

# **Appendice**

Trasparenti integrativi utilizzati  
per il modulo Ricerca Operativa A-L

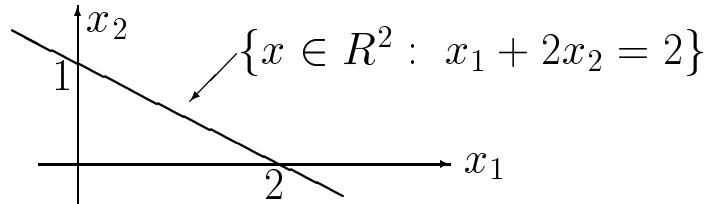
(La numerazione delle pagine fa riferimento alla loro collocazione nel testo)

## Politopi convessi

- Spazio  $R^d$ , vettore  $h \neq 0$ , scalare  $g$ :

$$\text{Iperpiano} = \{x \in R^d : h'x = g\}$$

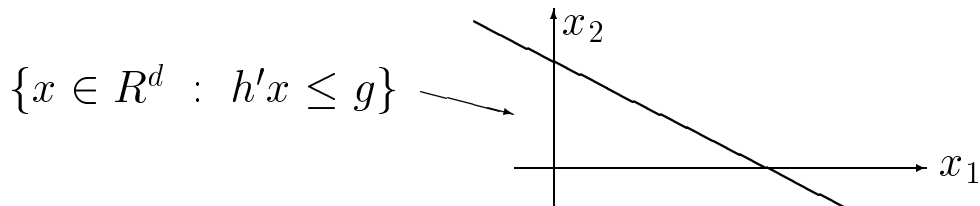
- Es:  $\cdot$  in  $R^2$  :  $h' = (1,2)$ ,  $g=2$



$\cdot$  in  $R^3$  un piano.

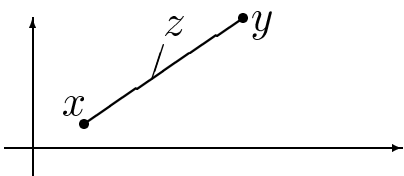
- Un iperpiano definisce 2 *semispazi*:

$$\{x \in R^d : h'x \geq g\} \longrightarrow$$



- Un *semispazio*  $S$  è un *insieme convesso*  
( $\forall$  2 punti  $\in S$ , il segmento che li unisce  $\in S$ ).
- L'*intersezione di più semispazi è convessa*  
(facilmente dimostrabile).
- *Politopo (Politopo convesso)* = intersezione di un numero finito di semispazi, quando sia limitata e non vuota.
- L'insieme dei vincoli di un LP (in forma canonica) definisce un politopo.

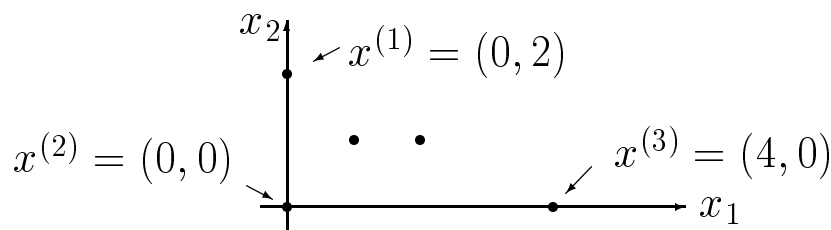
- *Combinazione convessa* di 2 punti  $x, y \in R^n$  = punto  $z \in R^n$ :

$$z = \lambda x + (1 - \lambda)y \quad (\text{con } 0 \leq \lambda \leq 1).$$


Al variare di  $\lambda$ ,  $z$  descrive tutti i punti del segmento  $[x, y]$ .

