Corso di

Modelli e metodi di ottimizzazione

(materiale integrativo)

Alberto Bemporad

http://www.dii.unisi.it/~bemporad

Università di Siena

Dipartimento di Ingegneria dell'Informazione

Programmazione Lineare

(*Linear Programming, LP*)

Modellistica

"Formulare in termini matematici un problema reale"

Occorre definire:

- Quali sono le **variabili** da decidere
- Qual'è la **funzione obiettivo** che ci interessa minimizzare o massimizzare
- Quali **vincoli** si hanno nella scelta

Sono una componente fondamentale del modello.
Dato il problema reale, occorre cercare di individuarli facendo riferimento a una casistica nota (e non solo)

Vincoli

1 - Upper & Lower Bounds

$$x_i \le X_{\max}$$
$$x_i \ge X_{\min}$$

2 - Flusso

$$\sum x_i \le F_{\max}$$

3 - Risorse

$$\sum_{i=1}^{N} R_{ji} x_i \le \text{max risorsa}_j$$

4 - Qualità

$$\frac{\sum_{i=1}^{N} c_i x_i}{\sum_{i=1}^{N} x_i} \le c$$

5 - Miscelazione

$$\begin{aligned} 0.3 &= x_1/(x_1+x_2+x_3) \\ 0.4 &= x_2/(x_1+x_2+x_3) \\ 0.3 &= x_3/(x_1+x_2+x_3) \end{aligned}$$

6 – Vincoli di contabilità (*accounting*)

("**Vincoli che non vincolano**")

Sono vincoli che aggiungono informazioni sulla **relazione** esistente fra variabili del modello

Esempi:

$$\sum \text{use}_j = \text{produce}$$

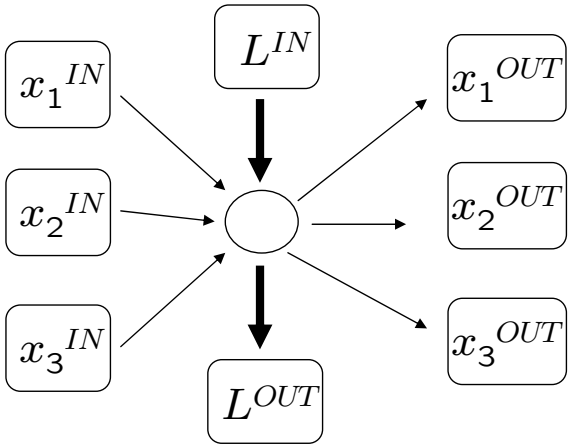
In generale possono essere espressi come vincoli di uguaglianza:

$$\sum_{i=1}^{N} a_i x_i = b_i$$

Sono usati per aumentare la **leggibilità** del modello

7 – Bilanciamento

I flussi in ingresso e in uscita devono bilanciarsi



$$L^{IN} + \sum x_i^{IN} = \sum x_i^{OUT} + L^{OUT}$$

$x_1{}^{IN}$, $x_2{}^{IN}$, $x_3{}^{IN}$, $x_1{}^{OUT}$, $x_2{}^{OUT}$, $x_3{}^{OUT}$ = variabili

È fondamentale per problemi con variabili dinamiche

Esempio: $\text{cassa}_{k+1} = \text{cassa}_k + \text{guadagno}_k - \text{spese}_k$

7 - Bilanciamento

Esempio:

Il numero di azioni di un certo tipo contenute nel portafoglio titoli ad un dato mese k è uguale al numero di azioni presenti nel mese precedente k-1, più il numero di azioni acquistate durante il mese k meno il numero di azioni vendute durante il mese k.

for $k = 1 \ \ldots \ T$

$$\text{stock}_k = \text{stock}_{k-1} + \text{buy}_k - \text{sell}_k$$

end

8 - Scelta del Modo di Produzione

Generalizzazione dei vincoli di miscelazione:
M ingredienti (=inputs) danno N prodotti (=outputs).
Le percentuali di miscelazione variano a seconda del **modo** di produzione.

	Modo 1	Modo 2	Modo 3
IN 1	34%	36%	44%
IN 2	28%	25%	24%
IN 3	38%	39%	32%
OUT 1	0.28	0.27	0.25
OUT 2	0.29	0.36	0.44

Significato: n kg di miscela composta al 34% da IN1, 28% da IN2 e 38% da IN3 viene trasformata, in un'ora di lavorazione e operando nel modo 1, in 0.28*n kg di OUT1 e 0.29*n kg di OUT2.

8 - Scelta del Modo di Produzione

ore_j = numero di ore impiegate nel j-esimo modo di produzione (=variabili di ottimizzazione), j=1,2,3

Sia f = kg di miscela di ingredienti processata ogni ora.

$\Longrightarrow$ f*ore_j = kg di miscela processata nel j-esimo modo

L'utilizzo complessivo delle risorse IN1, IN2, IN3 sarà:

$$\begin{aligned}
\text{IN}_1 &= 0.34 * f * \text{ore}_1 + 0.36 * f * \text{ore}_1 + 0.44 * f * \text{ore}_1 \\
\text{IN}_2 &= 0.28 * f * \text{ore}_2 + 0.25 * f * \text{ore}_2 + 0.24 * f * \text{ore}_2 \\
\text{IN}_3 &= 0.38 * f * \text{ore}_3 + 0.39 * f * \text{ore}_3 + 0.32 * f * \text{ore}_3
\end{aligned}$$

La produzione di OUT1 e OUT2 sarà:

$$\begin{aligned}
\text{OUT}_1 &= 0.28 * f * \text{ore}_1 + 0.27 * f * \text{ore}_2 + 0.25 * f * \text{ore}_3 \\
\text{OUT}_2 &= 0.29 * f * \text{ore}_1 + 0.36 * f * \text{ore}_2 + 0.44 * f * \text{ore}_3
\end{aligned}$$

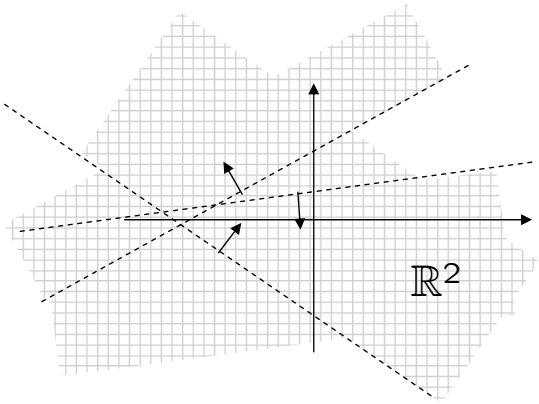
8 - Scelta del Modo di Produzione

Tipicamente si ha un vincolo sulla durata complessiva dell'operazione:

$$\text{ore}_1 + \text{ore}_2 + \text{ore}_3 \leq \text{Tempo}_{\text{tot}}$$

Inammissibilità

INAMMISSIBILITÀ
(**INFEASIBILITY**)



Il problema non ammette una soluzione, vale a dire nessuna scelta delle variabili è in grado di soddisfare tutti i vincoli. I vincoli imposti sono troppo stringenti.

Il risolutore non restituisce alcuna soluzione !

Nota: La non ammissibilità non dipende dalla funzione obiettivo ! Solo dai vincoli

Inammissibilità

L'inammissibilità del problema spesso è dovuta ad errori commessi in fase modellistica

Se sappiamo a priori che il problema ammette soluzione perché conosciamo una scelta x ammissibile, dobbiamo capire dov'è che il modello è errato. In particolare:

- Una direzione invertita in una diseguaglianza
- Dati del problema sbagliati

Come individuare i vincoli errati ?

Se conosciamo una scelta x che sappiamo dovere essere ammissibile, possiamo vedere subito quali vincoli sono violati

Inammissibilità

Se non sappiamo a priori se il problema ammette una soluzione, diventa molto più difficile individuare quali vincoli sono errati (ammesso che siano errati)

Nota: chiedersi "quale vincolo causa infeasibility ?" non ha molto senso. È la **combinazione** di vincoli che è infeasible !

Esempio:

$$\begin{aligned} &\text{prodotto}_1 \leq 3 \\ &\text{prodotto}_2 \leq 7 \\ &\text{prodotto}_3 \leq 1 \\ &\text{prodotto}_1 + \text{prodotto}_2 + \text{prodotto}_3 \geq 12 \end{aligned}$$

Quale di questi quattro vincoli causa infeasibility ? Tutti e nessuno ... Occorre guardare al significato "fisico" di ogni vincolo.

9 – Vincoli soft

Ammettiamo che il modello sia corretto, ma che non ammetta soluzioni ammissibili. Dobbiamo quindi modificare il modello per ottenere una soluzione.

Distinguiamo fra due tipi di vincoli:

- Vincoli "**HARD**": non possono essere assolutamente violati (es: capacità di un magazzino)
- Vincoli "**SOFT**": una loro violazione è tollerabile, anche se c'è una penale da pagare (es: il fattore di rischio di un investimento è maggiore del dovuto)

9 – Vincoli soft

Esempio: Abbiamo a disposizione 200 kg di legno, ma se necessario possiamo comprarne quantità aggiuntive al prezzo di 20 €/kg

"ammorbidimento" del vincolo

$$\sum \text{usa}_j \text{legno}_j \leq 200 + \text{legno}_{\text{add}}$$

$$\text{Costo} = \sum C_j * \text{usa}_j + 20 * \text{legno}_{\text{add}}$$

penalità

9 – Vincoli soft

La variabile $\text{legno}_{\text{add}}$ viene chiamata anche variabile **di panico**

Il risolutore tenderà a porre le variabili di panico a zero

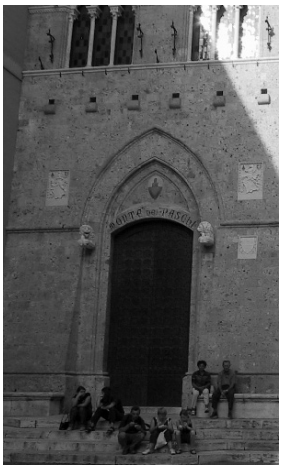
Qualora ciò non sia possibile, fornirà comunque una soluzione al problema, rendendo positive alcune variabili di panico

Se una certa variabile di panico è diversa da zero nella soluzione ottima, possiamo provare ad aumentare la penalità per capire se il vincolo corrispondente deve essere davvero violato per avere una soluzione

Problema del prestito

La nostra ditta, all'inizio della propria attività, pur disponendo di un capitale iniziale di 50 k€, deve decidere l'entità di un prestito da richiedere al Monte dei Paschi. Supponendo che l'interesse sia del 4% annuo, l'obiettivo è di minimizzare gli interessi restituiti alla banca.

Si ha una stima dei pagamenti e degli introiti che la nostra ditta dovrà effettuare nei prossimi 12 anni.



```
Spese (k€):    [40 40 40 40 25 20 20 20 20 20 37 40]
Profitti (k€):[30 30 20 22 22 40 40 40 40 40 40 40]
```


Modellistica

"decidere l'entità del prestito"

→ variabile di ottimizzazione

"minimizzare gli interessi restituiti"

→ funzione obiettivo

"interesse annuale del 4%"

È una funzione lineare ?

Modellistica

"Si ha una stima dei pagamenti e degli introiti"

vincoli di bilanciamento

Osservazione: la cassa non deve mai diventare negativa!

Lower bound:
il capitale ≥ 0 !

Dobbiamo esprimere l'evoluzione del capitale a disposizione della ditta sull'arco temporale dei 12 anni

Scelta delle variabili

prestito	Entità del prestito
rata_i	Rata pagata durante l'anno i-esimo
cassa_i	Capitale a disposizione nell'anno i-esimo

$i = 1 \ldots N$

N=12, I%=4

Interessi

Abbiamo supposto che gli interessi siano a tasso fisso

$$\text{rata}_i = \text{prestito} * \frac{\frac{I\%}{100}}{1-(1+\frac{I\%}{100})^{-N}}$$

sembra non lineare ...

In realtà, essendo $I\%$ e N parametri fissati, vale la relazione seguente, dove K è una costante:

$$\text{rata}_i = \text{prestito} * \boxed{K}$$

È un vincolo lineare (di contabilità)

Funzione obiettivo

Minimizzare gli interessi restituiti:

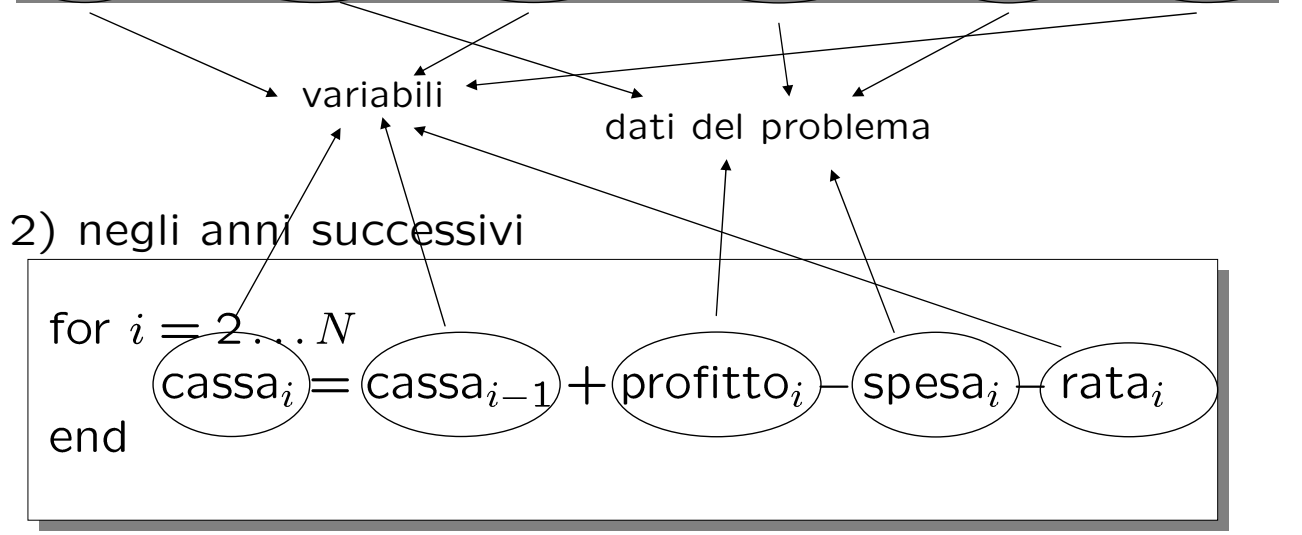
$$\text{Interessi} = \sum_{i=1}^{12} \text{rata}_i - \text{prestito}$$

Vincoli di bilanciamento

I flussi di cassa si devono bilanciare:

1) Alla fine del primo anno

$$\text{cassa}_1 = \text{iniziale} + \text{prestito} + \text{profitto}_1 - \text{spesa}_1 - \text{rata}_1$$



2) negli anni successivi

for $i = 2 \ldots N$

$$\text{cassa}_i = \text{cassa}_{i-1} + \text{profitto}_i - \text{spesa}_i - \text{rata}_i$$

end

Modello del problema

min Interessi

sogg. a

$$\text{Interessi} = \sum_{i=1}^{N} \text{rata}_i - \text{prestito}$$

$$\text{cassa}_1 = \text{iniziale} + \text{prestito} + \text{profitto}_1 - \text{spesa}_1 - \text{rata}_1$$

for $i = 2 \ldots N$

$$\text{cassa}_i = \text{cassa}_{i-1} + \text{profitto}_i - \text{spesa}_i - \text{rata}_i$$

end

for $i = 1 \ldots N$

$$\text{rata}_i = \text{prestito} * \frac{\frac{I\%}{100}}{1-(1+\frac{I\%}{100})^{-N})}$$

end

for $i = 1 \ldots N$

$$\text{cassa}_i \geq 0$$

end

$$N = 12, \quad I\% = 4$$

Modello MOSEL

mutuo.mos

```
model "Mutuo"
 uses "mmxprs"

declarations
  N = 12                          ! Arco temporale (anni)
  ANNI = 1..N
  SPESA: array(ANNI) of real      ! Spese
  PROFITTO: array(ANNI) of real   ! Profitti
  I: real                         ! Interesse annuale del mutuo (%)
  INIZIALE: real                  ! Capitale iniziale
  prestito: mpvar                 ! Importo del mutuo richiesto
  rata: array(ANNI) of mpvar      ! Rata del mutuo
  cassa: array(ANNI) of mpvar     ! Capitale disponibile in cassa
 end-declarations

 initializations from 'mutuo.dat'
  SPESA PROFITTO I INIZIALE
 end-initializations
```


Dati del problema

mutuo.dat

```
SPESA:     [40 40 40 40 25 20 20 20 20 20 37 40]
PROFITTO:  [30 30 20 22 22 40 40 40 40 40 40 40]
I:    4
INIZIALE: 40
```

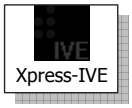
Modello MOSEL

Definizione dei vincoli

```
! Funzione obiettivo: interessi restituiti
 Interessi:= sum(i in ANNI) rata(i) - prestito
! Relazione fra mutuo e rata
forall (i in ANNI) do
 rata(i)=prestito*(I/100)/(1-((1+(I/100))^(-N)))
end-do
! Bilancio alla fine del primo anno
cassa(1) = prestito + INIZIALE - SPESA(1) + PROFITTO(1)-
   rata(1)
! Bilancio alla fine del generico anno k-esimo
forall(i in ANNI| i>1) do
   cassa(i) = cassa(i-1) - SPESA(i) + PROFITTO(i) - rata(i)
end-do
! Risolvi il problema: arricchire il meno possibile la banca
minimize(Interessi)
```


Risultati

File MOSEL: **mutuo.mos**

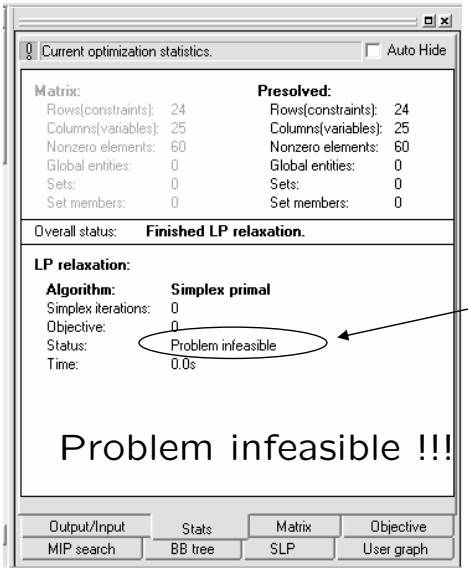



```
Importo del prestito: 44.9449
Capitale restituito: 12.5228
-------------------------
       rata       |  cassa
-------------------------
Anno  1: 4.78897  | 70.1559
Anno  2: 4.78897  | 55.3669
Anno  3: 4.78897  | 30.5779
Anno  4: 4.78897  | 7.78897
Anno  5: 4.78897  | 0
Anno  6: 4.78897  | 15.211
Anno  7: 4.78897  | 30.4221
Anno  8: 4.78897  | 45.6331
Anno  9: 4.78897  | 60.8441
Anno 10: 4.78897  | 76.0551
Anno 11: 4.78897  | 74.2662
Anno 12: 4.78897  | 69.4772
```

Infeasibility

Il problema può diventare non ammissibile ?

```
SPESA:    [40 40 40 40 25 20 20 20 20 20 37 40]
PROFITTO:  [30 30 20 22 22 30 20 20 30 20 30 40]
I:    4
INIZIALE: 40
```




```
Interessi restituiti: 0
Importo del prestito: 0
Rata del mutuo:
Anno  1: 0  - Cassa: 0
Anno  2: 0  - Cassa: 0
Anno  3: 0  - Cassa: 0
Anno  4: 0  - Cassa: 0
Anno  5: 0  - Cassa: 0
Anno  6: 0  - Cassa: 0
Anno  7: 0  - Cassa: 0
Anno  8: 0  - Cassa: 0
Anno  9: 0  - Cassa: 0
Anno 10: 0  - Cassa: 0
Anno 11: 0  - Cassa: 0
Anno 12: 0  - Cassa: 0
```


Vincoli soft

Rimuoviamo il vincolo che impone $\text{cassa}_i \geq 0$, supponendo di ricorrere se necessario a dei prestiti aggiuntivi. I prestiti sono molto costosi: per ogni euro preso a prestito, devono essere restituiti (1+F) euro.

