Ricerca Operativa

Massimo Pappalardo
Dipartimento di Informatica
Largo B. Pontecorvo 3, Pisa
massimo.pappalardo@unipi.it

Laurea in Ingegneria Informatica
Universitá di Pisa
A.A. 2020/'21

Riferimenti

Massimo Pappalardo

Dipartimento di Informatica

Largo B. Pontecorvo 3- Pisa

Edificio C - studio 289DE

tel. 050 2212750

e-mail: massimo.pappalardo@unipi.it

ricevimento: da concordare

Orario del corso

- lunedí 8.30-12.30
- martedí 8.30-10.30
- mercoledí 8.30-10.30
- venerdí 8.30-10.30

Materiale per il corso

Pagina Teams del corso

Testi

- M.Pappalardo, M.Passacantando, Ricerca Operativa, Edizioni Plus, 2010.
- F.S.Hillier, G.J.Lieberman, Ricerca Operativa, McGraw Hill, 2010.

Esame

Prova orale comprendente risoluzione di problemi.

Conoscenze richieste

- Operazioni tra matrici: somma, moltiplicazione e prodotto per uno scalare.
- Matrice Inversa.
- Sistemi lineari.
- Autovalori di una matrice.
- Derivate parziali, derivate direzionali, gradiente, matrice hessiana.
- Insiemi del piano definiti da curve di livello.

Obiettivo e argomenti del corso

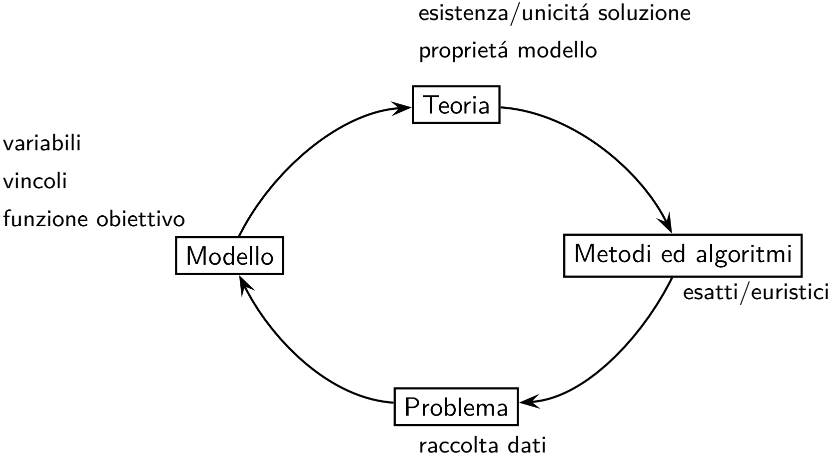
Obiettivo

Fornire conoscenze e metodi per la formulazione e risoluzione di problemi di ottimizzazione.

Argomenti

- Modelli matematici canonici di ottimizzazione per processi decisionali.
- Modelli matematici per la risoluzione di alcuni problemi di ottimizzazione: produzione, assegnamento, trasporto ottimo, localizzazione, caricamento, "bin packing", commesso viaggiatore, portafoglio.
- Elementi di teoria e metodi di Programmazione Matematica: Lineare (PL), Lineare Intera (PLI), Lineare su Reti e Non Lineare (PNL).
- MATLAB per la risoluzione di problemi di ottimizzazione.

Il processo decisionale



Modelli matematici di problemi decisionali

- Raccolta dati.
- Variabili decisionali, funzione obiettivo, vincoli.
- Costruzione del modello matematico.
- Teoria, metodi e algoritmi per la risoluzione del modello matematico.
- Software per la soluzione.
- Controllo della soluzione.

Un esempio: il problema del contadino

Un contadino ha 12 ettari di terra per coltivare pomodori e/o patate.

Ha anche 70 kg di semi di pomodoro, 18 t di tuberi e 160 t di letame.

Il guadagno per ettaro é 3000 euro per i pomodori e 5000 euro per le patate.

I pomodori necessitano di 7 kg di semi e 10 t di letame per ettaro, mente le patate richiedono 3 t di tuberi e 20 t di letame per ettaro.

Per massimizzare il guadagno come dividere la terra tra pomodori e patate?

Raccolta dati

Dati

- 12 ettari di terra.
- 160 t di letame, 70 Kg di semi, 18 t di tuberi.
-

	profitto/ett.	semi/ett.	letame/ett.	tuberi/ett.
pomodori	3000	7 kg	10 t	
patate	5000		20 t	3 t

- Come decidere?
  Quanti ettari devono essere assegnati ai pomodori e quanti alle patate.
- Quale é il nostro obiettivo? Massimizzare il guadagno.
- Quali sono le richieste per avere una soluzione ammissibile?
  Limiti sulle risorse disponibili.

Modello

Cosa si deve decidere? $\rightarrow$ variabili decisionali

- x_T = ettari di pomodori
- x_P = ettari di patate

Massimizzare il guadagno $\rightarrow$ funzione obiettivo

Guadagno $= 3000x_T + 5000x_P$
Massimizzare il guadagno $\rightarrow$ $\max\ 3x_T + 5x_P$

Modello

Richieste e disponibilitá di risorse $\rightarrow$ vincoli

- Disponibilitá di terreno: $x_T + x_P \leq 12$
- Semi di pomodoro disponibili: $7x_T \leq 70 \rightarrow x_T \leq 10$
- Tuberi di patate disponibili: $3x_P \leq 18 \rightarrow x_p \leq 6$
- Letame disponibile: $10x_T + 20x_P \leq 160 \rightarrow x_T + 2x_P \leq 16$
- Variabili non negative: $x_T \geq 0$, $x_P \geq 0$

Forma matriciale

$$
\begin{array}{rl}
\max\ 3x_T + 5x_P & \\
x_T + x_P & \leq 12 \\
x_T & \leq 10 \\
x_P & \leq 6 \\
x_T + 2x_P & \leq 16 \\
-x_T & \leq 0 \\
-x_P & \leq 0
\end{array}
$$

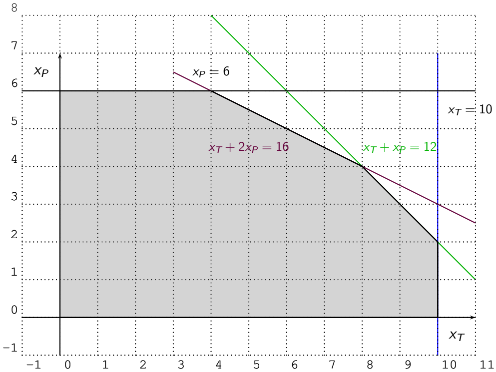
$$
\begin{cases} \max\ c^T x \\ Ax \leq b \end{cases}
$$

$$
A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 2 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 12 \\ 10 \\ 6 \\ 16 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 3 \\ 5 \end{pmatrix}
$$

$m =$ numero di vincoli
$n =$ numero di variabili
A matrice $m \times n$
b vettore m componenti
c vettore n componenti

Rappresentazione grafica della regione ammissibile

$$
\begin{aligned}
\max\ & 3x_T + 5x_P \\
x_T + x_P & \leq 12 \\
x_T & \leq 10 \\
x_P & \leq 6 \\
x_T + 2x_P & \leq 16 \\
-x_T & \leq 0 \\
-x_P & \leq 0
\end{aligned}
$$



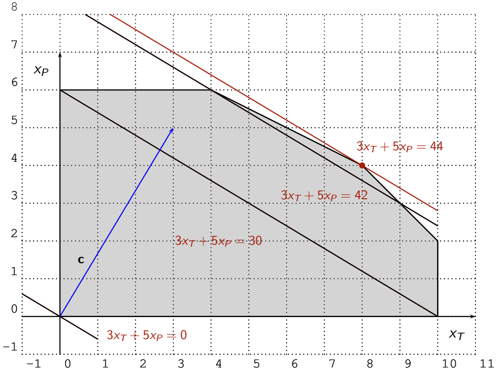
Soluzione grafica della PL per n=2

Le linee di isocosto o isoguadagno sono

$$L(v) = \{x \in \mathbb{R}^n : \; c^T x = v\}$$

dove $v \in \mathbb{R}$ é un numero reale.

$$\begin{aligned}
\max \; & 3x_T + 5x_P \\
x_T + x_P & \leq 12 \\
x_T & \leq 10 \\
x_P & \leq 6 \\
x_T + 2x_P & \leq 16 \\
-x_T & \leq 0 \\
-x_P & \leq 0
\end{aligned}$$



Soluzione ottima $x_T = 8, x_P = 4$

PL e la sua forma standard

Definizione

Un problema di Programmazione Lineare (PL) consiste nel trovare il massimo o il minimo di una funzione lineare soggetta ad un insieme finito di vincoli lineari di disuguaglianza o di uguaglianza:

$$\begin{cases} \max(\min)\ c^T x \\ A_1 x \leq b_1 \\ A_2 x \geq b_2 \\ A_3 x = b_3 \end{cases}$$

Definizione

Un problema nella forma

$$\begin{cases} \max\ c^T x \\ Ax \leq b \end{cases}$$

é detto problema di PL in formato primale standard.

PL e formato primale standard

Osservazione

Ogni problema di PL puó essere equivalentemente scritto in formato primale standard.

Dimostrazione. $\min \ c^T x = -\max \ (-c^T x)$

$a^T x \geq b$ é equivalente a $-a^T x \leq -b$

$a^T x = b$ é equivalente a $\begin{cases} a^T x \leq b \\ -a^T x \leq -b \end{cases}$.

Definizione

Un poliedro P é l'intersezione di un numero finito di semispazi chiusi o, equivalentemente, l'insieme $\{x \in \mathbb{R}^n : Ax \leq b\}$.

Definizione

Un punto x di un poliedro P é un vertice se non esistono due punti di P differenti da x tali che x appartenga al segmento generato da essi.

Esempi.

Vertici di $P_1 = \{x \in \mathbb{R}^2 : \ 1 \leq x_1 \leq 4, \quad 1 \leq x_2 \leq 3\}$ sono $(1,1)$, $(1,3)$, $(4,1)$ e $(4,3)$.

Vertici di $P_2 = \{x \in \mathbb{R}^2 : \ x_1 \geq 1, \quad x_2 \geq 1, \quad x_1 + x_2 \geq 3\}$ sono $(1,2)$ e $(2,1)$.

$P_3 = \{x \in \mathbb{R}^2 : \ 0 \leq x_2 \leq 1\}$ non ha vertici.

Consideriamo un problema di PL in forma primale standard:

$$\begin{cases} \max\ c^T x \\ x \in P = \{x \in \mathbb{R}^n :\ Ax \leq b\} \end{cases} \qquad (\mathcal{P})$$

Teorema fondamentale della PL

Se P é limitato e non vuoto, allora un vertice di P é ottimo.

Esempio. Consideriamo il problema

$$\begin{cases} \max\ -2x_1 - 3x_2 \\ x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \geq 3 \end{cases}$$

La soluzione ottima é $(2, 1)$.

Scarti complementari

Come riconoscere una soluzione ottima?

Teorema

Una soluzione $\bar{x}$ é ottima se e solo se esiste $\bar{y} \in \mathbb{R}^m_+$ tale che

$$\begin{cases} \bar{y}^T A = c^T \\ \bar{y}^T(b - A\bar{x}) = 0 \quad \text{(scarti complementari)} \end{cases}$$

Esempio. Consideriamo il problema di PL

$$\begin{cases} \max\ 3x_1 + 4x_2 \\ 2x_1 + x_2 \leq 3 \\ x_1 + 2x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases}$$

$\bar{x} = (1, 1)$ é ottima perché $\bar{y} = (2/3, 5/3, 0, 0)$ risolve il sistema.
$\bar{x} = (0, 0)$ non é ottima perché il sistema non ha soluzione.

Caratterizzazione algebrica dei vertici

Sappiamo che se $P \neq \emptyset$ e limitato, allora un vertice di P é ottimo.

Come trovare un vertice ottimo? Sevono proprietá algebriche dei vertici

Consideriamo un problema

$$\begin{cases} \max \ c^T x \\ Ax \leq b \end{cases}$$

Definizione

Una base é un insieme B di n indici di riga tali che $\det(A_B) \neq 0$.

$$A = \begin{pmatrix} A_B \\ A_N \end{pmatrix} \qquad b = \begin{pmatrix} b_B \\ b_N \end{pmatrix}$$

Data una base B, il vettore $\bar{x} = A_B^{-1} b_B$ é detto soluzione di base primale.

$\bar{x}$ é ammissibile se $A_N \bar{x} \leq b_N$.

Esempio. Consideriamo

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$

$B = \{1, 2\}$ é una base perché $A_B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ é invertibile.

La soluzione corrispondente é $\bar{x} = A_B^{-1} b_B = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.

$\bar{x}$ é ammissibile perché $A_N \bar{x} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \end{pmatrix} = b_N$.

Caratterizzazione algebrica dei vertici

$B = \{1, 3\}$ non é una base perché $A_B = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}$ non é invertibile.

$B = \{2, 4\}$ é una base e la corrispondente soluzione di base é inammissibile:

$$A_B = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix},\ \bar{x} = \begin{pmatrix} 3 \\ 0 \end{pmatrix},\ A_N \bar{x} = \begin{pmatrix} 3 \\ -3 \end{pmatrix} \not\leq \begin{pmatrix} 2 \\ 0 \end{pmatrix} = b_N.$$

Perché le soluzioni di base sono importanti?

Teorema

$\bar{x}$ é un vertice di P se e solo se $\bar{x}$ é una soluzione di base ammissibile.

Esempio. Consideriamo il precedente problema

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$\bar{x} = (2, 1)$ é una soluzione di base ammissibile corrispondente alla base $B = \{1, 2\}$.

Al problema primale in forma standard

$$\begin{cases} \max\ c^{\mathsf{T}} x \\ x \in P = \{x \in \mathbb{R}^n :\ A\,x \leq b\} \end{cases} \tag{$\mathcal{P}$}$$

associamo il seguente problema di PL:

$$\begin{cases} \min\ y^{\mathsf{T}} b \\ y \in D = \{y \in \mathbb{R}^m :\ y^{\mathsf{T}} A = c^{\mathsf{T}}, \quad y \geq 0\} \end{cases} \tag{$\mathcal{D}$}$$

che sará chiamato problema duale in forma standard.

