto. In corrispondenza della posizione passante per A' il vincolo 1 diventa ridondante, ma la configurazione della s.b.a. ottima non muta. Il termine noto b_1 può continuare ad aumentare indefinitamente senza che cambi la configurazione della s.b.a. ottima. Dunque $\Delta b_1^+ = \infty$.

Se invece b_1 subisce un decremento, lo spigolo A-B si sposta verso il basso. In corrispondenza della posizione A''-C il vincolo 2 diventa ridondante, ma non muta la configurazione della soluzione basica ammissibile ottima. Il termine noto b_1 può diminuire ancora fino alla posizione limite A'''-D del vincolo, in corrispondenza della quale diventa ridondante anche il vincolo 3 (decremento limite Δb_i^-). Se il termine noto b_1 diminuisse ulteriormente il punto di ottimo si sposterebbe lungo lo spigolo D-E e la configurazione della s.b.a. ottima muterebbe, diventando: $y_1 = y_4 = 0$, $x_1 > 0$, $x_2 > 0$, $y_2 > 0$, $y_3 > 0$, diventando degenere in E con $x_1 = 0$.

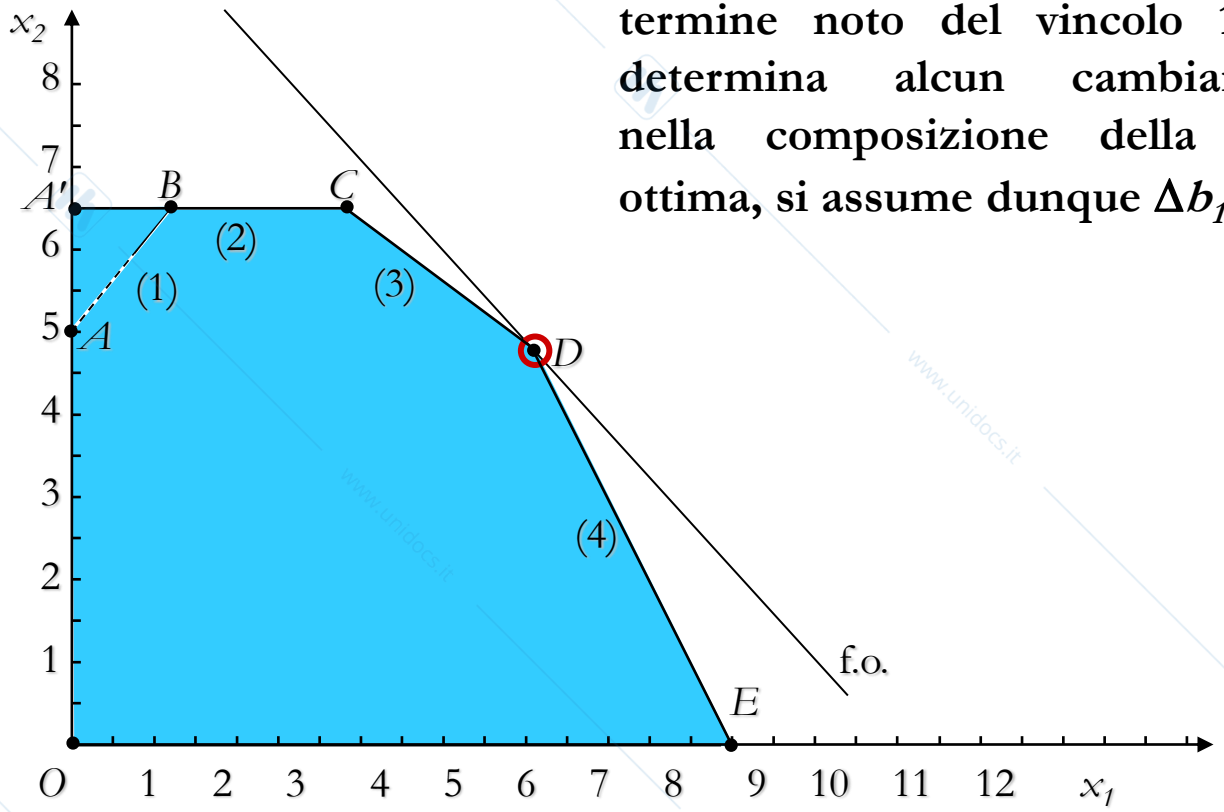
Analisi di stabilità rispetto al termine noto b_i

di un vincolo che non partecipa alla formazione dell'ottimo



Analisi di stabilità rispetto al termine noto b_i

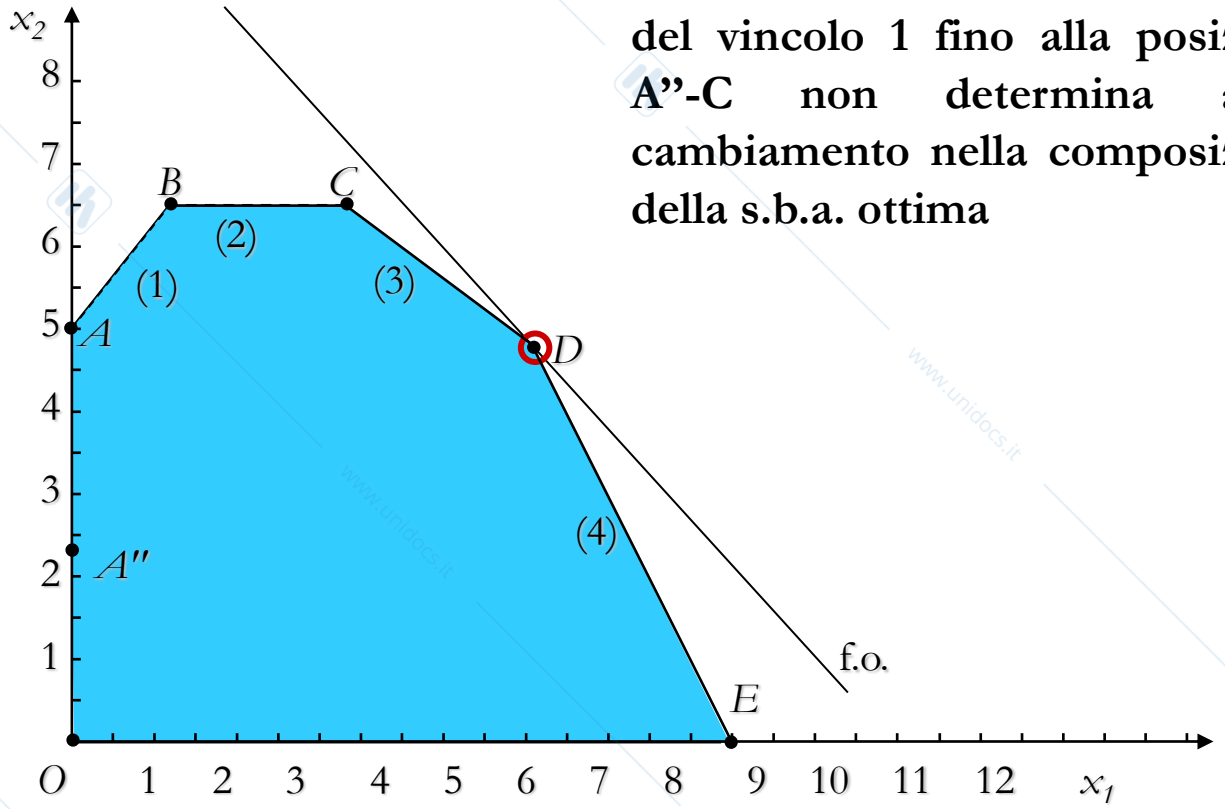
di un vincolo che non partecipa alla formazione dell'ottimo



Un incremento anche grande del termine noto del vincolo 1 non determina alcun cambiamento nella composizione della s.b.a. ottima, si assume dunque $\Delta b_1^+ = \infty$

Analisi di stabilità rispetto al termine noto b_i

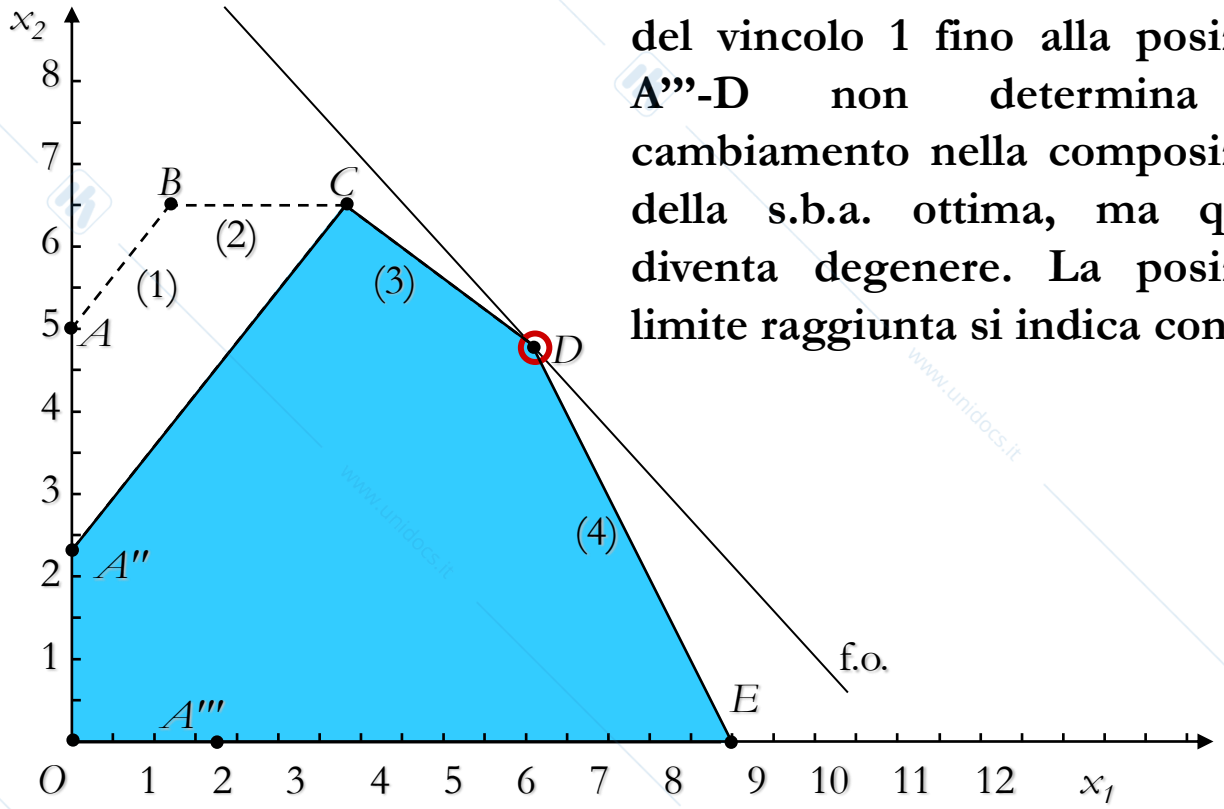
di un vincolo che non partecipa alla formazione dell'ottimo



Il decremento del termine noto del vincolo 1 fino alla posizione A"-C non determina alcun cambiamento nella composizione della s.b.a. ottima

Analisi di stabilità rispetto al termine noto b_i

di un vincolo che non partecipa alla formazione dell'ottimo



Il decremento del termine noto del vincolo 1 fino alla posizione A'''-D non determina un cambiamento nella composizione della s.b.a. ottima, ma questa diventa degenere. La posizione limite raggiunta si indica con Δb_1^-

Campo di variazione del termine noto b_i

Indicando con Δb_i^+ e Δb_i^- i valori limite, in incremento e decremento, del generico termine noto b_i. che "mantengono" la configurazione della s.b.a. ottima, si può dire che il campo di variazione del generico b_i è definito dalle posizioni limite che il vincolo può raggiungere, corrispondenti ai valori:

$$b_i^- = b_i - \Delta b_i^- \qquad\qquad b_i^+ = b_i + \Delta b_i^+$$

$$b_i^- = b_i - \Delta b_i^- \leq b_i \leq b_i + \Delta b_i^+ = b_i^+$$

I valori di Δb_i^+ e Δb_i^- possono essere calcolati con una semplice espressione algebrica che utilizza le informazioni contenute nell'ultima tabella del simplesso.

Questa espressione algebrica si costruisce a partire dalla condizione $\boldsymbol{x}_b^* = (\boldsymbol{B}^*)^{-1}\boldsymbol{b}$, come illustrato nelle slide successive e poi successivamente con riferimento all'esempio numerico precedente.

Procedura algebrica per la determinazione di Δb_i^+ e Δb_i^-

Si ricordi che ogni soluzione basica ammissibile è esprimibile attraverso la relazione

$$\boldsymbol{x}_b = \boldsymbol{B}^{-1}\boldsymbol{b}$$

Le s.b.a. sono sempre non negative quindi possiamo scrivere $\boldsymbol{x}_b = \boldsymbol{B}^{-1}\boldsymbol{b} \geq \boldsymbol{0}$.

Nel punto di ottimo vale dunque la seguente condizione di ammissibilità:

$$\boldsymbol{x}_b^* = (\boldsymbol{B}^*)^{-1}\boldsymbol{b} \geq \boldsymbol{0}$$

I valori limite di Δb_i^- e Δb_i^+ sono quelli in corrispondenza di quali non vale più questa condizione di ammissibilità.

L'espressione algebrica dei valori di Δb_i^+ e Δb_i^- si determina, a partire da questa condizione, in funzione di parametri noti presenti nella tabella finale del simplesso.

Procedura algebrica attraverso un esempio numerico

Calcolo di Δb_i^+

$$\Delta b_i^+ = \min_k [x_{kb}^* / (-\beta_{ki})] \ (\forall \beta_{ki} < 0)$$

Dunque per calcolare Δb_i^+

- bisogna fare i rapporti fra gli elementi di posto omologo
  - > del vettore finale dei termini noti e
  - > della colonna dei coefficienti della colonna β_i
    - relativa alla variabile (slack o artificiale) corrispondente al vincolo in esame, $\forall \ \beta_{ki} < 0$,
- prendere il valore minimo

Procedura algebrica per la determinazione di Δb_i^-

Si può effettuare la stessa analisi per calcolare Δb_i^-. Si indichi quindi con $\boldsymbol{b}_i^-$ il vettore dei termini noti contenente l'elemento $b_i^- = b_i - \Delta b_i^-$ e sia $\boldsymbol{x}_b^{*-}$ il vettore della soluzione ottima ottenuta con un decremento Δb_i^- del termine noto b_i.

Con la stessa procedura utilizzata per la determinazione di Δb_i^+ si determinano le espressioni scalari: $x_{kb}^* - \Delta b_i^- \beta_{ki} \geq 0 \quad (k = 1,\ldots, m)$

Se nella colonna $\boldsymbol{\beta}_i$ non ci sono elementi $\beta_{ki} > 0$ allora $\Delta b_i^- = \infty$.

Altrimenti il valore limite di Δb_i^- sarà espresso da:

$$\Delta b_i^- = x_{tb}^* / \beta_{ti} = \min_k [x_{kb}^* / \beta_{ki}] \quad (\forall \beta_{ki} > 0)$$

Calcolo di Δb_i^-

Quindi per calcolare Δb_i^- bisogna fare i rapporti fra gli elementi di posto omologo del vettore finale dei termini noti e della colonna dei coefficienti della colonna $\boldsymbol{\beta}_i$ (colonna della matrice $\boldsymbol{B}^{*-1}$ inversa di base) corrispondente al vincolo in esame, per $\beta_{ki} > 0$

Attenzione!

Per il calcolo dei rapporti $[\alpha_{kb}^{*}/\beta_{ki}]$ bisogna porre attenzione alla individuazione della colonna β_i

β_i è una colonna di B^{*-1} (inversa della matrice di base) e dunque:

- se il vincolo *i* per il quale si deve calcolare Δb_i^+ o Δb_i^- è del tipo ≤ essa sarà la colonna della variabile slack y_i corrispondente,

- se è del tipo = o ≥ la colonna β_i sarà quella della variabile artificiale h_i corrispondente.

Il modello duale

Il modello duale

Il modello duale e la teoria della dualità assumono una grande importanza nella teoria della programmazione matematica. **In questo capitolo il modello duale viene presentato con due esempi: un classico problema di produzione ed un problema di dieta alimentare.**

Nel problema di produzione **un'azienda vuol determinare il piano di produzione che massimizzi il profitto nel rispetto dei vincoli di disponibilità delle materie prime necessarie (modello primale), ed inoltre vuole determinare il prezzo «ombra» delle materie prime attraverso un modello per la minimizzazione del costo di un nuovo approvvigionamento (modello duale).**

Il problema della dieta consiste nella determinazione dell'insieme di alimenti che minimizzi il costo di acquisto nel rispetto dei vincoli di soddisfacimento del fabbisogno vitaminico. Il problema viene presentato sia dal punto di vista di un cliente del mercato che deve acquistare al minimo costo gli alimenti necessari ad assumere le vitamine (modello primale), sia dal punto di vista di un consulente farmaceutico che vuole massimizzare il ricavo della vendita di pillole di vitamine (modello duale).

In entrambi i problemi vengono mostrate le corrispondenze tra i due modelli con le relative regole di trasformazione.
Vengono poi illustrati i principali teoremi della dualità.

Il problema duale

L'azienda vede che le materie prime disponibili per il processo produttivo non hanno tutte la stessa importanza per il raggiungimento del punto di ottimo. Decide quindi di costruire un modello di ottimizzazione volto a calcolare il prezzo che essa è disposta a pagare per una ulteriore unità di ciascuna risorsa in un eventuale nuovo approvvigionamento.

V**uole calcolare quindi il loro "*prezzo ombra*", cioè il "valore" che l'azienda attribuisce ad esse in termini di variazione del valore di funzione obiettivo determinata da un incremento o un decremento unitario della loro disponibilità.** Il prezzo ombra rappresenta quindi **l'utilità marginale di una risorsa rispetto al punto di ottimo.**

Per costruire questo modello si utilizzano gli stessi dati presenti nel modello precedente, ma con altre variabili, u_1, u_2 ed u_3, che rappresentano i prezzi ombra delle risorse R1, R2 ed R3, e si definiscono variabili duali.

Il modello duale per i «prezzi ombra»

Il modello duale è il seguente:

$$\text{Min } v = 57\,u_1 + 135\,u_2 + 84\,u_3$$

s.a.

$$(1) \quad 3u_1 + 3u_2 + 12u_3 \geq 60$$

$$(2) \quad 6u_1 + 15u_2 + 3u_3 \geq 90$$

$$u_1, u_2, u_3 \geq 0$$

La funzione obiettivo rappresenta il costo totale di approvvigionamento (da minimizzare), espresso in funzione dei prezzi ombra. Dunque i coefficienti delle variabili nella funzione obiettivo sono le quantità delle materie prime necessarie, cioè i termini noti del modello primale costruito per la massimizzazione del profitto.

Ciascun vincolo si riferisce ad un prodotto ed esprime la condizione che il prezzo ombra totale delle risorse necessarie per produrre una unità del prodotto stesso deve essere almeno pari al profitto ricavabile da una unità di prodotto, cioè al corrispondente coefficiente della funzione obiettivo del modello primale.

Il dominio ammissibile del modello, tridimensionale, è nella slide seguente.

Il punto di ottimo è il vertice C con $u_1^*=14.3$, $u_2^*=0$, $u_3^*=1.43$, $v^* = 934.3$.

In esso entrambi i vincoli sono saturi (cioè soddisfatti con il segno =).