Vincoli soft:

for $i = 1 \ldots N$

$\text{cassa}_i + \text{emergenza}_i \geq 0$

end

variabile di panico

Vincoli soft

Il problema sarà sempre ammissibile, ma il costo si modifica:

$$\text{Interessi} = \sum_{i=1}^{12} \text{rata}_i - \text{prestito} + \sum_{i=1}^{12} F * \text{emergenza}_i$$

valore molto elevato

Modello con vincoli soft

min Interessi

sogg. a

$$\text{Interessi} = \sum_{i=1}^{N} \text{rata}_i - \text{prestito} + \sum_{i=1}^{12} F * \text{emergenza}_i$$

$$\text{cassa}_1 = \text{iniziale} + \text{prestito} + \text{profitto}_1 - \text{spesa}_1 - \text{rata}_1$$

for $i = 2 \dots N$

$$\text{cassa}_i = \text{cassa}_{i-1} + \text{profitto}_i - \text{spesa}_i - \text{rata}_i$$

end

for $i = 1 \dots N$

$$\text{rata}_i = \text{prestito} * \frac{\frac{I\%}{100}}{1-(1+\frac{I\%}{100})^{-N}}$$

end

for $i = 1 \dots N$

$$\text{cassa}_i + \text{emergenza}_i \geq 0$$

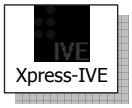
end

$N = 12, \quad I\% = 4$

Soluzione

```
SPESA:     [40 40 40 40 25 20 20 20 20 20 37 40]
PROFITTO:   [30 30 20 22 22 30 20 20 30 20 30 40]
I:     4      F:1.5
INIZIALE: 40
```

File MOSEL:
mutuo2.mos




```
Importo del prestito: 31.3703
Capitale restituito: 33.2079
-----------------------------------------
          rata    |  cassa   |  emergenza
-----------------------------------------
Anno  1: 3.34257  | 58.0277  | 0
Anno  2: 3.34257  | 44.6851  | 0
Anno  3: 3.34257  | 21.3426  | 0
Anno  4: 3.34257  | 0        | 0
Anno  5: 3.34257  | -6.34257 | 6.34257
Anno  6: 3.34257  | 0.314855 | 0
Anno  7: 3.34257  | -3.02772 | 3.02772
Anno  8: 3.34257  | -6.37029 | 6.37029
Anno  9: 3.34257  | 0.287137 | 0
Anno 10: 3.34257  | -3.05544 | 3.05544
Anno 11: 3.34257  | -13.398  | 13.398
Anno 12: 3.34257  | -16.7406 | 16.7406
```


Programmazione lineare mista-intera (MILP)

Modellistica MILP

Oltre alle variabili reali continue, si hanno a disposizione nuovi tipi di variabili:

intere
binarie
semicontinue
SOS (special ordered sets)
...

Si possono modellare una gamma molto estesa di nuovi vincoli ! In particolare:

vincoli logici

Variabili intere

Le variabili intere possono assumere valori soltanto nell'insieme dei numeri naturali $\mathbb{N} = \{0,1,2,...\}$

$$x \in \mathbb{N}$$

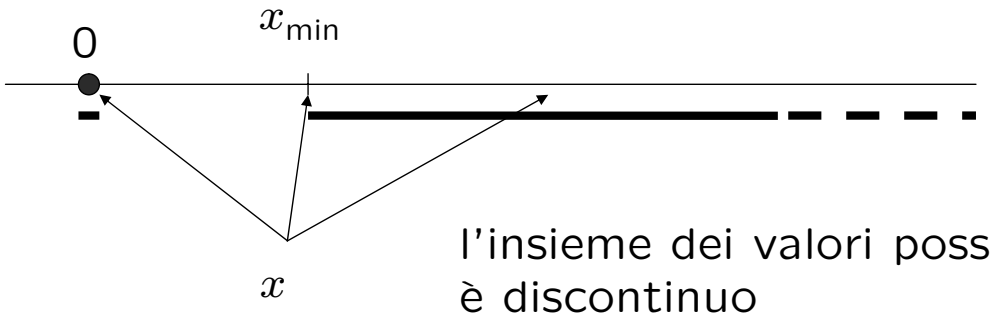
È un vincolo molto comune e del tutto naturale.

Esempi:

Numero di persone assegnate ad un progetto
Numero di impianti di produzione impiegati
Numero di lotti prodotti
...

Variabili semicontinue

Una variabile semicontinua o è nulla, oppure assume un valore maggiore di una certa quantità $x_{\min}$



l'insieme dei valori possibili è discontinuo

Permettono di esprimere delle situazioni reali molto frequenti. Esempi:

Se vendo, devo vendere almeno una certa quantità
Livelli di produzione minima per garantire un profitto

Altri tipi di variabile

Esistono altri tipi di variabile non reale. Ad esempio:

Special Ordered Sets SOS1 e SOS2

Riconoscerle è importante per facilitare la risoluzione del problema di ottimizzazione.

Dal punto di vista della sola modellistica, possiamo focalizzare l'attenzione soltanto sulle variabili continue e intere.

Il problema del ladro

Un malvivente decide di derubare un supermercato. La capacità del bagagliaio del suo furgoncino è di 500 litri. Dato che dei prodotti disponibili si conosce il prezzo e la quantità sugli scaffali, quali prodotti conviene scegliere e in quale quantità per ricavare un bottino il più redditizio possibile ?

(applicazione: problemi di stoccaggio)

Modellistica

"...quali prodotti e in quale quantità ..."

Variabili intere

$take_i \in N$

"...bottino il più costoso possibile..."

Funzione obiettivo

$Profit = \sum value_i * take_i$

"...bagagliaio di 500 litri..."

Vincoli di spazio

$\sum volume_i * take_i \leq 500$

"...quantità disponibile ..."

$take_i \leq quantity_i$

Prodotti disponibili

	Prezzo per unità (€)	Vol. per unità (litri)	pezzi disponibili
TV	120	80	2
DVD	60	20	3
Lavatrice	150	100	3
Detersivo	6	20	50
Cioccolata	0.55	0.3	80
Profumo	10	0.5	10
Conf. Latte (12 pezzi)	12.40	12	150
	$value_i$	$volume_i$	$quantity_i$

Modello del problema

max $Profit$

sogg.a

$N = 7$

$$Profit = \sum_{i=1}^{N} value_i * take_i$$

for $i = 1 \dots N$

$$items_i \leq quantity_i$$

end

$$\sum_{i=1}^{N} volume_i * take_i \leq 500$$

for $i = 1 \dots N$

$$take_i \in \mathbb{N}$$

end

Modello MOSEL

(file: **ladro.mos**)

```
declarations
        ITEMS = 1..7
        SMAX = 500
        !Data arrays
        VALUE:  array(ITEMS) of rea
        VOLUME: array(ITEMS) of rea
        QMAX:   array(ITEMS) of rea
        !Decision variables
        take:   array(ITEMS) of mpv
end-declarations

!Objective function
Profit:= sum(i in ITEMS) VALUE(i)*ta
!Constraints
sum(i in ITEMS) VOLUME(i)*take(i) <=
forall (i in ITEMS)
        take(i) <= QMAX(i)
!Decision variable type
forall (i in ITEMS)
        take(i) is_integer
!Solver call
maximize(Profit)
```

```
Il ladro ricava un bottino di...
921.85 €
portandosi via ...
TV: 1 (di 2)
DVD: 3 (di 3)
Lavatrice: 3 (di 3)
Detersivo: 0 (di 50)
Cioccolata: 63 (di 80)
Profumo: 10 (di 10)
Latte: 3 (di 150)
```

→ NEW

Xpress-IVE

Risoluzione di problemi misti-interi

$$\begin{aligned} \min \quad & f'x + d'\delta \\ \text{s.t.} \quad & Ax + B\delta \le c \\ & x \in \mathbb{R}^n, \ \delta \in \{0,1\}^m \end{aligned}$$

È un problema facile da risolvere ?

Alcune variabili sono continue, altre binarie

Proviamo ad enumerare tutte le possibili combinazioni di variabili intere e scegliamo quella migliore !

Dobbiamo risolvere un LP per ogni combinazione

Esplosione combinatoria

- Enumerare tutte le alternative possibili spesso è impossibile
- Si ha la cosiddetta "esplosione combinatoria" delle possibili alternative

Esplosione combinatoria - Esempio

- Trovare il minimo percorso che copre 10 località
- Supponiamo di avere un "super-computer" in grado di analizzare un milione di alternative al secondo
- La risposta viene determinata in circa 3 secondi ($10! \approx 3.6*10^6$)
- Adesso supponiamo di volere coprire 20 località
- Lo stesso computer impiegherebbe 77.000 anni per trovare la soluzione ! ($20! \approx 2.4*10^{18}$)

Risoluzione di problemi misti-interi

L'enumerazione non è una via percorribile:

Se abbiamo m variabili binarie, dovremmo risolvere ben 2^m problemi LP !

Esempio: supponiamo che in media riusciamo a risolvere un LP con 10 variabili in 0.03s e un LP con 20 variabili in 0.12s

Un MILP con 10 variabili reali + 10 binarie richiederebbe 2^{10} * 0.03s $\approx$ 30s

Algoritmi di Branch & Bound

Esiste il modo di evitare di esplorare tutte le combinazioni possibili di variabili intere.

Gli algoritmi di **branch & bound**, risolvendo una sequenza di problemi LP, evitano di enumerare tutte le possibili combinazioni, permettendo di poter risolvere problemi di ottimizzazione con variabili intere in maniera efficiente.

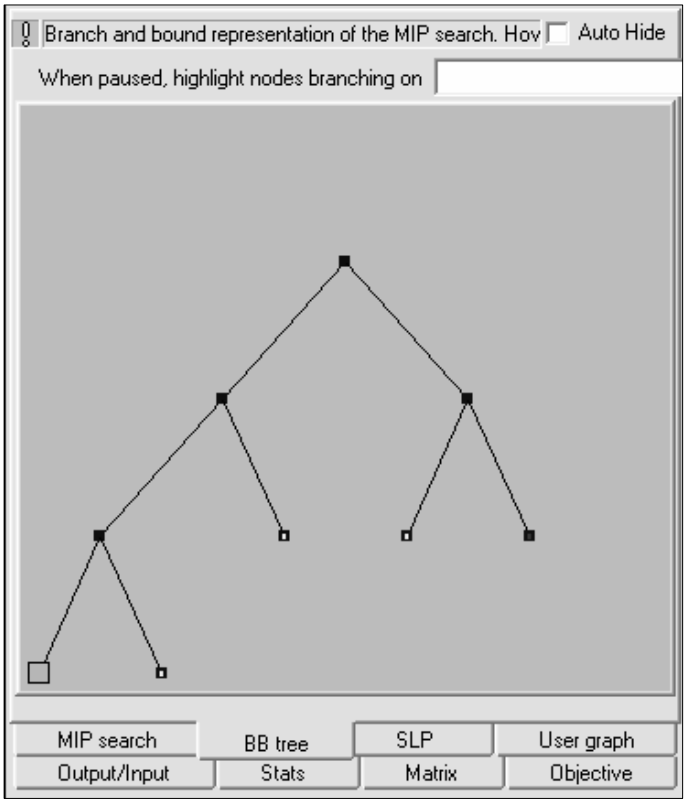
Purtroppo non sono altrettanto efficienti dei risolutori di problemi lineari a variabili continue.

Algoritmo di Branch & Bound

1. Viene risolto un problema "rilassato" in cui tutte le variabili binarie δ sono considerate come continue, $0 \leq \delta \leq 1$. La soluzione di tale problema fornisce una stima per difetto (**lower bound**) del valore ottimo della soluzione. Purtroppo, nella soluzione del problema rilassato può accadere che alcune delle variabili binarie δ abbiano valori frazionari.
2. Si dirama (**branch**) la ricerca spezzando il problema in due sottoproblemi: uno in cui una certa variabile binaria è posta $=0$, l'altro in cui è posta $=1$.
3. Ciascuno dei sottoproblemi viene rilassato a sua volta, e suddiviso ulteriormente se necessario scegliendo un'altra variabile binaria per diramare la ricerca

Vantaggio rispetto all'enumerazione completa: spesso si riescono ad eliminare molte diramazioni usando il bound del valore ottimo trovato via via.

Problema del ladro: albero di ricerca



Traduzione di vincoli logici in disuguaglianze (miste-)intere lineari

(Glover 1975, Williams 1977, Hooker 2000)

Any logic statement
$f(X)$ =TRUE

$$\bigwedge_{j=1}^{m} \left(\vee_{i \in P_j} X_i \vee_{i \in N_j} \neg X_i \right) \quad \text{(CNF)}$$
$$N_j, P_j \subseteq \{1, \ldots, n\}$$

$\Longrightarrow$

$$\begin{cases} 1 \leq \sum_{i \in P_1} \delta_i + \sum_{i \in N_1} (1 - \delta_i) \\ \vdots \\ 1 \leq \sum_{i \in P_m} \delta_i + \sum_{i \in N_m} (1 - \delta_i) \end{cases}$$

$$[\delta_e^i(k) = 1] \leftrightarrow [H^i x_c(k) \leq W^i] \quad \Longrightarrow \quad \begin{cases} H^i x_c(k) - W^i & \leq & M^i(1 - \delta_e^i) \\ H^i x_c(k) - W^i & > & m^i \delta_e^i \end{cases}$$

IF $[\delta = 1]$ THEN $z = a_1^T x + b_1^T u + f_1$
ELSE $z = a_2^T x + b_2^T u + f_2$ $\Longrightarrow$

$$\begin{cases} (m_1 - M_2)(1 - \delta) + z & \leq & a_1 x + b_1 u + f_1 \\ (m_2 - M_1)(1 - \delta) - z & \leq & -a_1 x - b_1 u - f_1 \\ (m_2 - M_1)\delta + z & \leq & a_2 x + b_2 u + f_2 \\ (m_1 - M_2)\delta - z & \leq & -a_2 x - b_2 u - f_2 \end{cases}$$

Formule logiche: Approccio simbolico

0. Data una formula logica $F(X_1, X_2, \ldots, X_n) = \text{TRUE}$

1. Si converte in Conjunctive Normal Form (CNF):

$$\bigwedge_{j=1}^{m} \left(\bigvee_{i \in P_j} X_i \bigvee_{i \in N_j} \bar{X}_i \right) = \text{TRUE}$$

2. Si trasforma in disuguaglianze:

$$\begin{array}{rcl} \sum_{i \in P_1} \delta_i + \sum_{i \in N_1} (1 - \delta_i) & \geq & 1 \\ \vdots & \vdots & \vdots \\ \sum_{i \in P_m} \delta_i + \sum_{i \in N_m} (1 - \delta_i) & \geq & 1 \end{array}$$

$\Longrightarrow$ poliedro $A\delta \leq b, \quad \delta \in \{0, 1\}^n$

Any logic proposition can be translated into linear integer ineqs.