- *Combinazione convessa di p punti*  $x^{(1)}, \dots, x^{(p)} \in R^n$ :

$$z = \sum_{i=1}^p \alpha_i x^{(i)} \quad (\text{con } \sum_{i=1}^p \alpha_i = 1, \alpha_i \geq 0 \forall i).$$



$$\cdot \quad \alpha = (\frac{1}{2}, 0, \frac{1}{2}) \quad : \quad z = \frac{1}{2}(0, 2) + 0(0, 0) + \frac{1}{2}(4, 0) = (2, 1);$$

$$\cdot \quad \alpha = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4}) \quad : \quad z = \frac{1}{2}(0, 2) + \frac{1}{4}(0, 0) + \frac{1}{4}(4, 0) = (1, 1).$$

- *Ogni punto di un politopo è combinazione convessa dei vertici e viceversa.* (Dim. omessa ( $\leftarrow$  Geometria)).
- *Un vertice non è combinazione convessa stretta* (cioè con  $0 < \lambda < 1$ ) *di due punti distinti del politopo.*

Dim (sufficienza): sia  $P$  un politopo,  $v \in P$  un vertice e supponiamo che esistano altri due punti  $y, w \in P$  tali che

$$v = \lambda y + (1 - \lambda)w;$$

$$v \text{ vertice} \Rightarrow \exists \text{ semispazio } HS = \{x : h'x \leq g\} : HS \cap P = v$$

$$\Rightarrow y, w \notin HS \quad \Rightarrow \quad h'y > g \text{ e } h'w > g$$

$$\Rightarrow h'v = h'(\lambda y + (1 - \lambda)w) > g \quad \Rightarrow \quad v \notin HS, \text{ assurdo. } \square$$

## Un diverso algoritmo del simplesso

- Simpleso = sequenza di pivoting,  
ossia di legittime operazioni elementari di riga,  
che trasformano un tableau,  
ossia un sistema di equazioni lineari,  
senza alterarlo.
- Struttura: tutti i valori in colonna 0 (righe  $1-m$ ) sono  $\geq 0$ ;  
in riga 0 (colonne  $1-n$ )  $\exists$  valore  $< 0$ ;
- ossia: si mantiene una soluzione ammissibile ( $\Leftarrow$  variabili  $\geq 0$ )  
ma non ottima ( $\exists$  costo relativo  $< 0$ )  
fino ad ottenere una soluzione ammissibile ed ottima.
- **Osservazione:** un tableau in cui  
 $y_{i0} \geq 0$  ( $i = 1, \dots, m$ ) e  $y_{0j} \geq 0$  ( $j = 1, \dots, n$ )  
rappresenta sempre una soluzione ottima.

Si supponga di avere uno speciale tableau “speculare”, cioè con:

- struttura: tutti i valori in riga 0 (colonne  $1-n$ ) sono  $\geq 0$ ;  
in colonna 0 (righe  $1-m$ )  $\exists$  valore  $< 0$ , ossia

**criterio di ottimalità soddisfatto    MA  
soluzione non ammissibile**

e si supponga di eseguire dei pivoting (legittime operazioni di riga)  
per renderlo ammissibile mantenendo tale struttura,

- ossia: si mantiene una soluzione ottima ( $\Leftarrow$  costi relativi  $\geq 0$ )  
ma non ammissibile ( $\exists$  variabile  $< 0$ )  
fino ad ottenere una soluzione ammissibile ed ottima.

## Algoritmo del simplesso duale:

- 1) Si inizia con una base “ottima” ( $y_{0j} \geq 0 \quad j = 1, \dots, n$ )  
ma non ammissibile ( $\exists y_{i0} < 0, i \geq 1$ ):

	$x_1$	$x_2$	$x_3$	$x_4$			$x_1$	$x_2$	$x_3$	$x_4$	
0	1	0	$-\frac{1}{2}$	0	$\rightarrow$	$-z$	2	0	0	$\frac{1}{2}$	1
2	2	1	-2	0		$x_1$	-2	1	0	$\textcircled{-1}$	-1
4	1	1	-1	1		$x_2$	6	0	1	0	2

- 2) Si sceglie una riga  $i$  corrispondente ad un  $y_{i0} < 0$ .  
3) Si sceglie la colonna  $s$  del pivot  $y_{is}$  tra gli  $y_{ij} < 0$   
( $\Leftarrow$  il pivoting deve rendere positiva  $y_{i0}$ ):

- il pivoting deve portare 0 in  $y_{0s}$ ,  $\Rightarrow$

$$\tilde{y}_{0j} := \underbrace{y_{0j}}_{\geq 0} - \underbrace{\left(\frac{y_{0s}}{y_{is}}\right)}_{\leq 0} y_{ij} \geq 0 \quad \Leftarrow (y_{0s} \geq 0, y_{is} < 0)$$

e  $\tilde{y}_{0j}$  deve restare  $\geq 0$  ( $j = 1, \dots, n$ )  $\Rightarrow$

$$\frac{y_{0j}}{y_{ij}} \leq \frac{y_{0s}}{y_{is}} \quad \forall j : y_{ij} < 0 \Rightarrow \text{scelta del pivot: } \max_{j: y_{ij} < 0} \left\{ \frac{y_{0j}}{y_{ij}} \right\}$$