Modelli primale e duale

$$\text{Max } z = 60x_1 + 90x_2$$

s.a

$$(R1)\quad 3x_1 + 6x_2 \leq 57$$
$$(R2)\quad 3x_1 + 15x_2 \leq 135$$
$$(R3)\quad 12x_1 + 3x_2 \leq 84$$
$$x_1, x_2 \geq 0$$

$$\text{Min } v = 57u_1 + 135u_2 + 84u_3$$

s.a

$$(1)\quad 3u_1 + 3u_2 + 12u_3 \geq 60$$
$$(2)\quad 6u_1 + 15u_2 + 3u_3 \geq 90$$
$$u_1, u_2, u_3 \geq 0$$

Modelli primale e duale

$$\text{Max } z = 60x_1 + 90x_2$$

s.a

$$(R1)\quad 3x_1 + 6x_2 \leq 57$$
$$(R2)\quad 3x_1 + 15x_2 \leq 135$$
$$(R3)\quad 12x_1 + 3x_2 \leq 84$$

$$x_1, x_2 \geq 0$$

$$\text{Min } v = 57u_1 + 135u_2 + 84u_3$$

s.a

$$(1)\ 3u_1 + 3u_2 + 12u_3 \geq 60$$
$$(2)\ 6u_1 + 15u_2 + 3u_3 \geq 90$$
$$u_1, u_2, u_3 \geq 0$$

Si possono notare le corrispondenze tra i due modelli, primale e duale.

-Il primo modello è a massimizzare, il secondo modello è a minimizzare.

-Le matrici dei coefficienti dei vincoli dei due modelli sono tra loro trasposte.

-Il vettore dei termini noti del modello primale diventa il vettore dei coefficienti della funzione obiettivo del modello duale

-Il vettore dei coefficienti della funzione obiettivo del modello primale diventa il vettore dei termini noti del modello duale. I vincoli del modello primale sono del tipo ≤, i vincoli del modello duale sono del tipo ≥.

Si noti che il ruolo dei due modelli è scambievole e quindi le corrispondenze possono essere invertite.

Significato delle variabili duali

Si può affermare che una variabile duale rappresenta il valore di una risorsa espresso in termini di variazione del valore di f.o. del modello primale.

- Se una variazione unitaria di termine noto (di risorsa) del primale determina una variazione di valore della funzione obiettivo allora quella risorsa "ha valore" e questo valore è pari alla variazione di valore indotta nella f.o..

- Se invece una variazione unitaria di termine noto (di risorsa) del primale non determina alcuna variazione di valore della funzione obiettivo allora si dice che quella risorsa "non ha valore".

Trasformazione primale max - duale min in forma standard

Si consideri il seguente **modello con *m* vincoli ed *n* variabili** in cui con $\boldsymbol{c}^T$ si indica il vettore riga dei coefficienti della funzione obiettivo:

$$\begin{array}{lrcl} \text{Max} & z & = & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.a} & \mathbf{A}\,\boldsymbol{x} & \leq & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & \boldsymbol{0} \end{array}$$

Questo modello può essere definito modello primale in forma *standard*.

Si definisce modello duale (in forma *standard*) del modello precedente il seguente **modello con *n* vincoli ed *m* variabili**:

$$\begin{array}{lrcl} \text{Min} & v & = & \boldsymbol{b}^T\boldsymbol{u} \\ \text{s.a} & \mathbf{A}^{\mathbf{T}}\boldsymbol{u} & \geq & \boldsymbol{c} \\ & \boldsymbol{u} & \geq & \mathbf{0} \end{array}$$

Corrispondenze Primale (max) - Duale (min) in forma standard

Primale		Duale
Funzione obiettivo		Funzione obiettivo
z max	⟷	v min
Vincoli i		Variabili u_i
$\leq$	⟷	$u_i \geq 0$
$\leq$	⟷	$u_i \geq 0$
$\leq$	⟷	$u_i \geq 0$
Variabili x_j		Vincoli j
$x_j \geq 0$	⟷	$\geq$
$x_j \geq 0$	⟷	$\geq$
$x_j \geq 0$	⟷	$\geq$
matrice $\boldsymbol{A}$	⟷	*matrice* $\boldsymbol{A^T}$
vettore coefficienti f.o. $\boldsymbol{c}^T$	⟷	*vettore termini noti* $\boldsymbol{c}$
vettore termini noti $\boldsymbol{b}$	⟷	*vettore coefficienti f.o.* $\boldsymbol{b}^T$

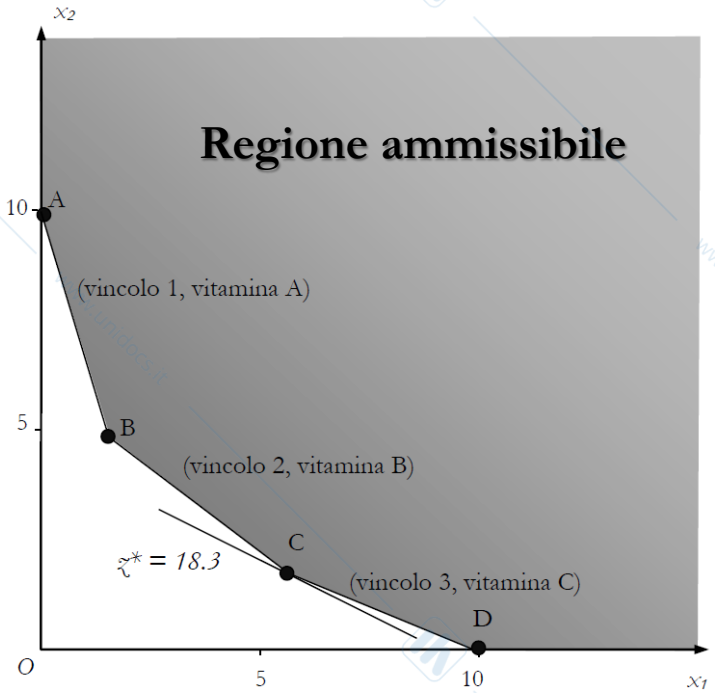
Modello del problema dell'acquirente (primale)

L'acquirente al mercato costruisce un modello per la determinazione delle quantità di alimenti da acquistare, con l'obiettivo di minimizzare il costo di acquisto, con i vincoli sul rispetto della dieta in termini di vitamine.

$$\text{Min } z = 2x_1 + 4x_2$$

s.a

$$(1)\quad 10x_1 + 3x_2 \geq 30$$
$$(2)\quad 6x_1 + 8x_2 \geq 48$$
$$(3)\quad 4x_1 + 10x_2 \geq 40$$
$$x_1, x_2 \geq 0$$



Punto di ottimo del modello primale

Il punto di ottimo è il vertice C al quale corrisponde il piano di acquisti $x_1^* = 5.71$, $x_2^* = 1.71$ con $z^* = 18.3$.

Sono saturi (cioè soddisfatti con il segno di uguale) i vincoli 2 e 3 corrispondenti alle vitamine B e C (che sono dunque assunte nella quantità minima richiesta).

Il vincolo 1 è invece soddisfatto con il segno di maggiore e dunque la vitamina A è assunta in quantità maggiore del minimo richiesto.

Modello (duale) del problema del venditore di pillole

Il venditore di pillole costruisce un modello per la determinazione del prezzo di una unità di ciascuna vitamina con l'obiettivo di massimizzare il suo ricavo, con i vincoli sulla «competizione» tra il prezzo delle vitamine e il costo dei prodotti alimentari al mercato.

$$\text{Max } v = 30\,u_A + 48\,u_B + 40\,u_C$$

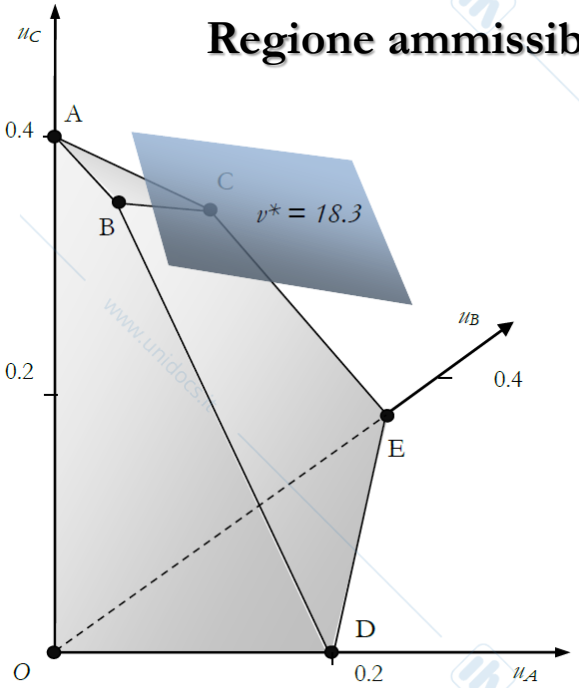
s.a

$$(1)\quad 10\,u_A + 6\,u_B + 4\,u_C \le 2$$

$$(2)\quad 3\,u_A + 8\,u_B + 10\,u_C \le 4$$

$$u_A, u_B, u_C \ge 0$$

Regione ammissibile



Punto di ottimo del modello duale

Il punto di ottimo è il vertice C al quale corrispondono i valori $u_A^* = 0$, $u_B^* = 0.143$, $u_C^* = 0.286$, con $v^* = 18.3$. In esso sono saturi (cioè soddisfatti con il segno di uguale) entrambi i vincoli corrispondenti ai due alimenti.

Si noti che il prezzo u_A della vitamina A risulta pari a 0. Ciò significa che il venditore non attribuisce valore a quella vitamina che nel piano di acquisto del signore è assunta in modo sovrabbondante.

Modelli primale e duale

$$\text{Min } z = \quad 2x_1 + 4x_2$$

s.a

$$(1) \quad 10x_1 + 3x_2 \geq 30$$
$$(2) \quad 6x_1 + 8x_2 \geq 48$$
$$(3) \quad 4x_1 + 10x_2 \geq 40$$
$$x_1, x_2 \geq 0$$

$$\text{Max } v = 30u_A + 48u_B + 40u_C$$

s.a

$$(1) \quad 10u_A + 6u_B + 4u_C \leq 2$$
$$(2) \quad 3u_A + 8u_B + 10u_C \leq 4$$
$$u_A, u_B, u_C \geq 0$$

Trasformazione primale min - duale max in forma standard

Si consideri il seguente modello con m vincoli ed n variabili in cui con $\boldsymbol{c}^T$ si indica il vettore riga dei coefficienti della funzione obiettivo:

$$\begin{array}{lrcl} \text{Min} & v & = & \boldsymbol{b^T u} \\ \text{s.a} & \boldsymbol{A^T u} & \geq & \boldsymbol{c} \\ & \boldsymbol{u} & \geq & \boldsymbol{0} \end{array}$$

Questo modello può essere definito modello primale in forma *standard*.

Si definisce modello duale (in forma *standard*) del modello precedente il seguente modello con n vincoli ed m variabili:

$$\begin{array}{lrcl} \text{Max} & z & = & \boldsymbol{c^T x} \\ \text{s.a} & \boldsymbol{Ax} & \leq & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & \boldsymbol{0} \end{array}$$

Un esempio numerico con 6 prodotti alimentari e due vitamine

Problema primale: *6 variabili e 2 vincoli*

$$\text{Min } z = 3.5x_1 + 3x_2 + 6x_3 + 5x_4 + 2.7x_5 + 2.2x_6$$

s.a

$$\begin{aligned} x_1 + 2x_3 + 2x_4 + x_5 + 2x_6 &\geq 9 \\ x_2 + 3x_3 + x_4 + 3x_5 + 2x_6 &\geq 19 \end{aligned}$$

$$x_j \geq 0 \qquad j = 1,\ldots,6$$

Problema duale: *2 variabili e 6 vincoli*

$$\text{Max } v = 9u_1 + 19u_2$$

s.a

$$\begin{aligned} u_1 &\leq 3.5 \\ u_2 &\leq 3 \\ 2u_1 + 3u_2 &\leq 6 \\ 2u_1 + u_2 &\leq 5 \\ u_1 + 3u_2 &\leq 2.7 \\ 2u_1 + 2u_2 &\leq 2.2 \end{aligned}$$

$$u_j \geq 0 \qquad j = 1,..,2$$

Trasformazione primale - duale in forma non - standard

Se nel modello primale ci sono: - vincoli di qualunque tipo ($\leq$, $=$, $\geq$) e

- variabili ≥ 0, ≤ 0 e non ristrette nel segno (n.r.s.)

il primale non è in forma standard e può essere messo in forma standard con semplici operazioni algebriche, relative a vincoli e variabili

Vincoli **:** - Un vincolo $\leq$ resta tale;

- Un vincolo $\geq$ diventa $\leq$ cambiando segno a tutti gli elementi ed invertendo il verso della disuguaglianza
- Un vincolo di $=$ si sdoppia in 2 vincoli, uno di $\leq$ e uno di $\geq$; il vincolo di $\geq$ diventa $\leq$ come sopra descritto.

Variabili : - Una variabile $x_j \geq 0$ resta tale

- per una variabile $x_j \leq 0$ si pone $x_j = -x'_j$ con $x'_j \geq 0$;
- per una variabile x_k n.r.s. si pone $x_k = x'_k - x''_k$, con $x'_k \geq 0$, $x''_k \geq 0$.

Il modello primale così ottenuto in forma standard genera un modello duale in forma standard che non è il duale del modello primale originario. Operazioni algebriche "opposte" a quelle descritte generano un modello non in forma standard, duale dell'originario primale in forma non standard. Le operazioni algebriche necessarie sono descritte nel testo di riferimento del corso. Le corrispondenze tra variabili e vincoli del primale e vincoli e variabili del duale sono descritte nelle due slide successive-

Corrispondenze Primale (max) - Duale (min) in forma non standard

Primale/Duale		Duale
Funzione obiettivo		Funzione obiettivo
z max	⟷	v min
Vincoli i		Variabili u_i
$\leq$	⟷	$u_i \geq 0$
$=$	⟷	u_i n.r.s.
$\geq$	⟷	$u_i \leq 0$
Variabili x_j		Vincoli j
$x_j \geq 0$	⟷	$\geq$
x_j n.r.s.	⟷	$=$
$x_j \leq 0$	⟷	$\leq$
matrice $\boldsymbol{A}$	⟷	*matrice* $\boldsymbol{A^T}$
vettore termini noti $\boldsymbol{b}$	⟷	*vettore coefficienti f.o.* $\boldsymbol{b}^T$
vettore coefficienti f.o. $\boldsymbol{c}^T$	⟷	*vettore termini noti* $\boldsymbol{c}$

Teoremi del duale

- Il duale del duale è il primale

- Teorema del duale in forma debole $z \leq v$
- Teorema del duale in forma forte $z^* = v^*$

- Corollario dello scarto complementare
  - $x_j^* r_j^* = 0$, *per ogni j*
  - $u_i^* y_i^* = 0$, *per ogni i*

Ottimizzazione Intera

Modelli e metodi di ottimizzazione intera

I modelli e i metodi di ottimizzazione intera intervengono quando il problema decisionale presenta variabili corrispondenti a prodotti e/o attività non frazionabili.

Infatti il punto di ottimo di un problema formulato con variabili continue non è necessariamente intero. È necessario pertanto inserire nel modello il vincolo addizionale che alcune o tutte le variabili decisionali devono assumere valori interi.

Questi vincoli si definiscono vincoli di interezza e per i modelli così formulati si utilizza il termine *programmazione intera* (P.I.) *o discreta*. Si utilizza inoltre il termine *programmazione intera mista* quando alcune variabili decisionali sono continue ed altre sono intere.

Le variabili intere possono essere di due tipi:

- variabili intere di tipo generale, che assumono i valori interi 0,1,2,3,....fino ad un valore limite definito dai vincoli del modello.

- variabili intere binarie, che possono assumere solo i valori 1 e 0, nei casi in cui siano presenti decisioni del tipo si/no, associabili ai valori 1 e 0 delle variabili binarie.

Nel seguito si presentano un problema di ottimizzazione intera generale ed un problema di ottimizzazione binaria. Si illustra poi la rappresentazione grafica del reticolo dei punti interi e il concetto di formulazione di un problema intero attraverso un semplice modello in due variabili

Ottimizzazione intera

Un modello di programmazione intera (PI) si presenta nella forma seguente:

$$\text{Max } z = \sum_{j=1,n} c_j x_j$$

$$\text{s.a} \quad \sum_{j=1,n} a_{ij} x_j \leq (\geq, =)\ b_i \qquad i = 1,\ldots, m$$

$$x_j \geq 0 \text{ intera} \qquad j = 1,\ldots, n$$

I vincoli di interezza definiscono un reticolo di punti, interni al dominio continuo definito dai vincoli del problema, che costituisce l'insieme delle soluzioni ammissibili del problema intero. Il dominio ammissibile continuo diventa discreto e la soluzione ottima non si trova più necessariamente sulla frontiera del dominio convesso definito dai vincoli, ma può trovarsi al suo interno.

Nel seguito si illustra il concetto di *formulazione* di un problema intero

Nella successiva lezione si illustrerà il metodo del *Piano di Taglio (Cutting Plane)* ed il metodo Branch and Bound.

Poliedro P e insieme X dei punti interi

Si consideri il problema:

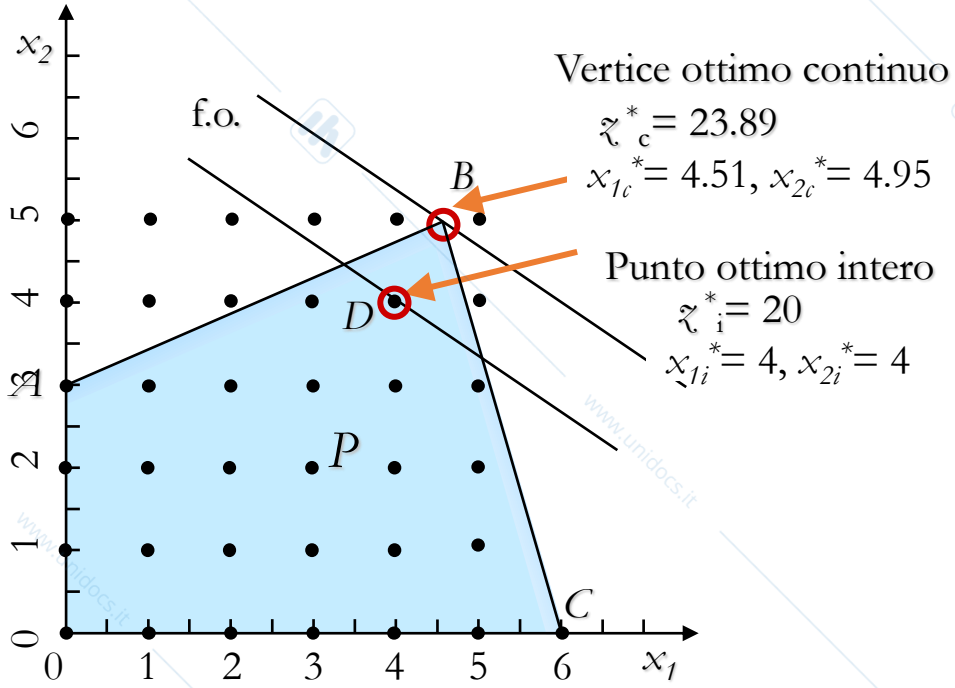
Max $z = 2x_1 + 3x_2$

s.a

$$-1.3x_1 + 3x_2 \leq 9$$

$$3x_1 + 0.9x_2 \leq 18$$

$x_1, x_2 \geq 0$ interi



Formulazioni del problema

Se non è possibile enumerare tutte le soluzioni o approssimare la soluzione continua, è necessario individuare metodi efficaci che consentano la soluzione dei problemi interi. A tale scopo è necessario premettere alcune considerazioni sul concetto di formulazione di un problema utilizzando un semplice problema in due variabili e la relativa regione ammissibile.