Formule logiche: Approccio simbolico

Esempio: $F(X_1, X_2, X_3) = [X_3 \leftrightarrow X_1 \wedge X_2]$

1. Si converte in Conjunctive Normal Form (CNF):
   (ad esempio: **http://www.oursland.net/aima/propositionApplet.html**
   oppure cerca [CNF + applet] su Google ...)

$$(X_3 \vee \neg X_1 \vee \neg X_2) \wedge (X_1 \vee \neg X_3) \wedge (X_2 \vee \neg X_3)$$

2. Si trasforma in disuguaglianze

$$\begin{cases} \delta_3 + (1-\delta_1) + (1-\delta_2) \geq 1 \\ \delta_1 + (1-\delta_3) \geq 1 \\ \delta_2 + (1-\delta_3) \geq 1 \end{cases}$$

Formule logiche: approccio geometrico

Formula booleana

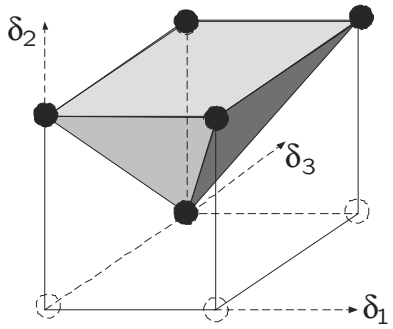
$F(X_1, X_2, \ldots, X_n) = \text{TRUE}$ $\Longrightarrow$ poliedro $A\delta \leq b, \quad \delta \in \{0,1\}^n$

Il politopo $P = \{\delta : \ A\delta \leq b\}$ è dato dall'inviluppo convesso (convex hull) delle righe della tabella di verità T associata alla formula $F(X_1, \ldots, X_N)$

T:

X_1	X_2	...	X_N
0	0	...	1
0	1	...	0
⋮	⋮	⋮	⋮
1	1	...	0

$\Longrightarrow$



$\Longrightarrow$ $P : \ \{\delta : \ A\delta \leq b\}, \ \delta \in \{0,1\}^n$

Algoritmi di convex hull: **cdd, lrs, qhull, chD, Hull, Porto**

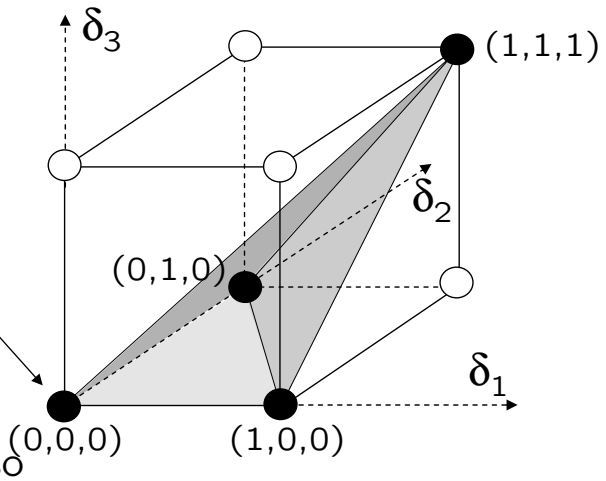
Formule logiche: approccio geometrico

Esempio: "AND" logico

$F(X_1, X_2, X_3) = [X_3 \leftrightarrow X_1 \wedge X_2]$

T:

X_1	X_2	X_3
0	0	0
0	1	0
1	0	0
1	1	1



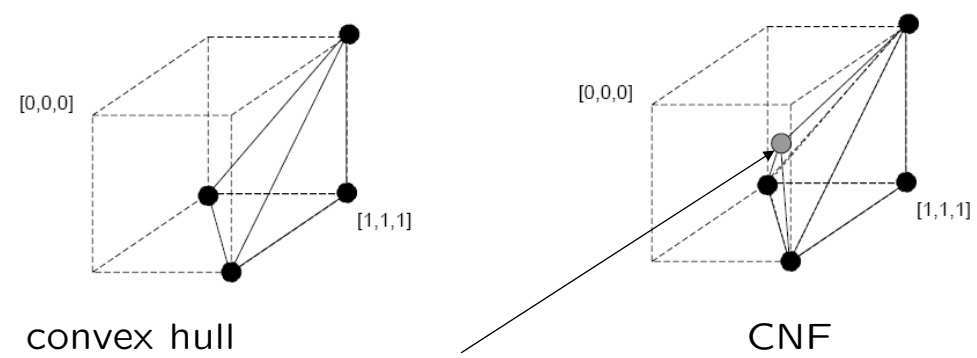
Idea chiave:
I punti bianchi non sono contenuti nell'inviluppo convesso dei punti neri

$$\text{conv}\left(\begin{bmatrix}0\\0\\0\end{bmatrix},\begin{bmatrix}0\\1\\0\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix},\begin{bmatrix}1\\1\\1\end{bmatrix}\right) = \left\{\delta : \begin{array}{r} -\delta_1 + \delta_3 \le 0 \\ -\delta_2 + \delta_3 \le 0 \\ \delta_1 + \delta_2 - \delta_3 \le 1 \end{array}\right\}$$

Confronti fra gli approcci

- Il politopo ottenuto con l'inviluppo convesso è minimale
- Il politopo ottenuto tramite CNF potrebbe essere più grande. Ad esempio:

$$(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3) = \text{TRUE}$$



convex hull

CNF

vertice spurio
in (0.5,0.5,0.5)

Nota: l'unico altro esempio con 3 vars è $(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee \bar{X}_3)$

Problema dello zaino

Stiamo pianificando un trekking in montagna per il fine settimana. Intenderemmo portare una serie di oggetti, ma ci troviamo di fronte al dilemma di dovere scegliere quali effettivamente inserire nello zaino, in modo tale da non superare i 9 kg.

oggetti disponibili

```
1) sacco a pelo      8) vivande
2) materassino       9) ricambi
3) bussola          10) sapone
4) giacca           11) asciugamano
5) fornello         12) pane
6) piatti           13) accendino
7) posate
```


Problema dello zaino

Assegniamo a ciascuno degli oggetti un certo valore. L'obiettivo è massimizzare il valore totale, senza eccedere però un peso complessivo massimo. Quali oggetti dobbiamo portare ?

Decisione binaria

Vincoli sulle risorse

funzione obiettivo

```
! Articolo:  1  2   3   4   5   6   7   8   9  10  11  12  13
  VALORE:= [11,  9, 12, 8,  8,  6,  2,  6,  1,  4,  3, 5,   6]
  PESO:=   [ 4,  2,0.2, 2,1.5,0.5,0.2,3.5,0.3,0.2,0.8, 1, 0.1]
```

Modello

massimizza $\text{Valore}_{\text{tot}}$

sogg. a

$$\text{Valore}_{\text{tot}} = \sum_{i=1}^{N} \text{valore}_i * \text{prendi}_i$$

$$\sum_{i=1}^{N} \text{peso}_i * \text{prendi}_i \leq \text{PESO}_{\text{max}}$$

$$\text{prendi}_i \in \{0,1\}, \forall 1 = 1 \ldots N$$

Niente di nuovo rispetto al problema del ladro ...

Risultato

(file: **zaino.mos**)

```
Soluzione:
 Valore ottimo: 118
Peso dello zaino: 8.5
 prendi(sacco a pelo): 1
 prendi(materassino): 0
 prendi(bussola): 1
 prendi(giacca): 1
 prendi(fornello): 0
 prendi(piatti): 1
 prendi(posate): 1
 prendi(vivande): 0
 prendi(ricambi): 1
 prendi(sapone): 1
 prendi(asciugamano): 0
 prendi(pane): 1
 prendi(accendino): 1
```

?

?

Occorre aggiungere dei vincoli logici !

Xpress-IVE

Vincoli logici

Aggiungiamo dei vincoli logici:

"se non prendo le vivande, non devo prendere neanche i piatti"

"se non prendo le vivande, non devo prendere neanche le posate"

"se non prendo le vivande, non devo prendere neanche il fornello"

```
prendi("piatti") <= prendi("vivande")
prendi("posate") <= prendi("vivande")
prendi("fornello") <= prendi("vivande")
```


Vincoli logici

"se prendo il fornello devo prendere anche l'accendino"

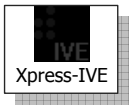
"se non prendo il sapone non prendo i ricambi"

```
prendi("accendino") >= prendi("fornello")
prendi("ricambi") <= prendi("sapone")
```

Risultato

(file: **zaino2.mos**)

```
Soluzione:
 Valore ottimo: 112
Peso dello zaino: 8.6
 prendi(sacco a pelo): 1
 prendi(materassino): 0
 prendi(bussola): 1
 prendi(giacca): 1
 prendi(fornello): 0
 prendi(piatti): 0
 prendi(posate): 0
 prendi(vivande): 0
 prendi(ricambi): 1
 prendi(sapone): 1
 prendi(asciugamano): 1
 prendi(pane): 1
 prendi(accendino): 1
```



Ulteriori modifiche al modello

Aggiungiamo un altro vincolo logico:

"se prendo sacco e materassino aumento di +10 il valore"

Definiamo una nuova variabile binaria:

```
dormibene = prendi("sacco a pelo") AND
            prendi("materassino")
```

```
A = B AND C
```

```
a ≤ b
a ≤ c
a ≥ b+c-1
```

```
dormibene <= prendi("sacco a pelo")
dormibene <= prendi("materassino")
dormibene >= prendi("sacco a pelo") +
             prendi("materassino") - 1
```

```
ValoreTot ← ValoreTot + 10*dormibene
```

Ulteriori modifiche al modello

Aggiungiamo un altro vincolo logico:

"se non prendo cibo diminuisco -10 il valore"

Definiamo una nuova variabile binaria:

```
cibo = prendi("vivande") OR
               prendi("pane")
```

A = B OR C

$\Downarrow$

$a \geq b$
$a \geq c$
$a \leq b+c$

```
cibo >= prendi("vivande")
cibo >= prendi("pane")
cibo <= prendi("vivande") +
      prendi("pane")
```

ValoreTot ← ValoreTot - 10*(1-cibo)

Risultato

(file: **zaino3.mos**)

```
Soluzione:
 Valore ottimo: 112
Peso dello zaino: 8.6
 prendi(sacco a pelo): 1
 prendi(materassino): 1
 prendi(bussola): 1
 prendi(giacca): 0
 prendi(fornello): 0
 prendi(piatti): 0
 prendi(posate): 0
 prendi(vivande): 0
 prendi(ricambi): 1
 prendi(sapone): 1
 prendi(asciugamano): 1
 prendi(pane): 1
 prendi(accendino): 1
```

Xpress-IVE

Problemi di trasporto (*ground transport problems*)

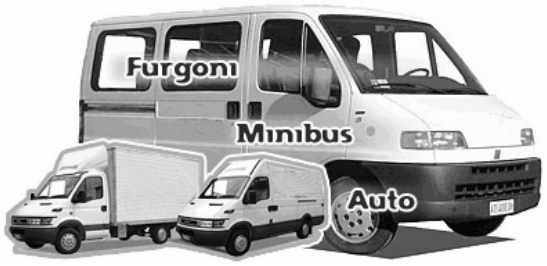
Problemi di Trasporto

Sono esempi classici in cui l'ottimizzazione fornisce soluzioni utili e in maniera molto efficiente

Tipicamente l'obiettivo è minimizzare il costo dovuto al trasporto di merci su gomma o rotaia, pianificando in maniera ottima
(1) le locazioni per i depositi e le fabbriche e
(2) gli itinerari da seguire.

Esempio: Autonoleggio

Una compagnia di autonoleggio ha una flotta di 146 veicoli distribuiti su 17 agenzie. Dopo alcuni mesi di movimenti di auto da un'agenzia all'altra, si vuole determinare come ridistribuire i veicoli in modo che ve ne siano un certo numero per agenzia, minimizzando i costi complessivi, assumendo un costo di 0.50 € per km,



Distanze fra città

	AN	AO	BA	BO	CB	FI	GE	AQ	MI	NA	PG	PZ	RC	RM	TO	UD	VE
Ancona	-	609	466	572	764	260	188	423	257	401	143	464	855	308	544	412	300
Aosta	609	-	1063	536	1292	471	785	783	354	940	622	1060	1383	740	113	556	449
Bari	466	1063	-	1026	348	662	407	877	711	255	567	140	439	412	998	866	754
Bologna	572	536	1026	-	923	101	393	206	206	570	253	668	1014	370	327	265	158
Campo-basso	764	1292	348	923	-	472	702	736	570	137	377	201	542	222	857	725	613
Firenze	260	471	662	101	472	-	230	315	295	468	151	607	912	268	400	365	258
Genova	188	785	407	393	702	230	-	555	253	707	389	846	1150	507	170	509	402
L'Aquila	423	783	877	206	736	315	555	-	433	238	175	377	681	122	720	588	476
Milano	257	354	711	206	570	295	253	433	-	764	447	875	1208	564	138	383	276
Napoli	401	940	255	570	137	468	707	238	764	-	373	158	462	219	869	834	727
Perugia	143	622	567	253	377	151	389	175	447	373	-	512	817	173	551	439	327
Potenza	464	1060	140	668	201	607	846	377	875	158	512	-	390	358	995	863	752
Reggio Calabria	855	1383	439	1014	542	912	1150	681	1208	462	817	390	-	662	1312	1278	1171
Roma	308	740	412	370	222	268	507	122	564	219	173	358	662	-	669	634	527
Torino	544	113	998	327	857	400	170	720	138	869	551	995	1312	669	-	512	405
Udine	412	556	866	265	725	365	509	588	383	834	439	863	1278	634	512	-	127
Venezia	300	449	754	158	613	258	402	476	276	727	327	752	1171	527	405	127	-

Agenzie di autonoleggio

Auto disponibili:

AN	AO	BA	BO	CB	FI	GE	AQ	MI	NA	PG	PZ	RC	RM	TO	UD	VE
8	13	4	8	12	2	14	11	15	7	10	9	3	12	11	4	3

Auto richieste:

AN	AO	BA	BO	CB	FI	GE	AQ	MI	NA	PG	PZ	RC	RM	TO	UD	VE
10	6	8	11	9	7	15	7	9	12	3	1	2	16	15	4	4

Modello matematico

Vincoli da soddisfare:

"... ve ne siano un certo numero per agenzia..."

Variabili di ottimizzazione

"...determinare come ridistribuire i veicoli ..."

Funzione obiettivo:

"... minimizzando i costi complessivi ..."

Modello matematico

Variabili di ottimizzazione

$$\text{move}_{ij} \in \{CIT, CIT\}$$

$CIT = \{1,2,\ldots,17\}$

move_{ij} = quante auto spostare dalla città $\#i$ alla città $\#j$

Vincolo di interezza: $\text{move}_{ij} \in \mathbb{N}$

Vincolo di bilanciamento

$$\text{stock}_i = \text{AVAIL}_i + \sum_{j \in CIT} \text{move}_{ji} - \sum_{j \in CIT} \text{move}_{ij}$$

$$\text{stock}_i \geq \text{NEED}_i \quad \forall i \in CIT$$

Variabili di contabilità

Nota che abbiamo una variabile di contabilità:

$$\text{stock}_i$$

Per come abbiamo definito il problema, le variabili stock_i non hanno necessità di essere definite come intere !

$$\text{stock}_i = \text{AVAIL}_i + \sum_{j \in CIT} \text{move}_{ji} - \sum_{j \in CIT} \text{move}_{ij}$$

Il vincolo di contabilità impone già automaticamente la natura intera di stock_i

Funzione obiettivo

Il costo totale per il trasporto dei veicoli da un'agenzia all'altra:

$$\text{Cost} = \sum_{j,i \in CIT} 0.5 * \text{Dist}_{ij} * \text{move}_{ji}$$

Formulazione del problema

$$\text{min} \quad \text{Cost} = \sum_{j,i \in CIT} 0.5 * \text{Dist}_{ij} * \text{move}_{ji}$$

$$\text{sogg. a} \quad \text{stock}_i = \text{AVAIL}_i + \sum_{j \in CIT} \text{move}_{ji} - \sum_{j \in CIT} \text{move}_{ij}$$

$$\text{stock}_i \geq \text{NEED}_i \quad \forall i \in CIT$$

$$\text{move}_{ij} \in \mathbb{N} \quad \forall i,j \in CIT$$

Modello MOSEL

```
declarations
        CIT = 1..17
        DIST : array(CIT,CIT) of integer
        AVAIL : array(CIT) of integer
        NEED : array(CIT) of integer
        move : array(CIT,CIT) of mpvar
        stock : array(CIT) of mpvar
end-declarations
initializations from 'ground.dat'
        DIST AVAIL NEED
end-initializations
Cost := sum(i,j in CIT) 0.5*move(i,j)*DIST(i,j)
forall (i in CIT)
        stock(i) = AVAIL(i) + sum(j in CIT) move(j,i)
                            - sum(j in CIT) move(i,j)
forall (i in CIT)
        stock(i) >= NEED(i)
forall (i,j in CIT)
        move(i,j) is_integer
minimize(Cost)
end-model
```

Solo queste sono dichiarate come variabili intere

Soluzione

```
Il costo totale e' 2207.5 euro
Mosse ottime:
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  0  0
  0  0  0  3  0  0  1  0  0  0  0  0  0  0  0  0  1
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  2  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0
  0  0  4  0  0  0  0  0  0  2  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
Dopo gli spostamenti:
in stock / richieste

10/10, 9/6, 8/8, 11/11,
9/9, 7/7, 15/15, 7/7,
10/9, 12/12, 3/3, 3/1,
3/2, 16/16, 15/15, 4/4,
4/4
```


Problemi di trasporto

I problemi di trasporto godono di una proprietà molto interessante:

"Il metodo del simplesso per risolvere un LP a variabili continue trova sempre una soluzione intera"

Questo vuol dire che i problemi di trasporto, sebbene siano definiti come problemi MILP (=problemi difficili), possono essere risolti come LP (=problemi facili).

MOSEL

```
declarations
        CIT = 1..17
        DIST : array(CIT,CIT) of integer
        AVAIL : array(CIT) of integer
        NEED : array(CIT) of integer
        move : array(CIT,CIT) of mpvar
        stock : array(CIT) of mpvar
end-declarations
initializations from 'ground.dat'
        DIST AVAIL NEED
end-initializations
Cost := sum(i,j in CIT) 0.5*move(i,j)*DIST(i,j)
forall (i in CIT)
        stock(i) = AVAIL(i) + sum(j in CIT) move(j,i)
                            - sum(j in CIT) move(i,j)
forall (i in CIT)
        stock(i) >= NEED(i)


minimize(Cost)
end-model
```

È stato eliminato il vincolo di interezza

Soluzione

Risultato MILP

Il costo totale e' **2207.5** euro

Mosse ottime:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	**4**	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	**3**	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	**3**	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	**5**	0	0	0	0	0	0	0	0	0	0	0
0	0	**4**	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0

Risultato LP

Il costo totale e' **2207.5** euro

Mosse ottime:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	**4**	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	**3**	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	**4**	0	0
0	0	0	**3**	0	0	1	0	0	0	0	0	0	0	0	0	**1**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	**5**	0	0	0	0	0	0	0	0	0	0	0
0	0	**4**	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0

i risultati sono identici

Linearizzazione di funzioni obiettivo

Funzioni obiettivo razionali

Abbiamo già visto vincoli sulla qualità e di miscelazione, che sono vincoli sul **rapporto di variabili**

Esempio: vincolo sulla qualità

$$\frac{\sum_{i=1}^{N} c_i x_i}{\sum_{i=1}^{N} x_i} \leq c_{\text{media}}$$

Supponendo $\sum_{i=1}^{N} x_i > 0$, è possibile riscrivere il vincolo come vincolo lineare:

$$\sum_{i=1}^{N} c_i x_i \leq c_{\text{media}} \sum_{i=1}^{N} x_i$$

Vediamo adesso come trattare rapporti di variabili nella funzione obiettivo

Funzioni obiettivo razionali

Esempio:

La ditta Metalli S.p.A. ha ricevuto un ordine di 500 tonnellate di acciaio da usare per la costruzione di una nave. L'acciaio deve avere determinate caratteristiche di composizione.

La ditta dispone di sette tipi diversi di materiale grezzo da utilizzare per la produzione di questo acciaio.

L'obiettivo è determinare la composizione che minimizza la percentuale di zolfo, soddisfacendo i vincoli sulle caratteristiche di composizione.