Il valore ottimo del problema $(\mathcal{D})$ verrá indicato con $v(\mathcal{D})$.

Poiché ogni problema di PL si puó trasformare in un problema primale standard di tipo $(\mathcal{P})$, allora ogni problema di PL ha un suo problema duale.

Osservazione

Ogni problema di PL si puó trasformare in un duale standard.

Dimostrazione. Una disuguaglianza

$$A_i^{\mathsf{T}} x \leq b_i$$

si puó trasformare nell'uguaglianza

$$A_i^{\mathsf{T}} x + s_i = b_i,$$

aggiungendo i vincoli $s_i \geq 0$, dove le s_i sono dette variabili di scarto.

Ogni numero reale é la differenza di due numeri non negativi quindi si puó sempre introdurre il vincolo di positivitá sulle variabili spezzando ogni variabile non vincolata in segno nella differenza di due variabili vincolate in segno e porre

$$x = x^+ - x^- \qquad \text{con} \qquad x^+ \geq 0, \quad \text{e} \quad x^- \geq 0.$$

Teorema

(Dualitá forte) Se i poliedri P e D sono non vuoti, allora

$$-\infty < v(\mathcal{P}) = v(\mathcal{D}) < +\infty$$

Data una base B, definiamo:

Soluzione di base duale: $\bar{y} = \begin{pmatrix} \bar{y}_B \\ \bar{y}_N \end{pmatrix}$ dove $\bar{y}_B^\mathsf{T} = c^\mathsf{T} A_B^{-1}, \quad \bar{y}_N = 0.$

Una soluzione di base duale puó essere:

	$\bar{y}$
ammissibile	per ogni $i \in B$ si ha $\bar{y}_i \geq 0$
non ammissibile	esiste $i \in B$ tale che $\bar{y}_i < 0$
degenere	esiste $i \in B$ tale che $\bar{y}_i = 0$
non degenere	per ogni $i \in B$ si ha $\bar{y}_i \neq 0$

Teorema

Due soluzioni di base complementari $\bar{x}$ e $\bar{y}$ sono in scarti complementari.

Dimostrazione:

$$\begin{aligned}
\bar{y}^{\mathsf{T}}(b - A\,\bar{x}) &= (\bar{y}_B^{\mathsf{T}}, \bar{y}_N^{\mathsf{T}}) \begin{pmatrix} b_B - A_B\,\bar{x} \\ b_N - A_N\,\bar{x} \end{pmatrix} \\
&= (c^{\mathsf{T}} A_B^{-1}, 0) \begin{pmatrix} b_B - A_B\,A_B^{-1}\,b_B \\ b_N - A_N\,A_B^{-1}\,b_B \end{pmatrix} \\
&= (c^{\mathsf{T}} A_B^{-1}, 0) \begin{pmatrix} 0 \\ b_N - A_N\,A_B^{-1}\,b_B \end{pmatrix} \\
&= 0
\end{aligned}$$

Dal teorema precedente e dal teorema degli scarti complementari possiamo dedurre le seguenti condizioni sufficienti di ottimalitá.

Teorema*(Condizioni sufficienti di ottimalitá per soluzioni di base)*

Date due soluzioni di base complementari $\bar{x}$ e $\bar{y}$, si ha:

$$\begin{bmatrix} \bar{x} \text{ é ammissibile per } (\mathcal{P}) \\ \text{e} \\ \bar{y} \text{ é ammissibile } (\mathcal{D}) \end{bmatrix} \qquad \Longrightarrow \qquad \begin{bmatrix} \bar{x} \text{ é ottima per } (\mathcal{P}) \\ \text{e} \\ \bar{y} \text{ é ottima } (\mathcal{D}) \end{bmatrix}$$

Algoritmo del simplesso

1. Trova una base B tale che la corrispondente soluzione di base $\bar{x} := A_B^{-1}\, b_B$ sia ammissibile.
2. Calcola
$$\bar{y} := \begin{pmatrix} \bar{y}_B \\ \bar{y}_N \end{pmatrix}, \qquad \text{with} \quad \bar{y}_B^T = c^T A_B^{-1}, \quad \bar{y}_N = 0.$$
3. **se** $\bar{y}_B \geq 0$ **allora** STOP ($\bar{x}$ é ottima) **altrimenti** trova l'indice uscente
$$h := \min\{i \in B : \ \bar{y}_i < 0\}$$
poniamo $W := -A_B^{-1}$, denotiamo W^h la h–ma colonna di W.
4. **se** $A_i\, W^h \leq 0$ per tutti gli indici $i \in N$ **allora** STOP (ottimo di $(\mathcal{P})$ é $+\infty$)
altrimenti calcola $\vartheta := \min \left\{ \dfrac{b_i - A_i\, \bar{x}}{A_i\, W^h} : \quad i \in N, \ A_i\, W^h > 0 \right\}$,
trova l'indice entrante
$$k := \min \left\{ i \in N : \ A_i\, W^h > 0, \ \frac{b_i - A_i\, \bar{x}}{A_i\, W^h} = \vartheta \right\},$$
aggiorna la base $B := B \setminus \{h\} \cup \{k\}$, vai al passo 2.

Teorema

L'algoritmo del simplesso si ferma dopo un numero finito di iterazioni.

Algoritmo del simplesso

Esempio. Partendo dalla base $B = \{3, 4\}$, risolviamo il problema:

$$\begin{cases} \max\ 2x_1 + x_2 \\ x_1 \leq 2 \\ x_1 + x_2 \leq 3 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \end{cases} \qquad A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Iterazione 1. $A_B = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ é ammissibile.

$\bar{y}_B^T = (2, 1) \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = (-2, -1)$, $h = 3$, $W^3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. $A_1 W^3 = 1$, $A_2 W^3 = 1$,
$\vartheta = \min\{2/1, 3/1\} = 2$, $k = 1$.

Iterazione 2. $B = \{1, 4\}$, $A_B = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = A_B^{-1}$, $\bar{x} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$,

$\bar{y}_B^T = (2, 1) \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = (2, -1)$, $h = 4$, $W^4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. $A_2 W^4 = 1$, $A_3 W^4 = 0$, $k = 2$.

Iterazione 3. $B = \{1, 2\}$, $A_B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, $\bar{x} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$, $\bar{y}_B^T = (1, 1) \geq 0$ stop $\bar{x}$ é ottima.

Simplesso duale

Descriviamo l'algoritmo del simplesso duale per risolvere un problema in forma duale standard:

$$\begin{cases} \min\ y^{\mathsf{T}} b \\ y^{\mathsf{T}} A = c^{\mathsf{T}} \\ y \geq 0 \end{cases} \qquad (\mathcal{D})$$

L'algoritmo é analogo al simplesso primale con la differenza che ad ogni passo si mantiene ammissibile la soluzione di base duale e si controlla l'ammissibilitá di quella primale.

Se la soluzione di base primale é ammissibile, allora abbiamo trovato una coppia primale/duale di soluzioni ottime.

Il cambio di base é definito in modo che, se la nuova soluzione di base duale é diversa da quella vecchia, il valore della funzione obiettivo diminuisce.

ALGORITMO DEL SIMPLESSO DUALE

1. Trova una base B che genera una soluzione di base duale ammissibile.
2. Calcola la soluzione di base primale $\bar{x} = A_B^{-1}\, b_B$ e la soluzione di base duale

$$\bar{y} = (\bar{y}_B, \bar{y}_N), \qquad \text{con} \quad \bar{y}_B^{\mathsf{T}} = c^{\mathsf{T}} A_B^{-1}, \quad \bar{y}_N = 0.$$

3. **Se** $b_N - A_N\,\bar{x} \geq 0$ **allora** STOP ($\bar{y}$ é ottima per $(\mathcal{D})$ e $\bar{x}$ é ottima per $(\mathcal{P})$).
   altrimenti calcola l'indice entrante

$$k = \min\{i \in N : \; b_i - A_i\,\bar{x} < 0\} \qquad \text{(regola anticiclo di Bland)}$$

   poni $W = -A_B^{-1}$ ed indica con W^i la i–esima colonna di W.
4. **Se** $A_k\, W^i \geq 0$ per ogni $i \in B$ **allora** STOP ($(\mathcal{D})$ ha valore $-\infty$).
   altrimenti calcola

$$\vartheta = \min\left\{ \frac{\bar{y}_i}{-A_k\, W^i} : \quad i \in B,\; A_k\, W^i < 0 \right\},$$

   calcola l'indice uscente

$$h = \min\left\{ i \in B : \quad A_k\, W^i < 0,\; \frac{\bar{y}_i}{-A_k\, W^i} = \vartheta \right\} \quad \text{(regola anticiclo di Bland)},$$

   aggiorna la base $B := B \setminus \{h\} \cup \{k\}$ e torna al passo 2.

Teorema

Il simplesso duale risolve $(\mathcal{D})$ in un numero finito di iterazioni.

Illustriamo ora una risoluzione di un problema di PL applicando il simplesso duale.

Esempio. Risolviamo il seguente problema

$$\begin{cases} \min \ 13\,y_3 + 9\,y_4 + 7y_5 \\ -y_1 + y_3 + y_4 + y_5 = 3 \\ -y_2 + 2\,y_3 + y_4 = -4 \\ y \geq 0 \end{cases} \tag{1}$$

con il simplesso duale partendo dalla base $B = \{2, 3\}$.

Iterazione 1.
La matrice di base e la sua inversa sono

$$A_B = \begin{pmatrix} 0 & -1 \\ 1 & 2 \end{pmatrix} \qquad A_B^{-1} = \begin{pmatrix} 2 & 1 \\ -1 & 0 \end{pmatrix},$$

la soluzione di base duale é

$$\bar{y}_B^{\mathsf{T}} = (3, -4) A_B^{-1} = (10, 3), \qquad \bar{y} = (0, 10, 3, 0, 0)^{\mathsf{T}},$$

e quella primale é $\bar{x} = (13, 0)^{\mathsf{T}}$ che non é ammissibile perché

$$b_1 - A_1\,\bar{x} = 13, \qquad b_4 - A_4\,\bar{x} = -4, \qquad b_5 - A_5\,\bar{x} = -6,$$

quindi l'indice che entra in base é 4. Calcoliamo i prodotti

$$A_4\,W^2 = -1, \qquad A_4\,W^3 = -1,$$

e i rapporti:

$$\frac{\bar{y}_2}{-A_4\,W^2} = 10, \qquad \frac{\bar{y}_3}{-A_4\,W^3} = 3,$$

quindi l'indice uscente dalla base é 3.

Iterazione 2.
La nuova base é $B = \{2, 4\}$, la matrice di base e la sua inversa sono

$$A_B = \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix} \qquad\qquad A_B^{-1} = \begin{pmatrix} 1 & 1 \\ -1 & 0 \end{pmatrix}$$

la soluzione di base duale é

$$\bar{y} = (0, 7, 0, 3, 0)^\mathsf{T},$$

e quella primale é $\bar{x} = (9, 0)^\mathsf{T}$ che non é ammissibile perché

$$b_1 - A_1\,\bar{x} = 9, \qquad b_3 - A_3\,\bar{x} = 4, \qquad b_5 - A_5\,\bar{x} = -2,$$

quindi l'indice che entra in base é 5. Calcoliamo i prodotti

$$A_5\,W^2 = -1, \qquad A_5\,W^4 = -1,$$

e i rapporti:

$$\frac{\bar{y}_2}{-A_5\,W^2} = 7, \qquad \frac{\bar{y}_4}{-A_5\,W^4} = 3,$$

quindi l'indice uscente dalla base é 4.

Iterazione 3.
La nuova base é $B = \{2, 5\}$, la matrice di base e la sua inversa sono

$$A_B = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \qquad\qquad A_B^{-1} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

la soluzione di base duale é

$$\bar{y} = (0, 4, 0, 0, 3)^{\mathsf{T}},$$

e quella primale é $\bar{x} = (7, 0)^{\mathsf{T}}$ che é ammissibile perché

$$b_1 - A_1\,\bar{x} = 7, \qquad b_3 - A_3\,\bar{x} = 6, \qquad b_4 - A_4\,\bar{x} = 2,$$

quindi $(0, 4, 0, 0, 3)$ é la soluzione ottima del problema.

Problema ausiliario duale

Consideriamo il problema in forma duale standard:

$$\begin{cases} \min \ y^{\mathsf{T}} b \\ y^{\mathsf{T}} A = c^{\mathsf{T}} \\ y \geq 0 \end{cases} \qquad (\mathcal{D})$$

Senza ledere la generalitá della trattazione possiamo supporre che $c \geq 0$, cambiando eventualmente segno alle equazioni.

Per trovare una base ammissibile di $(\mathcal{D})$ che sia la base di partenza del simplesso duale si costruisce il problema ausiliario duale:

$$\begin{cases} \min \ \sum_{i=1}^{n} \epsilon_i \\ y^{\mathsf{T}} A + \epsilon^{\mathsf{T}} = c^{\mathsf{T}} \\ y \geq 0 \\ \epsilon \geq 0 \end{cases} \qquad (\mathcal{D}_{aux})$$

La base formata dagli indici relativi alle variabili ausiliarie $\epsilon_1, \dots, \epsilon_n$, con la matrice identitá come matrice di base, é una base ammissibile per il problema ausiliario, infatti la corrispondente soluzione di base é $\bar{y} = 0$, $\bar{\epsilon} = c \geq 0$.

A partire da tale base ammissibile, possiamo applicare il simplesso duale per risolvere il problema ausiliario.

Il valore ottimo del problema ausiliario stabilisce se esiste una base ammissibile per il problema $(\mathcal{D})$.

Teorema

1. Se il valore ottimo di $(\mathcal{D}_{aux})$ é > 0 allora $(\mathcal{D})$ non ha soluzioni ammissibili.
2. Se il valore ottimo di $(\mathcal{D}_{aux})$ é $= 0$ allora c'é una base ammissibile per $(\mathcal{D})$ che si costruisce a partire da una base ottima per $(\mathcal{D}_{aux})$.

DATI: Un'azienda deve produrre due tipi di tessuto.

Per produrre un quintale del primo tessuto servono 28 kg di lana e 7 kg di cotone.

Per il secondo tipo servono 7 kg di lana e 14 kg di cotone.