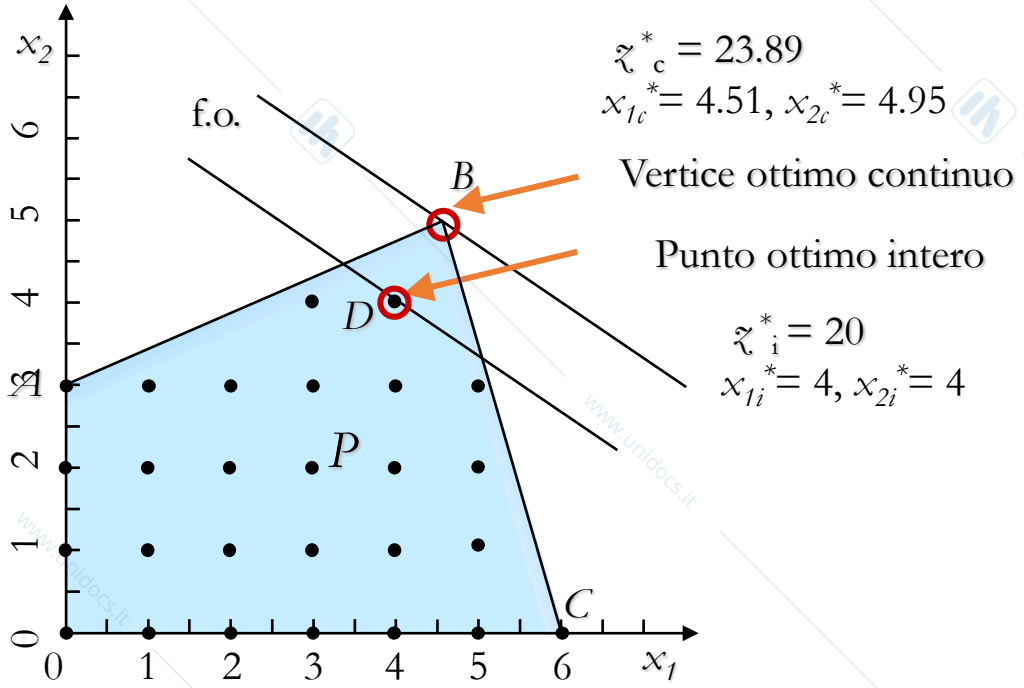
$$\text{Max } z = 2x_1 + 3x_2$$

s.a

$$-1.3x_1 + 3x_2 \leq 9$$

$$3x_1 + 0.9x_2 \leq 18$$

$$x_1, x_2 \geq 0 \quad \text{interi}$$



Si indichi con P il dominio convesso costituito dal poliedro dei vincoli.

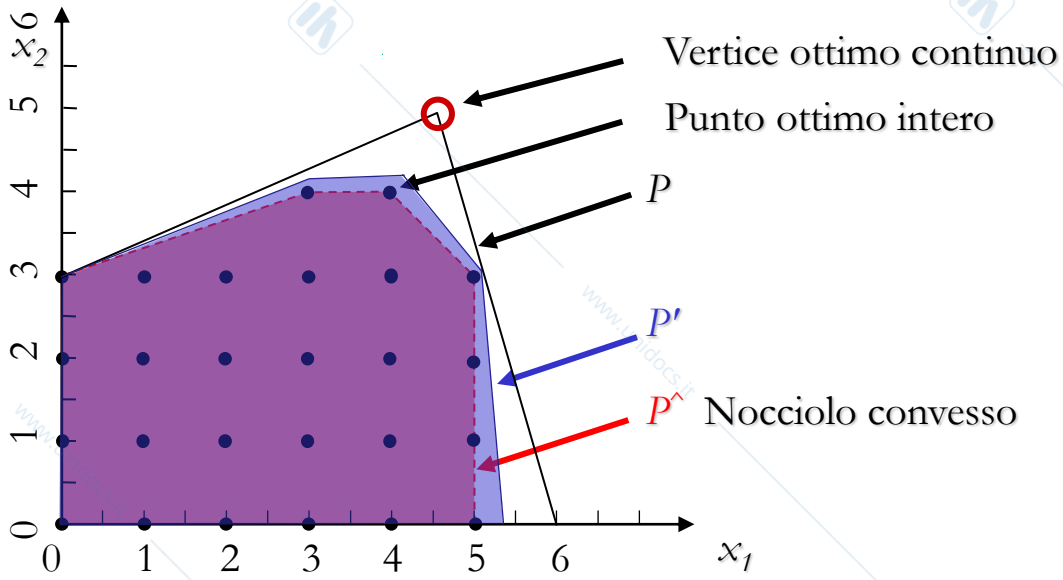
Si indichi con S l'insieme, discreto e finito, delle soluzioni ammissibili intere.

Formulazioni equivalenti e nocciolo convesso

$P = \{\boldsymbol{x} \geq 0 : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$

$P' = \{\boldsymbol{x} \geq 0 : \boldsymbol{A}'\boldsymbol{x} \leq \boldsymbol{b}'\}$

$P^{\wedge} = \{\boldsymbol{x} \geq 0 : \boldsymbol{A}^{\wedge}\boldsymbol{x} \leq \boldsymbol{b}^{\wedge}\}$



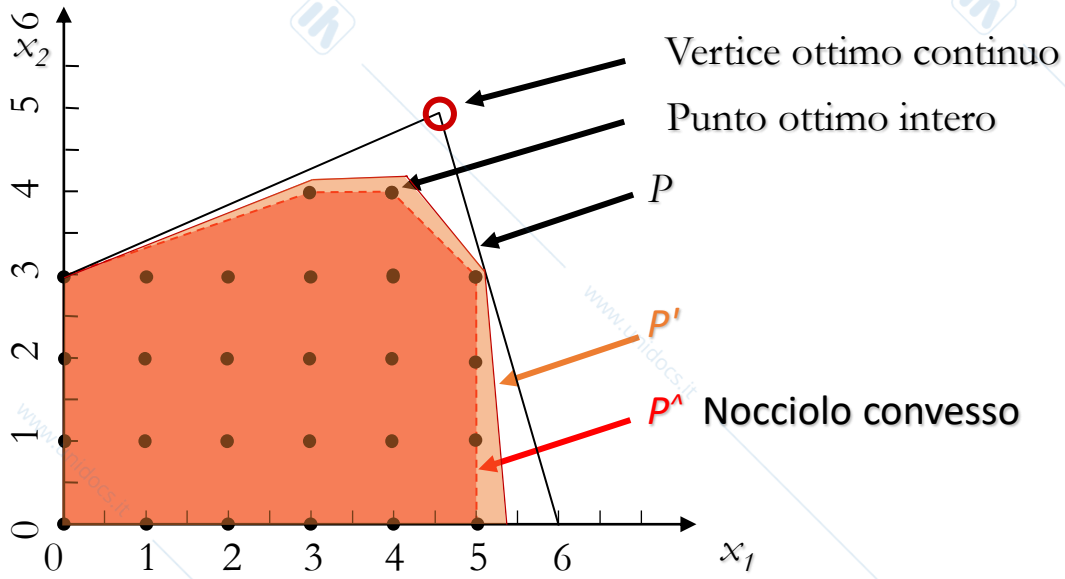
Formulazioni *P*, *P'* e nocciolo convesso

In figura oltre al poliedro *P* è riportato anche il poliedro *P'*, interno a *P*, corrispondente ad un altro insieme di vincoli. Si può verificare facilmente che anche *P'* mantiene al suo interno tutti i punti interi appartenenti ad *S*, ma è più "stringente" di quella di *P*.

$$P = \{\boldsymbol{x} \geq 0 : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$$

$$P' = \{\boldsymbol{x} \geq 0 : \boldsymbol{A}'\boldsymbol{x} \leq \boldsymbol{b}'\}$$

$$P^{\wedge} = \{\boldsymbol{x} \geq 0 : \boldsymbol{A}^{\wedge}\boldsymbol{x} \leq \boldsymbol{b}^{\wedge}\}$$



Le formulazioni di *P* e *P'* sono dunque equivalenti rispetto ai punti interi, ma le soluzioni prodotte con il rilassamento continuo dei due modelli sono fra loro diverse. Ciò dipende ovviamente dall'insieme di vincoli $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$, cioè dalla formulazione adottata per descrivere il problema.

Nocciolo convesso

Si potrebbe pensare di costruire una formulazione ancora più aderente all'insieme S dei punti interi della formulazione P'. Nella stessa figura 10.2 è riportato con linea tratteggiata il poliedro $P^{\wedge}$, interno a P, che si può immaginare corrispondente ad un altro insieme di vincoli. Si può verificare che anche $P^{\wedge}$ mantiene al suo interno tutti i punti interi appartenenti ad S. Il poliedro $P^{\wedge}$ è il *nocciolo convesso*:

"Dato un insieme S di punti interi, si definisce *nocciolo convesso* di S il più piccolo insieme convesso [*conv* (S)] che contiene S".

Il *nocciolo convesso* $P^{\wedge}$ consentirebbe dunque la determinazione immediata del punto di ottimo intero perché tutti i suoi vertici corrispondono a soluzioni intere e quindi si potrebbe trovare la soluzione ottima intera con il metodo (continuo) del Simplesso.

Si potrebbe pensare quindi di costruire la formulazione $\boldsymbol{A}^{\wedge}\boldsymbol{x} \leq (=, \geq)\boldsymbol{b}^{\wedge}$, perfettamente "aderente" all'insieme S dei punti interi, per garantire la determinazione della soluzione intera per qualunque funzione obiettivo lineare utilizzando l'algoritmo del Simplesso.

Purtroppo la formulazione del nocciolo convesso è una formulazione *ideale* e quindi bisogna ricorrere ad algoritmi dedicati alla programmazione intera.

Metodi di ottimizzazione intera

Cutting Plane

Branch and Bound

Il metodo del piano di taglio (Cutting plane)

Attraverso tecniche di generazione dei vincoli basate sui dati del modello, si aggiunge un vincolo. Per esempio:

$6.5x_1+12.5x_2 \leq 81.25$

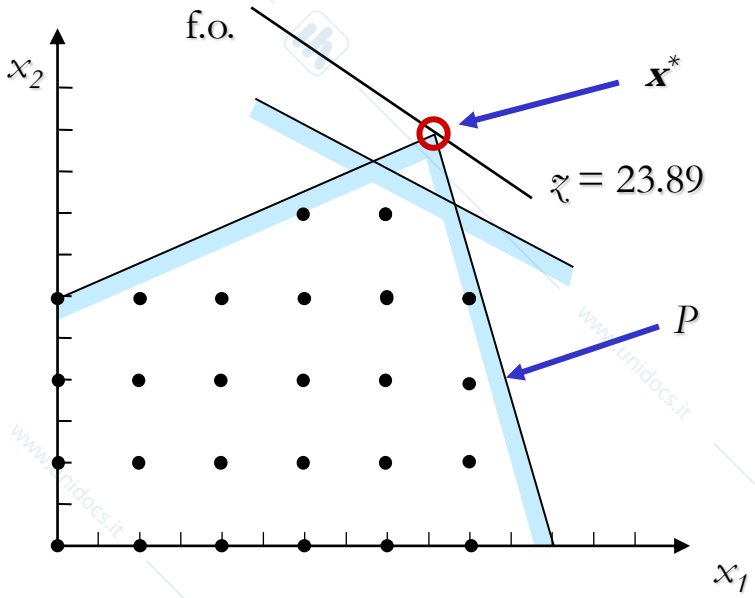
Questo vincolo esclude la soluzione ottima continua e non esclude nessuna soluzione intera

In termini formali, un vincolo

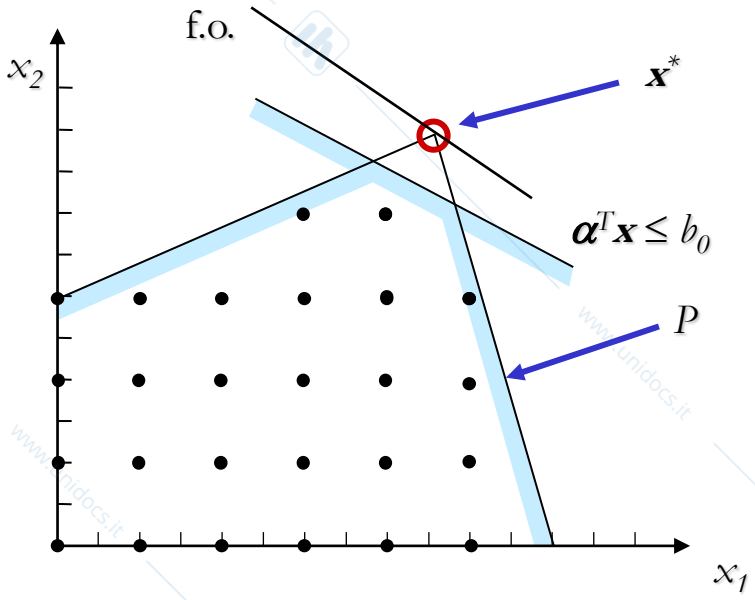
$\boldsymbol{\alpha}^T\boldsymbol{x} \leq b_0$ che:

- **esclude la soluzione ottima continua ($\boldsymbol{\alpha}^T\boldsymbol{x}^* > b_0$)**
- **non esclude nessuna soluzione intera ($\boldsymbol{\alpha}^T\boldsymbol{x} \leq b_0 \; \forall \; \boldsymbol{x} \in S$)**

Il vertice ottimo continuo si sposta

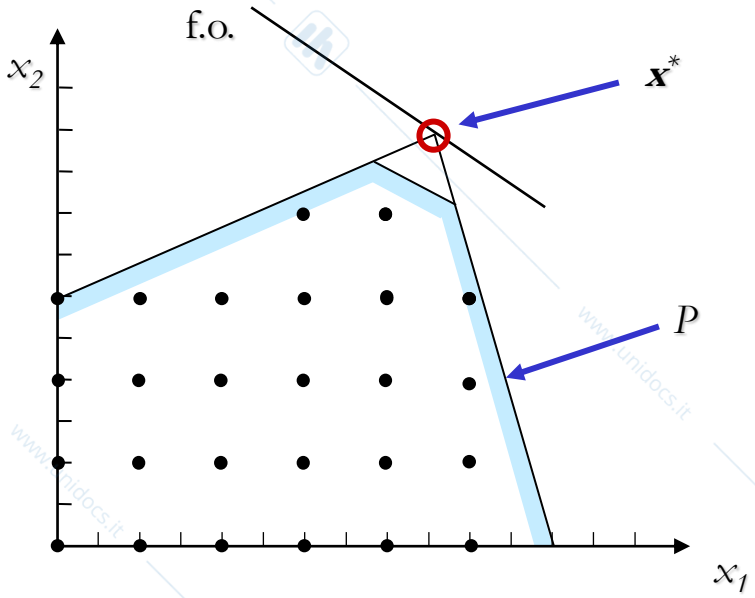


Piano di taglio $\alpha^T x \leq b_0$



Piano di taglio $\alpha^T x \leq b_0$

Il nuovo vertice ottimo è ancora continuo.



Piano di taglio $\alpha^T x \leq b_0$

Il nuovo vertice ottimo è ancora continuo.

Si aggiunge quindi un altro vincolo. Per esempio: $4.5x_1+25x_2 \leq 112.5$

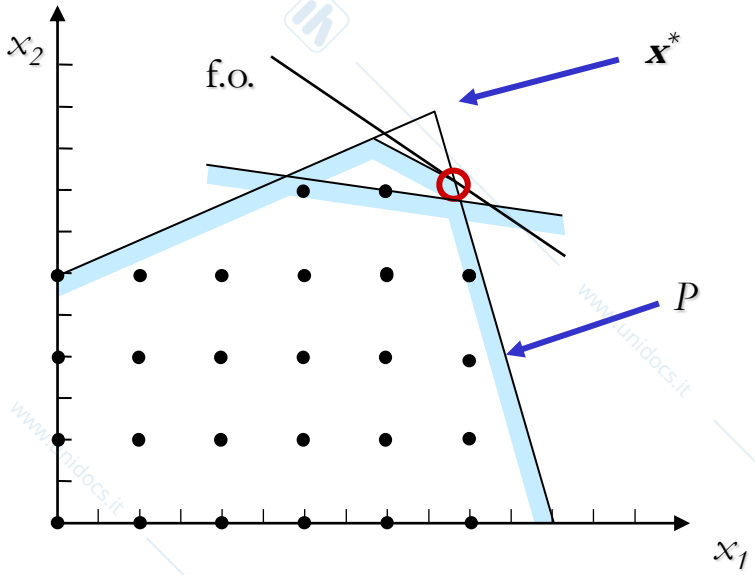
Questo vincolo esclude la nuova soluzione ottima continua e non esclude nessuna soluzione intera

In termini formali, un vincolo

$\alpha^T x \leq b'$ che:

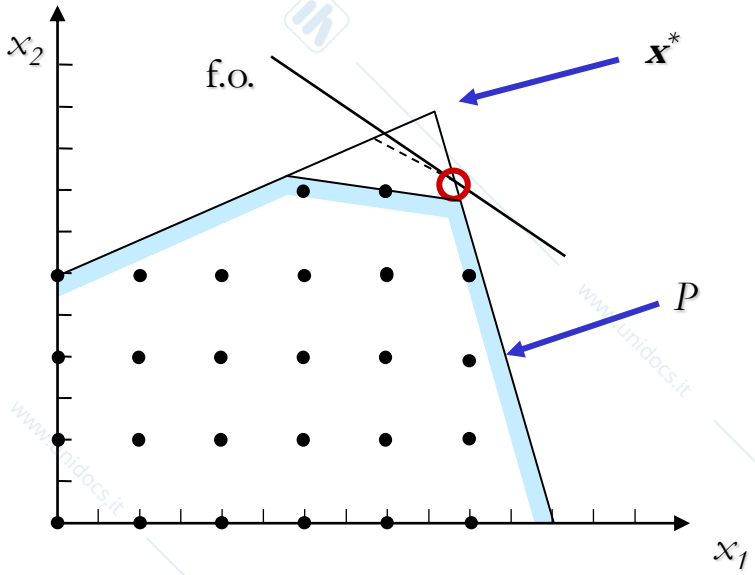
- **esclude la soluzione ottima continua ($\alpha^T x^* > b'$)**
- **non esclude nessuna soluzione intera ($\alpha^T x \leq b' \; \forall \; x \in S$)**

Il vertice ottimo continuo si sposta



Piano di taglio $\alpha^T x \leq b_0$

Il nuovo vertice ottimo è ancora continuo.



Piano di taglio $\alpha^T x \leq b_0$

Il vertice ottimo è ancora continuo.