Dati a disposizione

Mat. grezzo	C%	Cu%	Mn%	Tons	S%
Iron 1	2.5	0	1.3	400	0.5
Iron 2	3	0	0.8	300	0.76
Iron 3	0	0.3	0	600	0.3
Copper 1	0	90	0	500	1.2
Copper 2	0	96	1.2	200	1.1
Aluminium 1	0	0.4	1.2	300	0.3
Aluminium 2	0	0.6	0	250	0.45

Composizioni %, quantità disponibili, % di zolfo

Funzione obiettivo razionale

$$\min \text{Zolfo\%} = \frac{\sum S_j * \text{use}_j}{\sum \text{use}_j}$$

funzione non lineare delle variabili !

sogg. a: $\sum A_{ij}\text{use}_j \leq b_i$

rappresentano i vincoli originali del problema →

$\sum_{j=1}^{7} \text{use}_j = \text{produce}$
produce ≥ 500
for $j = 1 \ldots 7$
  $\text{use}_j \leq R_j$
end
for $i = 1 \ldots 3$
  $\sum_{j=1}^{7} P_{ji}\text{use}_j \geq Pmin_i * \text{produce}$
  $\sum_{j=1}^{7} P_{ji}\text{use}_j \leq Pmax_i * \text{produce}$
end

Cambio di variabili

$$d = \frac{1}{\sum \text{use}_j}$$
$$x_j = \text{use}_j * d$$

1) abbiamo aggiunto una nuova variabile: d

2) abbiamo abbandonato le vecchie variabili use_j per sostituirle con le nuove variabili x_j

Cambio di variabili

Considerando la variabile d come libera, otteniamo un vincolo lineare sulle variabili x_j:

$$\begin{array}{rcl} d & = & \frac{1}{\sum \text{use}_j} \\ x_j & = & \text{use}_j * d \end{array} \quad \Rightarrow \quad 1 = \sum \text{use}_j * d = \sum x_j$$

$$\Rightarrow \boxed{\sum x_j = 1}$$

nuovo vincolo lineare
da aggiungere al problema

Moltiplicando i vincoli originali a destra e a sinistra per la quantità d (d>0), otteniamo dei vincoli equivalenti in x,d:

$$\sum A_{ij} \text{use}_j \leq b_i \quad \Rightarrow \quad \sum A_{ij} x_j \leq b_i * d$$

Cambio di variabili

Funzione obiettivo:

$$\min \text{Zolfo\%} = \frac{\sum S_j * \text{use}_j}{\sum \text{use}_j} = \sum S_j * x_j$$

Tutti i vincoli (=quelli originali + il nuovo vincolo) possono essere espressi come un insieme di disuguaglianze lineari sulle variabili x_j, d.

$$\sum A_{ij} x_j \leq b_i * d$$

Programma Lineare

Costi variabili 1 (*economies of scale*)

Considera il seguente esempio: i prezzi del legno sono fissati come segue:

0-100 kg	0.99 € /kg
100-200 kg	0.95 €/kg
200-400 kg	0.90 €/kg
+400 kg	0.80 €/kg

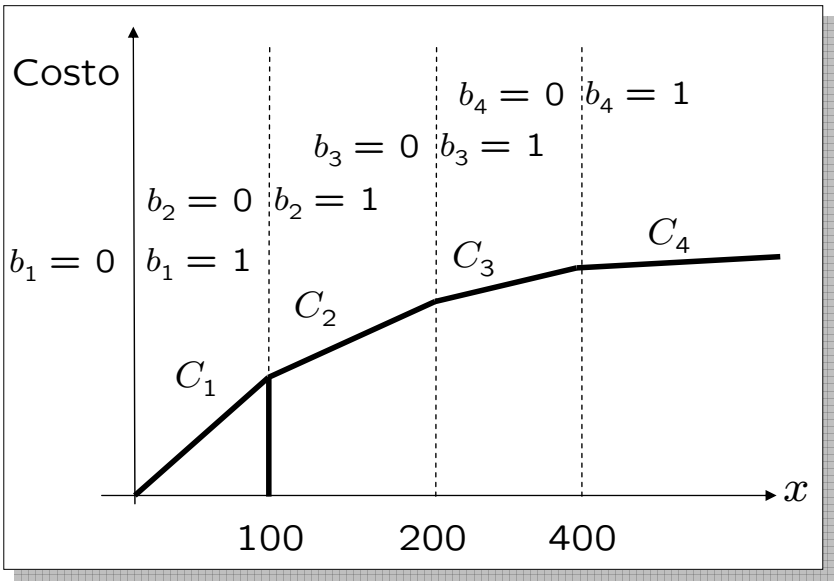
scala dei prezzi

Nota: non vengono effettuati sconti in blocco! I primi 100 kg sono sempre pagati 99 €, indipendentemente dal fatto che se ne comprino quantità maggiori

Come esprimere la funzione di costo ?

Costi variabili 1

Occorre introdurre delle variabili continue x_1, x_2, x_3, x_4 e binarie b_1, b_2, b_3, b_4



$x = x_1 + x_2 + x_3 + x_4$

x_i = quantità acquistata al prezzo C_i

Costi variabili 1

$$\text{Costo} = C_1x_1 + C_2x_2 + C_3x_3 + C_4x_4$$

$$\downarrow$$

$$\begin{gathered}
100 * b_2 \leq x_1 \leq 100 * b_1 \\
(200 - 100) * b_3 \leq x_2 \leq (200 - 100) * b_2 \\
(400 - 200) * b_4 \leq x_3 \leq (400 - 200) * b_3 \\
x_4 \leq X_{max} * b_4 \\
b_1 \geq b_2 \geq b_3 \geq b4 \\
x_1, x_2, x_3, x_4 \geq 0
\end{gathered}$$

Costi variabili 1

Esempio:

$$\begin{gathered}
100 * b_2 \leq x_1 \leq 100 * b_1 \\
(200 - 100) * b_3 \leq x_2 \leq (200 - 100) * b_2 \\
(400 - 200) * b_4 \leq x_3 \leq (400 - 200) * b_3 \\
x_4 \leq X_{max} * b_4 \\
b_1 \geq b_2 \geq b_3 \geq b4
\end{gathered}$$

[1000]
[1100]
[**1110**]
[1111]

x_1=100, x_2=100, x_3 minore di 200

Costi variabili 2

Considera il seguente esempio: i prezzi del legno sono fissati come segue:

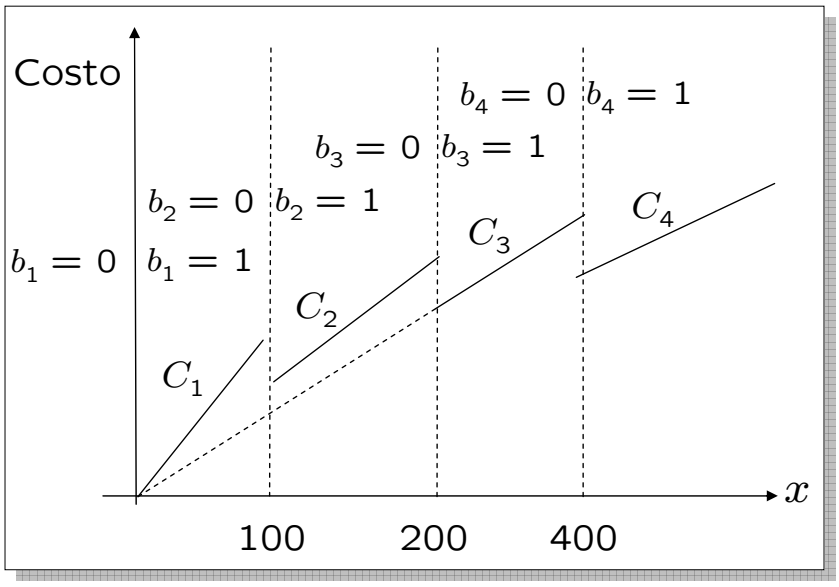
0-100 kg	0.99 € /kg
100-200 kg	0.95 €/kg
200-400 kg	0.90 €/kg
+400 kg	0.80 €/kg

scala dei prezzi

Nota: vengono effettuati sconti in blocco! Maggiore è la quantità acquistata, minore è il costo per kg

Come esprimere la funzione di costo ?

Costi variabili 2



$x = x_1 + x_2 + x_3 + x_4$

$x_i =$ quantità acquistata al prezzo C_i

Costi variabili 2

Definiamo le variabili b e x esattamente come prima:

$$
\begin{gathered}
100 * b_2 \leq x_1 \leq 100 * b_1 \\
(200-100) * b_3 \leq x_2 \leq (200-100) * b_2 \\
(400-200) * b_4 \leq x_3 \leq (400-200) * b_3 \\
x_4 \leq X_{max} * b_4 \\
b_1 \geq b_2 \geq b_3 \geq b4 \\
x_1, x_2, x_3, x_4 \geq 0
\end{gathered}
$$

e riscriviamo il costo come:

$$
\begin{aligned}
\text{Costo} = \; & C_1 x * (b_1 - b_2) + C_2 x * (b_2 - b_3) \\
& + C_3 x * (b_3 - b_4) + C_4 x * b_4
\end{aligned}
$$

Costi variabili 2

$$
\begin{aligned}
\text{Costo} = \; & C_1 x * (b_1 - b_2) + C_2 x * (b_2 - b_3) \\
& + C_3 x * (b_3 - b_4) + C_4 x * b_4
\end{aligned}
$$

Il vincolo b_1 , b_2 , b_3 , b_4 impone che le uniche combinazioni possibili siano [1000],[1100],[1110],[1111]

$\Rightarrow$ uno soltanto dei quattro termini sarà non nullo

OK, ma il costo non è lineare !

Abbiamo il **prodotto fra variabili continue e binarie**

Prodotto variabili continue-binarie

Tecnica del "big-M":

$$z = x * b$$

$x \geq 0$
$z \geq 0$
$b \in \{0,1\}$

può essere modellato come

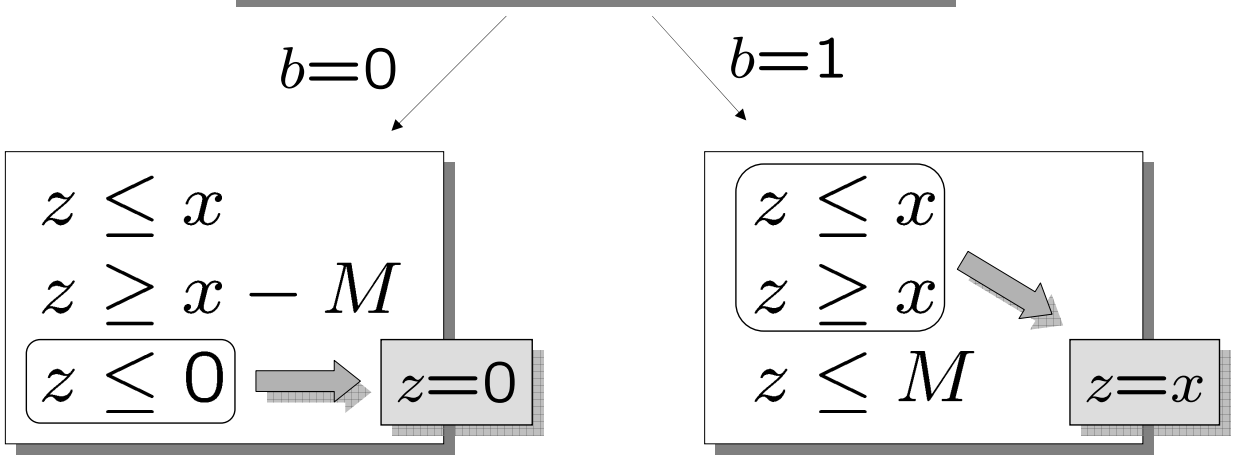
$$\begin{aligned} & z \leq x \\ & z \geq x - M * (1 - b) \\ & z \leq M * b \end{aligned}$$

sotto l'ipotesi che M sia scelto sufficientemente grande, in modo che sia sempre $x \leq M$

Prodotto variabili continue-binarie

$$\begin{aligned} & z \leq x \\ & z \geq x - M * (1 - b) \\ & z \leq M * b \end{aligned}$$

$x \geq 0$
$z \geq 0$
$b \in \{0,1\}$



Vincoli di cardinalità (*counting*)

È un vincolo (molto ricorrente) che limita il **numero** di scelte discrete che possiamo decidere.

Associamo ad ogni opzione i, i=1,2,...,N, una variabile binaria b_i, $b_i \in \{0,1\}$

$$\sum_{i=1}^{N} b_i \leq M$$ Possiamo scegliere al più M opzioni

$$\sum_{i=1}^{N} b_i = M$$ Dobbiamo scegliere esattamente M opzioni

$$\sum_{i=1}^{N} b_i \geq M$$ Dobbiamo scegliere almeno M opzioni

Problema del prestito

È un vincolo molto ricorrente. Facciamo un esempio:

Vogliamo aprire tre punti vendita, uno a Roma, uno a Parigi e uno a Madrid. Per realizzare il progetto abbiamo bisogno rispettivamente di 2 M€, 2.5 M€ e 1.8 M€. Dobbiamo scegliere a quali banche rivolgersi per un prestito. In base a calcoli sui fattori di rischio, ogni banca ha interessi diversi a seconda del paese nel quale il progetto verrà realizzato. Siamo però costretti a scegliere al più due banche per i prestiti. Ad una singola banca non possiamo chiedere più di 3.5 M€.

Vincolo di cardinalità

Problema del prestito

Interessi applicati dalle banche e capitali necessari

	Roma	Parigi	Madrid
Monte dei Paschi	5%	6.1%	6%
Cassa di Risparmio di Firenze	6.3%	4.3%	5.6%
Abbey National Bank	3%	5%	6%
Capitale:	2 M€	2.5 M€	1.8 M€

Problema del prestito

$$\min \sum_{i \in \text{Shops}} \sum_{j \in \text{Banche}} prestito_{ij} * \frac{I_{i,j}}{1-(1+I_{i,j})^{-N}}$$

soggetto a:

$$\sum_{j \in \text{Banche}} prestito_{ij} = \text{Prezzo}_i, \forall i \in \text{Shops}$$

$$\sum_{i \in \text{Shops}} prestito_{ij} \leq \text{Prestito}_{\max} * b_j, \forall j \in \text{Banche}$$

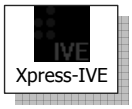
$$\sum_{j \in \text{Banche}} b_j \leq 2$$

Vincolo di cardinalità

Problema del prestito - Risultato

```
Interessi complessivi: 948171
-----------------------
Roma:   banca 3=2 MEUR;
Londra: banca 2=2.5 MEUR;
Madrid: banca 2=1 MEUR;   banca 3=0.8 MEUR;
```

(file: **prestiti.mos**)



Vincoli di "or" logico

Anche questi sono vincoli molto ricorrenti. Hanno bisogno di variabili binarie per essere modellati.

Esempio:

Siamo interessati ad investire del denaro per scopi pubblicitari, al fine di ottenere un buon impatto commerciale in TV o in radio...

Basta che sia soddisfatto almeno uno dei due vincoli

Vincoli di "or" logico

Impatto minimo = 1.5

	TV	Radio	Costo
Mediaset	0.5	0.3	250
RAI	0.4	0.35	200
Radio Capital	0.0	0.5	50
MTV	0.3	0.2	100

Vincoli di "or" logico

Vincoli sulle variabili $buy(i)$:

$$\sum_{i \in MEDIA} buy(i) * \text{Impact}(i, TV) \geq \text{Impact}_{\text{min}}$$

oppure

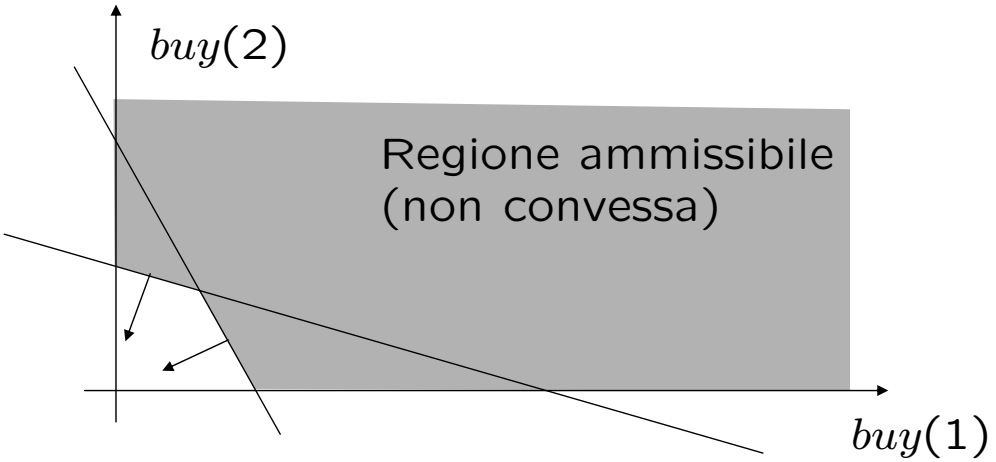
$$\sum_{i \in MEDIA} buy(i) * \text{Impact}(i, RADIO) \geq \text{Impact}_{\text{min}}$$

$$\begin{aligned} \sum_{i \in MEDIA} buy(i) * \text{Impact}(i, TV) &\geq b * \text{Impact}_{\text{min}} \\ \sum_{i \in MEDIA} buy(i) * \text{Impact}(i, RADIO) &\geq (1-b) * \text{Impact}_{\text{min}} \end{aligned}$$

$b \in \{0, 1\}$

Vincoli di "or" logico

Se almeno uno dei due vincoli è soddisfatto, allora esiste un valore della variabile binaria b ammissibile. Viceversa, se entrambe i vincoli non possono essere soddisfatti, non esistono valori ammissibili per b.



Problemi di assegnamento

Gestione del personale

Pianificazione dei turni di lavoro (*timetabling and personnel planning*)

La gestione del personale può essere vista come un problema di distribuzione di risorse.

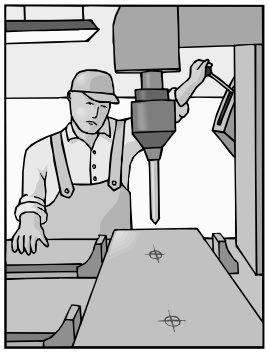
I dati a disposizione tipicamente sono:

- tassi di produttività
- ore lavorative
- vincoli sul personale
- ...