Per produrre i tessuti servono 3 ore di lavoro di un operaio specializzato per ogni quintale da produrre.

Ogni settimana sono disponibili 168 kg di lana, 84 kg di cotone e 42 ore di lavoro.

Siano 20 e 10 euro i guadagni (per quintale) per il tessuto 1 e per il tessuto 2.

VARIABILI: Indichiamo con x_1 e x_2 i quintali prodotti del primo e del secondo tessuto.

Problema di produzione

$$\begin{cases} \max\ 20\,x_1 + 10\,x_2 \\ 28\,x_1 + 7\,x_2 \leq 168 \\ 7\,x_1 + 14\,x_2 \leq 84 \\ 3\,x_1 + 3\,x_2 \leq 42 \\ x_1, x_2 \geq 0 \end{cases}$$

La soluzione ottima é: $(36/7,\ 24/7)$ per un guadagno di 137,14 euro.

Qualora il bene da produrre fosse stato un vestito anziché un chilogrammo di tessuto, la soluzione trovata non era ammissibile e si sarebbe dovuto aggiungere il vincolo di interezza.

In tal caso la soluzione ottima sarebbe stata (5,3) con un guadagno di 130 euro.

Problema di produzione

Supponiamo che si debbano produrre n oggetti, ognuno composto da m diverse materie prime.

Sia data una matrice di composizione A. In tale matrice l'elemento a_{ij} rappresenta la quantitá di materia prima i che serve per produrre l'oggetto j.

Sia dato il guadagno c_j ottenuto vendendo l'oggetto j e la disponibilitá b_i della materia prima i.

Introducendo le variabili x_j, che rappresentano le quantitá prodotta dell'oggetto j, il problema viene formulato nel modo seguente:

$$\begin{cases} \max \ \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \leq b_i & \text{per ogni } i = 1, \dots, m \\ x_j \geq 0 & \text{per ogni } j = 1, \dots, n \end{cases}$$

2. Problema di assegnamento

- Date n persone e n lavori.
- Ogni lavoro deve essere fatto da *esattamente una* persona.
- Ogni persona puó fare *al piú* un lavoro.
- Il *costo* della persona $j = 1, \ldots, n$ che fa il lavoro $i = 1, \ldots, n$ é c_{ij}.
- Vogliamo trovare un *assegnamento di costo minimo*.
- Possiamo associare una variabile 0-1 x_{ij} ad ogni possibile assegnamento(vale 1 se lavoro i assegnato alla persona j, 0 altrimenti).

Problema di assegnamento

Assegnamento non cooperativo:

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

Assegnamento cooperativo:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, \ldots, n) \\
& \sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, \ldots, n) \\
& x_{ij} \in [0, 1].
\end{aligned}
$$

Un assegnamento non cooperativo é una permutazione.

Un problema di assegnamento non cooperativo é un problema di PLI.

Un problema di assegnamento cooperativo é un problema di PL.

Teorema

I vertici del poliedro dell'assegnamento cooperativo hanno componenti intere.

Quindi anche il problema dell'assegnamento non cooperativo é un problema di PL.

Problema di assegnamento generalizzato

Ogni persona $j = 1, \ldots, n$ ha una capacitá b_j (per esempio ore di lavoro).

Ogni lavoro $i = 1, \ldots, m$ assegnato alla persona j usa w_{ij} della capacitá b_j.

Ogni lavoro deve essere assegnato ad una persona e ogni persona non puó superare la propria capacitá.

$$
\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} & \\
\text{s.t.} & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 & (i = 1, \ldots, m) \\
& \displaystyle\sum_{i=1}^{m} w_{ij}x_{ij} \leq b_j & (j = 1, \ldots, n) \\
& x_{ij} \in \{0, 1\}. &
\end{array}
$$

PLI: il problema di posizionare ambulanze

- Ambulanze possono essere posizionate in prefissati luoghi.
- Malati importanti devono essere raggiunti in al piú 8 minuti.

Problema

Come posizionare il minimo numero di ambulanze per arrivare in ogni posto in al piú 8 minuti?

- Dati:

$I =$ insieme di possibili posizioni delle ambulanze.

$J =$ insieme delle richieste.

$$a_{ij} = \begin{cases} 1 & \text{se é possibile andare da } i \text{ a } j \text{ in al piú 8 minuti} \\ 0 & \text{altrimenti} \end{cases}$$

Struttura del modello matematico

- Decisioni (dove porre le ambulanze) $\longrightarrow$ variabili

$$x_i = \begin{cases} 1 & \text{se un'ambulanza é posizionata al posto } i \\ 0 & \text{altrimenti} \end{cases}$$

- Il piú piccolo numero di ambulanze $\longrightarrow$ funzione obiettivo

- Richieste: tutte gli utenti raggiunti in al piú 8 minuti $\longrightarrow$ vincoli

Modello matematico

Funzione obiettivo

Minimizzare il numero di ambulanze

$$\min \sum_{i \in I} x_i$$

Vincoli

Per ogni posizione j almeno un'ambulanza deve arrivare in al piú 8 minuti

$$\sum_{i \in I} a_{ij} x_i \geq 1, \qquad \forall \; j \in J$$

Dominio delle variabili

$$x_i \in \{0, 1\}, \qquad \forall \; i \in I$$

Consideriamo un problema di programmazione lineare intera (PLI) in formato standard:

$$\begin{cases} \max \ c^T x \\ A x \leq b \\ x \in \mathbb{Z}^n \end{cases} \qquad (\mathcal{P})$$

Definizione

Il problema di PL

$$\begin{cases} \max \ c^T x \\ A x \leq b \end{cases} \qquad (RC)$$

é detto rilassamento continuo del problema $(\mathcal{P})$.

Quale é la relazione tra $(\mathcal{P})$ e (RC)?

Teorema

- Il valore ottimo di (RC) é una valutazione superiore per il valore ottimo di $(\mathcal{P})$.
- Se una soluzione ottima di (RC) é ammissibile per $(\mathcal{P})$, allora é ottima anche per $(\mathcal{P})$.

Usualmente la soluzione ottima di (RC) é inammissibile per $(\mathcal{P})$.

Relazioni tra PLI e PL

Per risolvere $(\mathcal{P})$, é sufficiente risolvere (RC) ed arrotondare la soluzione? NO

Esempio

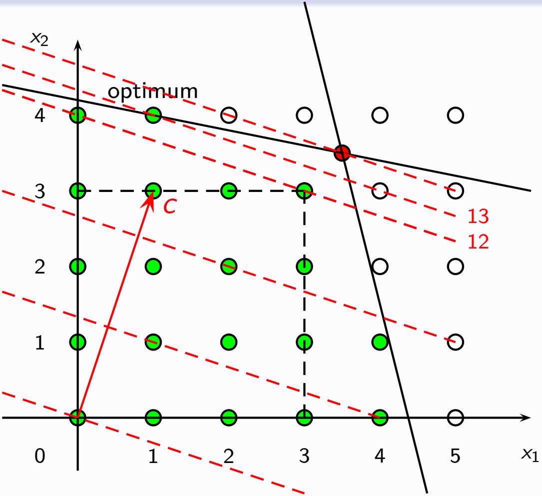
$$\begin{cases} \max \; x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

$\left(\frac{7}{2}, \frac{7}{2}\right)$ ottima per (RC)

arrotondamento $\rightarrow (3,3)$

$(3,3)$ non é ottima per $(\mathcal{P})$

$(1,4)$ é ottima per $(\mathcal{P})$



Relazioni tra PLI e PL

E' sufficiente risolvere (RC) e trovare la soluzione intera ammissibile piú vicina?
NO

Esempio

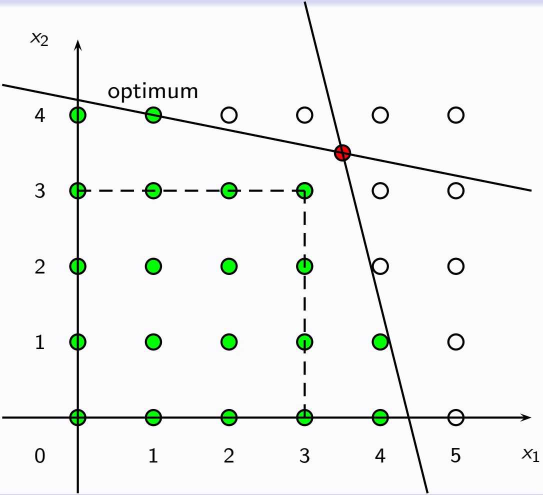
$$\begin{cases} \max \ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

$\left(\frac{7}{2}, \frac{7}{2}\right)$ ottima per (RC).

la soluzione intera ammissibile piú vicina é $(3,3)$.

$(3,3)$ non é ottima per $(\mathcal{P})$.

$(1,4)$ é ottima per $(\mathcal{P})$



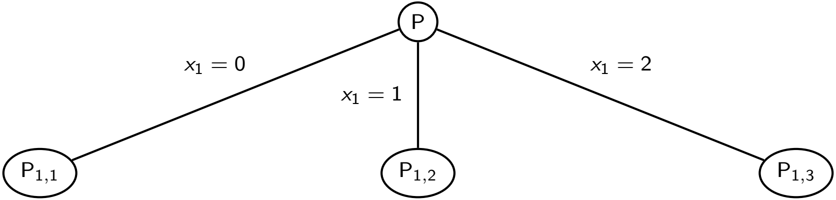
Consideriamo il problema di PLI:

$$\left\{\begin{array}{l} \max\ 5\,x_1 + 6\,x_2 \\ 3\,x_1 + 4\,x_2 \leq 7 \\ x_1 \geq 0,\ \ x_2 \geq 0 \\ x \in \mathbb{Z}^2 \end{array}\right. \qquad (\mathcal{P})$$

I vincoli implicano $x_1 = 0$ o 1 o 2.
Scriviamo una partizione della regione ammissibile Ω in tre sottoinsiemi:

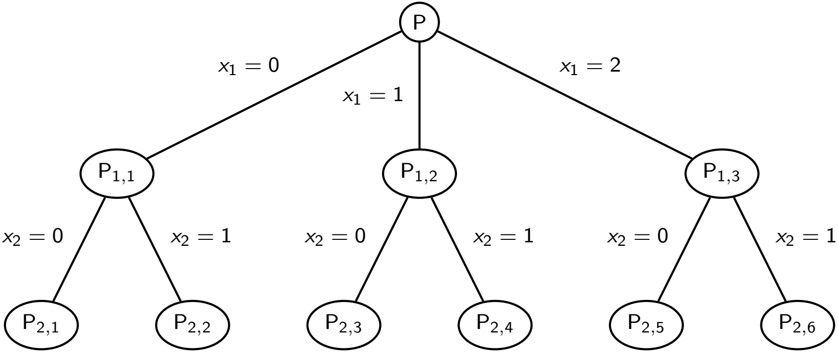
$$\Omega = (\Omega \cap \{x_1 = 0\}) \cup (\Omega \cap \{x_1 = 1\}) \cup (\Omega \cap \{x_1 = 2\})$$

corrispondente al primo livello dell'albero decisionale:



Enumerazione esplicita

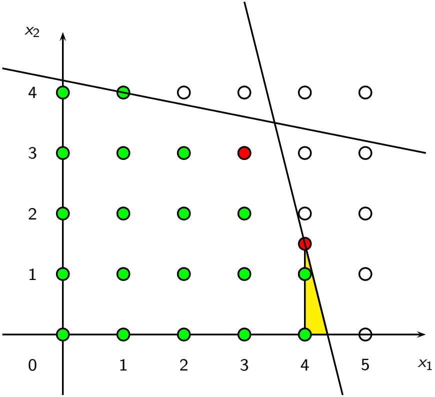
Similarmente, $x_2 = 0$ or 1. Quindi, l'albero delle decisioni completo é:



I nodi $P_{2,1}, \ldots, P_{2,5}$ corrispondono a soluzioni ammissibili di $(\mathcal{P})$, mentre il nodo $P_{2,6}$ corrisponde a $x = (2, 1)$ che é inammissibile. I valori della funzione obiettivo per i nodi $P_{2,1}, \ldots, P_{2,5}$ sono 0, 6, 5, 11, 10. Allora, la soluzione ottima é data da $P_{2,4}$, i.e., $x^* = (1, 1)$.

Enumerazione implicita

$$\begin{cases} \max\ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0,\ x \in \mathbb{Z}^2 \end{cases}$$



- Sappiamo che $(3,3)$ é ammissibile con valore 12.
- Consideriamo il vincolo aggiuntivo $x_1 \geq 4$: l'ottimo del rilassamento continuo del sottoproblema é $(4, 3/2)$ con valore 8.5 quindi le soluzioni ammissibili del sottoproblema sono peggiori di $(3,3)$
  $\rightarrow$ enumerazione implicita.

"Branch and Bound"

L'idea di base del *Branch and Bound* é:

- La regione ammissibile é partizionata, generando un albero di ricerca.
- Per ogni sottoregione (corrispondente ad un sottoproblema) il valore ottimo é approssimato tramite valutazioni.
- Le regioni che non possono contenere l'ottimo sono scartate.

"Branch and Bound"

Principali componenti

- *Branching*: come partizionare la regione ammissibile per generare sottoproblemi.
- *Bounding*: come stimare il valore dell'ottimo in ogni sottoregione.
- *Fathoming*: come depennare le sottoregioni che non contengono l'ottimo (come chiudere i nodi dell'albero di ricerca).

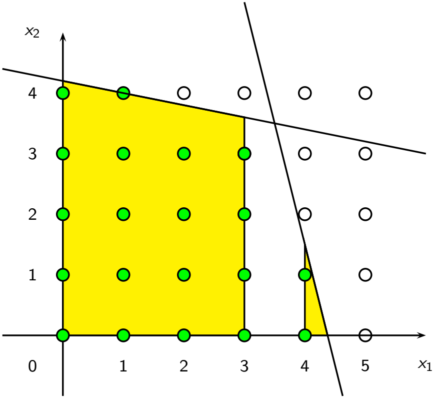
Come partizionare la regione ammissibile

- Le sottoregioni sono generate aggiungendo vincoli.
- Le sottoregioni devono essere una partizione della regione ammissibile, in modo tale da garantire che nessuna soluzione intera sia scartata.