Quindi si aggiunge un altro vincolo.
Per esempio: $7.5x_1+10x_2 \leq 75$

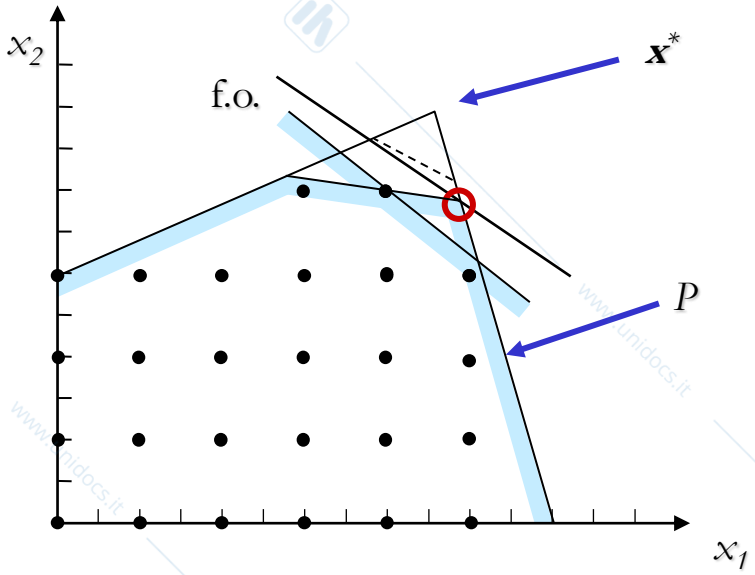
Questo vincolo esclude la nuova soluzione ottima continua e non esclude nessuna soluzione intera

In termini formali, un vincolo

$\alpha^T x \leq b$" che:

- **esclude la soluzione ottima continua ($\alpha^T x^* > b$")**
- **non esclude nessuna soluzione intera ($\alpha^T x \leq b$" $\forall\ x \in S$)**

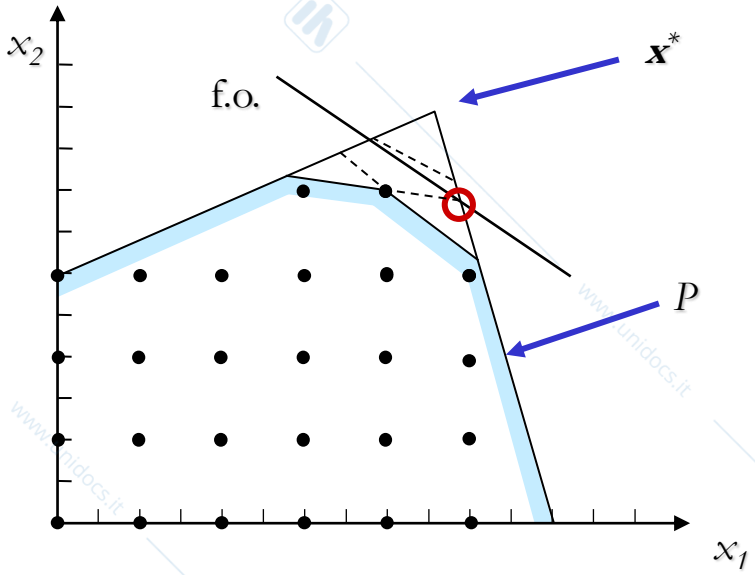
Il punto di ottimo continuo si sposta



Piano di taglio $\alpha^T x \leq b_0$

Il nuovo vertice ottimo ora è intero.

I vincoli che sono stati aggiunti hanno fatto sì che diventasse vertice del dominio e quindi ottenibile come soluzione dell'algoritmo del simplesso



Metodo del Cutting Plane

Il metodo del *piano di taglio* cerca di rafforzare la formulazione iniziale aggiungendo ad essa un vincolo (*piano di taglio*), generato con procedure di calcolo numerico, che la renda più stringente. Il piano di taglio deve avere quindi le seguenti proprietà:

1. escludere dalla regione ammissibile il vertice corrispondente alla soluzione ottima continua corrente.

2. non escludere dal dominio alcuna delle soluzioni ammissibili intere del problema originario.

Quindi, indicando con x^* la soluzione ottima continua del poliedro P corrente, con $\boldsymbol{\alpha}^T$ il vettore riga dei coefficienti a_{ij} del vincolo addizionale e con S l'insieme delle soluzioni ammissibili intere, il *piano di taglio* deve essere una disuguaglianza, per esempio del tipo $\boldsymbol{\alpha}^T\boldsymbol{x} \leq b_0$, tale che:

- $\boldsymbol{\alpha}^T\boldsymbol{x}^* > b_0$
- $\boldsymbol{\alpha}^T\boldsymbol{x} \leq b_0 \ \forall \ \boldsymbol{x} \in S$

Assegnato il punto $\boldsymbol{x}^*$, il problema di individuare un taglio del tipo $\boldsymbol{\alpha}^T\boldsymbol{x} \leq b_0$ viene detto *problema di separazione* di $\boldsymbol{x}^*$ da S.

Metodo del Cutting Plane

Il metodo del piano di taglio opera dunque con i seguenti passi:

1. si rilassano i vincoli di interezza e risolve il problema originario con l'algoritmo del Simplesso;
2. se la soluzione ottenuta rispetta i vincoli di interezza l'algoritmo termina, altrimenti si va al passo 3;
3. si modifica il sistema di vincoli iniziale introducendo un vincolo addizionale (*piano di taglio*, *cutting plane*) con le proprietà sopra descritte, che riduce la regione ammissibile del problema, ma non elimina nessuna soluzione intera;
4. si risolve il nuovo problema con il vincolo addizionale e senza i vincoli di interezza, e si torna al passo 2.

Se la nuova soluzione rispetta i vincoli di interezza la procedura termina, altrimenti si aggiunge un altro vincolo addizionale con le proprietà prima definite e si itera la procedura fino a determinare una soluzione intera.

L'algoritmo converge alla soluzione ottima intera del problema.

Le slide precedenti hanno mostrato lo sviluppo dell'algoritmo e la sua chiusura con l'aggiunta di due piani di taglio.

Metodo Branch and Bound

Per la soluzione di problemi di programmazione intera di dimensioni reali non è praticabile un approccio di enumerazione totale o esplicita e quindi bisogna ricorrere ad un approccio di "enumerazione parziale" o "implicita", basato sulla suddivisione di un problema in due o più sottoproblemi.

L'insieme delle soluzioni ammissibili viene partizionato in due o più sottoinsiemi la cui intersezione è nulla e la cui unione costituisce l'insieme di partenza. Ciascuno di questi sottoinsiemi può contenere la soluzione ottima del problema originario e viene a sua volta partizionato. Essi costituiscono quindi la lista dei sottoinsiemi "candidati".

In un generico stadio della procedura si individua con un opportuno criterio un sottoinsieme della lista e si calcola un limite per il valore di funzione obiettivo relativo alle soluzioni in esso contenute. Si sviluppa quindi un'analisi volta a decidere se il sottoinsieme in questione deve essere ulteriormente partizionato per determinare soluzioni ammissibili del problema, oppure può essere eliminato da ulteriori considerazioni.

Problemi di PLI intera e binaria risolti con il *Branch and Bound*

Nel seguito si illustrano alcuni noti problemi di PLI intera e binaria risolubili con il metodo *Branch and Bound* :

- un problema generale di programmazione lineare con variabili intere,

- il problema del *Cutting Stock*, cioè del taglio ottimo di un materiale, che presenta variabili intere,

- il problema dello *Zaino*, cioè del caricamento ottimo di un contenitore, nelle formulazioni con variabili intere e con variabili binarie,

- il problema dell'*Assegnamento,* con variabili binarie.

Ciascuno dei problemi è caratterizzato da una particolare "combinazione" delle quattro fasi fondamentali del metodo (limite, partizione, eliminazione e ricerca), alle quali si aggiungono particolari accorgimenti e artifici, tipici del problema stesso.

Metodo Branch and Bound

L'approccio metodologico descritto per la soluzione di problemi di ottimizzazione intera costituisce il metodo *Branch and Bound*, cosiddetto perché basato su un meccanismo di partizione e ramificazione (*branch*) degli insiemi di soluzioni, e sul calcolo di un valore limite (*bound*) della funzione obiettivo.

Le componenti fondamentali del metodo *Branch and Bound*, sono sostanzialmente quattro:

1. determinazione di un valore limite (*bound*) di funzione obiettivo per un generico insieme candidato (limite superiore per un problema di massimizzazione, limite inferiore per un problema di minimizzazione);

2. partizione di un insieme candidato e scelta della variabile di *branching*;

3. determinazione del sottoinsieme candidato da esplorare;

4. eliminazione degli insiemi candidati che non è più necessario esplorare.

Queste componenti sono descritte in dettaglio nel testo di riferimento del corso.

Nel seguito se ne riporta una sintesi, di ausilio allo studio sul testo, con riferimento ad un problema di massimo.

Valore limite della funzione obiettivo

Per valutare e confrontare i sottoinsiemi di soluzioni è necessario determinare per ciascun sottoinsieme di soluzioni un limite superiore al valore ottimo di z. Esso è basato in generale sul rilassamento dei vincoli. Il valore ottimo di z per il "problema rilassato" costituisce un limite superiore (LS) per il valore ottimo di z del problema originario.

Rilassamento continuo

Si elimina il vincolo di interezza delle variabili e si risolve il problema di continuo. Il valore ottimo di z è sicuramente non minore del valore ottimo di z del problema intero e quindi costituisce per esso un limite superiore.

Rilassamento di vincoli del modello

Si rilassano uno o più vincoli del modello, quelli cosiddetti "difficili", e si risolve il problema con i vincoli rimanenti. Il problema è ancora intero, ma è più "facile" e quindi più agevole da risolvere sotto l'aspetto computazionale. Anche in questo caso si ottiene un limite superiore per il valore ottimo della funzione obiettivo originaria.

Rilassamento lagrangiano

Si costruisce una nuova funzione obiettivo, nota come *funzione lagrangiana*, il cui valore ottimo è non minore del valore di funzione obiettivo della soluzione ottima intera e quindi ne costituisce un limite superiore

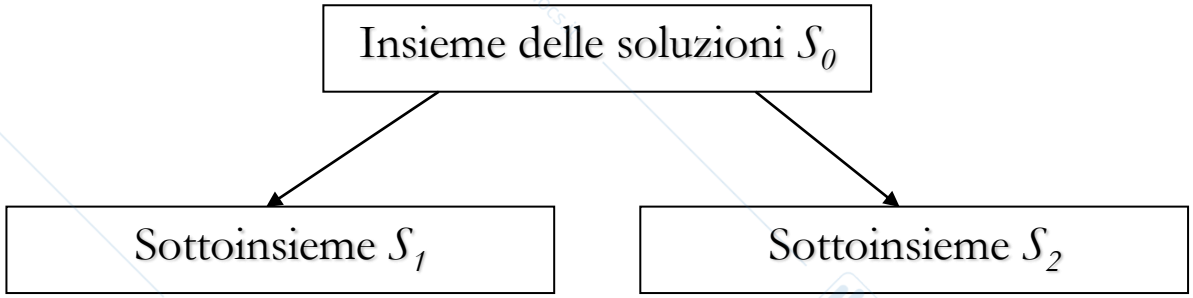
Partizione di un insieme

La partizione di un insieme si ottiene imponendo vincoli aggiuntivi al problema originario, per generare due o più sottoinsiemi disgiunti e tali che la loro unione costituisca l'insieme di partenza. A ciascun sotto insieme corrisponde un sottoproblema ($S_1 \cap S_2 = \varnothing$, $S_1 \cup S_2 = S_0$).

I criteri generalmente utilizzati per effettuare la partizione sono due:

- Partizione rispetto ad una variabile intera frazionaria
- Partizione rispetto ad una variabile binaria

Albero delle soluzioni



Partizione di un insieme e scelta della variabile di *branching*

Partizione rispetto a una variabile intera frazionaria

Se nella soluzione del problema rilassato una variabile x_j assume un valore frazionario $k.hh$ compreso tra k e k+1, con k intero, si possono generare due sottoproblemi, aggiungendo al problema di partenza i vincoli $x_j \leq k$ e $x_j \geq k+1$. La variabile x_j è la "variabile di *branching*".

Partizione rispetto a una variabile binaria

Questo criterio si applica quando nel problema candidato ci sono variabili binarie x_j. Aggiungendo ai vincoli di un problema il vincolo $x_j = 1$ oppure il vincolo $x_j = 0$ si generano due sottoproblemi e quindi due sottoinsiemi di soluzioni. La variabile x_j è la "variabile di *branching*".

Scelta della variabile di branching

Per effettuare la partizione é necessario quindi scegliere la variabile frazionaria rispetto alla quale effettuare la partizione. Un criterio può essere quello di scegliere la variabile con il valore frazionario più vicino al valore intero.

Determinazione del sottoinsieme da esplorare

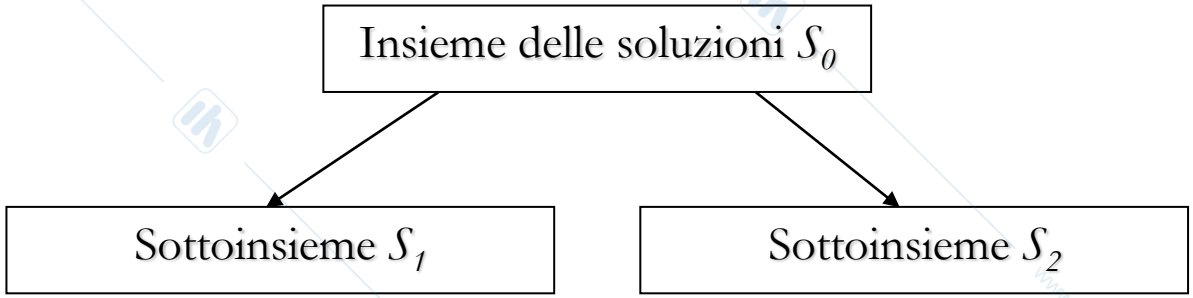
All'inizio della procedura l'insieme delle soluzioni intere del problema originario è l'unico insieme candidato. Ad esso si applica il metodo per la determinazione del limite. Se la soluzione corrispondente è ammissibile essa costituisce la soluzione ottima del problema e l'algoritmo termina, altrimenti si applica il criterio di partizione all'insieme originario. Per determinare quale degli insiemi generati deve essere esplorato è necessario definire una strategia di ricerca.

Le principali strategie di ricerca sono la strategia ***Best-first*** e la strategia ***Depth-first.***

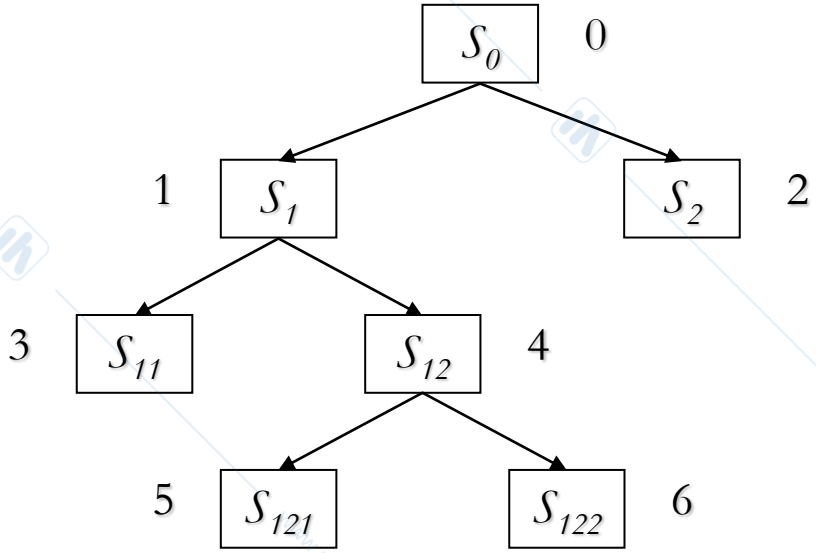
La strategia ***Best-first*** è la "strategia del limite migliore" (*best first* o *best bound*), perché si sceglie il sottoinsieme che presenta il limite migliore (il più alto per un problema di massimo e il più basso per un problema di minimo). Ad esso applica il criterio di partizione e la determinazione del limite superiore per i sottoinsiemi generati.

La strategia ***Depth-first*** esplora uno solo dei sottoinsiemi generabili con la partizione e ad esso applica la determinazione del limite superiore. L'albero di ricerca si sviluppa in profondità e quindi la strategia di ricerca viene definita "*depth-first*" o "*newest-bound*" (limite più recente).

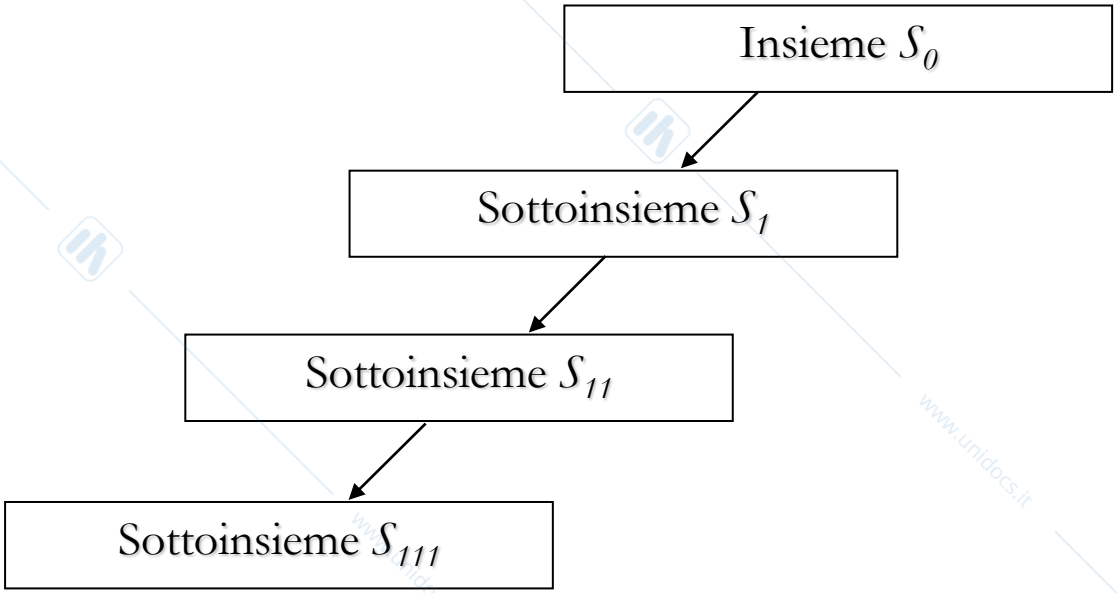
Albero delle soluzioni
ricerca best first