Le tecniche di ottimizzazione risultano molto utili per risolvere tali problemi, anche se occorre fare attenzione: gli uomini non sono risorse, quindi non dobbiamo sottovalutare eventuali risvolti psicologici

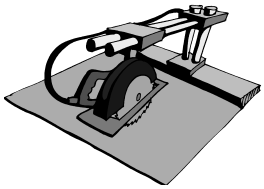
Assegnamento operatori/macchine

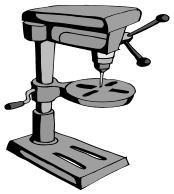
In un'officina esistono sei macchine, a ciascuna delle quali occorre assegnare un operatore. Sono stati prescelti sei operai, per ognuno dei quali è stato effettuato un test di produttività su ciascuna macchina. Le macchine lavorano in parallelo, quindi la produttività complessiva è la somma delle produttività di ciascuna macchina. L'obiettivo è assegnare il personale in modo da massimizzare la produttività

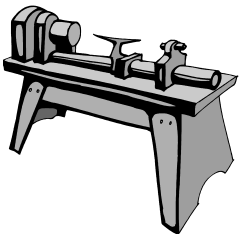


Assegnamento operatori/macchine

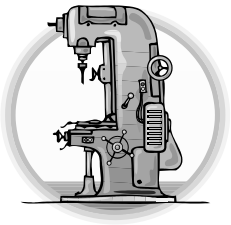
diversi operai ...

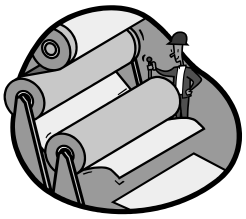






... per diverse macchine





Dati di produttività

Macchina

Operaio	1	2	3	4	5	6
1	13	24	31	19	40	29
2	18	25	30	15	43	22
3	20	20	27	25	34	33
4	23	26	28	18	37	30
5	28	33	34	17	38	20
6	19	36	25	27	45	24

Pezzi prodotti all'ora

Unità?

Problema dei dati

Soluzione euristica

Molti problemi di ottimizzazione ammettono soluzioni di tipo euristico in grado di soddisfare i vincoli imposti.

Soluzione euristica:

> "Iniziando dalla macchina numero 1, assegna a ogni macchina l'operaio che ha il tasso di produttività più alto"

La soluzione euristica è un algoritmo molto semplice che può essere implementato anche "a mano"

NOTA: la soluzione dipende dall'ordine scelto per le macchine. Inoltre, molto probabilmente, la soluzione non sarà ottimale.

Soluzione euristica in MOSEL

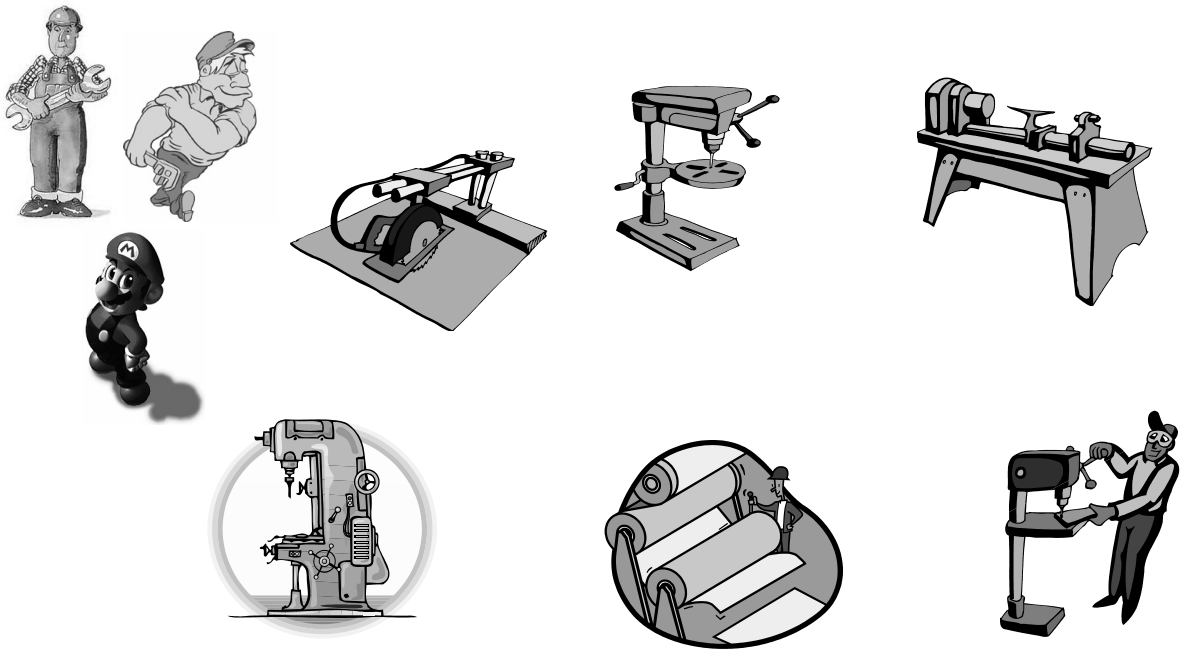
```
declarations
   ALLP, ALLM: set of integer        ! Copia degli insiemi OPERAI e MACCHINE
   HProd: integer                    ! Valore totale di produttivita'
   pmax,omax,mmax: integer
  end-declarations
 ! Copia gli insiemi di operai e macchine
  forall(p in OPERAI) ALLP+={p}
  forall(m in MACCHINE) ALLM+={m}
 ! Assegna operai alle macchine finche' esistono macchine
  while (ALLP<>{}) do
   pmax:=0; mmax:=0; omax:=0
 ! Trova la produttivita' piu' elevata fra quelle
   forall(p in ALLP, m in ALLM)
    if PRODUT(p,m) > omax then
     omax:=PRODUT(p,m)
     pmax:=p; mmax:=m
    end-if

   HProd+=omax                       ! Lo aggiunge alla produttivita' totale
   ALLP-={pmax}; ALLM-={mmax}        ! Rimuove la macchina e l'operai dai rispettivi insiemi

   writeln("  ",pmax, " e' assegnato alla macchina ", mmax, " (", omax, ")")
  end-do

  writeln("  Produttivita' totale: ", HProd)
```

NON CI SONO VARIABILI DI OTTIMIZZAZIONE

Nota: La soluzione euristica può essere implementata in qualsiasi linguaggio di programmazione, ad esempio in MATLAB

Soluzione ottima

Utilizziamo un modello matematico per risolvere il problema di assegnamento in maniera ottimale



Modello matematico

"... sei macchine, a ciascuna delle quali occorre assegnare un operatore..." — Variabili di ottimizzazione

"... Sono stati prescelti sei operai ..." — Vincoli

"... per ognuno dei quali è stato effettuato un test di produttività ..." — Dati del problema

"... massimizzare la produttività ..." — Funzione obiettivo

Modello matematico

Prima di tutto dobbiamo scegliere le variabili di ottimizzazione:

"...sei macchine, a ciascuna delle quali occorre assegnare un operatore..."

"...Sono stati prescelti sei operai..."

Difficile rappresentare poi il vincolo di assegnare soltanto UN operatore per macchina

~~Reali~~ ?

~~Intere~~ ?

$\text{macchina}_p \in \{1,2,3,4,5,6\}$

Binarie?

$\text{assegna}_{pm} \in \{0,1\}$

$p = 1,\ldots,6 \quad m = 1,\ldots,6$

Modello matematico

Vincolo:

... ogni macchina ha un solo operaio ...

$$\forall m \in MACCHINE : \sum_{p \in OPERAI} \text{assegna}_{pm} = 1$$

... ogni operaio ha assegnata una sola macchina ...

$$\forall p \in OPERAI : \sum_{m \in MACCHINE} \text{assegna}_{pm} = 1$$

vincolo logico di "or" esclusivo

Modello matematico

... ogni macchina ha un solo operaio ...

$$\forall m \in MACCHINE: \sum_{p \in OPERAI} \text{assegna}_{pm} = 1$$

Variabili binarie (0,1)

Almeno una variabile deve essere posta $= 1$

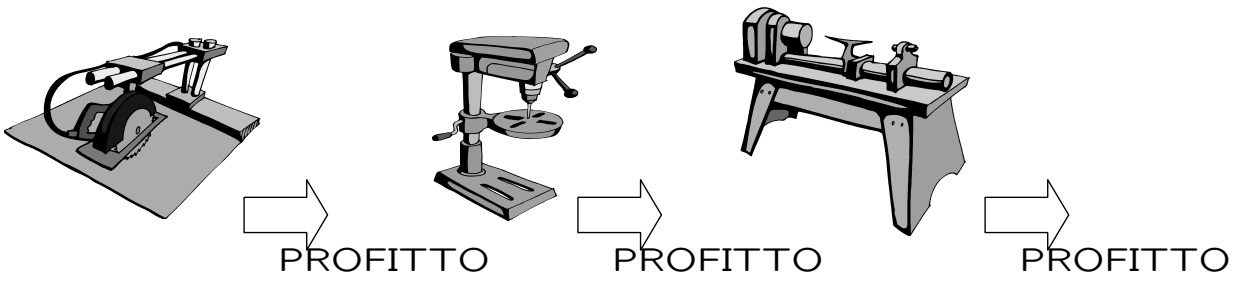
Soltanto una variabile può essere posta $= 1$

Idem per il vincolo "...ogni operaio ha assegnata una sola macchina..."

Funzione obiettivo

$$\text{Profitto}: \sum_p \sum_m \text{prod}_{pm} * \text{assegna}_{pm}$$

Lavorazione in parallelo ...



Problema di ottimizzazione

$$\max \text{Profitto} : \sum_p \sum_m \text{prod}_{pm} * \text{assegna}_{pm}$$

$$\text{sogg. a} \quad \forall m \in MACCHINE : \sum_{p \in OPERAI} \text{assegna}_{pm} = 1$$

$$\forall p \in OPERAI : \sum_{m \in MACCHINE} \text{assegn}_{pm} = 1$$

$$\text{assegna}_{pm} \in \{0, 1\}$$

È un **problema di assegnamento** (**assignment problem**)

Formulazione MOSEL

```
declarations
 OPERAI = 1..6                          ! Personale
 MACCHINE = 1..6                        ! Macchine
 PRODUT: array(OPERAI,MACCHINE) of integer  ! Produttivita'

 assegna: array(OPERAI,MACCHINE) of mpvar  ! 1 se l'operaio e' assegnato
                                           !   alla macchina, 0 altrimenti
end-declarations

! Obiettivo: produttivita' totale
 TotalProd:= sum(p in OPERAI, m in MACCHINE) PRODUT(p,m)*assegna(p,m)

! Una macchina per operaio
 forall(p in OPERAI) sum(m in MACCHINE) assegna(p,m) = 1
! Un operaio per macchina
 forall(m in MACCHINE) sum(p in OPERAI) assegna(p,m) = 1

 forall(p in OPERAI, m in MACCHINE) assegna(p,m) is_binary

! Risoluzione del problema
 maximize(TotalProd)
```

Soluzione

```
Soluzione euristica (macchine
parallele):
  6 e' assegnato alla macchina 5 (45)
  5 e' assegnato alla macchina 3 (34)
  3 e' assegnato alla macchina 6 (33)
  4 e' assegnato alla macchina 2 (26)
  1 e' assegnato alla macchina 4 (19)
  2 e' assegnato alla macchina 1 (18)
  Produttivita' totale: 175
```

```
Soluzione esatta (macchine parallele):
  1 e' assegnato alla macchina 3 (31)
  2 e' assegnato alla macchina 5 (43)
  3 e' assegnato alla macchina 4 (25)
  4 e' assegnato alla macchina 6 (30)
  5 e' assegnato alla macchina 1 (28)
  6 e' assegnato alla macchina 2 (36)
  Produttivita' totale: 193
```

La soluzione ottima è migliore di quella euristica

(e non può che essere cosí !)

Problemi di assegnazione

È uno strumento molto utile per risolvere problemi del tipo:

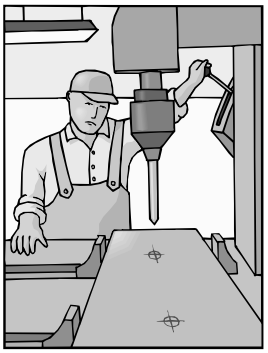
Orari dei corsi
Orari degli esami
Schedulazione dei turni di lavoro
...

Esistono molti esempi di applicazione in vari settori (vedi ad esempio nel testo "Applications of optimization with Xpress-MP")

Nota: quando non si hanno funzioni obiettivo da ottimizzare ma basta solo trovare una soluzione ammissibile, esistono anche altre tecniche modellistiche e risolutive (es: Constraint Logic Programming)

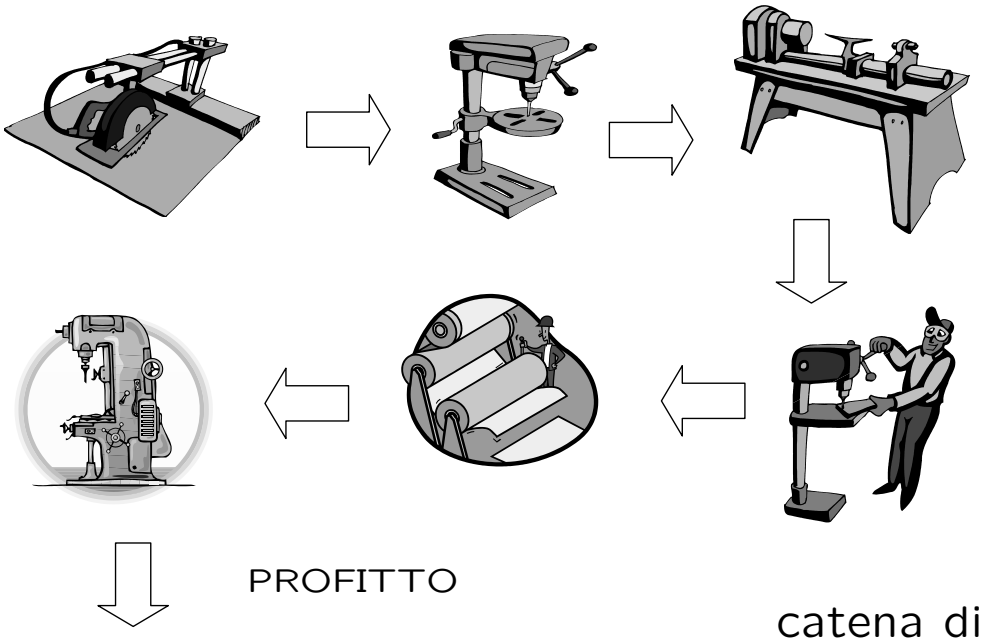
Assegnazione operatori/macchine

In un'officina esistono sei macchine, a ciascuna delle quali occorre assegnare un operatore. Sono stati prescelti sei operai, per ognuno dei quali è stato effettuato un test di produttività su ciascuna macchina. Le macchine lavorano in **serie**, quindi la produttività complessiva è quella della macchina con minor tasso di produttività. L'obiettivo è assegnare il personale in modo da massimizzare la produttività complessiva



Macchine in serie

Produzione in serie



PROFITTO

catena di produzione

Modello matematico

"... sei macchine, a ciascuna delle ... occorre assegnare un operatore ..."

Variabili di ottimizzazione

"... Sono stati prescel... operai ..."

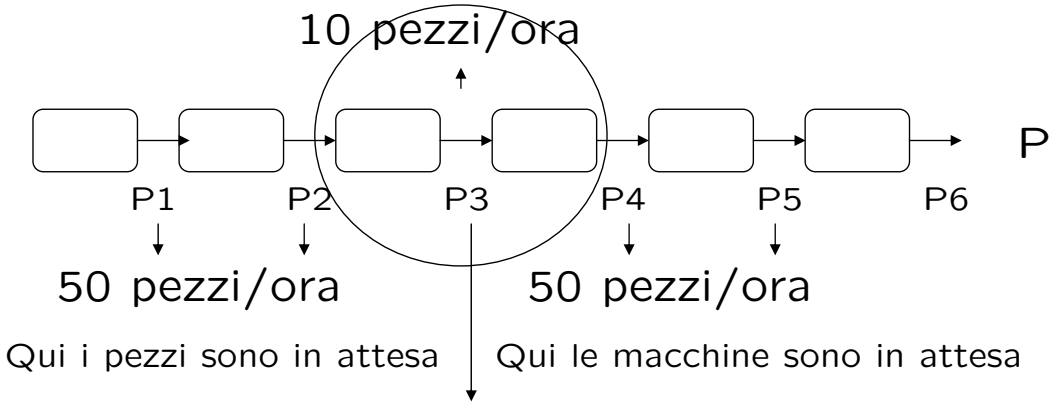
Vincoli

"... per ognun... quali è stato effettuato un test di produttività ..."

Dati del problema

NESSUNA MODIFICA

Funzione obiettivo



Situazione di **Bottleneck**

massimizzare la produttività complessiva =
massimizzare la minima produttività

È un problema di **MaxMin**

Funzione obiettivo MaxMin

$$\forall m \in MACCHINE: \sum_{p \in OPERAI} \text{assegna}_{pm} * prod_{pm} \geq \text{Prod}$$

Per ogni macchina, la sua produttività deve essere maggiore della produttività complessiva della fabbrica

È del tutto simile al caso MinMax: si massimizza una variabile (Prod) che rappresenta un lower bound della produttività di ciascuna macchina (e quindi è minore o uguale al minimo). All'ottimo, tale variabile sarà esattamente la produttività minima (si dimostra per assurdo)

Modello matematico

max Prod

sogg. a $\forall m \in MACCHINE: \sum_{p \in OPERAI} \text{assegna}_{pm} * prod_{pm} \geq \text{Prod}$

$$\forall m \in MACCHINE: \sum_{p \in OPERAI} \text{assegna}_{pm} = 1$$

$$\forall p \in OPERAI: \sum_{m \in MACCHINE} \text{assegn}_{pm} = 1$$

$$\text{assegna}_{pm} \in \{0, 1\}$$

COME PRIMA

Formulazione MOSEL

```
declarations
 OPERAI = 1..6                                ! Personale
 MACCHINE = 1..6                              ! Macchine
 PRODUT: array(OPERAI,MACCHINE) of integer    ! Produttivita'

 assegna: array(OPERAI,MACCHINE) of mpvar     ! 1 se l'operaio e' assegnato
                                              !   alla macchina, 0 altrimenti
 MinProd: mpvar                               ! Produttivita' complessiva
end-declarations

! Produttivita' minima
 forall(m in MACCHINE) sum(p in OPERAI) PRODUT(p,m)*assegna(p,m) >= MinProd

! Una macchina per operaio
 forall(p in OPERAI) sum(m in MACCHINE) assegna(p,m) = 1
! Un operaio per macchina
 forall(m in MACCHINE) sum(p in OPERAI) assegna(p,m) = 1

 forall(p in OPERAI, m in MACCHINE) assegna(p,m) is_binary

! Obiettivo: produttivita' totale
 maximize(MinProd)
```


Soluzione

```
Soluzione esatta (macchine serie):
  1 e' assegnato alla macchina 6 (29)
  2 e' assegnato alla macchina 3 (30)
  3 e' assegnato alla macchina 5 (34)
  4 e' assegnato alla macchina 2 (26)
  5 e' assegnato alla macchina 1 (28)
  6 e' assegnato alla macchina 4 (27)
  Produttivita' totale: 26
```

Esercizio

è sposata con Paolo, mentre Paola è la moglie del fratello di Giovanna.