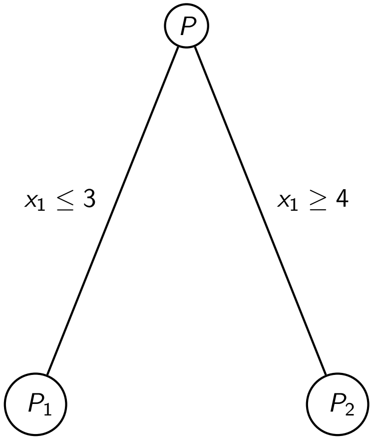
Regole di "Branching"

"Branching" bipartito

Ogni regione é partizionata in due sottoregioni, aggiungendo i vincoli $x_1 \leq 3$ e $x_1 \geq 4$



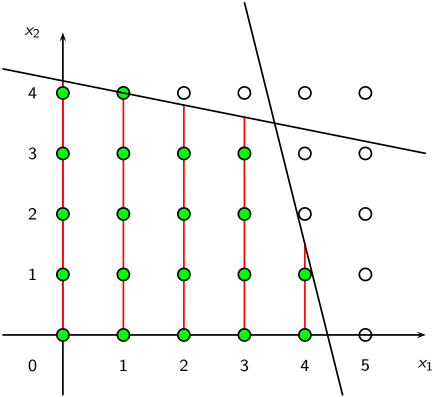
Albero di "branching" corrispondente



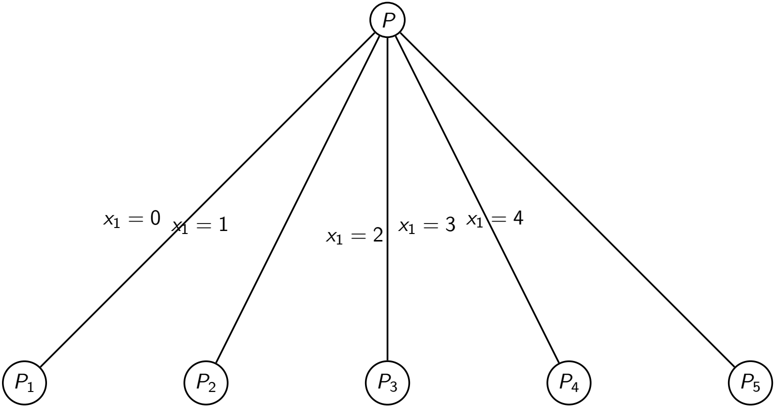
Regole di "Branching"

k-partito "branching"

Ogni regione é divisa in k sottoregioni, aggiungendo k vincoli E.g. $x_1 = 0$, $x_1 = 1$, $x_1 = 2$, $x_1 = 3$, e $x_1 = 4$



Albero di "branching" corrispondente



Valutazioni per problemi di massimo

Valutazioni inferiori LB date da soluzioni ammissibili.

Valutazioni superiori UB date da un *rilassamento*:

- Rilassamento continuo (eliminazione del vincolo di interezza)

  $x \in \{0, 1\} \Rightarrow 0 \leq x \leq 1$

  $x \in \mathbb{Z}_+ \Rightarrow x \geq 0$

- Eliminazione di uno o piú vincoli.

Criteri di ”fathoming” o ”pruning”

Un nodo dell’albero di decisione puó essere chiuso se una delle seguenti condizioni sussite:

- il sottoproblema é inammissibile.
- UB del sottoproblema é $\leq LB$ di $(\mathcal{P})$.
- UB del sottoproblema é $> LB$ e la soluzione ottima del sottoproblema é ammissibile per $(\mathcal{P})$. In tal caso noi aggiorniamo $LB = UB$.

Schema del metodo

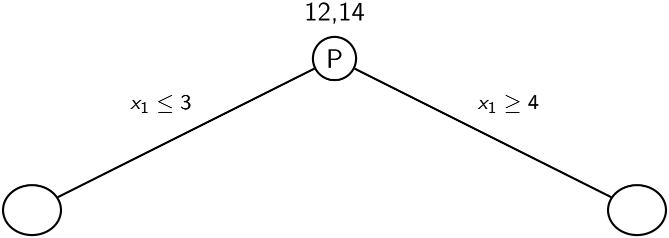
1. Calcola una valutazione inferiore LB del valore ottimo di $(\mathcal{P})$
2. Se tutti i nodi sono stati esplorati allora STOP
3. Seleziona il nodo (sottoproblema) da esplorare
4. Calcola una valutazione superiore UB del valore ottimo del sottoproblema
5. Controlla i criteri:
   se il sottoproblema é inammissibile allora chiudi il nodo
   se $UB \leq LB$, allora chiudi il nodo
   se $UB > LB$ e la soluzione ottima del sottoproblema é ammissibile per $(\mathcal{P})$, allora chiudi il nodo ed aggiorna $LB = UB$
6. se il nodo non é chiuso allora scendi
7. vai al passo 2

Esempio

Applica il "Branch and Bound" per risolvere il problema

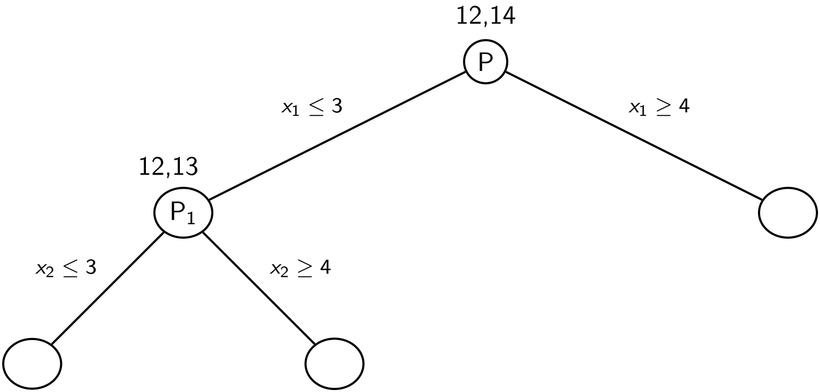
$$\begin{cases} \max\ x_1 + 3\,x_2 \\ x_1 + 5\,x_2 \leq 21 \\ 8\,x_1 + 2\,x_2 \leq 35 \\ x \geq 0,\ x \in \mathbb{Z}^2. \end{cases} \qquad (P)$$

Usiamo un albero bipartito.
Sappiamo che la soluzione ottima del rilassamento continuo é $(7/2, 7/2)$ quindi $UB(P) = 14$.
Conosciamo la soluzione ammissibile $(3, 3)$ quindi $LB = 12$.



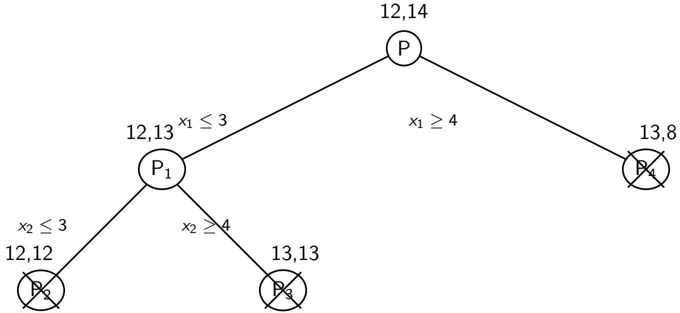
Esempio

La soluzione ottima del rilassamento continuo di P_1 é $(3, 18/5)$.
E' inammissibile con valore 13.8, quindi $UB(P_1) = 13 > 12 = LB$.
Il nodo P_1 rimane aperto.
Partizione: $x_2 \leq 3$ e $x_2 \geq 4$.



Esempio

La soluzione ottima del rilassamento continuo di P_2 é $(3, 3)$, quindi $UB(P_2) = 12 = LB$, chiudiamo il nodo P_2.
La soluzione ottima del rilassamento continuo di P_3 é $(1, 4)$ e $UB(P_3) = 13 > 12 = LB$.Poiché $(1, 4)$ é ammissibile per P, aggiorniamo $LB = 13$ e chiudiamo il nodo P_3.
La soluzione ottima del rilassamento continuo di P_4 é $(4, 3/2)$ con valore 8.5, quindi $UB(P_4) = 8 < 13 = LB$. Chiudiamo il nodo P_4.



Tutti i nodi sono chiusi, la soluzione ottima di P é $(1, 4)$ e il valore é 13.

Problema

Dati: un contenitore di capacitá C, n oggetti di valore $v_1, \ldots, v_n$ e peso $p_1, \ldots, p_n$. Quali oggetti inserisco nel contenitore, rispettando la sua capacitá, in modo da massimizzare il valore totale?

Esempio

Budget 100. Scegliere tra 9 investimenti possibili:

Investimento	1	2	3	4	5	6	7	8	9
Ricavo atteso	50	65	35	16	18	45	45	40	25
Costo	40	50	25	10	10	40	35	30	20

Modello dello zaino binario

Variabili: $x_j = \begin{cases} 1 & \text{se oggetto } j \text{ viene inserito,} \\ 0 & \text{altrimenti.} \end{cases}$

$$\begin{aligned} \max \ & \sum_{j=1}^{n} v_j x_j \\ & \sum_{j=1}^{n} p_j x_j \le C \\ & x_j \in \{0,1\} \quad \forall\, j = 1, \dots, n \end{aligned}$$

Metodi *greedy* per trovare una soluzione ammissibile.

Metodo 1

Esamino gli oggetti in ordine di valore decrescente.
Ogni oggetto viene inserito purché sia rispettato il vincolo di capacitá.

Esempio

Oggetto	1	2	3	4	5	6	7	8	9	
Valore	50	65	35	16	18	55	45	40	25	
Peso	31	39	26	21	25	28	29	27	23	$C = 100$

$x_2 = 1$, $x_6 = 1$, $x_1 = 1$, $x_7 = 0$, $x_8 = 0$, $x_3 = 0$, $x_9 = 0$, $x_5 = 0$, $x_4 = 0$.
Quindi $v_I(P) = 170$.

Metodo 2

Esamino gli oggetti in ordine di peso crescente.
Ogni oggetto viene inserito purché sia rispettato il vincolo di capacitá.

Esempio

Oggetto	1	2	3	4	5	6	7	8	9	
Valore	50	65	35	16	18	55	45	40	25	
Peso	31	39	26	21	25	28	29	27	23	$C = 100$

$x_4 = 1$, $x_9 = 1$, $x_5 = 1$, $x_3 = 1$, $x_8 = 0$, $x_6 = 0$, $x_7 = 0$, $x_1 = 0$, $x_2 = 0$.
Quindi $v_I(P) = 94$.

Metodo 3

Esamino gli oggetti in ordine di rendimento (valore/peso) decrescente.
Ogni oggetto viene inserito purché sia rispettato il vincolo di capacitá.

Esempio

Oggetto	1	2	3	4	5	6	7	8	9	
Valore	50	65	35	16	18	55	45	40	25	
Peso	31	39	26	21	25	28	29	27	23	$C = 100$

$x_6 = 1$, $x_2 = 1$, $x_1 = 1$, $x_7 = 0$, $x_8 = 0$, $x_3 = 0$, $x_9 = 0$, $x_4 = 0$, $x_5 = 0$.
Quindi $v_I(P) = 170$.

Teorema

Supponiamo che le variabili siano in ordine di rendimento decrescente.

Sia h l'indice tale che $\sum_{j=1}^{h} p_j \leq C$ e $\sum_{j=1}^{h+1} p_j > C$.

Il rilassamento continuo $\begin{cases} \max \ \sum_{j=1}^{n} v_j x_j \\ \sum_{j=1}^{n} p_j x_j \leq C \\ 0 \leq x_j \leq 1 \end{cases}$ ha come soluzione ottima

$$\bar{x}_1 = 1, \ \dots, \ \bar{x}_h = 1, \ \bar{x}_{h+1} = \frac{C - \sum_{j=1}^{h} p_j}{p_{h+1}}, \ \bar{x}_{h+2} = 0, \ \dots, \ \bar{x}_n = 0$$

Esempio

Sia dato il seguente problema:

$$\begin{cases} \max \; 10\,x_1 + 13\,x_2 + 18\,x_3 + 24\,x_4 \\ \qquad 2\,x_1 + 3\,x_2 + 4\,x_3 + 6\,x_4 \le 7 \\ x_j \in \{0,1\} \end{cases}$$

Disponiamo le variabili in ordine di rendimento decrescente:

variabili	1	3	2	4
rendimenti	5	4.5	4.33	4

Applicando il terzo algoritmo *greedy* otteniamo la soluzione ammissibile $(1,0,1,0)$ e quindi $v_I(P) = 28$.

L'ottimo del rilassamento continuo é $\left(1, \frac{1}{3}, 1, 0\right)$, quindi $v_S(P) = 32$.

Problema dello zaino a variabili intere

Problema

Dati: n oggetti, ognuno di valore v_j peso p_j, un contenitore di capacitá C. Quanti oggetti di ogni tipo inserisco nel contenitore per massimizzare il valore totale?

Modello

$x_j =$ numero (intero) di oggetti di tipo j inseriti nel contenitore

$$\max \sum_{j=1}^{n} v_j x_j$$

$$\sum_{j=1}^{n} p_j x_j \leq C$$

$$x_j \in \mathbb{N} \quad \forall\, j = 1, \dots, n$$

Metodi *greedy* per trovare una soluzione ammissibile.

Metodo 1

Esamino gli oggetti in ordine di valore decrescente.
Inserisco ogni oggetto nella massima quantitá possibile, purché sia rispettato il vincolo di capacitá.

Esempio

Oggetto	1	2	3	4	5	6	7	8	9	
Valore	50	65	35	16	18	55	45	40	25	
Peso	31	39	26	21	25	28	29	27	23	$C = 100$

$x_2 = 2$, $x_6 = 0$, $x_1 = 0$, $x_7 = 0$, $x_8 = 0$, $x_3 = 0$, $x_9 = 0$, $x_5 = 0$, $x_4 = 1$.
Quindi $v_I(P) = 146$.

Metodo 2

Esamino gli oggetti in ordine di peso crescente.
Inserisco ogni oggetto nella massima quantitá possibile, purché sia rispettato il vincolo di capacitá.