	$x_1$	$x_2$	$x_3$	$x_4$	
$-z$	1	$\frac{1}{2}$	0	0	$\frac{1}{2}$
$x_3$	2	-1	0	1	1
$x_2$	6	0	1	0	2

ammissibile  $\Rightarrow$  ottima

- Cosa avviene del valore della soluzione ( $z$ ) ?

$$-z = \tilde{y}_{00} = y_{00} - \underbrace{\left(\frac{y_{0s}}{y_{is}}\right)}_{\leq 0} \underbrace{y_{i0}}_{< 0}$$

ossia  $z$  aumenta (peggiora), come era da aspettarsi  
perchè stiamo spostando verso l'ammissibilità  
una soluzione inammissibile e “più che” ottima.

- La scelta del pivot garantisce il minimo aumento di  $z$ . Infatti:

$$-z = \tilde{y}_{00} = y_{00} - \overbrace{\left(\frac{y_{0s}}{y_{is}}\right) y_{i0}}^{\text{aumento di } z (\geq 0)}$$

$$\underbrace{\left(\frac{y_{0s}}{y_{is}}\right)}_{\substack{\uparrow \\ \max \leq 0}} \underbrace{y_{i0}}_{< 0} < 0$$

$\max \leq 0 \Leftrightarrow \text{minimo valore assoluto}$

- Cosa succede se al passo 3), dove scelta del pivot:

$$\max_{j: y_{ij} < 0} \left\{ \frac{y_{0j}}{y_{ij}} \right\},$$

non esistono  $y_{ij} < 0$  ?

Significa sostanzialmente che non è possibile rendere positiva  $y_{i0}$   
 $\Rightarrow$  il problema rappresentato dal tableau è impossibile  
 $\Rightarrow$  il problema originale è impossibile.

**procedure DUAL\_SIMPLEX:**

**comment:** sia dato un tableau con base che soddisfa  
il criterio di ottimalità ( $\Leftrightarrow y_{0j} \geq 0, j = 1, \dots, n$ )  
ma non è ammissibile ( $\Leftrightarrow \exists i > 0 : y_{i0} < 0$ );

**begin**

*optimal* := false;

*infeasible* := false;

**while** *optimal* = false **and** *infeasible* = false **do**

**if**  $y_{i0} \geq 0$  for  $i = 1, \dots, m$  **then** *optimal* = true

**else**

**begin**

      scegli un  $i > 0 : y_{i0} < 0$  ;

**if**  $y_{ij} \geq 0$  for  $j = 1, \dots, n$  **then**

*infeasible* := true

**else**

**begin**

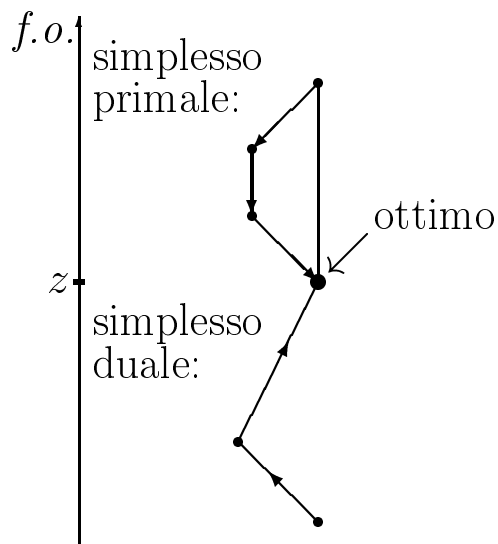
$$\vartheta := \max_{j > 0 : y_{ij} < 0} \left\{ \frac{y_{0j}}{y_{ij}} \right\} = \frac{y_{0s}}{y_{is}};$$