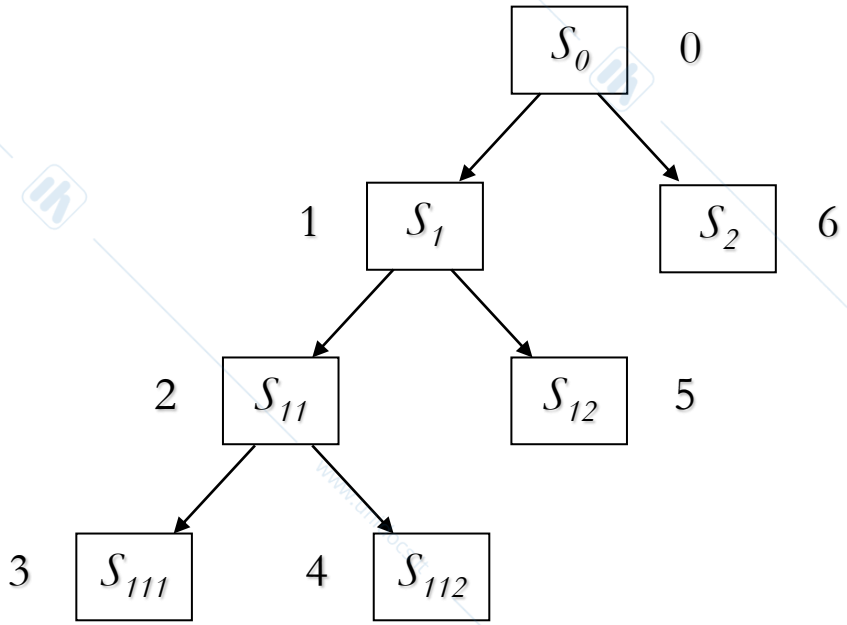
Albero di ricerca *best-first*



Albero di ricerca *depth-first*

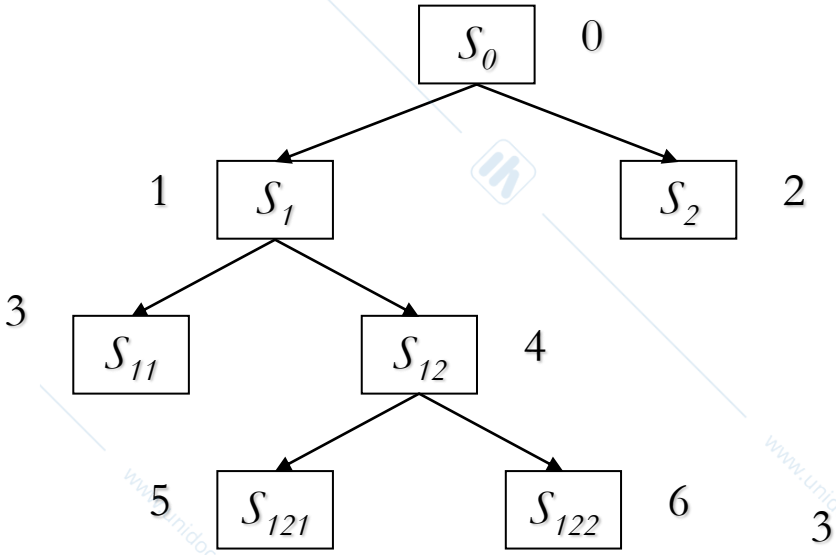


Albero di ricerca *depth-first*

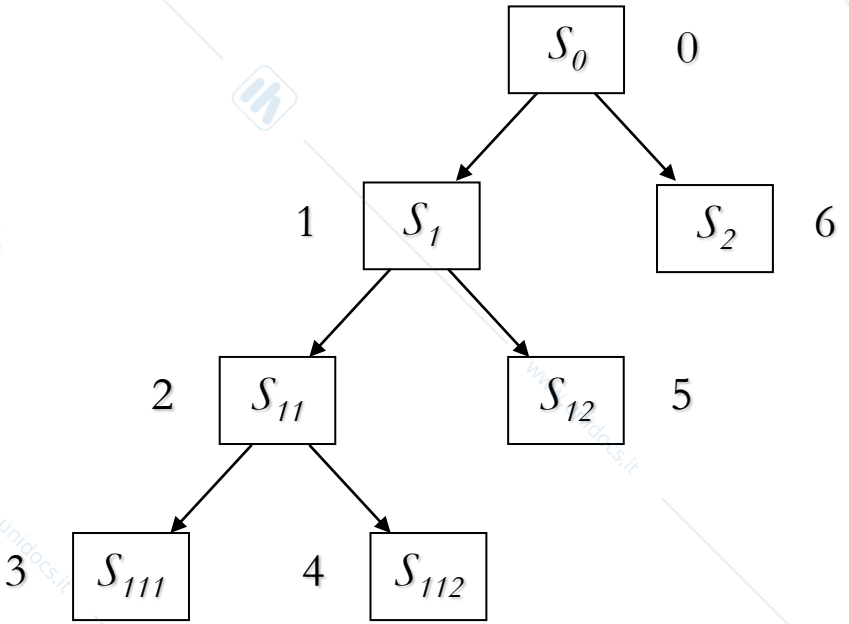


Albero di ricerca

best-first



depth-first



Test di eliminazione

Nella procedura di enumerazione parziale del *Branch and Bound* è fondamentale definire criteri di eliminazione degli insiemi di soluzioni dalla lista degli insiemi candidati. In assenza di eliminazioni l'albero di ricerca sarebbe generato per intero e si avrebbe una enumerazione totale delle soluzioni.

Test di inammissibilità

Dall'esame di un sottoinsieme candidato si può dedurre che esso non può contenere soluzioni ammissibili.

Test di ammissibilità

Se la soluzione corrispondente al limite superiore LS calcolato per un sottoinsieme soddisfa tutti i vincoli del problema originario (compresi i vincoli di interezza) è una soluzione ammissibile ed è proprio la soluzione ottima del sottoproblema in esame che viene pertanto eliminato dalla lista. La soluzione ottima ad esso corrispondente viene memorizzata come soluzione ottima corrente ed il relativo valore di z diventa un limite inferiore LI per il valore di funzione obiettivo del problema originario.

Test di limite

Si confronta il limite superiore LS_k di un sottoproblema k con il limite inferiore generale LI. Poiché $z^*_k \leq LS_k$, se si verifica $LS_k \leq LI$ sarà anche $z^*_k \leq LI$ e dunque il sottoinsieme k può essere eliminato.

Se non si verifica nessuna di queste condizioni il sottoinsieme esaminato viene a sua volta partizionato in due o più sottoinsiemi, che vengono aggiunti alla lista degli insiemi candidati, e si ripete la scelta di un insieme candidato da esplorare. La procedura si itera fino a quando non ci sono più sottoinsiemi da esplorare. La migliore soluzione ammissibile trovata costituisce la soluzione ottima del problema originario.

Problemi di Ottimizzazione Intera

Il problema del Cutting Stock

Problemi di ottimizzazione intera

In questa lezione si illustrano alcuni classici problemi di ottimizzazione intera:

- il problema del *Cutting Stock*, cioè del taglio ottimo di un materiale, che presenta variabili intere,
- il problema dello *Zaino*, cioè del caricamento ottimo di un contenitore, con variabili intere e con variabili binarie,
- il problema dell'*Assegnamento*, cioè dell'assegnamento ottimo addetti - mansioni, con variabili binarie.

Cutting Stock

Cutting Stock

Il problema del *Cutting Stock* è nato in ambito industriale per le decisioni relative a lavorazioni di taglio di un semilavorato monodimensionale, cioè con una dimensione prevalente (tubi, aste di legno . . .) o bidimensionale, con una superficie (lamiere, lastre di vetro . . .), per ottenere item di dimensioni ridotte, prefissate.

Ha trovato poi applicazione anche nel settore dell'informazione per la partizione ottima delle risorse di memoria e di calcolo.

Nel seguito si farà riferimento al caso monodimensionale in ambito industriale.

Cutting Stock

L'operazione di taglio richiede preliminarmente di determinare un insieme di possibili *schemi di taglio.*

Uno schema di taglio è una combinazione del numero di pezzi che si possono realizzare per tutti i prodotti, a partire da un'asta, con una singola operazione di taglio.

Nel taglio si possono produrre scarti che generano un costo. In tabella è riportato lo scarto associato a ciascuno schema di taglio.

Tra tutti gli schemi generati si devono individuare quali schemi adottare ed in quale numero per ciascuno di essi.

In generale il numero di possibili schemi di taglio può essere elevato. Nel seguito si presenta un semplice problema con 2 schemi di taglio e poi un problema con 4 e 23 schemi di taglio

Gli obiettivi possono essere:

- la minimizzazione dello scarto totale,

- la minimizzazione del numero di aste utilizzate

- la minimizzazione della sovrapproduzione di pezzi.

Cutting Stock

Definiti n schemi di taglio, bisogna determinare quali schemi bisogna adottare per produrre i pezzi richiesti per ciascun prodotto e quante volte bisogna applicare ciascuno degli schemi individuati.

A tale scopo si può costruire un modello in programmazione intera nel quale la variabile decisionale x_j ($j = 1,\ldots, n$) rappresenta il numero di volte che lo schema di taglio j deve essere ripetuto. Se risulta $x_j = 0$ lo schema j non deve essere utilizzato.

Vincoli

Sia a_{ij} il numero di pezzi del prodotto i ottenibili con lo schema di taglio j.

Pertanto $\sum_{j=1,n} a_{ij} x_j$ rappresenta il numero di pezzi del prodotto i realizzati con tutti gli schemi di taglio.

Se b_i è la quantità richiesta di prodotto i si costruisce per ciascun prodotto un vincolo di $\geq$ che impone un limite inferiore al numero di pezzi da produrre per ciascun prodotto i (A, B, C, D):

$$\sum_{j=1,n} a_{ij} x_j \geq b_i \quad i = 1,\ldots, m$$
$$x_j \geq 0 \text{ intera} \qquad j = 1,\ldots, n$$

Cutting Stock

Funzione obiettivo

Nel problema del cutting stock è possibile definire più obiettivi dell'ottimizzazione.

- ***Minimizzazione dello scarto totale***, ovvero, se c_j è lo scarto relativo allo schema j:

$$\text{Min } z = \sum_{j=1,n} c_j x_j$$

- ***Minimizzazione del numero totale di aste di legno*** necessarie per realizzare la produzione. Poiché in ogni taglio si utilizza un'asta di legno, il numero di aste necessarie è pari al numero totale di tagli da effettuare:

$$\text{Min } z = \sum_{j=1,n} x_j$$

- ***Minimizzazione del numero totale dei pezzi sovraprodotti:***

$$\text{Min } z = \sum_{i=1,m} \left(\sum_{j=1,n} a_{ij} x_j - b_i \right)$$

Il termine all'interno della parentesi è in effetti la variabile *slack* dell'i-esimo vincolo di ≥. La minimizzazione del numero totale di pezzi sovraprodotti equivale quindi alla minimizzazione della somma delle variabili *slack*.

Problemi di Ottimizzazione Intera

Il problema dello Zaino

Problema dello zaino

Il problema dello zaino (*Knapsack Problem*) è un classico problema di ottimizzazione combinatoria che può essere così schematizzato:

- è necessario caricare uno "zaino", cioè un contenitore, che può sopportare al massimo un peso p_{max} ;
- si hanno a disposizione n prodotti, per ciascuno dei quali si conoscono peso e valore dell'unità di prodotto, e numero di unità disponibili per ciascuno di essi.
- si vuol determinare quali, e quanto, di questi prodotti bisogna caricare per massimizzare il valore totale dello zaino, rispettando il vincolo di peso massimo e la condizione di indivisibilità di ciascuna unità di prodotto.

- Nel seguito si opera la distinzione tra due casi:
- per ciascun prodotto sono disponibili più unità di prodotto
- per ciascun prodotto è disponibile una sola unità.

Nel primo caso si formula un modello con variabili intere, nel secondo caso un modello con variabili binarie.

Problema dello zaino intero

Dati del problema:

- i pesi unitari dei prodotti $p_1,\ldots,p_n$
- il valori unitari dei prodotti $v_1,\ldots,v_n$
- le disponibilità dei prodotti $k_1,\ldots,k_n$
- il peso massimo $Pmax$ $Pmax \geq p_j \quad j = 1, \ldots, n$

La variabile x_j intera rappresenta il numero di unità del prodotto j.

$$\text{Max } z = \sum_{j=1,n} v_j x_j$$

$$\text{s.a } \sum_{j=1,n} p_j x_j \leq P_{max}$$
$$x_j \leq k_j \qquad j = 1,\ldots,n$$
$$x_j \geq 0 \text{ intera} \qquad j = 1,\ldots,n$$

Questo modello si risolve con il metodo Branch and Bound per la PLI

Problema dello zaino binario

Se è disponibile una sola unità per ogni prodotto, il modello assume la seguente configurazione:

$$\text{Max } z = \sum_{j=1,\,n} v_j x_j$$

$$\text{s. a } \sum_{j=1,\,n} p_j x_j \leq pmax$$

$$x_j = \{0, 1\}\ j = 1, \ldots, n$$

Questo modello può essere risolto con un algoritmo *Branch and Bound* particolarizzato per il caso binario.

Problemi di Ottimizzazione Intera

Il problema dell'Assegnamento

Il problema dell'Assegnamento

Il problema dell'assegnamento è un classico problema di ottimizzazione combinatoria per il quale si può formulare un modello di PLI 0/1. Il problema può essere schematizzato nel modo seguente:

- è necessario assegnare n addetti a n mansioni;

- ciascun addetto può essere assegnato ad una sola mansione e ciascuna mansione può essere svolta da un solo addetto;

- ciascun addetto i è in grado di svolgere ciascuna mansione j con un costo di assegnamento c_{ij}.

- è nota la matrice dei costi di assegnamento c_{ij} positivi:

$\boldsymbol{C} = \{c_{ij}\}$ di dimensioni $[n \times n]$.

Si vuole determinare l'insieme di accoppiamenti addetto-mansione che realizzi il costo di assegnamento minimo.

Si può associare una variabile x_{ij} binaria a ciascun accoppiamento ij :

$x_{ij} = 1$ se l'addetto i è assegnato alla mansione j, $x_{ij} = 0$ altrimenti

Il problema dell'Assegnamento

Step dell'algoritmo euristico

1. Si sceglie una riga della matrice $\boldsymbol{C}$ e si individua l'elemento c_{ij} di costo minimo
2. Si pone $x_{ij} = 1$ e si elimina la riga i e la colonna j nella matrice $\boldsymbol{C}$
3. Se la matrice $\boldsymbol{C}$ residua contiene almeno un elemento si torna al passo 1, altrimenti l'algoritmo termina.

La qualità della soluzione determinata dipende dal criterio che si adotta per la scelta dell'elemento ij della matrice $\boldsymbol{C}$ al passo 1.

Proprietà della matrice A

La matrice ***A*** dell'assegnamento gode della proprietà di unimodularità totale.

Inoltre i termini noti dei vincoli sono pari ad 1 e quindi interi.

Si può dimostrare che:
in virtù della proprietà di unimodularità totale di ***A*** e della interezza dei termini noti, la soluzione ottima del modello di assegnamento risulta essere sempre intera.

Pertanto se il vincolo $x_{ij} = 0/1$ viene sostituito dai vincoli $x_{ij} \leq 1$, per determinare la soluzione esatta del problema dell'assegnamento si può usare l'algoritmo del Simplesso invece di un algoritmo per l'ottimizzazione intera.

E' agevole verificare che la soluzione del modello di assegnamento è fortemente degenere.

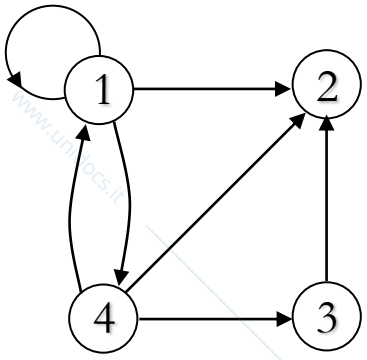
Elementi di Teoria dei Grafi

Grafo parziale

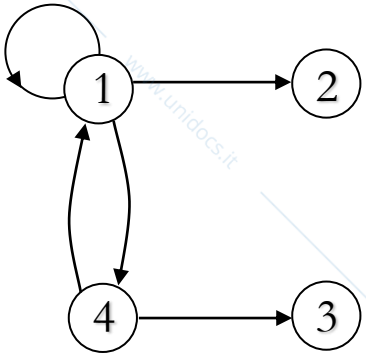
Sottografo

Dato un grafo orientato G si dirà che G' è un *grafo parziale* di G se esso è costituito dagli stessi vertici e da un sottoinsieme degli archi di G.

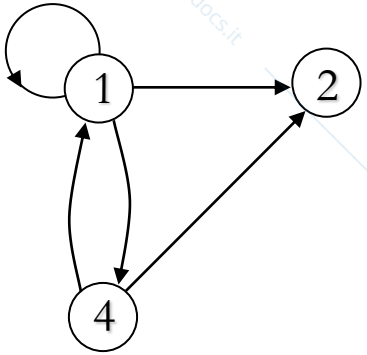
Dato un grafo orientato G si dirà che $G^{\wedge}$ è un *sottografo* di G se esso è costituito da una parte dei vertici di G e da tutti gli archi di G ad essi relativi.



Grafo orientato G



G': Grafo parziale di G

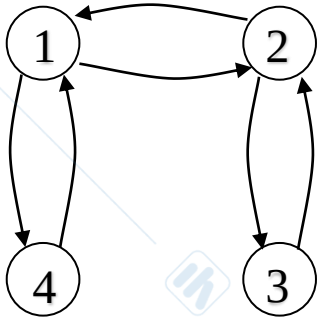


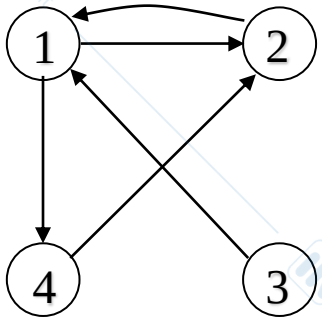
G^ : Sottografo di G

Grafo simmetrico

Grafo non simmetrico.

Un *grafo* si dice *simmetrico* se per ogni arco (v_i, v_j) esiste il suo simmetrico (v_j, v_i), in caso contrario si dice *non simmetrico*. Si dice *antisimmetrico* se per ogni arco (v_i, v_j) non esiste il suo simmetrico.

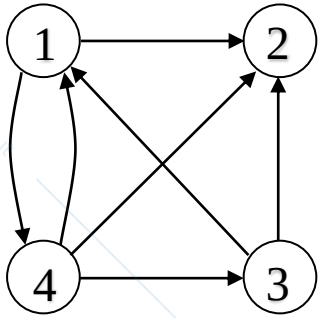


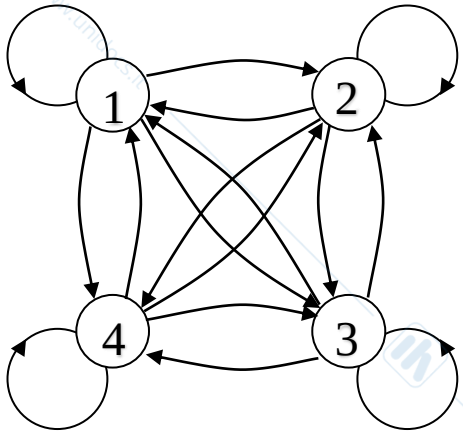


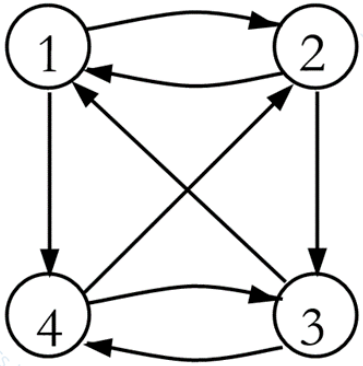
Grafo completo

Grafo pieno

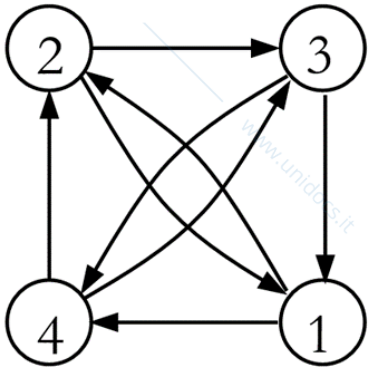
Un grafo si dice *completo* se esiste almeno un arco per tutte le coppie di vertici individuabili sul grafo. Un grafo si dice *pieno* se esiste un arco per tutte le coppie ordinate di vertici del grafo.