Esempio

Oggetto	1	2	3	4	5	6	7	8	9	
Valore	50	65	35	16	18	55	45	40	25	
Peso	31	39	26	21	25	28	29	27	23	$C = 100$

$x_4 = 4$, $x_9 = 0$, $x_5 = 0$, $x_3 = 0$, $x_8 = 0$, $x_6 = 0$, $x_7 = 0$, $x_1 = 0$, $x_2 = 0$.
Quindi $v_I(P) = 64$.

Metodo 3

Esamino gli oggetti in ordine di rendimento decrescente.
Inserisco ogni oggetto nella massima quantitá, rispettando il vincolo di capacitá.

Esempio

Oggetto	1	2	3	4	5	6	7	8	9	
Valore	50	65	35	16	18	55	45	40	25	
Peso	31	39	26	21	25	28	29	27	23	$C = 100$

$x_6 = 3$, $x_2 = 0$, $x_1 = 0$, $x_7 = 0$, $x_8 = 0$, $x_3 = 0$, $x_9 = 0$, $x_4 = 0$, $x_5 = 0$.
Quindi $v_I(P) = 165$.

Calcoliamo una $v_S(P)$ risolvendo il rilassamento continuo.

Teorema

Se $\max\limits_j \{ \frac{v_j}{p_j} \} = \frac{v_r}{p_r}$, allora il rilassamento continuo

$$\begin{cases} \max \sum\limits_{j=1}^{n} v_j x_j \\ \sum\limits_{j=1}^{n} p_j x_j \leq C \\ x \geq 0 \end{cases}$$

ha come soluzione ottima

$$\bar{x}_1 = 0, \ \ldots, \ \bar{x}_{r-1} = 0, \ \bar{x}_r = \frac{C}{p_r}, \ \bar{x}_{r+1} = 0, \ \ldots, \ \bar{x}_n = 0$$

e valore ottimo $C\, v_r / p_r$.

Esempio

Consideriamo il problema:

$$\begin{cases} \max\ 4\,x_1 + 20\,x_2 + 27\,x_3 + 26\,x_4 \\ \qquad 4\,x_1 + 19\,x_2 + 16\,x_3 + 14\,x_4 \leq 32 \\ x_j \in \mathbb{N} \end{cases} \tag{P}$$

Disponiamo le variabili in ordine di rendimento decrescente:

variabili	4	3	2	1
rendimenti	1.85	1.68	1.05	1

Il terzo algoritmo *greedy* trova la soluzione $(1, 0, 0, 2)$ con $v_I(P) = 56$.

La soluzione ottima del rilassamento continuo é $(0, 0, 0, \frac{32}{14})$, quindi $v_S(P) = 59$.

4. Problema di trasporto ottimo

Supponiamo di avere m luoghi di produzione collegati con n luoghi di raccolta.

Per fissare le idee si puó pensare alla distribuzione giornaliera su un territorio di un prodotto come ad esempio un carburante.

Supponiamo che siano note le capacitá produttive o_i, per $i = 1, \ldots, m$, le domande d_j, per $j = 1, \ldots, n$, ed il costo di trasporto da ogni luogo di produzione ad ogni luogo di destinazione.

Si voglia determinare un piano di trasporto compatibile con la produzione e con la richiesta e che minimizzi il costo totale.

Problema di trasporto ottimo

Supponiamo che il costo di spedizione sia proporzionale (lineare) e quindi esista il costo unitario c_{ij} del trasporto da i a j.

Indichiamo con x_{ij} la quantitá di merce da trasportare da i a j.

Fissato j, sommando su i le x_{ij} si ottiene la quantitá di merce che arriva al luogo di raccolta j e viceversa, fissato i, sommando su j le x_{ij} si ottiene la quantitá di merce spedita dal luogo di produzione i.

Il modello matematico é il seguente:

$$\begin{cases} \min \ \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{i=1}^{m} x_{ij} \geq d_j \qquad \text{per ogni } j = 1, \dots, n \\ \sum_{j=1}^{n} x_{ij} \leq o_i \qquad \text{per ogni } i = 1, \dots, m \\ x_{ij} \geq 0 \end{cases} \tag{2}$$

Naturalmente il problema potrebbe non avere alcuna soluzione qualora

$$\sum_{j=1}^{n} d_j > \sum_{i=1}^{m} o_i,$$

cioé se la domanda totale supera l'offerta totale. Nel caso in cui ci sia, invece, un eccesso di produzione, cioé

$$\sum_{j=1}^{n} d_j < \sum_{i=1}^{m} o_i,$$

si puó pensare di aggiungere un luogo di raccolta fittizio a cui spedire (a costo nullo) l'eccesso di produzione.

A meno di aggiungere un nodo fittizio di raccolta, potremmo supporre che nel modello precedente i vincoli di produzione e di domanda siano tutti vincoli di uguaglianza, cioé:

$$\begin{cases} \min \ \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{j=1}^{n} x_{ij} = o_i \qquad \text{per ogni } i = 1, \dots, m \\ \sum_{i=1}^{m} x_{ij} = d_j \qquad \text{per ogni } j = 1, \dots, n \\ x_{ij} \geq 0 \end{cases} \tag{3}$$

È evidente che il modello é compatibile con il trasporto di merce che sia divisibile (tipo carburante), in quanto la soluzione del modello matematico potrebbe non essere a componenti intere anche con vettori $(o_1, \dots, o_m)$ e $(d_1, \dots d_n)$ a componenti intere.

Se il bene da trasportare fosse indivisibile (tipo elettrodomestici, mobili, etc.) bisognerebbe aggiungere nel problema il vincolo

$$x_{ij} \in \mathbb{Z} \qquad \text{per ogni } i = 1, \dots, m \quad \text{e} \quad \text{per ogni } j = 1, \dots, n.$$

Esempio

Un'azienda elettrica possiede tre stabilimenti che devono soddisfare le esigenze di 4 cittá.

Ogni stabilimento puó fornire un certo numero di kWh di elettricitá: 35 milioni lo stabilimento 1, 50 milioni lo stabilimento 2 e 40 milioni lo stabilimento 3.

Il picco di domanda delle cittá che avviene verso le 2 del pomeriggio é di 45 milioni per la cittá 1, 20 milioni per la cittá 2, di 30 milioni per la cittá 3 e di 30 milioni per la cittá 4.

Il costo per mandare 1 milione di kWh dipende dalla distanza che l'elettricitá deve percorrere ed é indicato nella tabella seguente:

	cittá 1	cittá 2	cittá 3	cittá 4
stabilimento 1	8	6	10	9
stabilimento 2	9	12	13	7
stabilimento 3	14	9	16	5

Esempio

Sia x_{ij} il numero di kWh (in milioni) prodotto dallo stabilimento i per la cittá j.

$$\begin{cases} \min\ 8\,x_{11} + 6\,x_{12} + 10\,x_{13} + 9\,x_{14} + 9\,x_{21} + 12\,x_{22} + 13\,x_{23} + 7\,x_{24} + \\ \qquad + 14\,x_{31} + 9\,x_{32} + 16\,x_{33} + 5\,x_{34} \\ x_{11} + x_{12} + x_{13} + x_{14} = 35 \\ x_{21} + x_{22} + x_{23} + x_{24} = 50 \\ x_{31} + x_{32} + x_{33} + x_{34} = 40 \\ x_{11} + x_{21} + x_{31} = 45 \\ x_{12} + x_{22} + x_{32} = 20 \\ x_{13} + x_{23} + x_{33} = 30 \\ x_{14} + x_{24} + x_{34} = 30 \\ x \geq 0 \end{cases}$$

La soluzione ottima é:

$$\begin{array}{llll} x_{11} = 0 & x_{12} = 10 & x_{13} = 25 & x_{14} = 0 \\ x_{21} = 45 & x_{22} = 0 & x_{23} = 5 & x_{24} = 0 \\ x_{31} = 0 & x_{32} = 10 & x_{33} = 0 & x_{34} = 30 \end{array}$$

ed il costo totale é di 1020 milioni di euro.

5. Problema del *bin packing*

Problema

Dati: n oggetti di peso $p_1, \dots, p_n$ e m contenitori ognuno di capacitá C.

Trovare il minimo numero di contenitori in cui inserire tutti gli oggetti.

Modello del bin packing

Variabili: $x_{ij} = \begin{cases} 1 & \text{se oggetto } j \text{ inserito contenitore } i, \\ 0 & \text{altrimenti,} \end{cases}$ $\quad y_i = \begin{cases} 1 & \text{se } i \text{ é usato,} \\ 0 & \text{altrimenti.} \end{cases}$

$$\min \sum_{i=1}^{m} y_i$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall\ j = 1, \ldots, n \tag{4}$$

$$\sum_{j=1}^{n} p_j x_{ij} \leq C y_i \quad \forall\ i = 1, \ldots, m \tag{5}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\ i, j$$

$$y_i \in \{0, 1\} \qquad \forall\ i$$

(4): ogni oggetto é inserito in un solo contenitore.
(5): capacitá contenitori.

Metodi euristici

Per calcolare una soluzione ammissibile (e quindi una valutazione superiore) descriviamo 3 algoritmi *greedy*.

Algoritmo Next-Fit Decreasing (NFD)

Esamina gli oggetti in ordine di peso decrescente.
Il primo contenitore é il contenitore corrente.
Se possibile, assegna un oggetto al contenitore corrente;
altrimenti assegnalo ad un nuovo contenitore, che diventa quello corrente.

Esempio

j	1	2	3	4	5	6	7	8	9	
p_j	70	60	50	33	33	33	11	7	3	$C = 100$

Contenitori	Oggetti	Capacitá residua
1	1	30
2	2	40
3	3 4	50 17
4	5 6 7 8 9	67 34 23 16 13

Algoritmo First-Fit Decreasing (FFD)

Esamina gli oggetti in ordine di peso decrescente.
Assegna ogni oggetto al primo contenitore usato che puó contenerlo.
Se nessuno di essi puó contenerlo, assegna l'oggetto ad un nuovo contenitore.

Esempio

j	1	2	3	4	5	6	7	8	9	
p_j	70	60	50	33	33	33	11	7	3	$C = 100$

Contenitori	Oggetti	Capacitá residua
1	1 7 8 9	30 19 12 9
2	2 4	40 7
3	3 5	50 17
4	6	67

Metodi euristici

Algoritmo Best-Fit Decreasing (BFD)

Esamina gli oggetti in ordine di peso decrescente.
Tra tutti i contenitori usati che possono contenere un oggetto, scegli quello con la minima capacitá residua.
Se nessuno di essi puó contenerlo, assegna l'oggetto ad un nuovo contenitore.

Esempio

j	1	2	3	4	5	6	7	8	9	
p_j	70	60	50	33	33	33	11	7	3	$C = 100$

Contenitori	Oggetti	Capacitá residua
1	1	30
2	2 4 8	40 7 0
3	3 5 7 9	50 17 6 3
4	6	67

Rilassamento continuo

Per calcolare una valutazione inferiore $v_I(P)$ usiamo il rilassamento continuo.

Teorema

Il rilassamento continuo di (P):

$$\begin{cases} \min \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_{ij} = 1 \quad \forall\ j = 1, \ldots, n \\ \sum_{j=1}^{n} p_j x_{ij} \leq C y_i \quad \forall\ i = 1, \ldots, n \\ 0 \leq x_{ij} \leq 1 \quad \forall\ i, j \\ 0 \leq y_i \leq 1 \quad \forall\ i \end{cases} \tag{RC}$$

ha come soluzione ottima $x_{ij} = \begin{cases} 1 & \text{se } i = j, \\ 0 & \text{se } i \neq j, \end{cases} \quad y_i = \dfrac{p_i}{C}$ per ogni i.

$L = \left\lceil \sum_{i=1}^{n} p_i / C \right\rceil$ é una $v_I(P)$.

Esempio

Sia dato il seguente problema:

j	1	2	3	4	5	6	7	
p_j	99	98	64	45	40	23	17	$C = 100$

Calcoliamo una $v_I(P)$:

$$L = \left\lceil \frac{386}{100} \right\rceil = 4$$

Calcoliamo una $v_S(P)$:

Oggetti	1	2	3	4	5	6	7
Algoritmo NFD	1	2	3	4	4	5	5
Algoritmo FFD	1	2	3	4	4	3	5
Algoritmo BFD	1	2	3	4	4	3	5

Quindi $v_S(P) = 5$.

6. Problema del commesso viaggiatore (TSP)

Problema

Grafo (N, A) completo; c_{ij} = costi sugli archi.
Trovare un ciclo di costo minimo che passi su tutti i nodi una ed una sola volta (ciclo hamiltoniano).

Quante sono le soluzioni ammissibili? n!

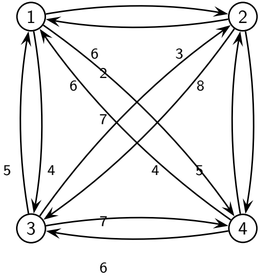
Applicazioni

- trasporti, logistica: (N', A') rete stradale. $S \subseteq N'$, cerco ciclo di costo minimo che passi su tutti i nodi di S. Il problema é un TSP sul grafo (N, A), dove $N = S$, $A = S \times S$, c_{ij} = costo cammino minimo da i a j sul grafo (N', A').
- produzione di circuiti integrati
- data analysis
- sequenze DNA

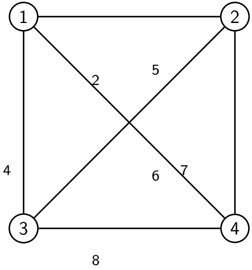
TSP simmetrico e asimmetrico

Se la matrice dei costi é simmetrica, cioé $c_{ij} = c_{ji}$ per ogni arco (i, j), il problema é detto simmetrico; altrimenti é detto asimmetrico.

TSP asimmetrico



TSP simmetrico



Prima tratteremo il problema asimmetrico (piú generale) e poi quello simmetrico (caso particolare).