Come si chiama la moglie di Livio?

Vedete poi la soluzione pubblicata a pagina 46

53107. CALCOLO... ENIGMATICO

A segno uguale corrisponde cifra uguale.

Risolvere il problema utilizzando MOSEL

(file: `calcolo_enigmatico-assegnamento.mos`)

Programmazione Quadratica

Programmazione lineare

minimizza o massimizza $\sum_{j=1}^{N} c_j x_j$ funzione obiettivo

soggetto a

$$\sum_{j=1}^{N} a_{ij} x_j \leq b_i \quad \text{per } i = 1 \dots M_1$$

$$\sum_{j=1}^{N} a_{ij} x_j \geq b_i \quad \text{per } i = M_1 + 1 \dots M_2$$

$$\sum_{j=1}^{N} a_{ij} x_j = b_i \quad \text{per } i = M_2 + 1 \dots M_3$$

$$x_j \geq 0 \quad \text{per } j = 1 \dots N$$

Programmazione quadratica

minimize or maximize $\sum_{j=1}^{N} \sum_{i=1}^{N} H_{ji} x_j x_i + \sum_{j=1}^{N} c_j x_j$ funzione obiettivo

soggetto a

$$\sum_{j=1}^{N} a_{ij} x_j \leq b_i \quad \text{per } i = 1 \dots M_1$$

$$\sum_{j=1}^{N} a_{ij} x_j \geq b_i \quad \text{per } i = M_1 + 1 \dots M_2$$

$$\sum_{j=1}^{N} a_{ij} x_j = b_i \quad \text{per } i = M_2 + 1 \dots M_3$$

$$x_j \geq 0 \quad \text{per } j = 1 \dots N$$

STESSI VINCOLI DELL'LP

Funzione obiettivo

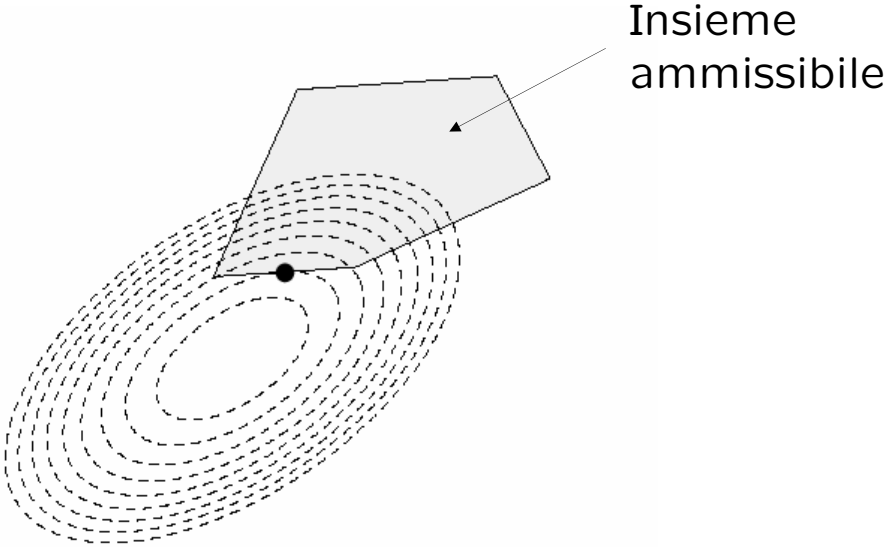
Si hanno dei termini quadratici:

$$\sum_{j=1}^{N}\sum_{i=1}^{N} H_{ji}x_j x_i + \sum_{j=1}^{N} c_j x_j$$

Solitamente, per programmazione quadratica (QP) si intende quella convessa, ovvero quella per cui $\sum_{j=1}^{N}\sum_{i=1}^{N} H_{ji}x_j x_i > 0$ per ogni $[x_1 \ \ldots \ x_n] \neq [0 \ \ldots \ 0]$

Programmazione quadratica

$$\begin{aligned} \min \quad & \frac{1}{2}x'Hx + c'x \\ \text{s.t.} \quad & Ax \leq b, \ x \in \mathbb{R}^n \end{aligned}$$



Programmazione quadratica

Anche la programmazione quadratica è molto diffusa nelle applicazioni e vari risolutori QP sono disponibili in molti pacchetti software per l'ottimizzazione

Xpress-MP, Cplex, Matlab, OOQP, LOQO, NAG, ...

H.D. Mittelmann and P. Spellucci, "Decision Tree for Optimization Software", World Wide Web, http://plato.asu.edu/guide.html

Applicazioni:

- Economia (es: selezione di portafoglio titoli)
- Ingegneria (es: automazione di processo)
- ...

Matlab Optimization Toolbox QP

quadprog

Purpose

Solve the quadratic programming problem

$$\min_x \ \frac{1}{2}x^T Hx + f^T x \quad \text{such that} \quad \begin{array}{l} A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub \end{array}$$

where H, A, and Aeq are matrices, and f, b, beq, lb, ub, and x are vectors.

Syntax

```
x = quadprog(H,f,A,b)
x = quadprog(H,f,A,b,Aeq,beq)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)
[x,fval] = quadprog(...)
[x,fval,exitflag] = quadprog(...)
[x,fval,exitflag,output] = quadprog(...)
[x,fval,exitflag,output,lambda] = quadprog(...)
```

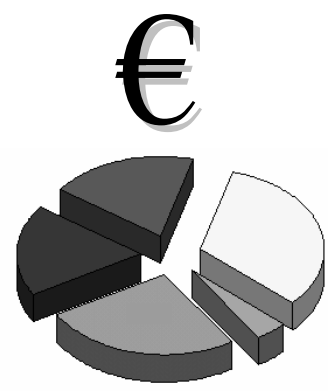
Description

`x = quadprog(H,f,A,b)` returns a vector `x` that minimizes `1/2*x'*H*x + f'*x` subject to `A*x <= b`.

Esistono risolutori QP molto più efficienti !

Scelta portafoglio titoli
(Mean Variance Portfolio Selection)

Decidiamo di investire una certa somma di denaro in titoli di quattro tipi diversi: buoni del tesoro, una fabbrica di calcolatori, una software company, e una produzione cinematografica ad alto rischio.



Definiamo adesso la funzione obiettivo ...

Scelta portafoglio titoli

I vantaggi di questi investimenti non sono indipendenti, per esempio hardware e software sono direttamente correlati, mentre cinema e software sono inversamente correlati (quando gli acquirenti vanno molto al cinema comprano meno videogiochi).

Vogliamo investire la nostra somma in maniera tale da massimizzare il profitto atteso minimizzando però la varianza del profitto stesso (e quindi il rischio).

↓

Espressione quadratica ...

Dati del problema di portfolio

	Hardware	Software	Cinema	BOT
Valore atteso	8	9	12	7
Hardware	4	3	-1	3
Software	3	6	1	-1
Cinema	-1	1	10	-2
BOT	3	-1	-2	3

Termini di covarianza

Funzione obiettivo

$$\sum_{j=1}^{N} \sum_{i=1}^{N} VAR_{ji} * \text{frac}_j * \text{frac}_i$$

frac_i = quota di denaro da investire nell' i-esimo titolo

Modello matematico

Vincoli:

$$\sum_{j=1}^{N} \text{frac}_j * Exp_j \geq E_{min}$$

Vogliamo avere almeno un certo profitto minimo

$$\sum_{j=1}^{N} \text{frac}_j = 1$$

Vogliamo investire tutta la somma

$$\text{frac}_j \geq 0$$

Vincoli di non negatività

Modello matematico

$$\min \sum_{j=1}^{N} \sum_{i=1}^{N} VAR_{ji} * \text{frac}_j * \text{frac}_i$$

$$\text{sogg. a} \quad \sum_{j=1}^{N} \text{frac}_j * Exp_j \geq E_{min}$$

$$\sum_{j=1}^{N} \text{frac}_j = 1$$

$$\text{frac}_j \geq 0$$

MOSEL

```
uses "mmxprs","mmquad"
declarations
        RETMIN = 9
        SECS = 1..4
        RET: array(SECS) of real
        VAR: array(SECS,SECS) of real
        frac: array(SECS) of mpvar
end-declarations
initializations from 'meanvar.dat'
        RET VAR
end-initializations
Variance:= sum(i,j in SECS) VAR(i,j)*frac(j)*frac(i)
sum(i in SECS) frac(i) = 1
sum(i in SECS) frac(i)*R
minimize(Variance)
end-model
```

Usiamo un nuovo tipo di risolutore

```
La varianza del portafoglio e' 1.26601
Il portafoglio ottimo e' costituito
da:
0% di titoli di tipo 1
23.6453% di titoli di tipo 2
30.5419% di titoli di tipo 3
45.8128% di titoli di tipo 4
```

meanvar.mos

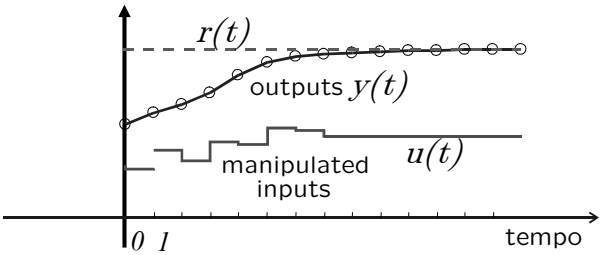
Ottimizzazione dinamica e controllo a orizzonte recessivo

Problema di controllo ottimo

- Dato il modello dinamico

$$\begin{aligned} x(t+1) &= f(x(t),u(t)) \\ y(t) &= g(x(t),u(t) \end{aligned}$$

$$x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ y \in \mathbb{R}^p$$



determinare la sequenza di variabili u manipolate che ottimizza

$$\begin{aligned} \min \quad & \sum_{t=0}^{T} \|y(t)-r(t)\| + \rho\|u(t)\| \\ \text{s.t.} \quad & u_{\min} \le u(t) \le u_{\max} \\ & y_{\min} \le y(t) \le y_{\max} \end{aligned}$$

- Può essere riscritto come problema di ottimizzazione
- Problemi: complessità elevata se T è grande;
  soluzione "open-loop"

Filosofia *Receding Horizon*

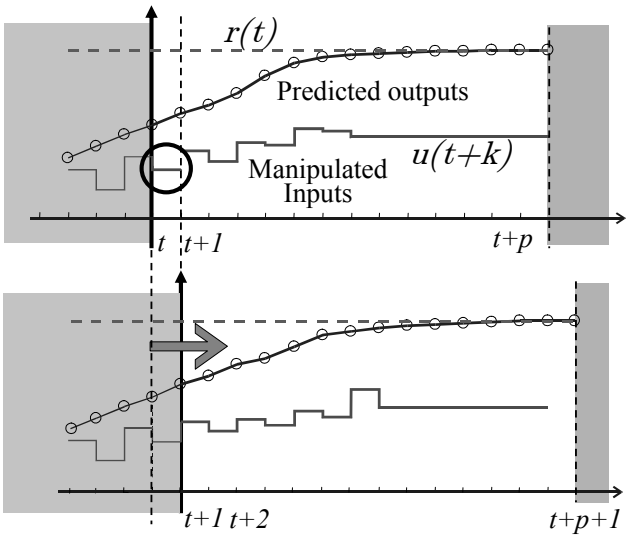
- Al tempo t:

  Si risolve un problema di controllo ottimo su un orizzonte futuro di p passi:

  – minimizza $f(|y-r|,|u|)$

  – soggetto ai vincoli

$$\begin{aligned} u_{\min} &\le u \le u_{\max} \\ y_{\min} &\le y \le y_{\max} \end{aligned}$$

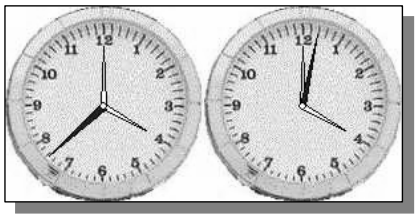


- Si applica soltanto la prima della sequenza ottima di mo$u^*(t)$
- Ottenute nuove misure, si ripete l'ottimizzazione al tempo t+1

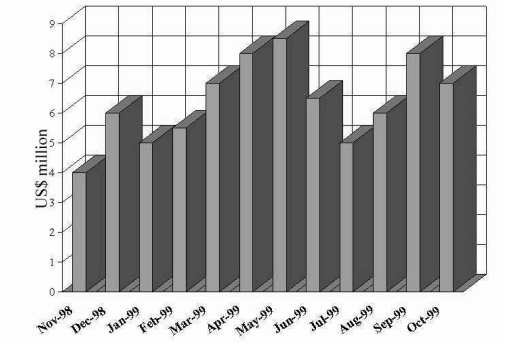
Vantaggio dell'ottimizzazione in linea: **FEEDBACK!**

Receding Horizon - Esempi

- MPC is like playing chess !



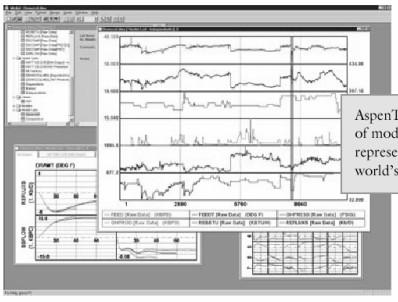
- MPC and investing money



MPC nell'industria

- History: 1979 Dynamic Matrix Control (DMC) by Shell (Motivation: multivariable, constrained)

DMCplus™



The new GUI-based system makes DMCplus easy to use.

AspenTech's installed base of model predictive control represents over 50% of the world's applications.

New Generation Controller

DMCplus is the "new generation" multivariable control product devel- optimization technology and thus also for AspenTech's plant-wide optimiza-

- Particularly suited for problems with
  - many inputs and outputs
  - constraints on inputs, outputs, states
  - varying objectives and limits on variables

  (e.g. because of faults)

Controllo ottimo LQ non vincolato

• Modello lineare:

$$\begin{cases} x(t+1) = Ax(t)+Bu(t) \\ y(t) = Cx(t) \end{cases}$$

$x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$
$y \in \mathbb{R}^p$

• Obiettivo: trova la sequenza $u^*(0)$, $u^*(1)$, ..., $u^*(N-1)$che minimizza

$$J(x(0),U) = \sum_{k=0}^{N-1} x'(k)Qx(k)+u'(k)Ru(k)+x'(N)Px(N)$$

$$U = [u'(0)\ u'(1)\ \ldots\ u'(N-1)]'$$

$u^*(0)$, $u^*(1)$, ..., $u^*(N-1)$ è la sequenza di ingressi che porta lo stato verso l'origine x=0 in maniera "ottima"

Controllo ottimo LQ non vincolato

Il problema di controllo ottimo LQ può essere riscritto come

$$\min J(x(0),U) = \frac{1}{2}U'HU + x'(0)FU + \frac{1}{2}x'(0)Yx(0)$$

L'ottimo viene ottenuto azzerando il gradiente:

$\nabla_U J(x(0),U) = HU + F'x(0) = 0$, e quindi

$$U^* = \begin{bmatrix} u^*(0) \\ u^*(1) \\ \vdots \\ u^*(T-1) \end{bmatrix} = -H^{-1}F'x(0)$$

(questo metodo è conosciuto anche come "*batch least squares*")

Alternativa: trovare la soluzione ottima utilizzando le iterazioni di Riccati (programmazione dinamica)

MPC Lineare

- Modello lineare:

$$\begin{cases} x(t+1) = Ax(t)+Bu(t) \\ y(t) = Cx(t) \end{cases} \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ y \in \mathbb{R}^p$$

- Vincoli:

$$\begin{cases} u_{\min} \le u(t) \le u_{\max} \\ y_{\min} \le y(t) \le y_{\max} \end{cases}$$

- Problema di controllo ottimo (con indice di prestazione quadratico):

$$\min_{u(0),\ldots,u(N-1)} \sum_{k=0}^{N-1} \left[x'(k)Qx(k)+u'(k)Ru(k)\right] + x'(N)Px(N)$$

$$\text{s.t. } u_{\min} \le u(k) \le u_{\max},\ k=0,\ldots,N-1$$
$$y_{\min} \le y(k) \le y_{\max},\ k=1,\ldots,N$$

$$Q=Q' \succeq 0,\ R=R' \succ 0,\ P \succeq 0$$

MPC Lineare

- Sostituzione: $x(k) = A^k x(0) + \sum_{j=0}^{k-1} A^j B u(k-1-j)$

- Problema di ottimizzazione:

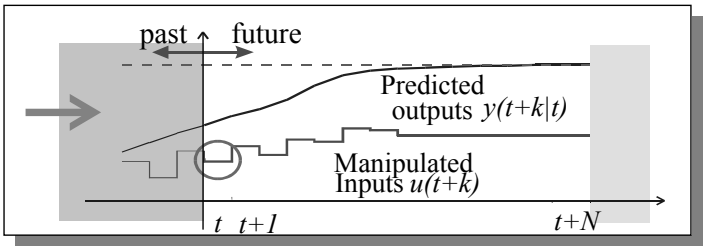
$$V(x(0)) = \tfrac{1}{2}x'(0)Yx(0) + \min_U \tfrac{1}{2}U'HU + x'(0)FU \quad \text{(quadratico)}$$
$$\text{s.t. } GU \le W + Sx(0) \quad \text{(lineare)}$$

Convex QUADRATIC PROGRAM (QP)

- $U \triangleq [u'(0),\ldots,u'(N-1)]' \in \mathbb{R}^s$, $s \triangleq Nm$, è il vettore di ottimizzazione

- $H=H' \succ 0$, e H, F, Y, G, W, S sono ottenuti dai pesi Q, R, P e dalle matrici del modello A, B, C