        esegui un pivoting su  $y_{is}$

**end**

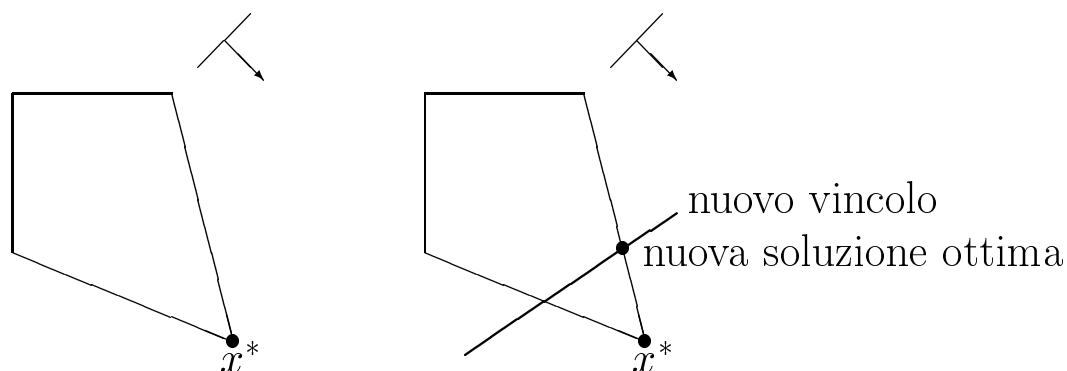
**end**

**end.**



Il simplesso duale ha importanza pratica quando

- è dato il tableau ottimo di un  $LP$ ;
- si aggiunge un vincolo che rende la soluzione attuale non più ammissibile (ma “più che ottima”);



- si vuole la nuova soluzione ottima, senza ripartire da capo.

( $\Rightarrow$  Non serve metodo delle 2 fasi per avere il tableau iniziale.)

SIMPLESSO	PRIMALE	DUALE
ci si sposta	da $SBA$ a $SBA$	da $SB$ inammissibile a $SB$ inammissibile
$z$ parte da un valore	superiore all'ottimo	inferiore all'ottimo
$z$	decresce	cresce
si sceglie prima cioè la variabile che	la colonna entra in base	la riga lascia la base
poi si sceglie cioè la variabile che	la riga lascia la base	la colonna entra in base
pivot dato da un	min tra valori $> 0$	max tra valori $< 0$



## Algoritmi approssimati

- Un algoritmo branch-and-bound può richiedere un tempo di calcolo inaccettabile, se il bound non elimina molti nodi.
- Ad esempio, l'algoritmo branch-and-bound per *KP01* può generare, nel caso peggiore,  $2^n$  nodi, e quindi il tempo di calcolo può crescere, con  $n$ , proporzionalmente a  $2^n$ . Si dice che l'algoritmo ha *complessità*  $O(2^n)$  (*di ordine*  $2^n$ ).
- Supponiamo ad esempio che un algoritmo richieda tempo  $2^n$  e risolva un problema con  $n = 10$  in 0.001 secondi:
  - per  $n = 20$  richiederà 1 secondo ( $2^{10} \approx 1000$ ,  $2^{20} \approx 1000000$ );
  - per  $n = 30$  richiederà 17.9 minuti;
  - per  $n = 40$  richiederà 12.7 giorni;
  - per  $n = 50$  richiederà 35.7 anni;
  - per  $n = 60$  richiederà 366 secoli.
- In molti casi occorrono anche *algoritmi approssimati* in grado di fornire, in tempi di calcolo accettabili, una soluzione ammissibile (che rispetti i vincoli) e sufficientemente buona (anche se non ottima).
- Un algoritmo branch-and-bound può essere usato anche come algoritmo approssimato:
  - si arresta l'esecuzione dopo un numero prefissato di iterazioni (es: di nodi esplorati, di bound calcolati);
  - si prende la miglior soluzione trovata fino a quel momento.
- Se però si è interessati solo ad una soluzione approssimata, è generalmente preferibile progettare ed implementare algoritmi specifici, normalmente più semplici.

- **Algoritmo approssimato per KP01:**

- **procedure GREEDY**

**begin**

ordina gli elementi per  $p_j/w_j$  decrescenti;

$\bar{c} = c; z^g = 0;$

**for**  $j := 1$  **to**  $n$  **do**

**if**  $w_j \leq \bar{c}$  **then**  $x_j := 1, \bar{c} := \bar{c} - w_j, z^g := z^g + p_j$

**else**  $x_j := 0$

**end.**

Tempo:  $O(n \log n)$  per l'ordinamento (+  $O(n)$  per il ciclo).