Modello

Variabili: $x_{ij} = \begin{cases} 1 & \text{se arco } (i,j) \in \text{ciclo hamiltoniano,} \\ 0 & \text{altrimenti.} \end{cases}$

$$
\begin{aligned}
\min \sum_{(i,j)\in A} c_{ij}\, x_{ij} & \\
\sum_{i\in N\setminus\{j\}} x_{ij} = 1 & \qquad \forall\ j \in N \\
\sum_{j\in N\setminus\{i\}} x_{ij} = 1 & \qquad \forall\ i \in N \\
\sum_{(i,j)\in A:\ i\in S,\ j\notin S} x_{ij} \geq 1 & \qquad \forall\ S \subseteq N, \quad S \neq \emptyset, N \\
x_{ij} \in \{0,1\} & \qquad \forall\ (i,j) \in A
\end{aligned}
$$

Eliminando i vincoli di connessione dal modello si ottiene un problema di assegnamento di costo minimo che é quindi una valutazione inferiore:

$$\min \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}$$
$$\sum_{i \in N} x_{ij} = 1 \quad \forall\ j \in N$$
$$\sum_{j \in N} x_{ij} = 1 \quad \forall\ i \in N$$
$$x_{ij} \in \{0, 1\} \quad \forall\ i, j \in N$$

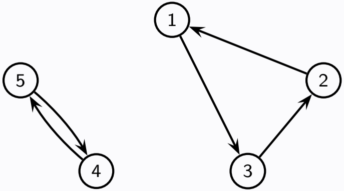
che é facile da risolvere.

La soluzione ottima di tale rilassamento é, a priori, una famiglia di cicli orientati che coprono tutti i nodi.

Esempio

Consideriamo la seguente matrice dei costi:

	1	2	3	4	5
1	–	33	13	25	33
2	33	–	46	58	76
3	39	33	–	12	30
4	35	29	12	–	23
5	60	54	30	23	–



La soluzione ottima del rilassamento é formata da due cicli:

$$x_{13} = 1, \quad x_{32} = 1, \quad x_{21} = 1, \quad x_{45} = 1, \quad x_{54} = 1,$$

e ha valore $125 = v_I(P)$.

Osservazione

Quando la matrice dei costi é fortemente asimmetrica, il valore ottimo del problema di assegnamento é di solito una buona stima del valore ottimo del TSP.

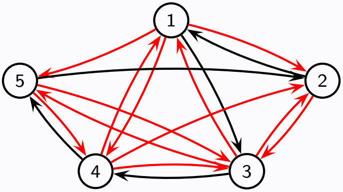
Quando la matrice dei costi é simmetrica, l'assegnamento di costo minimo solitamente é formato da un gran numero di cicli orientati.

Metodo *greedy* sugli archi

Dispongo gli archi in ordine crescente di costo.
Seguendo l'ordine, inserisco un arco se vengono rispettati tutti i vincoli.

Esempio

	1	2	3	4	5
1	–	33	13	25	33
2	33	–	46	58	76
3	39	33	–	12	30
4	35	29	12	–	23
5	60	54	30	23	–



$x_{34} = 1$, $x_{43} = 0$, $x_{13} = 1$, $x_{45} = 1$, $x_{54} = 0$, $x_{14} = 0$, $x_{42} = 0$, $x_{35} = 0$, $x_{53} = 0$, $x_{12} = 0$, $x_{15} = 0$, $x_{21} = 1$, $x_{32} = 0$, $x_{41} = 0$, $x_{31} = 0$, $x_{23} = 0$, $x_{52} = 1$.

Il ciclo trovato é 3-4-5-2-1-3 di costo 135.

Algoritmo del nodo piú vicino

Parti da un nodo *i*. Il nodo successivo é il piú vicino a *i* tra quelli non ancora visitati.

Esempio

	1	2	3	4	5
1	–	33	13	25	33
2	33	–	46	58	76
3	39	33	–	12	30
4	35	29	12	–	23
5	60	54	30	23	–

Partendo dal nodo 1 si ottiene il ciclo 1–3–4–5–2–1 di costo 135.

Partendo dal nodo 5 si ottiene il ciclo 5–4–3–2–1–5 di costo 134.

Algoritmi di inserimento

Costruisco un ciclo su un sottoinsieme di nodi. Estendo questo ciclo inserendo uno alla volta i nodi rimanenti fino ad inserire tutti i nodi.

Questo schema dipende da:

- come costruisco il ciclo iniziale: ciclo qualsiasi, ciclo sui 3 nodi che formano il triangolo piú grande, ciclo che segue l'involucro convesso dei nodi (quando $c_{ij} =$ distanza euclidea tra i e j), ...
- come scelgo il nodo da inserire: il piú vicino al ciclo, il piú lontano dal ciclo, quello il cui inserimento causa il minimo incremento nella lunghezza del ciclo, ...
- dove inserisco il nodo scelto: di solito é inserito nel punto che causa il minimo incremento nella lunghezza del ciclo

Esempio

	1	2	3	4	5
1	–	33	13	25	33
2	33	–	46	58	76
3	39	33	–	12	30
4	35	29	12	–	23
5	60	54	30	23	–

Scelgo un ciclo: 1-2-3-1.
Il nodo 4 ha distanza 12 dal ciclo, mentre il nodo 5 ha distanza 30. Scelgo il nodo piú vicino al ciclo: 4.
Dove inserisco il nodo 4?
Se inserisco 4 tra 1 e 2, la lunghezza del ciclo aumenta di $25 + 29 - 33 = 21$
Se inserisco 4 tra 2 e 3, la lunghezza del ciclo aumenta di $58 + 12 - 46 = 24$
Se inserisco 4 tra 3 e 1, la lunghezza del ciclo aumenta di $12 + 35 - 39 = 8$
Quindi inserisco il nodo 4 tra 3 e 1. Il ciclo diventa 1-2-3-4-1.
Dove inserisco il nodo 5? Conviene inserirlo tra 3 e 4, il ciclo hamiltoniano é 1-2-3-5-4-1 di costo 167.

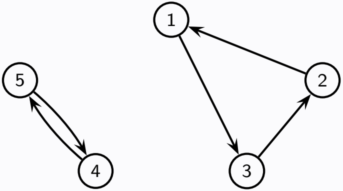
Algoritmo basato sulla soluzione ottima del rilassamento.

Algoritmo delle toppe (*patching*)

1. L'assegnamento di costo minimo é formato da una famiglia di cicli orientati $F = \{C_1, \ldots, C_p\}$.
2. Per ogni coppia di cicli $C_i, C_j \in F$, calcola l'incremento di costo γ_{ij} corrispondente alla fusione di C_i e C_j nel modo piú conveniente possibile.
3. Effettua la fusione dei due cicli C_i e C_j ai quali corrisponde il minimo valore di γ_{ij}. Aggiorna F.
4. Se F contiene un solo ciclo allora STOP altrimenti torna al passo 2.

Esempio

	1	2	3	4	5
1	–	33	13	25	33
2	33	–	46	58	76
3	39	33	–	12	30
4	35	29	12	–	23
5	60	54	30	23	–



La soluzione ottima del rilassamento é formata da due cicli: 1–3–2–1 e 4–5–4, con $v_I(P) = 125$.

Le possibili fusioni dei due cicli sono le seguenti:

sostituire gli archi	con gli archi	si ottiene il ciclo	di costo
$(1,3)$ e $(4,5)$	$(1,5)$ e $(4,3)$	1-5-4-3-2-1	134
$(1,3)$ e $(5,4)$	$(1,4)$ e $(5,3)$	1-4-5-3-2-1	144
$(2,1)$ e $(4,5)$	$(2,5)$ e $(4,1)$	1-3-2-5-4-1	180
$(2,1)$ e $(5,4)$	$(2,4)$ e $(5,1)$	1-3-2-4-5-1	187
$(3,2)$ e $(4,5)$	$(3,5)$ e $(4,2)$	1-3-5-4-2-1	128
$(3,2)$ e $(5,4)$	$(3,4)$ e $(5,2)$	1-3-4-5-2-1	135

La fusione piú conveniente trova il ciclo 1-3-5-4-2-1 di costo 128.

TSP simmetrico

Il modello per il TSP asimmetrico é valido ovviamente anche per il TSP simmetrico.
Vediamo ora un altro modello valido solo per il TSP simmetrico.
Nel TSP simmetrico possiamo supporre che il grafo (N, A) sia non orientato.
Quanti sono i cicli hamiltoniani? n!/2

Modello

Variabili: $x_{ij} = \begin{cases} 1 & \text{se } \{i,j\} \in \text{ciclo hamiltoniano,} \\ 0 & \text{altrimenti,} \end{cases}$ con $i < j$.

$$\min \sum_{i \in N} \sum_{\substack{j \in N \\ j > i}} c_{ij} x_{ij}$$

$$\sum_{\substack{j \in N \\ j > i}} x_{ij} + \sum_{\substack{j \in N \\ j < i}} x_{ji} = 2 \qquad \forall\ i \in N \tag{6}$$

$$\sum_{i \in S} \sum_{\substack{j \notin S \\ j > i}} x_{ij} + \sum_{i \notin S} \sum_{\substack{j \in S \\ j > i}} x_{ij} \geq 1 \qquad \forall\ S \subset N, \tag{7}$$

$$x_{ij} \in \{0,1\} \quad \forall\ i,j \in N, \quad i < j$$

(6): due archi incidenti su ogni nodo (cioé ogni nodo ha grado 2).
(7): eliminazione di sottocicli.

- Assegnamento di costo minimo (come nel caso asimmetrico)
- r-albero di costo minimo

Fissiamo un nodo r nel grafo. Un r–albero é un insieme di n archi di cui

2 sono incidenti sul nodo r.

$n - 2$ formano un albero di copertura sui nodi diversi da r.

Ogni ciclo hamiltoniano é un r–albero.

Il problema dell'r–albero di costo minimo é facile da risolvere: basta trovare

2 archi di costo minimo incidenti sul nodo r

un albero di copertura di costo minimo sui nodi diversi da r

Problema

Dato un grafo (N, A) non orientato e connesso, con costi c_{ij} sugli archi, trovare un albero (insieme di archi connesso e privo di cicli) di copertura (che connetta tutti i nodi) di costo minimo.

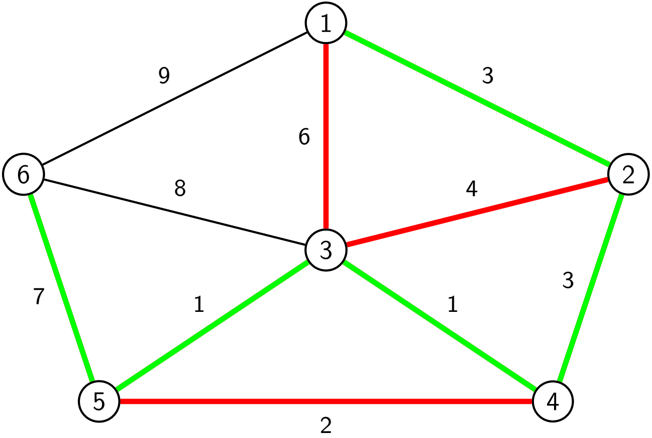
Algoritmo di Kruskal

È un algoritmo di tipo greedy:

- crea la soluzione aggiungendo un arco per volta.
- non ridiscute decisioni giá prese.

1. Ordina gli archi $a_1, \dots, a_m$ in ordine crescente di costo. $T = \emptyset$, $h = 1$.
2. **se** $T \cup \{a_h\}$ non contiene un ciclo **allora** inserisci a_h in T
3. **se** $|T| = n - 1$ **allora** STOP
   altrimenti $h = h + 1$ e torna al passo 2

Algoritmo di Kruskal: esempio



$\{1,2\}$	$\{1,3\}$	$\{1,6\}$	$\{2,3\}$	$\{2,4\}$	$\{3,4\}$	$\{3,5\}$	$\{3,6\}$	$\{4,5\}$	$\{5,6\}$
3	6	9	4	3	1	1	8	2	7

$\{3,4\}$	$\{3,5\}$	$\{4,5\}$	$\{1,2\}$	$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\{5,6\}$	$\{3,6\}$	$\{1,6\}$
1	1	2	3	3	4	6	7	8	9

archi in ordine crescente di costo

$\{3,4\}$	$\{3,5\}$	$\{4,5\}$	$\{1,2\}$	$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\{5,6\}$	$\{3,6\}$	$\{1,6\}$
1	1	2	3	3	4	6	7	8	9

$\{3,4\}$ non forma un ciclo con $T = \emptyset$

$\{3,4\}$	$\{3,5\}$	$\{4,5\}$	$\{1,2\}$	$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\{5,6\}$	$\{3,6\}$	$\{1,6\}$
1	1	2	3	3	4	6	7	8	9

$\{3,5\}$ non forma un ciclo con T

$\{3,4\}$	$\{3,5\}$	$\{4,5\}$	$\{1,2\}$	$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\{5,6\}$	$\{3,6\}$	$\{1,6\}$
1	1	2	3	3	4	6	7	8	9

$\{4,5\}$ forma un ciclo con T!

$\{3,4\}$	$\{3,5\}$	$\{4,5\}$	$\{1,2\}$	$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\{5,6\}$	$\{3,6\}$	$\{1,6\}$
1	1	2	3	3	4	6	7	8	9

$\{1,2\}$ non forma un ciclo con T

$\{3,4\}$	$\{3,5\}$	$\{4,5\}$	$\{1,2\}$	$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\{5,6\}$	$\{3,6\}$	$\{1,6\}$
1	1	2	3	3	4	6	7	8	9

$\{2,4\}$ non forma un ciclo con T

$\{3,4\}$	$\{3,5\}$	$\{4,5\}$	$\{1,2\}$	$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\{5,6\}$	$\{3,6\}$	$\{1,6\}$
1	1	2	3	3	4	6	7	8	9

$\{2,3\}$ forma un ciclo con T!

$\{3,4\}$	$\{3,5\}$	$\{4,5\}$	$\{1,2\}$	$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\{5,6\}$	$\{3,6\}$	$\{1,6\}$
1	1	2	3	3	4	6	7	8	9

$\{1,3\}$ forma un ciclo con T!