Algoritmo MPC



<u>Al tempo t:</u>

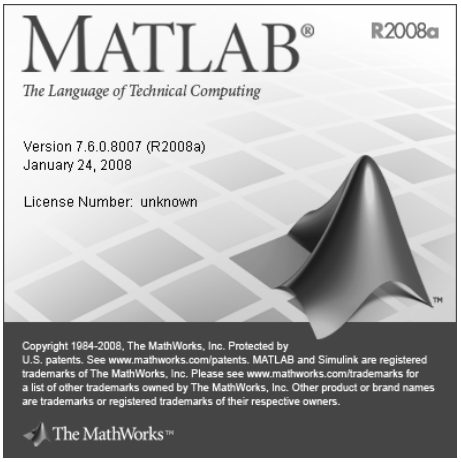
- Misura o stima lo stato corrente $x(t)$
- Risolvi il problema QP $\min_U \ \frac{1}{2}U'HU + x'(t)FU$

  s.t. $GU \le W + Sx(t)$

  e sia U=$\{u^*(0),\ldots,u^*(N\text{-}1)\}$ la soluzione (=controllo ottimo vincolato ad anello aperto su orizzonte finito)
- Applica soltanto $u(t) = u^*(0)$ al processo e scarta gli ingressi ottimi futuri rimanenti
- Vai al tempo t+1

Model Predictive Control Toolbox

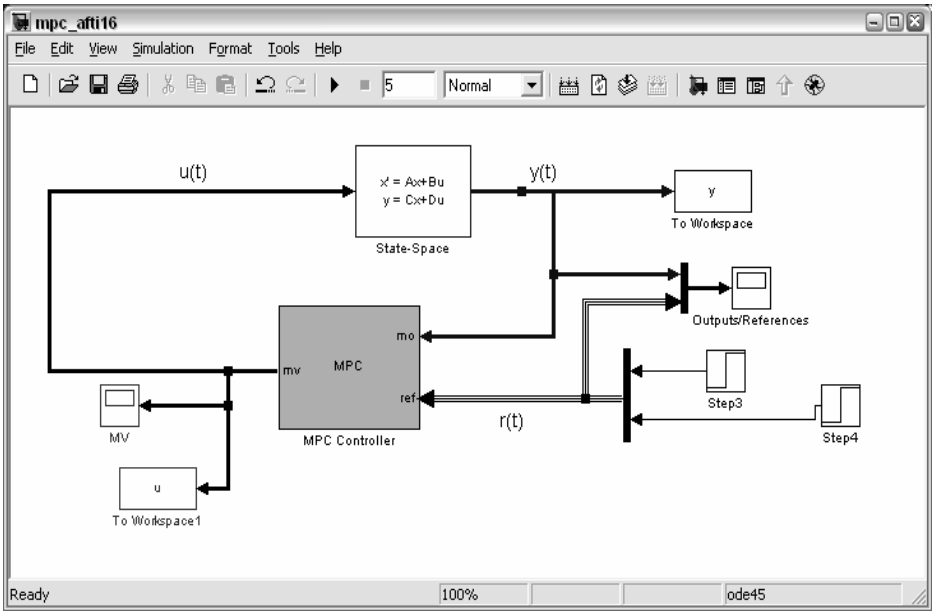
- **MPC Toolbox 2.0** (Bemporad, Ricker, Morari, 1998-today):
  - Object-oriented implementation (MPC object)
  - MPC Simulink Library
  - MPC Graphical User Interface
  - RTW extension (code generation)
  - Linked to OPC Toolbox v2.0.1



Only linear models are handled

http://www.mathworks.com/products/mpc/

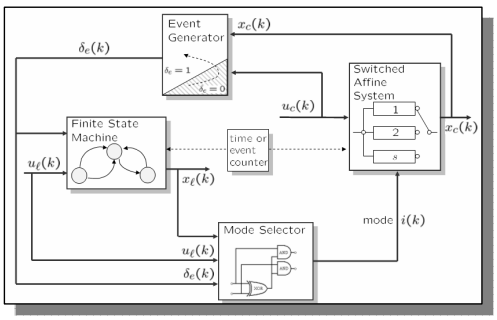
MPC Simulink Library



Software e documentazione: **http://www.mathworks.com/products/mpc/**

Mixed Logical Dynamical Systems

Discrete Hybrid Automaton



HYSDEL
(Torrisi, Bemporad, 2004)

Mixed Logical Dynamical (MLD) Systems (Bemporad, Morari 1999)

$$\begin{aligned} x(t+1) &= Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) + B_5 \\ y(t) &= Cx(t) + D_1u(t) + D_2\delta(t) + D_3z(t) + D_5 \\ E_2\delta(t) + E_3z(t) &\leq E_4x(t) + E_1u(t) + E_5 \end{aligned}$$

Continuous and binary variables

$x \in \mathbb{R}^{n_r} \times \{0,1\}^{n_b}$, $u \in \mathbb{R}^{m_r} \times \{0,1\}^{m_b}$
$y \in \mathbb{R}^{p_r} \times \{0,1\}^{p_b}$, $\delta \in \{0,1\}^{r_b}$, $z \in \mathbb{R}^{r_r}$

- Computationally oriented (mixed-integer programming)

MIQP Formulation of MPC

(Bemporad, Morari, 1999)

$$\min_\xi J(\xi, x(0)) = \sum_{t=0}^{T-1} y'(t)Qy(t) + u'(t)Ru(t)$$

$$\text{subject to } \begin{cases} x(t+1) = Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) + B_5 \\ y(t) = Cx(t) + D_1u(t) + D_2\delta(t) + D_3z(t) + D_5 \\ E_2\delta(t) + E_3z(t) \le E_4x(t) + E_1u(t) + E_5 \end{cases}$$

- Optimization vector:

$$\xi = [u(0), \dots, u(T-1), \delta(0), \dots, \delta(T-1), z(0), \dots z(T-1)]'$$

$$\Rightarrow \quad \min_\xi \frac{1}{2}\xi' H \xi + x(0)' F \xi + \frac{1}{2}x'(0) Y x(0)$$
$$\text{subj. to} \quad G\xi \le W + Sx(t)$$

Mixed Integer Quadratic Program (MIQP)

$$u \in \mathbb{R}^{n_u},\ \delta \in \{0,1\}^{n_\delta},\ z \in \mathbb{R}^{n_z} \Rightarrow \xi \in \mathbb{R}^{(n_u+n_z)T} \times \{0,1\}^{n_\delta T}$$

ξ has both real and $\{0,1\}$ components

MILP Formulation of MPC

(Bemporad, Borrelli, Morari, 2000)

$$\min_\xi \ J(\xi, x(0)) = \sum_{t=0}^{T-1} \|Qy(t)\|_\infty + \|Ru(t)\|_\infty$$
$$\text{subject to MLD model}$$

- Basic trick: introduce slack variables

$$\min_x |x| \quad \Rightarrow \quad \min_{x,\epsilon} \ \epsilon \quad \text{s.t.} \ \epsilon \ge x, \ \epsilon \ge -x$$

$$\begin{cases} \epsilon_k^x \ge \|Qy(t+k|t)\|_\infty \\ \epsilon_k^u \ge \|Ru(t+k)\|_\infty \end{cases} \Rightarrow \begin{cases} \epsilon_k^x \ge [Qy(t+k|t)]_i & i=1,\dots,p \quad k=1,\dots,T-1 \\ \epsilon_k^x \ge -[Qy(t+k|t)]_i & i=1,\dots,p \quad k=1,\dots,T-1 \\ \epsilon_k^u \ge [Ru(t+k)]_i & i=1,\dots,m \quad k=0,\dots,T-1 \\ \epsilon_k^u \ge -[Ru(t+k)]_i & i=1,\dots,m \quad k=0,\dots,T-1 \end{cases}$$

- Optimization vector:

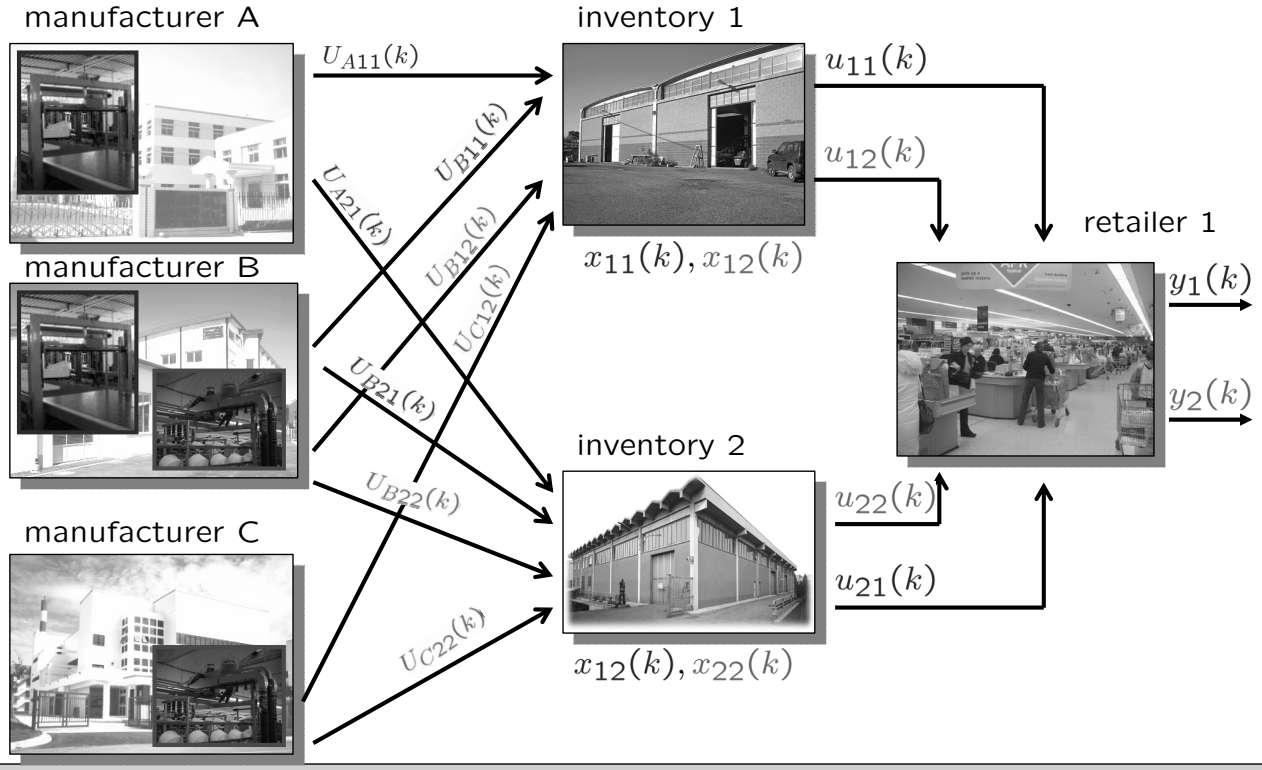
$$\xi = [\epsilon_1^x, \dots, \epsilon_{T-1}^x, \epsilon_0^u, \dots, \epsilon_{T-1}^u, u(0), \dots, u(T-1), \delta(0), \dots, \delta(T-1), z(0), \dots z(T-1)]'$$

$$\Rightarrow \quad \min_\xi \ J(\xi, x(0)) = \sum_{k=0}^{T-1} \epsilon_k^x + \epsilon_k^u \quad \text{s.t.} \ G\xi \le W + Sx(0)$$

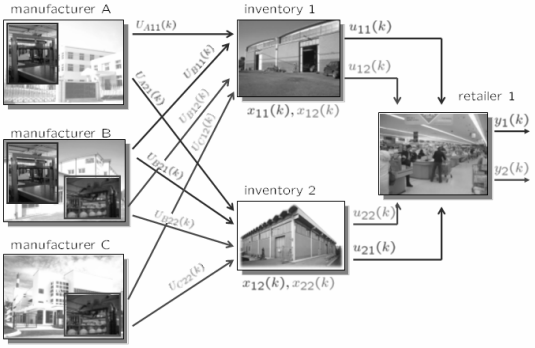
Mixed Integer Linear Program (MILP)

ξ has both real and $\{0,1\}$ components

A Simple Example in Supply Chain Management



System Variables



- <u>continuous states:</u>

  $x_{ij}(k)$ = amount of j hold in inventory i at time k (i=1,2, j=1,2)

- <u>continuous outputs:</u>

  $y_j(k)$ = amount of j sold at time k (i=1,2)

- <u>continuous inputs:</u>

  $u_{ij}(k)$ = amount of j taken from inventory i at time k (i=1,2, j=1,2)

- <u>binary inputs:</u>

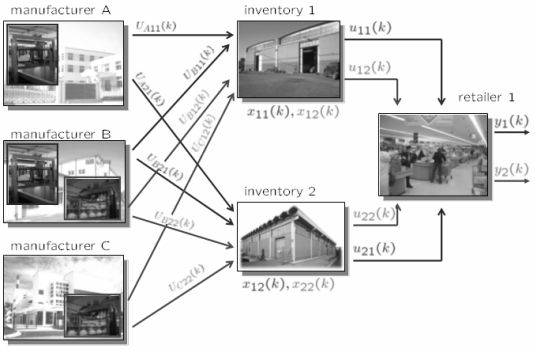
  $U_{Xij}(k)$ = 1 if manufacturer X produces and send j to inventory i at time k

Constraints

- Max capacity of inventory i:

  $0 \leq \sum_j x_{ij}(k) \leq x_{Mi}$

  Numerical values:
  x_{M1}=10, x_{M2}=10



- Max transportation from inventories:

  $0 \leq u_{ij}(k) \leq u_M$

- A product can only be sent to one inventory:

  UA11(k) and UA21(k) cannot be =1 at the same time
  UB11(k) and UB21(k) cannot be =1 at the same time UB12(k) and UB22(k) cannot be =1 at the same time
  UC12(k) and UC22(k) cannot be =1 at the same time

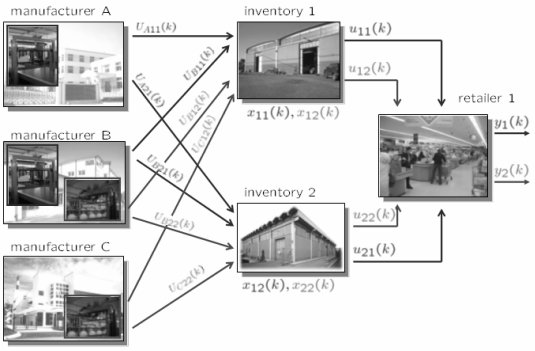
- A manufacturer can only produce one type of product at one time:

  [UB11(k)=1 or UB21(k)=1] and [UB12(k)=1 or UB22(k)=1] cannot be true

Dynamics

$P_{A1}, P_{B1}, P_{B2}, P_{C2}$ = amount of type 1(2) produced by A (B,C) in one time interval

Numerical values:
P_{A1}=4, P_{B1}=6, P_{B2}=7, P_{C2}=3

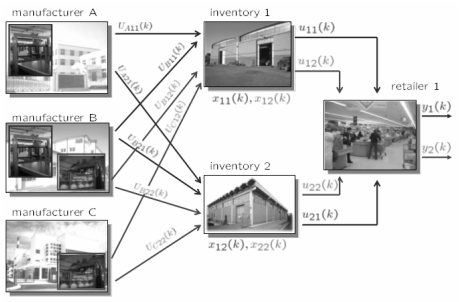


- Level of inventories:

$$\begin{cases} x_{11}(k+1) &= x_{11}(k) + P_{A1}U_{A11}(k) + P_{B1}U_{B11}(k) - u_{11}(k) \\ x_{12}(k+1) &= x_{12}(k) + P_{B2}U_{B12}(k) + P_{C2}U_{C12}(k) - u_{12}(k) \\ x_{21}(k+1) &= x_{21}(k) + P_{A1}U_{A21}(k) + P_{B1}U_{B21}(k) - u_{21}(k) \\ x_{22}(k+1) &= x_{22}(k) + P_{B2}U_{B22}(k) + P_{C2}U_{C22}(k) - u_{22}(k) \end{cases}$$

Hybrid Dynamical Model

```
SYSTEM supply_chain{
INTERFACE {
        STATE { REAL x11   [0,10];
                REAL x12   [0,10];
                REAL x21   [0,10];
                REAL x22   [0,10]; }

        INPUT { REAL u11 [0,10];
                REAL u12 [0,10];
                REAL u21 [0,10];
                REAL u22 [0,10];
                BOOL UA11,UA21,UB11,UB12,UB21,UB22,UC12,UC22; }

        OUTPUT {REAL y1,y2;}

        PARAMETER { REAL PA1,PB1,PB2,PC2,xM1,xM2;}
}
IMPLEMENTATION {

        AUX { REAL zA11, zB11, zB12, zC12, zA21,
                  zB21, zB22, zC22;}


        DA { zA11 = {IF UA11 THEN PA1 ELSE 0};
             zB11 = {IF UB11 THEN PB1 ELSE 0};
             zB12 = {IF UB12 THEN PB2 ELSE 0};
             zC12 = {IF UC12 THEN PC2 ELSE 0};
             zA21 = {IF UA21 THEN PA1 ELSE 0};
             zB21 = {IF UB21 THEN PB1 ELSE 0};
             zB22 = {IF UB22 THEN PB2 ELSE 0};
             zC22 = {IF UC22 THEN PC2 ELSE 0}; }
```




```
        CONTINUOUS {x11 = x11 + zA11 + zB11 - u11;
                    x12 = x12 + zB12 + zC12 - u12;
                    x21 = x21 + zA21 + zB21 - u21;
                    x22 = x22 + zB22 + zC22 - u22;  }

        OUTPUT {    y1 = u11 + u21;
                    y2 = u12 + u22; }

        MUST {  ~(UA11 & UA21);
            ~(UC12 & UC22);
            ~((UB11 | UB21) & (UB12 | UB22));

            ~(UB11 & UB21);
            ~(UB12 & UB22);
            x11+x12 <= xM1;
            x11+x12 >=0;
            x21+x22 <= xM2;
            x21+x22 >=0; }
} }
```

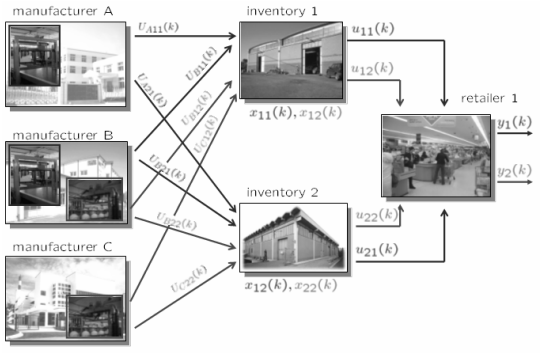
/demos/hybrid/supply_chain.m

Objectives

- Meet customer demand as much as possible:

  $y_1 \approx r_1,\ y_2 \approx r_2$

- Minimize transportation costs



- Fulfill all constraints

Performance Specs

penalty on demand tracking error

$$\min \sum_{k=0}^{N-1} 10(|y_1(k) - r_1(k)| + |y_2(k) - r_2(k)|) +$$

cost for shipping from inv.#1 to market

$$4(|u_{11}(k)| + |u_{12}(k)|) +$$

cost for shipping from inv.#2 to market

$$2(|u_{21}(k)| + |u_{22}(k)|) +$$

cost from A to inventories

$$1(|U_{A11}(k)| + |U_{A21}(k)|) +$$

$$4(|U_{B11}(k)| + |U_{B12}(k)| + |U_{B21}(k)| + |U_{B22}(k)|) +$$

$$10(|U_{C12}(k)| + |U_{C22}(k)|)$$

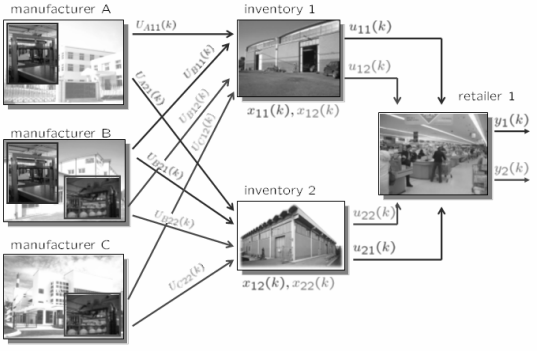
cost from B to inventories

cost from C to inventories

Hybrid MPC - Example

```
>>refs.y=[1 2];       % weights output2 #1,#2
>>Q.y=diag([10 10]); % output weights
…
>>Q.norm=Inf;         % infinity norms
>>N=2;                % optimization horizon
>>limits.umin=umin;   % constraints
>>limits.umax=umax;
>>limits.xmin=xmin;
>>limits.xmax=xmax;
```

```
>>C=hybcon(S,Q,N,limits,refs);
```

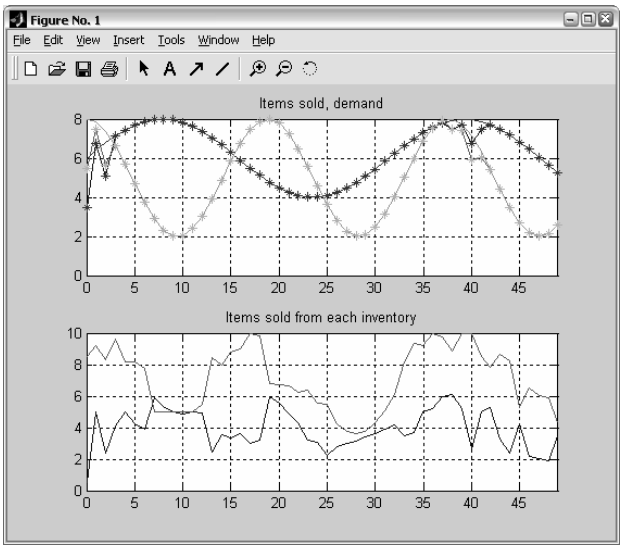


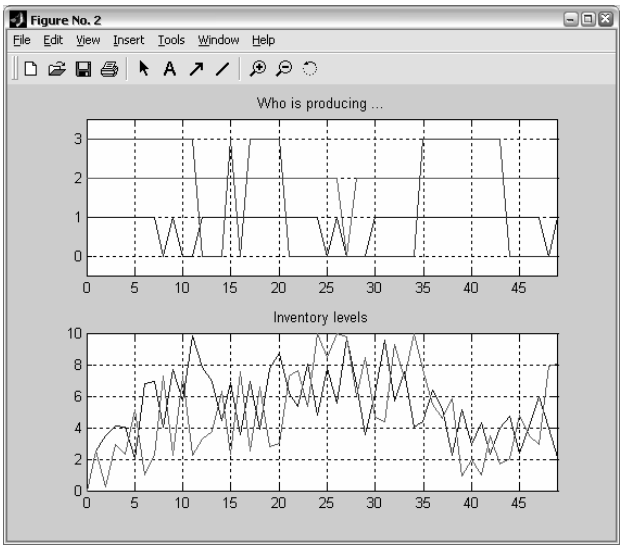

```
>> C

Hybrid controller based on MLD model S <supply_chain.hys> [Inf-norm]

  4 state measurement(s)
  2 output reference(s)
 12 input reference(s)
  0 state reference(s)
  0 reference(s) on auxiliary continuous z-variables

 44 optimization variable(s) (28 continuous, 16 binary)
176 mixed-integer linear inequalities
sampling time = 1, MILP solver = 'glpk'

Type "struct(C)" for more details.

>>
```

Hybrid MPC - Example

```
>>x0=[0;0;0;0];                          % Initial condition
>>r.y=[6+2*sin((0:Tstop-1)'/5)           % Reference trajectories
       5+3*cos((0:Tstop-1)'/3)];
```

```
>>[XX,UU,DD,ZZ,TT]=sim(C,S,r,x0,Tstop);
```





CPU time: ≈ 30ms per time step (using GLPK on this machine)

Programmazione stocastica

Incertezza dei dati del problema

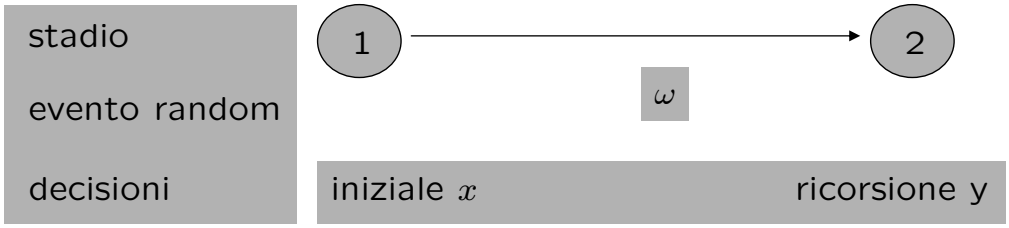
- Finora abbiamo ipotizzato che i dati del problema di ottimizzazione (es. A,b,c di un problema LP) fossero noti con precisione

- Spesso però in pratica si hanno molte situazioni in cui tali dati non sono *deterministici*, ma piuttosto **stocastici**, cioè descritti da funzioni di probabilità

- Questo accade ad esempio quando i dati del problema saranno noti soltanto nel futuro (es: domanda di un cliente, prezzo del materiale, ecc.)

Programmazione stocastica (2 stadi)

- Nella modellazione a 2 stadi, la sequenza delle decisioni può essere rappresentata come segue:



- ω = variabile random
- x = variabile di decisione presa *prima* che l'evento ω si sia manifestato
- y = variabile di decisione presa *dopo* che l'evento ω si sarà manifestato, e quindi da esso dipendente. Permette di effettuare *hedging*.

Programmazione stocastica (2 stadi)

Esempio: Significato delle variabili in problemi gerarchici di pianificazione (*hierarchical planning problems*)

- **Stadio 1 (variabile x)**: decisioni strategiche e globali, con impatto a lungo termine.

  Esempi: investimenti, capacità produttive, ecc.
  In questo caso la domanda del cliente può essere trattata come incerta.

- **Stadio 2 (variabile y)**: variabili operative, decisioni a breve termine in risposta a come le variabili stocastiche ω si sono realizzate (ad esempio la domanda del cliente).

  Esempi: quantitativi di produzione, schedulazione delle macchine, ecc.

Programmazione stocastica (2 stadi)

- Considera il seguente problema di programmazione lineare stocastica a due-stadi con ricorsione

$$\begin{array}{ll} \max & E[a'x + c(\omega)'y(\omega)] \\ \text{s.t.} & Ax = b \\ & B(\omega)x + C(\omega)y(\omega) = d(\omega) \\ & x \geq 0,\ y(\omega) \geq 0 \end{array}$$

$x \in \mathbb{R}^n$
$y \in \mathbb{R}^m$
$\omega \in \mathbb{R}^p$

- ω = variabile random
- x = variabile di decisione presa *prima* che l'evento ω si sia manifestato
- y = variabile di decisione presa *dopo* che l'evento ω si sarà manifestato, e quindi da esso dipendente

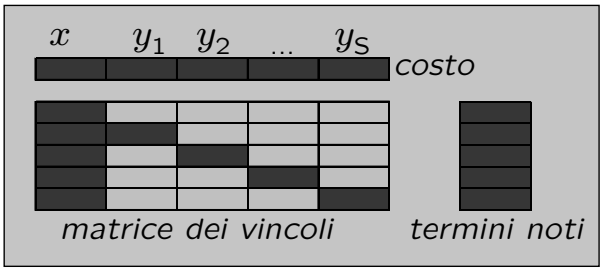
Nota: $\max E[a'x + c(\omega)'y(\omega)] = \max a'x + E[\max_{y(\omega)} c(\omega)'y(\omega)]$

Programmazione stocastica (2 stadi)

- Supponiamo che il vettore ω possa assumere solo un numero finito di valori possibili (detti "*scenari*") $\omega \in \{\omega_1, \omega_2, \ldots, \omega_S\}$
- Siano $\{p_1, p_2, \ldots, p_S\}$ le rispettive probabilità (se equiprobabili p_i=1/S)
- Allora, possiamo riscrivere $E[\max_{y(\omega)} c(\omega)'y(\omega)] = \sum_{j=1}^{S} \max_{y(\omega_j)} c(\omega_j)'y(\omega_j)$

  e quindi il problema di ottimizzazione stocastica come

$$\begin{aligned} \max \quad & a'x + \textstyle\sum_{j=1}^{s} p_j c_j' y_j \\ \text{s.t.} \quad & Ax = b \\ & B_1 x + C_1 y_1 = d_1 \\ & \vdots \\ & B_S x + C_s y_s = d_S \\ & x, y_1, \ldots, y_s \geq 0 \end{aligned}$$



La struttura particolare (e sparsa) del problema può essere sfruttata dai risolutori

Esempio

- Un municipalizzata deve gestire gli acquisti di gas per l'anno in corso e il prossimo.
- Il gas può essere comprato, venduto e immagazzinato
- A seconda delle condizioni climatiche, i prezzi di acquisto per quantitativo di gas e la domanda cambiano:

Clima	Probabilità p	Costo gas c	Domanda d
Normale	1/3	5	100
Freddo	1/3	7	120
Molto freddo	1/3	9	130

- Supponiamo che nell'anno in corso ci sia un clima "normale", mentre il clima dell'anno prossimo è incerto
- Quanto gas dobbiamo comprare quest'anno ? Quanto dobbiamo comprarne per immagazzinarlo ? Quanto dovremo comprarne l'anno prossimo ?

http://stoprog.org/spintroduction.html

Esempio

- È un problema di programmazione lineare stocastica a due stadi
- Variabili:

- ω = variabile random "clima" (3 possibili valori)
- x = variabile di decisione presa *prima* che l'evento ω si sia manifestato

  x_1 = acquisto di gas da rivendere subito (anno 1)

  x_2 = acquisto di gas da immagazzinare (anno 1)
- y = variabile di decisione presa *dopo* che l'evento ω si sarà manifestato

  y_{1i} = utilizzo di gas immagazzinato (anno 2)

  y_{2i} = acquisto di ulteriore gas da rivendere subito (anno 2)

  j=1,2,3

Esempio

- Funzione costo:

$$\min 5x_1 + (5+1)x_2 + E[c(\omega)y_2(\omega)]$$

$$\min 5x_1 + (5+1)x_2 + \frac{1}{3}5y_{21} + \frac{1}{3}7y_{22} + \frac{1}{3}9y_{23}$$

- Vincoli:
  - Soddisfacimento domanda: $x_1 \geq 100$

    $y_{1j} + y_{2j} \geq d_j,\ j = 1,2,3$
  - Disponibilità di gas immagazzinato:

    $y_{1j} \leq x_2,\ j = 1,2,3$
  - Il gas si può solo comprare o prelevare dallo stoccaggio: $x_1, x_2, y_{1j}, y_{2j} \geq 0,\ j = 1,2,3$

Esempio

- Risultato: costo = 1236.67, x_1=100, x_2=100
  y_{11}=100, y_{21}=0, y_{12}=100, y_{22}=20, y_{13}=100, y_{23}=30

- Calcoliamo il costo per ogni realizzazione di scenario:

Clima	Azione stage 2	Costo totale
Normale	nessun acquisto aggiuntivo	1100
Freddo	compra 30 unità al costo di 6 per unità	1240
Molto freddo	compra 36.67 unità al costo di 7.5 per unità	1370

- Conferma che il costo medio è (1100+1240+1370)/3=1236.67

Esempio

- Soluzione euristica: rimpiazzare i coefficienti con i rispettivi valori attesi:

$$\begin{array}{ll}\min & 5x_1+(5+1)x_2+E[c(\omega)]y_2 \\ \text{s.t.} & x_1 \geq 100 \\ & y_1+y_2 \geq E[d(\omega)]\end{array} \quad \Rightarrow \quad \begin{array}{ll}\min & 5x_1+(5+1)x_2+7y_2 \\ \text{s.t.} & x_1 \geq 100 \\ & y_1+y_2 \geq 116.67\end{array}$$

- Risultato: x_1=100, x_2=116.67, y_1=116.67, y_2=0.
  La soluzione però non è ammissibile nel caso "freddo" e "molto freddo". In ogni caso conviene ricalcolare y_2 dopo che l'evento stocastico si è palesato

Clima	Azione stage 2	Costo totale
Normale	nessuna (eccesso di 16.67 in magazzino)	1200
Freddo	compra 3.33 unità al costo di 7 per unità	1223.33
Molto freddo	compra 13.33 unità al costo di 9 per unità	1320

- Il costo medio è 1247.78

Esempio

- Soluzione euristica: ottimizzare ogni scenario, poi prendere come variabili decisionali la media degli ottimizzatori:

$$x^*(j), y^*(j) = \arg\min \; 5x_1 + (5+1)x_2 + c(j)y_2 \quad \text{s.t.} \; x_1 \geq 100,\; y_1 + y_2 \geq d(j) \quad , \; j = 1,2,3$$

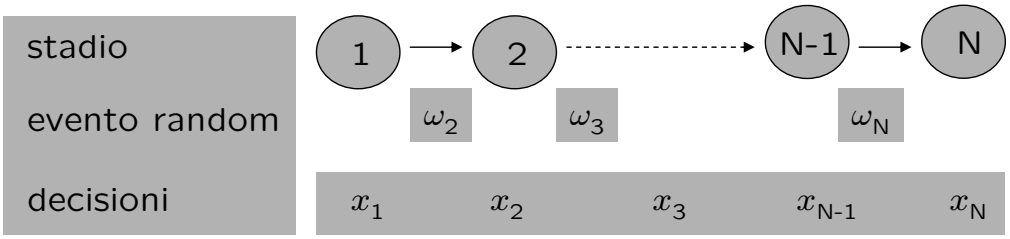
- Risultato: x_1=100, x_2=83.33, y_1=83.33, y_2=33.33.
  La soluzione però non è ammissibile nel caso "molto freddo".
  In ogni caso conviene ricalcolare y_2 dopo che l'evento stocastico si è palesato

Clima	Azione stage 2	Costo totale
Normale	compra 16.67 unità al costo di 5 per unità	1083.33
Freddo	compra 36.67 unità a 7 per unità	1256.67
Molto freddo	compra 46.67 unità a 9 per unità	1420

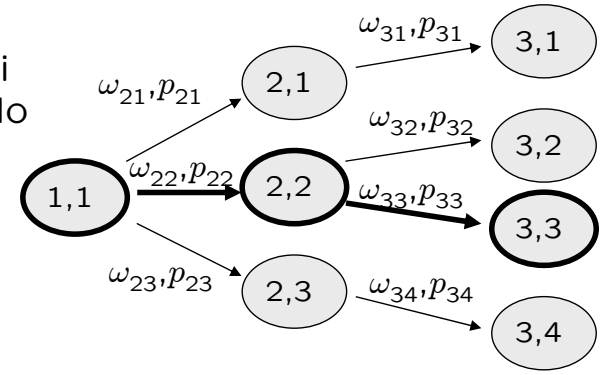
- Il costo medio è 1253.33

Programmazione stocastica (multi-stadio)

- Nella modellazione multi-stadio con ricorsione, la sequenza delle decisioni può essere rappresentata come segue:



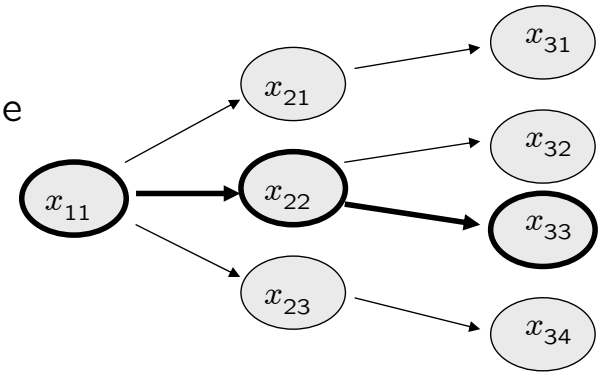
- In questo caso si suppone che i possibili valori assumibili da ω siano organizzati secondo un "*albero*"
- Gli scenari sono dati da tutti i percorsi possibili sull'albero, ad esempio [ω_{22}, ω_{33}] con probabilità p=$p_{22}p_{33}$.



Programmazione stocastica (multi-stadio)

- Ad ogni nodo dell'albero viene assegnata una variabile di decisione x_{ij} che rappresenta la decisione che sarebbe presa se ci venissimo a trovare nel nodo (i,j) dell'albero degli scenari

- x_{11} rappresenta la decisione presa prima che l'evento random si inizi a manifestare



- Con questo costrutto viene imposto un vincolo di **causalità** (o **non-anticipatività**) delle decisioni: la scelta x_{ij} presa allo stadio i, non conoscendo ancora quale sarà la realizzazione futura dell'evento random, è indipendente da quale percorso verrà seguito negli stadi successivi i+1,...,N.

Linguaggi di modellistica per programmazione stocastica

- **MOSEL + Xpress-SP**, commerciale
- **SMI** (Stochastic Modeling Interface), open-source
- La maggior parte dei linguaggi di modellistica menzionati in precedenza (AMPL, GAMS, ecc.) supporta estensioni per la rappresentazione di problemi di ottimizzazione stocastica
- **SMPS** (Stochastic Mathematical Programming System), è una versione di MPS per rappresentare problemi di ottimizzazione stocastica

Bibliografia:

[1] J.R. Birge and F. Louveaux, "*Introduction to Stochastic Programming*", Springer Verlag, 1997

Fine