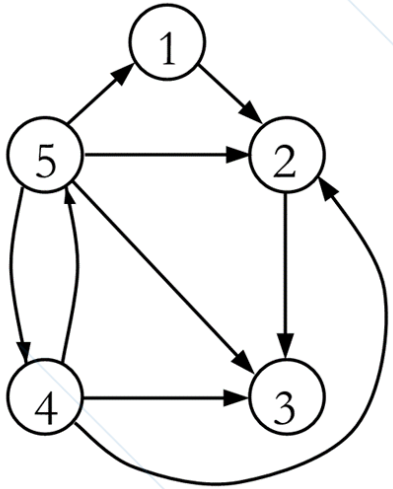


Grafi *isomorfi*

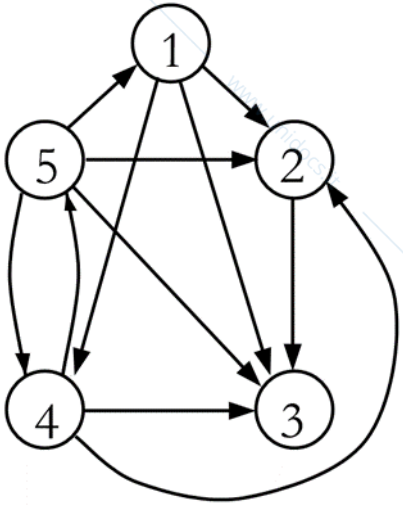


Due grafi si dicono *isomorfi* se presentano le medesime relazioni tra i vertici indipendentemente dalla posizione dei vertici e degli archi.

Un grafo si dice *planare* se esso o un suo isomorfo gode della proprietà che due archi non si toccano mai in punti diversi dai vertici, si dice *non planare* se non gode di questa proprietà.



Grafo *planare*

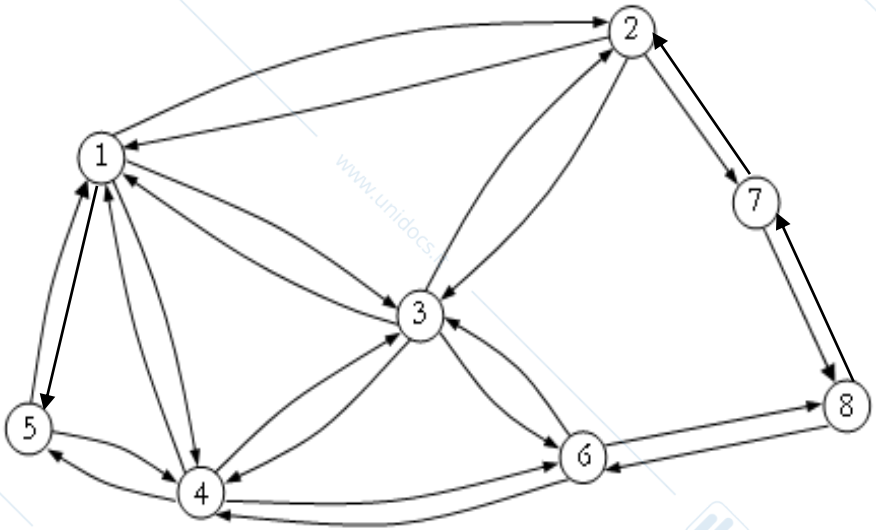


Grafo *non planare*

Percorso

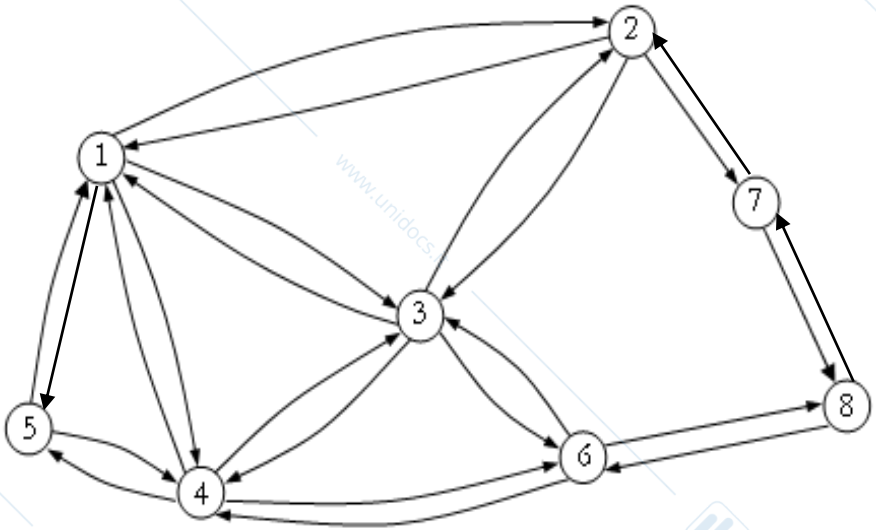
Si definisce *percorso* (o *cammino*) di un grafo orientato, da un vertice origine *o* ad un vertice destinazione *d*, una sequenza finita di archi $P = \{a_1,\ldots, a_r\}$ tale che, per ciascun $k = 1,\ldots, r-1$, il vertice destinazione dell'arco a_k coincide con il vertice origine dell'arco a_{k+1}. In figura, per esempio, 1-2-3-6-8. Si definisce *ordine* di un percorso il numero di archi che lo compongono.

Un grafo si dice *fortemente connesso* se per ciascuna coppia ordinata di vertici *o-d* esiste almeno un percorso che li congiunge. Il grafo in figura è fortemente connesso.



Percorso

Un percorso è detto *semplice* se attraversa una sola volta ciascuno degli archi che lo compongono (in figura 1-2-7-8, ma anche 1-2-3-2-7-8) altrimenti si dice *composto* (in figura 1-4-6-3-4-6-8).
Un percorso è detto *elementare* se attraversa una sola volta ciascuno dei suoi vertici (1-4-3-6-8), altrimenti si dice *non elementare* (1-4-3-4-6-8). Un percorso semplice non è necessariamente elementare. Un percorso elementare è anche semplice. Un percorso composto non può essere elementare.
Un percorso *o-d* si dice *hamiltoniano* se tocca tutti i vertici del grafo, ciascuno una sola volta (5-1-4-6-3-2-7-8). Un percorso di dice *euleriano* se attraversa tutti gli archi del grafo, ciascuno una sola volta.



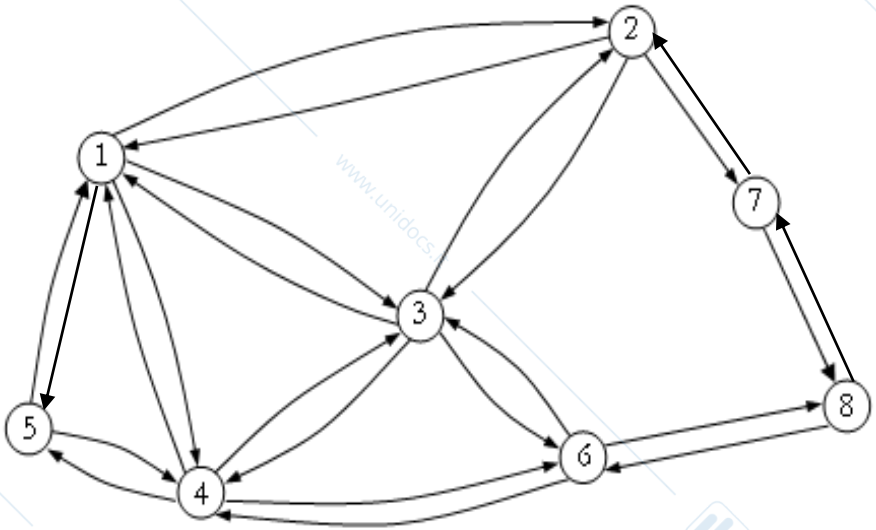
Circuito

Si definisce *circuito* un percorso *P* tale che il vertice destinazione del percorso coincida con il vertice origine (1-2-3-4-1 in figura)

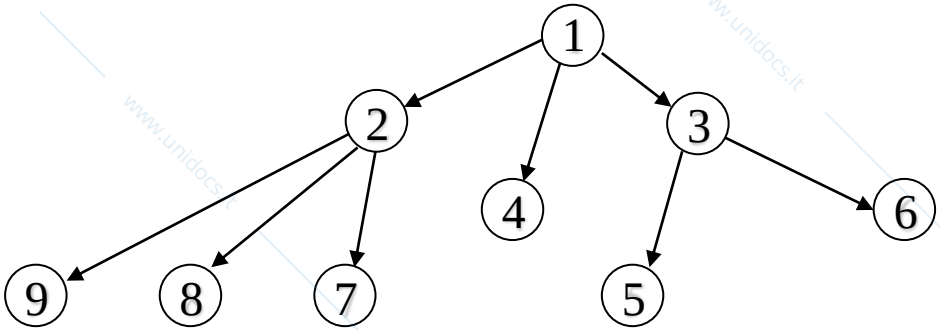
Un circuito si dice *semplice* se ciascuno degli archi che lo compongono é percorso una sola volta (1-2-3-4-1 in figura).

Un circuito si dice *hamiltoniano* se tocca tutti i vertici del grafo, ciascuno una sola volta (1-2-7-8-6-3-4-5-1 in figura).

Un circuito si dice *euleriano* se passa per tutti gli archi del grafo, ciascuno una sola volta.



Arborescenza

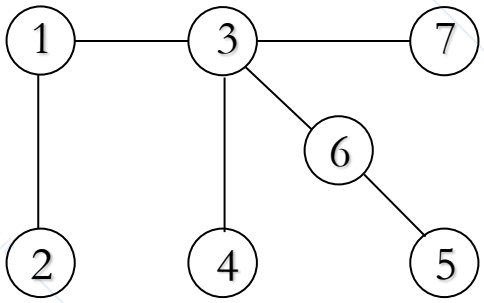


Una arborescenza è un grafo orientato nel quale:

- c'è un solo vertice che è solo origine di archi (vertice radice);
- ciascuno degli altri vertici è destinazione di un solo arco e può essere origine di uno o più archi. Se non è origine di alcun arco è detto foglia dell'arborescenza.

L'arborescenza ha v vertici e $v-1$ archi e non presenta circuiti.

Albero



Per ogni coppia di vertici esiste un solo percorso. L'aggiunta di un arco tra due vertici non adiacenti crea una maglia. L'eliminazione di un arco fa perdere la connessione.

Un *albero* è un grafo non orientato connesso e senza cicli. Ha v vertici e $v-1$ spigoli. È consolidato l'uso di utilizzare il termine *albero* anche per un'arborescenza e in generale per un grafo connesso orientato e senza circuiti, con v vertici e $v-1$ archi comunque orientati (dunque non necessariamente una arborescenza).

Visita di un grafo

Con l'espressione *visita di un grafo* si intende una procedura di raggiungimento sistematico di tutti i vertici del grafo a partire da un vertice individuato come origine della visita.

La visita di un grafo è variamente utilizzata in molti contesti applicativi, sia in maniera esplicita, quando il grafo è rappresentativo di un sistema nel quale è necessario raggiungere tutti i vertici, sia in maniera implicita, all'interno di alcuni algoritmi di ottimizzazione su rete (per esempio nei problemi di minimo percorso e massimo flusso) al fine di evitare la enumerazione esaustiva delle soluzioni con tecniche di etichettamento che operano in modo contestuale alla "visita" dei vertici stessi.

Le tecniche di visita possibili sono fondamentalmente due, descritte nel seguito: visita in larghezza e visita in profondità.

La visita di un grafo genera un albero, che viene detto *albero di visita*. Su quest'albero è possibile identificare un percorso dall'origine a ciascuno degli altri vertici.

Costo di un percorso

Il costo su un percorso tra o e d viene in generale espresso come somma dei costi (costanti o variabili) associati agli archi che compongono il percorso.

Indicando con p il percorso, con $C_{p,od}$ il costo del percorso p tra o e d e con P_{od} l'insieme dei percorsi p tra o e d, si può scrivere:

$\forall p \in P_{od} \quad C_{p,od} = \sum_{ij \in p} c_{ij}$ nel caso di costi costanti

oppure $C_{p,od} = \sum_{ij \in p} c_{ij}(f_{ij})$ nel caso di costi variabili

Problemi di percorso

Problemi di percorso

I problemi di determinazione di percorsi ottimi sono tra i più noti problemi di ottimizzazione su rete. Si incontrano frequentemente in numerosi campi applicativi, nella gestione del traffico e nella pianificazione dei trasporti, nell'ingegneria dell'informazione e della comunicazione, nell'ingegneria industriale.

In questa lezione, dopo la classificazione dei problemi e degli algoritmi di minimo percorso, si formula il modello relativo ad una singola coppia origine/destinazione (o/d) e vengono descritti i principali algoritmi utilizzabili per determinare il minimo percorso da un vertice o a tutti gli altri vertici, oppure per tutte le coppie di vertici o/d.

Nel contesto dei problemi di minimo percorso vengono brevemente introdotti il problema del k-esimo minimo percorso ed i problemi di minimo percorso vincolato.

Il modello del minimo percorso per una singola coppia o-d

Il problema può essere formulato quindi attraverso il seguente modello in programmazione intera binaria, nel quale $\sum_i$ sta per $\sum_{i:ij \in A}$:

$$\text{Min } z = \sum_{ij \in A} c_{ij}\, x_{ij}$$

$$\text{s.a} \quad \sum_k x_{ok} = 1$$

$$\sum_i x_{id} = 1$$

$$\sum_k x_{jk} - \sum_i x_{ij} = 0 \qquad (\forall j \neq o,d)$$

$$x_{ij} = 0/1, \ \forall ij \in A$$

Le soluzioni corrispondenti ai percorsi riportati in tabella rispettano questi vincoli. In effetti anche altre soluzioni, corrispondenti ai percorsi non semplici e/o non elementari, soddisfano questi vincoli, ma hanno sicuramente un valore di funzione obiettivo maggiore di quello dei percorsi semplici ed elementari e quindi vengono scartate dall'algoritmo

Algoritmi di minimo percorso per i problemi di classe b e c

I problemi delle classi (b) e (c) vengono risolti mediante algoritmi "ad hoc", disponibili in gran numero. La distinzione che più di frequente si adotta è quella tra ***algoritmi arborescenti*** e ***algoritmi matriciali***.

Gli **algoritmi arborescenti** si definiscono in questo modo perché una singola implementazione dell'algoritmo genera l'arborescenza (l'albero) dei minimi percorsi da un vertice origine a tutti gli altri vertici assunti come destinazione (classe (b)). La costruzione dell'arborescenza dei minimi percorsi con origine nel vertice *i* generico fornisce la riga *i*-esima della matrice ***C*** delle distanze minime. L'applicazione di un algoritmo arborescente per ciascun vertice della rete assunto come origine genera tutta la matrice ***C***.

Gli **algoritmi matriciali** risolvono la classe (c) dei problemi e dunque forniscono in una singola implementazione la matrice ***C*** dei costi minimi per tutte le coppie di vertici individuabili sulla rete.

Non c'è una rigida corrispondenza tra classe di problemi e algoritmi idonei a risolverli. È possibile infatti applicare un algoritmo arborescente per ciascuno dei vertici origine della rete per risolvere il problema (c) del cammino minimo per tutte le coppie di vertici, così come sarebbe possibile applicare un algoritmo matriciale ed utilizzare solo parte dei risultati ottenuti per dare risposta ad un problema del tipo (a) o (b).

Algoritmo di Dantzig

Algoritmo di Dantzig

L'algoritmo di Dantzig è un algoritmo arborescente del tipo *label setting*. Sia o il vertice origine. Si supponga di conoscere al $k°$ stadio della procedura i cammini minimi dal vertice o a k vertici. Si indichi questo insieme di vertici con S. Poiché gli algoritmi arborescenti generano progressivamente un albero di minimi percorsi, i vertici di S sono collegati in modo da costituire un albero parziale. L'insieme V dei vertici può essere dunque suddiviso in due sottoinsiemi: S, contenente i vertici etichettati definitivamente con il valore di minimo costo dall'origine o, e S', contenente i vertici per i quali il minimo percorso non è stato ancora determinato e quindi non hanno ancora una etichetta definitiva. Si indichi con:

i	il generico vertice foglia in S;
δ_i	la distanza minima di i dall'origine o;
j_i	il vertice più vicino ad i, non in S, collegato ad i da un arco i, j_i;
a_i	la lunghezza dell'arco i, j_i .

Tra tutti i vertici j_i si inserirà in S il vertice j_r tale che $\delta_r + a_r = \text{Min}_{i=1,k}\ (\delta_i + a_i)$.
Questo valore fornisce il valore minimo del percorso da o a j_r in quanto ogni altro percorso da o a j_r utilizzerebbe un vertice j_t non in S, avente una distanza minima da o pari a $\delta_t + a_t > \delta_r + a_r$. A maggior ragione quindi si verificherebbe $\delta_t + a_t + a_{t,r} > \delta_r + a_r$.

Questo *step* fondamentale dell'algoritmo si itera inserendo ogni volta uno o più nuovi vertici nell'insieme S. L'algoritmo termina quando tutti i vertici della rete fanno parte di S.

Algoritmo di Dijkstra

L'algoritmo di Dijkstra assegna a ciascuno dei v vertici del grafo dei valori di tentativo, che costituiscono dei limiti superiori al valore del cammino minimo dal vertice origine ad essi.

Si assegna inizialmente al vertice origine il valore definitivo 0 ed a tutti gli altri vertici il valore di tentativo ∞. Si parte dall'origine e si va negli altri vertici calcolando il valore del costo del percorso come somma dell'etichetta dell'origine e del costo dell'arco utilizzato (∞ qualora l'arco non esista). Tra i v–1 valori di tentativo ottenuti si sceglie il minore e lo si assume come etichetta definitiva del vertice cui corrisponde. A differenza di quanto avviene nell'algoritmo di Dantzig, in ogni iterazione ci si muove dall'ultimo vertice etichettato definitivamente e non da tutto l'insieme dei vertici etichettati definitivamente. Si supponga che sia k il vertice etichettato definitivamente nel passo precedente. Si riparte dal vertice k e si va negli altri vertici non ancora etichettati definitivamente, calcolando il valore del costo del percorso come somma dell'etichetta del vertice k e del costo dell'arco utilizzato (∞ qualora l'arco non esista). Per ciascun vertice si confronta il valore così calcolato con l'etichetta precedente ed il valore minimo viene assunto come nuovo valore di tentativo. Tra i v–2 valori di tentativo ottenuti si sceglie il minore e lo si assume come etichetta definitiva del vertice cui corrisponde, per esempio j, assumendolo come punto di partenza per un ulteriore *step* della procedura. Dopo un numero di iterazioni pari a quello dei vertici destinazione le etichette diventano tutte definitive e forniscono i valori dei cammini minimi dal vertice origine a tutti gli altri. Questo algoritmo può essere definito un algoritmo arborescente del tipo *label setting-correcting*. La slide successiva mostra il riepilogo delle operazioni effettuate sulla rete già utilizzata per l'algoritmo di Dantzig.

Algoritmi matriciali

Il calcolo della matrice ***C*** dei minimi percorsi può essere effettuato utilizzando un algoritmo di tipo matriciale, in grado di calcolare tutti gli elementi della matrice delle distanze minime.

ALGORITMO DI FLOYD

L'algoritmo matriciale più comunemente usato è l'algoritmo di Floyd. Esso lavora su una matrice ***C***, di dimensioni pari al numero di vertici. Tale matrice, inizialmente coincidente con la matrice di adiacenza vertice-vertice, viene modificata nelle successive iterazioni. La *k*-esima di tali matrici può essere interpretata come quella che fornisce il costo dei cammini minimi per tutte le coppie di vertici della rete, caratterizzati dalla proprietà di utilizzare soltanto i vertici numerati da 1 a *k*. Alla fine del procedimento la matrice contiene quindi i valori dei cammini minimi tra tutte le coppie di vertici del grafo che utilizzano tutti i vertici del grafo. Un algoritmo matriciale è quindi un algoritmo *label correcting*. Nelle slide successive si riportano gli step dell'algoritmo di Floyd, dalla matrice di adiacenza iniziale alla matrice dei costi minimi finale

Problemi di percorso vincolato

In questa slide si introducono alcuni problemi di minimo percorso vincolato: attraverso specificati vertici, con finestre temporali, con risorse limitate. Questi problemi, di grande interesse applicativo, intervengono quando è necessario determinare uno o più percorsi o circuiti per instradare persone, merci, dati o informazioni da un insieme di impianti, sedi del flusso in partenza, a un insieme di destinazioni nelle quali il flusso generato è richiesto.