$\{3,4\}$	$\{3,5\}$	$\{4,5\}$	$\{1,2\}$	$\{2,4\}$	$\{2,3\}$	$\{1,3\}$	$\{5,6\}$	$\{3,6\}$	$\{1,6\}$
1	1	2	3	3	4	6	7	8	9

$\{5,6\}$ non forma un ciclo con T. STOP

Rilassamenti: r-albero di costo minimo

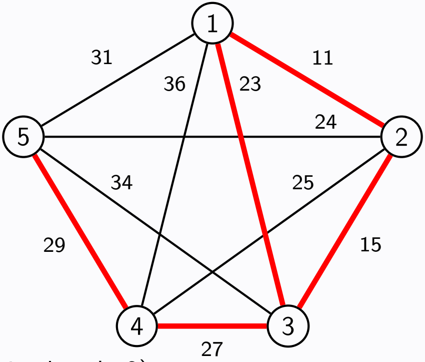
Per ottenenere il rilassamento, nel modello del TSP simmetrico possiamo eliminare i vincoli in rosso.

$$\begin{cases} \min \ \sum_{i} \sum_{j>i} c_{ij} x_{ij} \\ \sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} = 2 \qquad \forall \ i \neq r \\ \sum_{j>r} x_{rj} + \sum_{j<r} x_{jr} = 2 \\ \sum_{i \in S} \sum_{\substack{j \notin S \\ j>i}} x_{ij} + \sum_{i \notin S} \sum_{\substack{j \in S \\ j>i}} x_{ij} \geq 1 \qquad \forall \ S \subset N, \quad S \neq \emptyset, N \\ x_{ij} \in \{0, 1\} \quad i < j \end{cases}$$

Eliminando quindi i vincoli sul grado dei nodi diversi da r si ottiene l'r-albero di costo minimo.

Esempio

Il 2–albero di costo minimo sul grafo



$\{1,2\}$, $\{2,3\}$ (incidenti sul nodo 2)
$\{1,3\}$, $\{3,4\}$, $\{4,5\}$ (albero di copertura sui nodi diversi da 2).
Ha costo $105 = v_I(P)$.

Per trovare un ciclo hamiltoniano possiamo usare i metodi visti per il TSP asimmetrico:

- metodo *greedy* sugli archi.
- algoritmo del nodo piú vicino.
- algoritmi di inserimento.

Dopo aver trovato una soluzione ammissibile, provo a migliorarla.

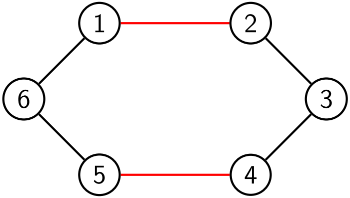
1. Trovo una soluzione ammissibile x
2. Genero un insieme $N(x)$ di soluzioni "vicine" ad x (intorno)
3. Se in $N(x)$ esiste una soluzione ammissibile x' migliore di x
   allora $x := x'$ e torno al passo 2

4 altrimenti STOP (x é una soluzione ottima locale)

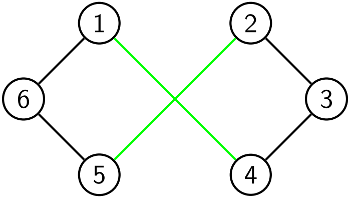
Nel caso del TSP, come definisco un intorno $N(x)$?

Una possibile scelta é:

$$N(x) = \{\text{cicli hamiltoniani che hanno 2 archi diversi da } x\}.$$



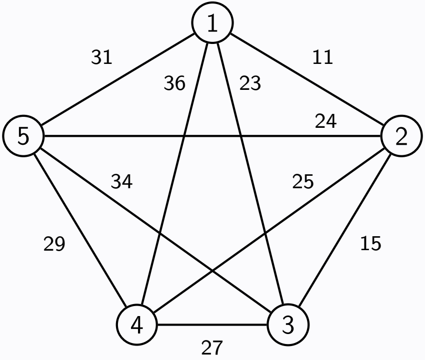
x



$x' \in N(x)$

Esempio

Consideriamo il TSP sul grafo



Effettuiamo la ricerca locale a partire dal ciclo 1-3-5-2-4 di costo 142.

Esempio

x	costo	elimino archi	inserisco archi	x'	costo
1-3-5-2-4-1	142	(1,3) (5,2)	(1,5) (3,2)	1-5-3-2-4-1	141
1-5-3-2-4-1	141	(1,5) (2,4)	(1,2) (5,4)	1-2-3-5-4-1	125
1-2-3-5-4-1	125	(1,2) (3,5)	(1,3) (2,5)	1-3-2-5-4-1	127
		(1,2) (5,4)	(1,5) (2,4)	1-5-3-2-4-1	141
		(2,3) (5,4)	(2,5) (3,4)	1-2-5-3-4-1	132
		(2,3) (4,1)	(2,4) (3,1)	1-3-5-4-2-1	122
1-3-5-4-2	122	...	...	...	...

7. Problemi di localizzazione

I problemi di *facility location* riguardano il posizionamento ottimale di servizi.
Molte applicazioni sia nel settore pubblico che in quello privato:

- porti, aeroporti, scuole, ospedali, fermate autobus, stazioni metropolitana, piscine, ...
- stabilimenti produttivi, magazzini, punti vendita, centri di calcolo ...

Problema di copertura

Problema

Dati:
$I =$ insieme di nodi domanda
$J =$ insieme di siti candidati ad ospitare il servizio.
$c_j =$ costo per aprire il servizio nel sito j
$d_{ij} =$ distanza tra nodo di domanda i e sito candidato j
$D =$ distanza di copertura, cioé i é coperto da j se $d_{ij} \leq D$

Decidere in quali siti aprire il servizio in modo da coprire tutti i nodi di domanda con l'obiettivo di minimizzare il costo totale.

Esempio

Posizionamento di servizi di emergenza (ambulanze, caserme dei pompieri, ...)

Modello di copertura

Definiamo $a_{ij} = \begin{cases} 1 & \text{se } d_{ij} \leq D, \\ 0 & \text{altrimenti.} \end{cases}$ Variabili:
$x_j = \begin{cases} 1 & \text{se nel sito } j \text{ apriamo il servizio,} \\ 0 & \text{altrimenti.} \end{cases}$

$$\begin{aligned} \min \sum_{j \in J} c_j \, x_j \\ \sum_{j \in J} a_{ij} \, x_j \geq 1 \quad \forall \ i \in I \\ x_j \in \{0, 1\} \quad \forall \ j \in J \end{aligned} \tag{8}$$

(8): ogni nodo di domanda deve essere coperto dal servizio.

Esempio

Una ASL deve decidere dove posizionare alcune ambulanze nella cittá.
Ci sono 5 siti diversi, ognuno dei quali puó ospitare un'ambulanza ed é necessario servire 10 zone della cittá.
I tempi (in minuti) per raggiungere le zone da ogni sito sono riassunti nella seguente tabella:

	siti				
zone	1	2	3	4	5
1	2	7	13	11	12
2	5	13	9	12	11
3	7	15	2	16	4
4	19	7	6	18	2
5	11	2	12	8	19
6	12	18	9	10	11
7	13	17	6	18	7
8	18	12	12	13	6
9	11	6	7	8	9
10	8	14	5	6	13

Esempio

I costi per posizionare le ambulanze nei vari siti sono:

siti	1	2	3	4	5
costo	6	9	10	8	7

L'ASL deve decidere quante ambulanze posizionare e dove posizionarle in modo che ogni zona possa essere raggiunta da un'ambulanza entro 10 minuti, con l'obiettivo di minimizzare i costi.

Problema di massima copertura

Quando il budget é limitato, si puó non riuscire a coprire tutti i nodi di domanda.

Problema

Dati:
$I =$ insieme di nodi domanda
$h_i =$ domanda associata al nodo i
$J =$ insieme di siti candidati ad ospitare il servizio
$p =$ numero prefissato di posti dove aprire il servizio
$d_{ij} =$ distanza tra nodo di domanda i e sito candidato j
$D =$ distanza di copertura

Decidere in quali p siti aprire il servizio in modo da massimizzare la domanda coperta.

Modello

Variabili: $x_j = \begin{cases} 1 & \text{se apriamo il servizio nel sito } j, \\ 0 & \text{altrimenti.} \end{cases}$ $z_i = \begin{cases} 1 & \text{se il nodo } i \text{ é coperto,} \\ 0 & \text{altrimenti.} \end{cases}$

$$\max \sum_{i \in I} h_i \, z_i$$

$$\sum_{j \in J} a_{ij} \, x_j \geq z_i \quad \forall \; i \in I \tag{9}$$

$$\sum_{j \in J} x_j = p \tag{10}$$

$$x_j \in \{0, 1\} \quad \forall \; j \in J$$

$$z_i \in \{0, 1\} \quad \forall \; i \in I$$

(9): per coprire il nodo i dobbiamo aprire almeno un servizio in un posto che lo possa coprire.
(10): deve essere aperto il servizio in p luoghi.

Esempio

Una ASL ha a disposizione un budget sufficiente per posizionare 2 centri di prenotazione (CUP) nella cittá e sa che ci sono 5 siti che possono ospitare tali centri.

La cittá é divisa in 10 zone e i tempi (in minuti) per raggiungere le zone da ogni sito sono indicati nella tabella dell'esempio precedente.

La tabella seguente indica il numero di abitanti che vivono nelle 10 zone della cittá:

zone	1	2	3	4	5	6	7	8	9	10
abitanti	1500	3000	1000	1800	2500	2100	1900	1700	2200	2700

Il dirigente deve decidere dove posizionare i 2 centri in modo da massimizzare il numero di abitanti che possono raggiungere un CUP in al piú 10 minuti.

Metodo greedy per soluzione ammissibile

1. $I = \{1, \dots, m\}$, $J = \{1, \dots, n\}$, $x := 0$.
2. Per ogni $j \in J$ calcola la domanda u_j coperta da j:

$$u_j = \sum_{i \in I: d_{ij} \leq D} h_i$$

3. Trova l'indice $k \in J$ tale che $u_k = \max_{j \in J} u_j$

   Poni $x_k := 1$, da J elimina k, da I elimina $\{i : \; d_{ik} \leq D\}$
4. Se $\sum_{j=1}^{n} x_j = p$ allora STOP; altrimenti torna al passo 2.

Applichiamo il metodo greedy al problema precedente.

La domanda coperta da ogni sito é

siti	1	2	3	4	5
domanda coperta	8200	8000	14700	9500	8600

Scegliamo il sito 3 ($x_3 = 1$) e copriamo le zone 2,3,4,6,7,9,10. Per le zone rimanenti 1,5,8 la domanda coperta dai 4 siti rimanenti é:

siti	1	2	4	5
domanda coperta	1500	4000	2500	1700

Scegliamo il sito 2 ($x_2 = 1$) e copriamo le zone 1 e 5. In totale copriamo 18700 abitanti (la zona 8 con 1700 abitanti rimane scoperta).

Problemi di p-center

Supponiamo che la distanza (in minuti) di copertura non sia fissata a priori.

Problema

Dati:
$I =$ insieme di nodi domanda
$h_i =$ domanda associata al nodo i
$J =$ insieme di siti candidati ad ospitare un servizio
$p =$ numero prefissato di servizi da aprire
$d_{ij} =$ distanza tra nodo di domanda i e sito candidato j

Decidere in quali p siti aprire i servizi in modo che sia minima la distanza massima tra ogni nodo di domanda e il luogo piú vicino dove é aperto il servizio.

Modello

Variabili: $x_j = \begin{cases} 1 & \text{se apriamo il servizio nel sito } j \\ 0 & \text{altrimenti} \end{cases}$

$y_{ij} = \begin{cases} 1 & \text{se assegniamo il nodo } i \text{ al servizio nel sito } j \\ 0 & \text{altrimenti} \end{cases}, \qquad D \geq 0$

$$\min \; D$$

$$D \geq \sum_{j \in J} h_i \, d_{ij} \, y_{ij} \qquad \forall \; i \in I \tag{11}$$

$$\sum_{j \in J} x_j = p \tag{12}$$

$$\sum_{j \in J} y_{ij} = 1 \qquad \forall \; i \in I \tag{13}$$

$$y_{ij} \leq x_j \quad \forall \; i \in I, \forall \; j \in J \tag{14}$$

$$x_j, \; y_{ij} \in \{0, 1\} \quad \forall \; i \in I, \forall \; j \in J$$

(11): $D \geq$ massima distanza tra un nodo di domanda e la facility a cui é assegnato.
(12): devono essere aperti i servizi in p luoghi.
(13): ogni nodo di domanda deve essere assegnato ad un servizio.
(14): se i é assegnato al servizio nel sito j, allora quest'ultimo deve essere aperto.

Esercizio

L'assessore comunale allo sport ha a disposizione un budget sufficiente per costruire 2 piscine e sa che ci sono 5 siti possibili in cui costruirle. La cittá é divisa in 10 zone e i tempi (in minuti) per raggiungere i siti da ogni zona sono indicati nella tabella, mentre il numero di abitanti che vivono nelle 10 zone della cittá é indicato nell'altra tabella.

Il dirigente deve decidere dove costruire le 2 piscine in modo da minimizzare la massima distanza tra ogni zona della cittá e la piscina piú vicina.

Questo é un problema di p-center.

Schema generale di localizzazione

Insieme $I = \{1, \ldots, m\}$.
Famiglia $S_1, \ldots, S_n$ di sottoinsiemi di I, ognuno ha costo (valore) c_j.

Problema di copertura: determinare una sottofamiglia $\mathcal{F}$ di costo minimo tale che ogni elemento di I appartenga ad almeno un sottoinsieme di $\mathcal{F}$.

Problema di partizione: determinare una sottofamiglia $\mathcal{F}$ di costo minimo tale che ogni elemento di I appartenga esattamente ad un sottoinsieme di $\mathcal{F}$.

Problema di riempimento: determinare una sottofamiglia $\mathcal{F}$ di valore massimo tale che ogni elemento di I appartenga ad al piú un sottoinsieme di $\mathcal{F}$.