- Es:  $p' = (12, 12, 7, 6, 2)$   
 $w' = (4, 5, 3, 3, 2)$   
 $c = 10$

Soluzione ottima:  $x' = (1, 0, 1, 1, 0), z = 25$ .

Soluzione greedy:  $x' = (1, 1, 0, 0, 0), z = 24$ .

- Gli algoritmi con complessità *esponenziale* (es:  $O(2^n)$ ) possono essere estremamente lenti;  
gli algoritmi con complessità *polinomiale* ( $O(n \log n), O(n^2), \dots$ ) sono in generale molto più veloci.
- Supponiamo ad esempio che un algoritmo richieda tempo  $n^3$  e risolva un problema con  $n = 10$  in 0.001 secondi:  
per  $n = 20$  richiederà 0.008 secondi;  
per  $n = 30$  richiederà 0.027 secondi;  
per  $n = 40$  richiederà 0.064 secondi;  
per  $n = 50$  richiederà 0.125 secondi;  
per  $n = 60$  richiederà 0.216 secondi.

## Generazione di valori pseudo-casuali

- I linguaggi di simulazione possono generare valori pseudo-casuali secondo qualunque distribuzione di probabilità.
- Principali distribuzioni di probabilità usate in simulazione:

Distribuzione uniforme tra  $a$  e  $b$ :

- Funzione *densità di probabilità*:

$$f(x) = \frac{1}{b-a} \quad (a \leq x \leq b);$$

- $E = \frac{a+b}{2}$  (*valore medio*).

Distribuzione esponenziale:

- $f(x) = \alpha e^{-\alpha x} \quad (\alpha > 0, x \geq 0);$

- $E = \frac{1}{\alpha}.$

- In un *processo di Poisson* di valor medio  $\alpha$ :
  - l'intervallo di tempo tra due eventi consecutivi ha distribuzione esponenziale di valor medio  $\frac{1}{\alpha}$ .
  - I processi di Poisson vengono spesso utilizzati per simulare eventi “rari” ed indipendenti (es: arrivi casuali:  
 $\alpha$  = numero medio di arrivi nell'unità di tempo).

## Descrizione statica del sistema

Configurazioni di dati  $\leftrightarrow$  **stato** del sistema.

I processi dinamici alterano i valori dei dati  $\Rightarrow$  alterano lo stato.

### Oggetti:

Terminologia	Esempi	SIMSCRIPT
Entità	auto	AUTO
Attributi	istante di ingresso in coda  TIC(AUTO)	TIC(AUTO)
Insiemi	Coda  inserisci l'auto in coda  estrai la prima auto dalla coda	CODA  FILE AUTO IN CODA  REMOVE FIRST AUTO FROM CODA

### Generazione e distruzione di oggetti:

Terminologia	Esempi	SIMSCRIPT
Generazione (ingresso nel sistema): riserva un'area di memoria	genera l'auto	CREATE AUTO
Distruzione (uscita dal sistema): rilascia l'area di memoria	distruggi l'auto	DESTROY AUTO

- Attributi del sistema:

$\lambda$  = numero medio di arrivi nell'unità di tempo;

NLG = numero letti nel reparto malati gravi;

NLN = numero letti nel reparto malati normali;

PG = probabilità che un malato sia grave;

PS = probabilità che un malato grave guarisca;

DMIG = durata minima di una degenza grave;

DMAG = durata massima di una degenza grave;

DMIN = durata minima di una degenza normale;

DMAN = durata massima di una degenza normale;

NT = numero di malati da simulare;

NLOG = numero letti occupati nel reparto malati gravi;

NLON = numero letti occupati nel reparto malati normali;

NMR = numero di malati gravi rifiutati;

NMD = numero di malati gravi deceduti;

NMG = numero di malati gravi guariti;

NMN = numero di malati normali guariti;

NMA = numero di malati che hanno atteso in coda;

NTOT = numero totale di malati simulati;

TTG = tempo totale trascorso dai malati gravi nel sistema;

TTN = tempo totale trascorso dai malati normali nel sistema;

TTA = tempo totale trascorso dai malati in coda.