Minimo percorso attraverso specificati vertici

In questo problema è necessario determinare il minimo percorso per una coppia di vertici *(o-d)* che passi per tutti o parte dei vertici della rete in una sequenza non prefissata.

Minimo percorso con finestre temporali

Questo problema si pone quando a ciascun nodo della rete è associato l'intervallo temporale del tempo di arrivo e partenza nel nodo stesso. In altri termini questo intervallo definisce la finestra temporale nella quale è possibile attraversare il nodo, se esso fa parte del percorso.

Minimo percorso con risorse limitate

In questo problema si deve determinare un minimo percorso o-d che utilizza risorse (disponibilità finanziaria, numero di addetti, carico di un veicolo) sottoposte a vincoli di disponibilità sugli archi.

Per la soluzione del problema generale sono stati proposti in letteratura numerosi algoritmi, per i quali si rimanda al corso di Ricerca Operativa 2 per la laurea magistrale.

Tecniche Reticolari per la Programmazione, Gestione e Controllo di un Progetto

P.E.R.T.
(Program Evaluation and Review Technique)

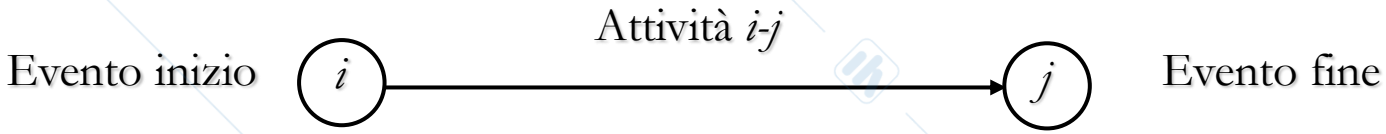
L'approccio alla gestione di un progetto, noto con il termine P.E.R.T. (*Program Evaluation and Review Technique*), è stato sviluppato negli anni '50.
La metodologia PERT assume che il tempo sia il parametro fondamentale del processo gestionale del progetto e che siano note le durate delle singole attività del progetto.
Le forme di rappresentazione reticolare di un progetto sono due:
- rappresentazione "attività – arco"
- rappresentazione "attività – nodo".

Negli anni successivi sono state sviluppate tecniche orientate alla valutazione dei costi e delle risorse disponibili, descritte in dettaglio nei testi specialistici di *Project Management*. Successivamente, nel contesto delle problematiche di gestione delle risorse limitate, hanno assunto rilievo le tecniche orientate alla schedulazione delle risorse disponibili per l'attuazione di un progetto (*Resource Allocation and Scheduling*).

In ultima analisi, i parametri rilevanti che è necessario analizzare per la gestione di un progetto sono la durata, il costo e le risorse impiegate per lo sviluppo delle attività che costituiscono il progetto stesso. Le procedure di calcolo disponibili sono basate su metodi di ottimizzazione intera e su metodi euristici.

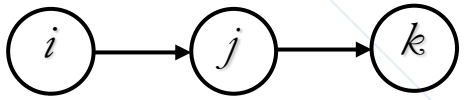
Rappresentazione attività - arco

Ciascuna attività del progetto è rappresentata da un arco, con un nodo origine (evento inizio dell'attività) ed un nodo destinazione (evento fine dell'attività).



In questo modo si possono rappresentare le sequenze di attività e le relazioni di precedenza e successione.

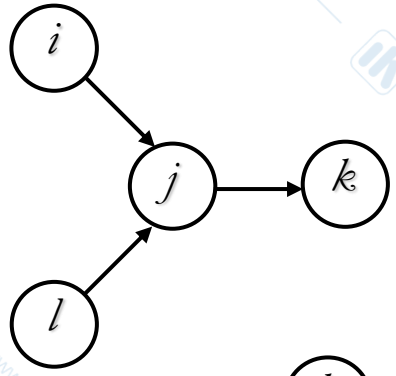
Nella figura l'attività *i-j* precede l'attività *j-k* e deve terminare prima che *j-k* possa iniziare.



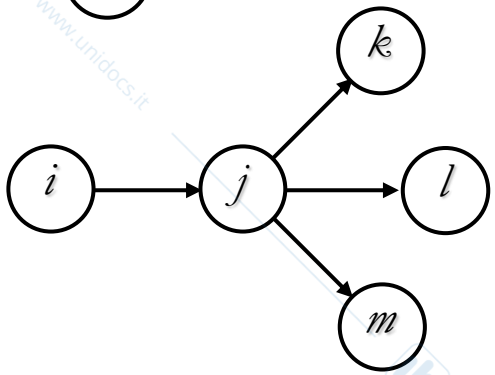
Costruzione della rete PERT

Sulla base della corrispondenza tra attività e arco si possono rappresentare le relazioni tra le attività.

Nella figura le attività *i-j* e *l-j* precedono l'attività *j-k* e devono terminare prima che *j-k* possa iniziare.

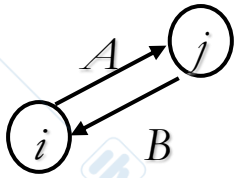


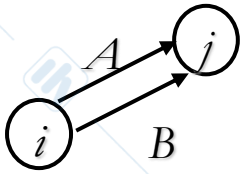
Nella seconda figura l'attività *i-j* precede le attività *j-k*, *j-l* e *j-m* e deve terminare prima esse possano iniziare.



Costruzione della rete PERT

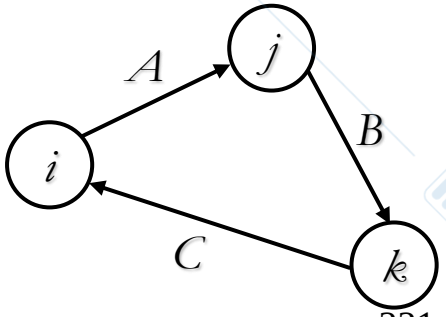
Non è possibile avere archi doppiamente orientati. Nel caso in figura l'evento i è l'evento-inizio dell'attività i-j. Quindi non può essere evento-fine di un'attività che inizia in j. Inoltre non è possibile avere due o più archi in parallelo tra due nodi.





No!

Non è possibile avere circuiti. Nel caso in figura l'attività A precede l'attività B, che precede l'attività C, che precede l'attività A, situazione palesemente assurda.



Determinazione della durata del progetto e del percorso critico

L'algoritmo si divide in due passi, tradizionalmente indicati come *forward step* e *backward step*.

FORWARD STEP

In questo *step* si calcola il tempo al più presto di ogni evento, la durata del progetto ed il percorso "critico" del progetto.

La durata totale del progetto corrisponde alla sequenza di archi che presenta il tempo di esecuzione maggiore per andare dall'evento iniziale all'evento finale del progetto, dunque è la durata del percorso massimo dal nodo origine al nodo destinazione della rete, che si definisce percorso critico. La sua determinazione si effettua dunque con un algoritmo di percorso massimo.

BACKWARD STEP

In questo *step* si calcola il tempo al più tardi di ogni evento, necessario per calcolare gli scorrimenti degli eventi e delle attività.

Determinazione della durata del progetto e del percorso critico

Si indichi con:

t_{ij} la durata dell'attività ij

$tpp(i)$ il tempo al più presto dell'evento i
$tpt(i)$ il tempo al più tardi dell'evento i

s_i lo scorrimento dell'evento i

$tipp(ij)$ il tempo di inizio al più presto per l'attività ij
$tipt(ij)$ il tempo di inizio al più tardi per l'attività ij

$tfpp(ij)$ il tempo di fine al più presto per l'attività ij
$tfpt(ij)$ il tempo di fine al più tardi per l'attività ij

s_{ij} lo scorrimento dell'attività ij

FORWARD STEP

Tempo al più presto di un evento

Il tempo al più presto di un evento è il suo tempo di accadimento al più presto, cioè il tempo necessario affinché l'evento possa verificarsi, dopo che tutte le attività che lo precedono sono state completate. Il calcolo viene effettuato utilizzando la regola di etichettamento dell'algoritmo di percorso massimo: si può etichettare un nodo solo se esso è destinazione di archi il cui nodo origine sia già etichettato.

Il tempo al più presto dell'evento inizio del progetto viene fissato a priori, ed in genere viene posto pari a 0.

Il tempo al più presto di un evento *j* si ottiene sommando, per ciascun arco *ij* incidente in esso, il tempo al più presto dell'evento inizio *i* e la durata dell'attività *ij*, scegliendo il valore massimo e registrando il nodo *i* (predecessore) a partire dal quale esso è stato determinato:

$$tpp(j) = \max_i [tpp(i) + t_{ij}]$$

Il tempo al più presto dell'evento fine del progetto fornisce la durata totale del progetto. Esso corrisponde alla lunghezza del percorso massimo dall'origine alla destinazione della rete, cioè alla durata della sequenza di attività che presenta il tempo di esecuzione maggiore. Questo percorso è definito *percorso critico* del progetto. Esso può essere individuato a ritroso, dalla destinazione all'origine, attraverso il vettore dei predecessori.

BACKWARD STEP

Tempo al più tardi di un evento

Il tempo al più tardi di un evento è il suo tempo di accadimento al più tardi, cioè è il tempo massimo in cui l'evento può verificarsi per rispettare il vincolo che la durata del progetto non venga incrementata.

Il calcolo dei tempi al più tardi viene eseguito a ritroso, partendo dall'evento fine del progetto, per giungere all'evento inizio. La regola di etichettamento è la seguente: si può etichettare un nodo solo se esso è origine di archi il cui nodo destinazione sia già etichettato.

Il tempo al più tardi dell'evento fine del progetto viene assunto in generale pari al suo tempo al più presto, cioè pari alla durata totale del progetto.

Il tempo al più tardi di un evento *i* si ottiene sottraendo la durata di ciascuna attività *ij* con origine in esso al tempo al più tardi dell'evento fine *j* dell'attività *ij* e scegliendo il valore minimo:

$$tpt(i) = \min_j [tpt(j) - t_{ij}]$$

Se il tempo al più tardi dell'evento fine è posto pari al suo tempo al più presto il tempo al più tardi dell'evento inizio del progetto deve risultare pari a 0.

Calcolo degli scorrimenti

Scorrimenti degli eventi

Lo scorrimento di un evento è il margine di tempo disponibile per il suo "accadimento". Esso è quindi pari alla differenza tra il suo tempo al più tardi ed il suo tempo al più presto:

$$s(i) = tpt(i) - tpp(i)$$

Gli eventi con scorrimento nullo sono detti *eventi critici* in quanto uno spostamento del loro tempo di realizzazione comporterebbe un ritardo nella conclusione del progetto.

Scorrimenti delle attività

Lo scorrimento di una attività è pari alla differenza tra il tempo totalmente disponibile per la sua esecuzione e la sua durata t_{ij}. Dunque lo scorrimento $s(ij)$ di una attività ij è pari a:

$$s(ij) = [tpt(j) - tpp(i)] - t_{ij}$$

Le attività del *percorso critico* già definito hanno scorrimento nullo e si definiscono attività critiche. Il percorso critico corrisponde, come già detto, al percorso massimo determinato nel *forward step*, la cui lunghezza fornisce la durata totale del progetto tra l'evento inizio e l'evento fine del progetto. Esso è costituito dalle attività del progetto che hanno scorrimento nullo e dunque vanno controllate con maggiore attenzione se si vogliono evitare ritardi nel completamento del progetto. È possibile, in generale, che in una rete siano presenti più percorsi critici, cui è ovviamente associata la medesima durata. Pertanto mentre un aumento di durata di una attività critica determina sicuramente un aumento della durata del progetto, una riduzione di durata di una attività critica non determina necessariamente una riduzione della durata del progetto.

Problemi di Flusso

Modello di flusso single commodity

Tipologie di flusso

Single-commodity e Multi-commodity

Si opera una distinzione fondamentale tra:

- Modelli di flusso *single-commodity,* applicabili quando il flusso sulla rete è di un solo tipo, e quindi può essere espresso per ogni arco da un'unica variabile di flusso f_{ij}

- Modelli di flusso *multi-commodity,* da applicare quando sulla rete ci sono più tipologie di flusso, per ciascuna delle quali è necessario adottare una variabile f_{ij}^{k}, indicando con k la generica tipologia o *commodity*

Parametri della rete

Costo c_{ij} sugli archi

Costo di spostamento c_{ij} per unità di flusso sugli archi

Un altro parametro importante è il costo di spostamento c_{ij} per unità di flusso sull'arco ij, già introdotto nei problemi di percorso. Si può assumere che esso sia:

- costante, cioè indipendente dal valore di flusso sull'arco o
- variabile con il flusso sull'arco.

La prima assunzione è valida quando il flusso sull'arco è basso rispetto alla capacità dell'arco stesso. Se invece il flusso sull'arco si avvicina al valore di capacità dell'arco, la movimentazione del flusso genera il fenomeno della congestione, cioè dell'aumento del costo di spostamento per unità di flusso all'aumentare del flusso sull'arco stesso.

In questo caso è opportuno assumere che il costo c_{ij} sia funzione del flusso f_{ij}. Questa funzione è non decrescente, con asintoto verticale in corrispondenza del valore di capacità dell'arco, come già mostrato nelle lezioni precedenti.

Parametri della rete

Capacità m_{ij} degli archi

Capacità m_{ij} degli archi

Un parametro importante da prendere in considerazione per studiare i problemi di flusso su rete è la capacità m_{ij} dell'arco, cioè il valore massimo di flusso che può attraversarlo nell'unità di tempo. Per esempio la capacità di un arco stradale si misura in veicoli/ora, quella di una condotta idrica in m^3/secondo e di una fibra di comunicazione in Mbyte/secondo.

Condizioni di funzionamento

Sottosaturazione

Si opera in generale una distinzione tra:

- modelli di flusso che assumono che la capacità degli archi sia infinita (in questo caso il costo di spostamento per unità di flusso si assume costante)
- modelli di flusso che impongono un vincolo di capacità sugli archi e dunque sui percorsi da essi costituiti.

Se si assume che ci sia un vincolo di capacità sugli archi è importante valutare le condizioni di saturazione della rete.

Si dice che la rete è in condizioni di ***sottosaturazione*** quando i flussi rispettano i vincoli di capacità sugli archi utilizzati, senza che si creino fenomeni di coda e dunque attese ai nodi della rete.

I classici modelli di flusso simulano il funzionamento della rete in condizioni di sottosaturazione.

Condizioni di funzionamento

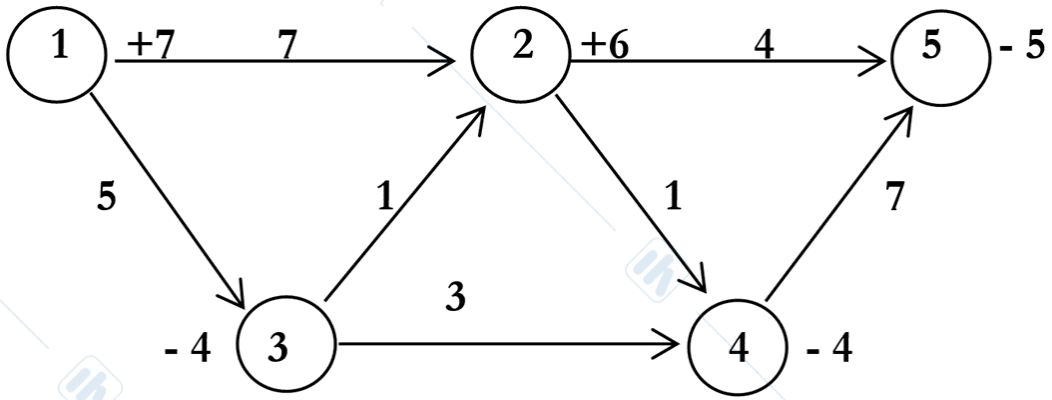
Sovrasaturazione

Si dice invece che la rete opera in condizioni di ***sovrasaturazione*** quando il flusso supera la capacità disponibile sugli archi (e quindi sui percorsi) utilizzabili per gli spostamenti dalle origini alle destinazioni, con la conseguente formazione di code ai nodi della rete. Si pensi per esempio al caso di una rete stradale sulla quale i tempi di verde di un ciclo semaforico non sono sufficienti a smaltire i flussi di traffico in attesa e quindi si formano code alle intersezioni.

Se la rete è *sovrasatura* esiste un limite alla quantità di flusso globale che può spostarsi sulla rete, dai vertici generatori di flusso (vertici origine) ai vertici attrattori di flusso (vertici destinazione). Il calcolo di questo valore di flusso massimo, nelle condizioni "limite" della saturazione, cioè utilizzando tutta la capacità disponibile sugli archi impiegati per il trasferimento del flusso, si effettua attraverso il modello del massimo flusso, descritto nel seguito.

I classici modelli di flusso, formulati per simulare il funzionamento della rete in condizioni di sottosaturazione, non restituiscono una corretta configurazione dei flussi sugli archi della rete. E' necessario quindi ricorrere ad altri modelli di funzionamento della rete, orientati alla determinazione del tempo di spostamento di una assegnata quantità di flusso dai vertici origine ai vertici destinazione, su uno o più percorsi disponibili. Questo argomento viene trattato più ampiamente nel corso di Ricerca Operativa 2.

Un problema di flusso single commodity



$$\text{Min } z = 7f_{12} + 5f_{13} + f_{24} + 4f_{25} + f_{32} + 3f_{34} + 7f_{45}$$

$$f_{12} + f_{13} = 7$$

$$f_{24} + f_{25} - f_{12} - f_{32} = 6$$

$$f_{32} + f_{34} - f_{13} = -4$$

$$f_{45} - f_{24} - f_{34} = -4$$

$$-f_{25} - f_{45} = -5$$

$$fij \geq 0 \ \forall \ ij \in A$$

Soluzione del modello di flusso single commodity

La struttura e le proprietà del modello del problema di flusso *single-commodity* con costi costanti e senza vincoli di capacità consentono una particolare implementazione dell'algoritmo del Simplesso che non utilizza lo schema tabellare, ma opera sulla rete. Esso prende perciò il nome di Algoritmo del Simplesso su rete e presenta gli stessi passi del simplesso tabellare:

- determinazione di una prima soluzione basica ammissibile;
- calcolo dei coefficienti di costo modificati c_{ij}';
- test di ottimalità;
- determinazione della nuova soluzione basica ammissibile, attraverso:
    - determinazione della variabile entrante;
    - determinazione della variabile uscente;
    - modifica della composizione della base.

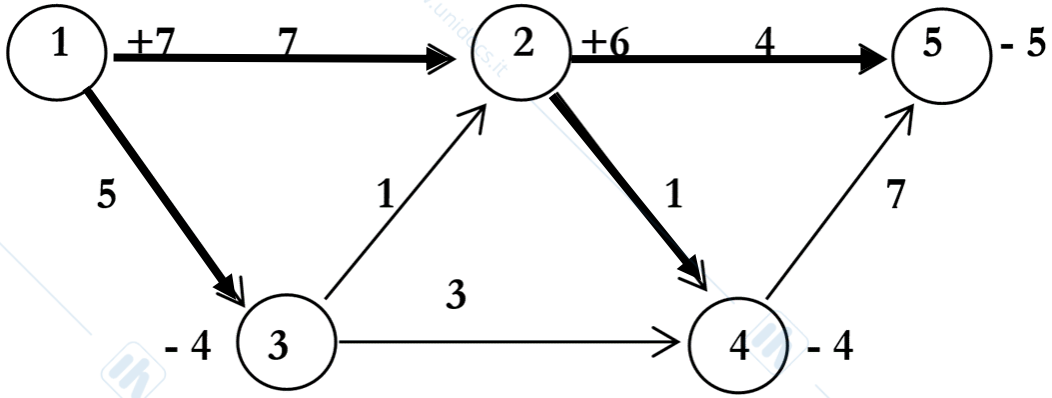
Ciascuno dei passi viene però effettuato come algoritmo su rete, basato sulla corrispondenza albero-base, la cui dimostrazione è disponibile sul libro di testo.

Prima s.b.a. del modello di flusso single commodity

Si consideri per esempio la rete riportata in figura. I nodi 1 e 2 sono generatori di flusso, rispettivamente di 7 e 6 unità di flusso, indicate per convenzione con il segno +. I nodi 3, 4 e 5 sono attrattori di flusso, rispettivamente di 4, 4 e 5 unità di flusso, indicate per convenzione con il segno -. Si noti che è rispettata la precondizione che la somma dei flussi generati sia pari alla somma dei flussi attratti ($\sum_{i \in O} g_i = \sum_{j \in F} a_j = 13$).

Il primo passo è la determinazione di una soluzione basica ammissibile, cioè una prima soluzione su albero, necessaria per innescare la procedura del Simplesso su rete.

A tale scopo si costruisce un albero ammissibile, costituito cioè da archi tali che ci sia almeno un arco uscente da ciascun nodo generatore di flusso ed almeno un arco entrante in ciascun nodo attrattore di flusso, rappresentato in grassetto.



Proprietà della matrice A

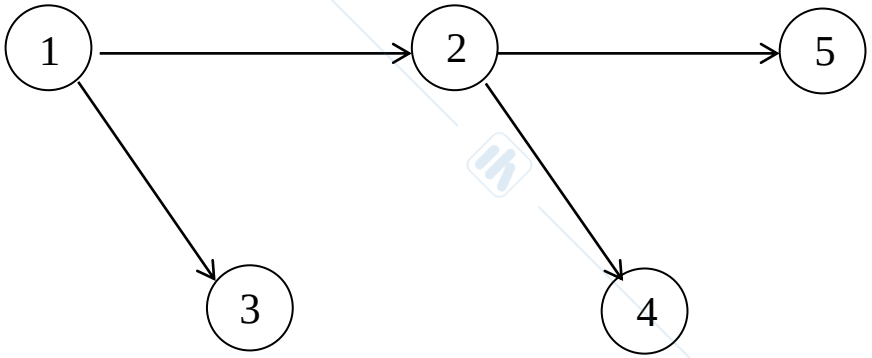
Il **rango** della matrice $\boldsymbol{A}$ non è massimo perché la somma di tutte le righe è il vettore nullo. Si può dimostrare che il rango di $\boldsymbol{A}$ è pari ad $v - 1$. Per fare ciò è necessario individuare in essa un minore di dimensioni $(v-1)\cdot(v-1)$ che sia non singolare. Questo risultato si ottiene dimostrando che la sottomatrice di $\boldsymbol{A}$ costituita dalle colonne corrispondenti agli archi di un albero estratto dalla rete G ha rango $(v-1)$.

Una matrice è totalmente unimodulare se il determinante di ogni minore da essa estratto ha valore −1, 0 o +1. Nel caso del modello in esame, poiché gli elementi della matrice $\boldsymbol{A}$ sono 0, 1 o −1, ogni minore di dimensione [1x1] ha determinante di valore 0, 1 o −1. Inoltre ogni minore [vxv] ha il determinante di valore 0 poiché il rango di $\boldsymbol{A}$ è pari a $v-1$. Si può dimostrare che ogni minore $\boldsymbol{A}_k$ di dimensioni [kxk] ($1<k<n$) ha determinante di valore −1, 0 o +1.

Corrispondenza albero-base

Utilizzando una sottomatrice $\boldsymbol{A}_t$ associata ad un albero estratto dalla rete G si può dimostrare che il rango di $\boldsymbol{A}$ è pari ad $v-1$. Dunque questa sottomatrice $\boldsymbol{A}_t$ corrisponde ad una soluzione basica ammissibile per il modello di flusso. Quindi un albero corrisponde ad una soluzione basica ammissibile. È necessario a questo punto dimostrare l'inverso, cioè che ogni soluzione basica ammissibile corrisponde ad un albero. Questo si dimostra verificando che nessuna s.b.a. corrisponda ad una rete in cui siano presenti circuiti.

Albero T



Matrice A_t

	1-2	1-3	2-4	2-5
1	1	1		
2	-1		1	1
3		-1		
4			-1	
5				-1

Il problema di flusso *single-commodity* con costi costanti e con vincoli di capacità

Se la capacità degli archi non è infinita è necessario aggiungere al modello i vincoli di capacità. Indicando con m_{ij} la capacità dell'arco ij, $\forall ij \in A$, il modello di flusso single-commodity diventa:

$$\text{Min } z = \sum_{ij \in A} c_{ij} f_{ij}$$

$$\sum_k f_{jk} - \sum_i f_{ij} = \begin{cases} g_j & \forall j \in O \\ -a_j & \forall j \in F \\ 0 & Altrimenti \end{cases}$$

$$f_{ij} \leq m_{ij} \qquad \forall ij \in A$$

$$f_{ij} \geq 0$$

Un particolare problema di flusso single commodity:

il problema del trasporto

Un particolare problema di flusso single commodity: il problema del trasporto

Con l'espressione "problema del trasporto" si indica tradizionalmente un problema di flusso *single-commodity* a costi costanti definito su una rete bipartita (lezione sulla teoria dei grafi). A ciascun arco ij è associato un costo c_{ij} per unità di flusso. I vertici origine degli archi sono generatori di flusso. Sia g_i la disponibilità (generazione) in ciascun vertice origine i ($g_i>0$, $i \in O$). I vertici destinazione sono attrattori di flusso. Sia a_j la richiesta (attrazione) in ciascun vertice j ($a_j>0$, $j \in F$). Si assume che sia soddisfatta la condizione $\sum_{i \in O} g_i = \sum_{j \in F} a_j$. Si vuol determinare la configurazione dei flussi sugli archi ij che soddisfi disponibilità e richieste al minimo costo.

Il modello di flusso *single-commodity* già descritto assume dunque, con queste ipotesi, la seguente configurazione:

$$\text{Min } z = \sum_{ij \in A} c_{ij} f_{ij}$$

$$\text{s.a} \quad \sum_{j \in F} f_{ij} = g_i \qquad i \in O$$

$$\sum_{i \in O} f_{ij} = a_j \qquad j \in F$$

$$f_{ij} \geq 0 \qquad \forall\, ij \in A$$

Il problema del trasporto

Prima soluzione basica ammissibile

I metodi per la determinazione di una prima soluzione basica ammissibile per il problema del trasporto sono basati su una tecnica di sostituzione direttamente sulla tabella del trasporto, attraverso la determinazione di una variabile $f_{rs} = \min(a_s, g_r)$, aggiornando in modo conseguente i valori di disponibilità e richiesta.

Uno dei più noti è il metodo dell'angolo di Nord-Ovest (N.O.), così denominato perché assegna valore alla variabile posta in alto a sinistra tra quelle che non sono state ancora determinate.

Il problema del trasporto

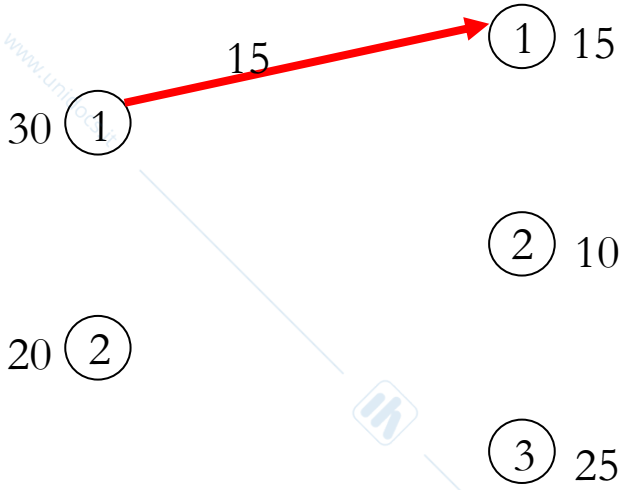
Metodo dell'angolo di nord-ovest per la prima s.b.a.

Il metodo dell'angolo di N.O. effettua le seguenti operazioni:

f_{11} = min (30, 15) = 15: la richiesta a_1 è esaurita, la disponibilità g_1 diventa pari a 15.

--	1	2	3	
1	15			30 → 15
2	----			20
	15 → 0	10	25	

Albero della prima s.b.a.



Il problema del trasporto

Metodo dell'angolo di nord-ovest per la prima s.b.a.

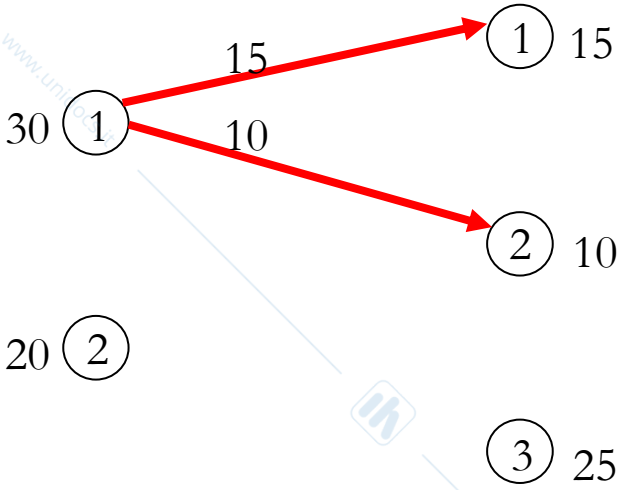
Il metodo dell'angolo di N.O. effettua le seguenti operazioni:

f_{11} = min (30, 15) = 15: la richiesta a_1 è esaurita, la disponibilità g_1 diventa pari a 15.

f_{12} = min (15, 10) = 10: la richiesta a_2 è esaurita, la disponibilità g_1 diventa pari a 5.

	1	2	3	
1	15	10		30 → 15 → 5
2	----	----		20
	15 → 0	10 → 0	25	

Albero della prima s.b.a.



Il problema del trasporto

Metodo dell'angolo di nord-ovest per la prima s.b.a.

Il metodo dell'angolo di N.O. effettua le seguenti operazioni:

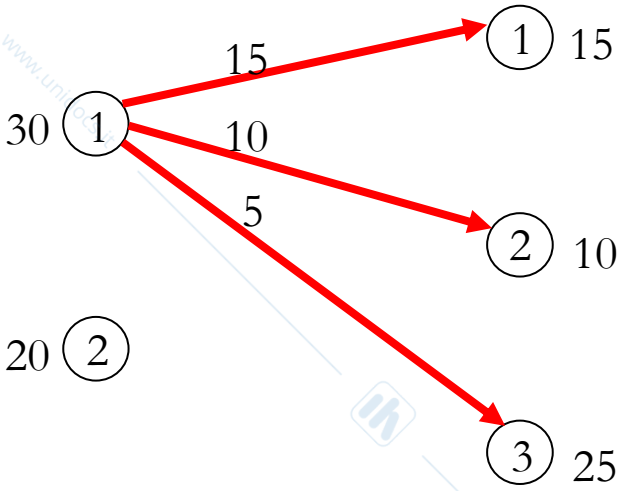
f_{11} = min (30, 15) = 15: la richiesta a_1 è esaurita, la disponibilità g_1 diventa pari a 15.

f_{12} = min (15, 10) = 10: la richiesta a_2 è esaurita, la disponibilità g_1 diventa pari a 5.

f_{13} = min (5, 25) = 5: la disponibilità g_1 è esaurita, la richiesta a_3 diventa pari a 20.

	1	2	3	
1	15	10	5	30 → 15 → 5 → 0
2			20	20
	15	10	25 → 20	

Albero della prima s.b.a.



Il problema del trasporto

Metodo dell'angolo di nord-ovest per la prima s.b.a.

Il metodo dell'angolo di N.O. effettua le seguenti operazioni:

f_{11} = min (30, 15) = 15: la richiesta a_1 è esaurita, la disponibilità g_1 diventa pari a 15.

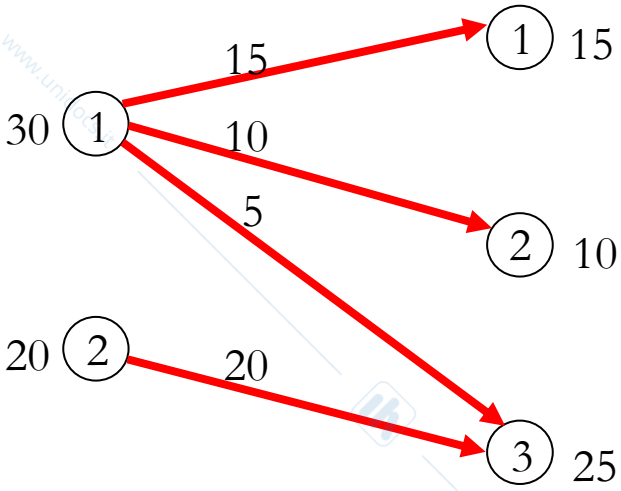
f_{12} = min (15, 10) = 10: la richiesta a_2 è esaurita, la disponibilità g_1 diventa pari a 5.

f_{13} = min (5, 25) = 5: la disponibilità g_1 è esaurita, la richiesta a_3 diventa pari a 20.

f_{23} = min (20, 20) = 20: la richiesta a_3 e la disponibilità g_2 sono esaurite.

	1	2	3	
1	15	10	5	30 → 15 → 5 → 0
2			20	20
	15	10	25 → 20	

Albero della prima s.b.a.



Problema del massimo flusso

Il problema del massimo flusso

Si consideri un grafo $G=(V, A)$ connesso e orientato e sia $M = \{m_{ij}\}$ l'insieme delle capacità sugli archi. Si supponga che sulla rete ci sia un solo vertice origine o con una quantità illimitata di flusso (persone, materiali o informazioni), ed un solo vertice destinazione d nel quale trasferire il flusso. Poiché esiste un vincolo di capacità su ciascun arco ($f_{ij} \leq m_{ij}$, $\forall ij \in A$), sarà possibile trasportare dall'origine o alla destinazione d soltanto una parte del flusso disponibile in o. Si configura pertanto il problema della determinazione del massimo flusso che può partire da o e arrivare in d, nel rispetto dei vincoli di capacità. Questo problema può essere formulato in programmazione lineare con un modello che assume la seguente configurazione:

$$
\begin{aligned}
& \text{Max } z = \sum_j f_{oj} = \sum_i f_{id} \\
\text{s.a} \quad & \sum_k f_{jk} - \sum_i f_{ij} = 0 && \forall\, j \neq o,d \\
& f_{ij} \leq m_{ij} && \forall\, ij \in A \\
& f_{ij} \geq 0 && \forall\, ij \in A
\end{aligned}
$$

La funzione obiettivo esprime la massimizzazione del flusso uscente dall'origine o ($\sum_j f_{oj}$) oppure, equivalentemente, del flusso entrante nella destinazione d ($\sum_i f_{id}$).

I vincoli esprimono il bilancio del flusso nei nodi, tipica dei modelli di flusso, ed i vincoli di capacità sugli archi.

Percorsi aumentanti flusso (p.a.f.)

E' importante riflettere sul fatto che la successione con la quale i percorsi equiversi vengono utilizzati può condizionare la soluzione ottenuta.

Si supponga infatti di utilizzare la visita in profondità invece della visita in larghezza.

Vedremo che il risultato cambia e quindi l'algoritmo descritto non è corretto.

Esso va modificato attraverso l'uso di percorsi aumentanti flusso controversi, per evitare che la successione con la quale i percorsi equiversi vengono generati possa condizionare la soluzione ottenuta.

Algoritmo di Ford e Fulkerson

Per risolvere un problema di massimo flusso si utilizzano algoritmi ad hoc. L'algoritmo di Ford e Fulkerson si basa su una procedura per la individuazione di percorsi aumentanti flusso (p.a.f.), costituita da una visita della rete e un opportuno etichettamento dei vertici. L'algoritmo costruisce una sequenza di p.a.f. la cui sovrapposizione determina il massimo flusso sulla rete.

L'algoritmo opera attraverso *step* successivi. Si assegnano inizialmente agli archi *ij* flussi f_{ij} nulli e capacità "residue" r_{ij} pari alle capacità iniziali m_{ij}.

In ciascuno *step* si sviluppa il processo di etichettamento, che parte dall'origine *o* e cerca di raggiungere la destinazione *d*, individuando un percorso aumentante flusso da *o* a *d* sul quale sia possibile trasportare una certa quantità di flusso δ. Quando viene raggiunta la destinazione *d* lo *step* termina e si aggiornano i valori di flusso e di capacità residua sugli archi che costituiscono il percorso individuato. Nello *step* successivo viene ripetuto il processo di etichettamento da *o* a *d*, determinando un nuovo percorso sul quale sia possibile trasportare una nuova quantità di flusso δ. La procedura si itera fino a quando non è più possibile raggiungere la destinazione *d*. Il flusso massimo è pari alla somma delle quantità δ trasportate da *o* a *d*.

Algoritmo di Ford e Fulkerson

- Con le regole di etichettamento enunciate si sviluppa una procedura di visita, in larghezza o in profondità.
- Lo *step* continua fino a quando viene raggiunta la destinazione d, con $[e_d', e_d''] = [\delta, j]$, e si ferma anche se non sono stati etichettati tutti i vertici.
- Si è determinato in questo modo un percorso aumentante flusso dalla origine alla destinazione, lungo il quale è possibile trasportare una quantità di flusso δ.
- L'individuazione del percorso e l'aggiornamento delle capacità residue e dei flussi sugli archi del p.a.f. viene effettuata nel modo seguente ripercorrendo a ritroso il cammino da d a o attraverso le etichette e_k'':
- per ciascun vertice k con $e_k' > 0$ si aggiornarnano flusso e capacità sull'arco lk orientato da l a k, incrementando il flusso e riducendo la capacità residua: $f_{lk} = f_{lk} + \delta$, $r_{lk} = r_{lk} - \delta$
- per ciascun vertice k con $e_k' < 0$ si aggiornarnano flusso e capacità sull'arco kl orientato da k a l, riducendo il flusso e recuperando capacità residua: $f_{kl} = f_{kl} - \delta$, $r_{kl} = r_{kl} + \delta$
- Questa operazione si effettua fino a quando viene raggiunto, a ritroso, il vertice origine 1. A partire da esso si riparte per una nuova iterazione con flussi e capacità aggiornate.
- La procedura di determinazione di un p.a.f. si itera fino a quando non è più possibile etichettare vertici con le regole stabilite e quindi non è possibile raggiungere la destinazione. Ciò significa che non è più possibile trovare percorsi aumentanti flusso. A questo punto l'algoritmo termina.
- Il flusso massimo è pari alla somma delle quantità δ trasportate da o a d.
- L'insieme dei vertici etichettati costituisce l'insieme N_1. L'insieme dei vertici non etichettati costituisce l'insieme N_2.
- Il taglio minimo è l'insieme degli archi con origine in N_1 e destinazione di N_2.

Nel seguito si un altro esempio numerico nel quale si utilizza un percorso controverso.