Un esempio:
$I = \{1, 2, 3, 4, 5, 6\}$,
$S_1 = \{1, 3\}$, $S_2 = \{2, 4\}$, $S_3 = \{2, 5, 6\}$, $S_4 = \{5, 6\}$, $S_5 = \{3, 4\}$.

copertura: $\mathcal{F} = \{S_1, S_2, S_3\}$
partizione: $\mathcal{F} = \{S_1, S_2, S_4\}$
riempimento: $\mathcal{F} = \{S_3, S_5\}$

Definiamo la matrice: $a_{ij} = \begin{cases} 1 & \text{se } i \in S_j, \\ 0 & \text{altrimenti.} \end{cases}$

Ad ogni sottofamiglia $\mathcal{F}$ associamo una variabile x, dove $x_j = \begin{cases} 1 & \text{se } S_j \in \mathcal{F}, \\ 0 & \text{altrimenti.} \end{cases}$

Copertura

$$\begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall\ i \\ x_j \in \{0,1\} \quad \forall\ j \end{cases}$$

Partizione

$$\begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j = 1 \quad \forall\ i \\ x_j \in \{0,1\} \quad \forall\ j \end{cases}$$

Riempimento

$$\begin{cases} \max \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad \forall\ i \\ x_j \in \{0,1\} \quad \forall\ j \end{cases}$$

Dislocazione mezzi soccorso

Insieme U di siti in cui sono presenti utenti.
Insieme M di siti dove é possibile dislocare ambulanze.
t_{ij} = tempo necessario utente in $i \in U$ sia raggiunto da ambulanza posta in $j \in M$.
c_j = costo di attivazione del sito $j \in M$, T = massima attesa per ogni utente.

Obiettivo: decidere in quali siti dislocare ambulanze in modo che ogni utente sia raggiunto in al piú T minuti, minimizzando il costo totale.

Puó essere formulato come problema di copertura:
sottoinsiemi $S_j = \{$utenti raggiunti entro T minuti da ambulanza posta in $j\}$.

Ricerca di informazioni

Bisogna soddisfare m richieste di dati.
n archivi a disposizione, ognuno contiene alcune delle informazioni richieste.
$c_j =$ costo per interrogare archivio j.

Obiettivo: selezionare un sottoinsieme di archivi capace di soddisfare tutte le richieste e che richieda il minimo costo totale.

Puó essere formulato come problema di copertura:
$I = \{\text{richieste}\}$, sottoinsiemi $S_j =$ archivi.

Pianificazione di equipaggi

Compagnia aerea ha m rotte (punto partenza, punto arrivo, durata volo).
n combinazioni possibili di rotte (*round trip*) che puó fare un equipaggio in un giorno.
$c_j =$ costo del *round trip* j.

Obiettivo: determinare un insieme di round trip con il minimo costo totale in modo che ogni rotta sia coperta da esattamente un *round trip*.

Puó essere formulato come problema di partizione:
$I = \{\text{rotte}\}$, sottoinsiemi $S_j =$ *round trip*.

Formazione di distretto elettorale

Un territorio formato da vari comuni deve essere diviso in un insieme di distretti elettorali.

Sono specificate n possibili suddivisioni che rispondono a determinati requisiti.

Se si sceglie di costruire il distretto j si paga un costo c_j.

Ogni comune deve essere incluso esattamente in un solo distretto elettorale.

La matrice A ha una riga per ciascun comune ed una colonna per ciascun distretto.

Puó essere formulato come problema di partizione:

$I = \{\text{comuni}\}$, sottoinsiemi $S_j = \{\text{insiemi dei possibili distretti }\}$.

Squadre di operai

Insieme di operai, insieme di squadre di operai.
La squadra j é in grado di svolgere un'attivitá che fornisce profitto p_j.
Le squadre devono lavorare contemporaneamente.

Obiettivo: formare le squadre in modo da massimizzare il profitto.

Puó essere formulato come problema di riempimento:
$I = \{\text{operai}\}$, sottoinsiemi $S_j =$ squadre di operai.

Esempio

Localizzazione di impianti ad elevato impatto ambientale

Insieme di cittá, gruppi di cittá inquinate dal sito j.

Obiettivo: Localizzare gli impianti massimizzando il profitto ma con il vincolo che ogni cittá subisca al piú un impianto inquinante.

Puó essere formulato come problema di riempimento:
$I = \{\text{cittá}\}$, sottoinsiemi $S_j =$ di cittá che subiscono disagi dall'impianto j.

Esempio

$$A = \begin{array}{|c|cccc|}
\hline
c_j & 7 & 6 & 4 & 2 \\
\hline
{}_i\backslash^j & 1 & 2 & 3 & 4 \\
\hline
1 & 1 & 1 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 \\
3 & 1 & 1 & 1 & 0 \\
4 & 0 & 0 & 0 & 1 \\
5 & 1 & 0 & 0 & 1 \\
6 & 1 & 1 & 0 & 0 \\
7 & 1 & 0 & 0 & 1 \\
8 & 1 & 0 & 0 & 0 \\
9 & 0 & 1 & 0 & 0 \\
\hline
\end{array}$$

La colonna 1 significa che posizionare l'impianto nel sito 1 si guadagna 7 e si creano disagi alle cittá $1, 3, 5, 6, 7, 8$

Ogni problema di riempimento puó essere trasformato in un problema di partizione cambiando segno ai costi e aggiungendo variabili di scarto:

$$\begin{cases} \max \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad \forall\ i \\ x_j \in \{0,1\} \quad \forall\ j \end{cases} \longrightarrow \begin{cases} \min\ -\sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j + s_i = 1 \quad \forall\ i \\ x_j \in \{0,1\} \quad \forall\ j \\ s_i \in \{0,1\} \quad \forall\ i \end{cases}$$

ossia aggiungendo i sottoinsiemi $S_i = \{i\}$ di costo nullo, per ogni $i = 1, \ldots, m$.

Teorema

Ogni problema di partizione puó essere trasformato in un problema di copertura.

Siano $\bar{c}_j = c_j + M \sum_{i=1}^{m} a_{ij}$, dove $M > \sum_{j: c_j > 0} c_j - \sum_{j: c_j < 0} c_j$.

Allora ogni soluzione ottima del problema di copertura

$$\begin{cases} \min \ \sum_{j=1}^{n} \bar{c}_j \, x_j \\ \sum_{j=1}^{n} a_{ij} \, x_j \geq 1 \quad \forall \ i \\ x_j \in \{0, 1\} \quad \forall \ j \end{cases}$$

é una soluzione ottima del problema di partizione

$$\begin{cases} \min \ \sum_{j=1}^{n} c_j \, x_j \\ \sum_{j=1}^{n} a_{ij} \, x_j = 1 \quad \forall \ i \\ x_j \in \{0, 1\} \quad \forall \ j \end{cases}$$

Riduzione di un problema di copertura

Si possono eliminare variabili e vincoli applicando le seguenti regole:

1. Se $c_j \leq 0$ allora pongo $x_j = 1$ ed elimino tutti i vincoli in cui compare x_j (perché esiste una soluzione ottima con $x_j = 1$).
2. Se esiste un vincolo r tale che $a_{rj} = \begin{cases} 1 & \text{se } j = p, \\ 0 & \text{altrimenti,} \end{cases}$ allora pongo $x_p = 1$ ed elimino tutti i vincoli in cui compare x_p (perché ogni sol. ammissibile ha $x_p = 1$).
3. Se esistono 2 vincoli r, s tali che $a_{rj} \leq a_{sj}$ per ogni j, allora elimino il vincolo s (perché il vincolo r implica il vincolo s).
4. Se esiste un sottoinsieme S_p e una famiglia F tali che

$$\begin{aligned} a_{ip} &\leq \sum_{j \in F} a_{ij} \qquad \text{per ogni } i \in I \\ c_p &\geq \sum_{j \in F} c_j \end{aligned}$$

allora pongo $x_p = 0$ (perché gli elementi di S_p sono coperti anche dalla famiglia F ed il costo di S_p non é inferiore a quello di F).

Riduzione di un problema di copertura: esempio

Consideriamo il problema di copertura:

$$
A = \begin{array}{|c|cccccccc|}
\hline
c_j & 10 & 6 & 4 & 2 & 8 & 5 & 3 & 7 \\
\hline
{}_i\backslash^j & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
3 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
5 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
6 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
7 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
8 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
9 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
\hline
\end{array}
$$

Dalla riga 4 si ottiene $x_5 = 1$ e si eliminano le righe 2, 4 e 9.

Riduzione di un problema di copertura: esempio (segue)

Il problema ridotto diventa:

$A =$

c_j	10	6	4	2	5	3	7
$_i\backslash^j$	1	2	3	4	6	7	8
1	1	1	0	0	1	1	0
3	1	1	1	0	0	0	1
5	1	0	0	1	0	1	0
6	1	1	0	0	1	1	1
7	1	0	0	1	1	0	0
8	1	0	0	0	1	1	1

Poiché $a_{1j} \leq a_{6j}$ per ogni j, si può eliminare la riga 6.

$A =$

c_j	10	6	4	2	5	3	7
$_i\backslash^j$	1	2	3	4	6	7	8
1	1	1	0	0	1	1	0
3	1	1	1	0	0	0	1
5	1	0	0	1	0	1	0
7	1	0	0	1	1	0	0
8	1	0	0	0	1	1	1

La colonna 1 é dominata dalle colonne 3, 4 e 7, quindi si pone $x_1 = 0$.

$A =$

c_j	6	4	2	5	3	7
$_i\backslash^j$	2	3	4	6	7	8
1	1	0	0	1	1	0
3	1	1	0	0	0	1
5	0	0	1	0	1	0
7	0	0	1	1	0	0
8	0	0	0	1	1	1

Metodi per problemi di copertura

D'ora in poi consideriamo un problema di copertura:

$$\begin{cases} \min \ \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall \ i \\ x_j \in \{0,1\} \quad \forall \ j \end{cases}$$

Per applicare un metodo risolutivo é necessario:

- calcolare le $v_I(P_i)$
- calcolare una $v_S(P)$

Metodi euristici: algoritmo di Chvátal

$v_S(P)$: un algoritmo *greedy*.

Algoritmo di Chvátal

1. $I = \{1, \ldots, m\}$, $J = \{1, \ldots, n\}$, $x := 0$.
2. Per ogni $j \in J$ calcola i costi unitari di copertura
$$u_j = \frac{c_j}{\sum_{i \in I} a_{ij}}$$
3. Trova un indice $k \in J$ tale che $u_k = \min_{j \in J} u_j$
   Poni $x_k := 1$, da J elimina k, da I elimina $\{i : \ a_{ik} = 1\}$
4. Se $I = \emptyset$ allora STOP (x é una copertura)
5. Se $J = \emptyset$ allora STOP (non esistono coperture)
   altrimenti torna al passo 2.

Esempio

$I = \{1, \ldots, 9\}$, $J = \{1, \ldots, 8\}$.

	c_j	10	6	4	2	8	5	3	7
	$i \backslash j$	1	2	3	4	5	6	7	8
A	1	1	1	0	0	0	1	1	0
	2	0	0	1	0	1	0	0	1
	3	1	1	1	0	0	0	0	1
	4	0	1	0	0	1	0	0	0
	5	1	0	0	1	0	0	1	0
	6	1	1	0	0	0	1	1	1
	7	1	0	0	1	0	1	0	0
	8	1	0	0	0	0	1	1	1
	9	0	1	0	0	1	0	1	0

Costi unitari:

j	1	2	3	4	5	6	7	8
u_j	$\frac{10}{6}$	$\frac{6}{5}$	2	1	$\frac{8}{3}$	$\frac{5}{4}$	$\frac{3}{5}$	$\frac{7}{4}$

$u_7 = \min u_j \rightarrow x_7 = 1$

Elimino la colonna 7 e le righe 1,5,6,8,9:

c_j	10	6	4	2	8	5	7
$_i\backslash^j$	1	2	3	4	5	6	8
2	0	0	1	0	1	0	1
3	1	1	1	0	0	0	1
4	0	1	0	0	1	0	0
7	1	0	0	1	0	1	0

Costi unitari:

j	1	2	3	4	5	6	8
u_j	5	3	2	2	4	5	$7/2$

$u_3 = \min u_j \to x_3 = 1$.

In seguito si trova $x_4 = 1$ e $x_2 = 1$. L'algoritmo di Chvátal trova la copertura $(0, 1, 1, 1, 0, 0, 1, 0)$ di costo 15.

Metodi euristici: arrotondamento

Un altro metodo euristico é risolvere il rilassamento continuo e arrotondare per eccesso le componenti frazionarie.

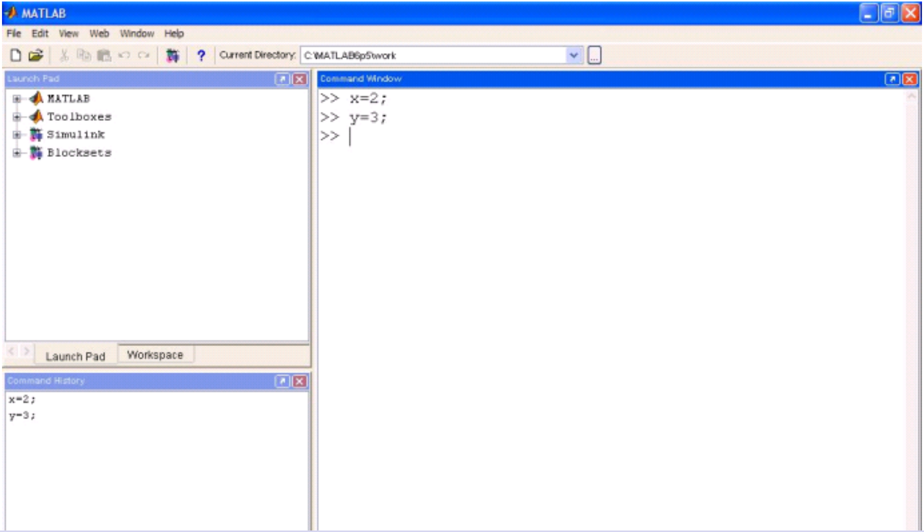
Esempio

	c_j	10	6	4	2	8	5	3	7
	$_i\backslash^j$	1	2	3	4	5	6	7	8
	1	1	1	0	0	0	1	1	0
	2	0	0	1	0	1	0	0	1
	3	1	1	1	0	0	0	0	1
	4	0	1	0	0	1	0	0	0
A	5	1	0	0	1	0	0	1	0
	6	1	1	0	0	0	1	1	1
	7	1	0	0	1	0	1	0	0
	8	1	0	0	0	0	1	1	1
	9	0	1	0	0	1	0	1	0

L'ottimo del rilassamento continuo é $\left(0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0\right)$,quindi $(0, 1, 1, 1, 1, 1, 1, 0)$ é una copertura di costo 28.

Introduzione all'Optimization Toolbox di MATLAB

Per avviare MATLAB é sufficiente fare doppio clic sull'icona MATLAB



Barra dei menu

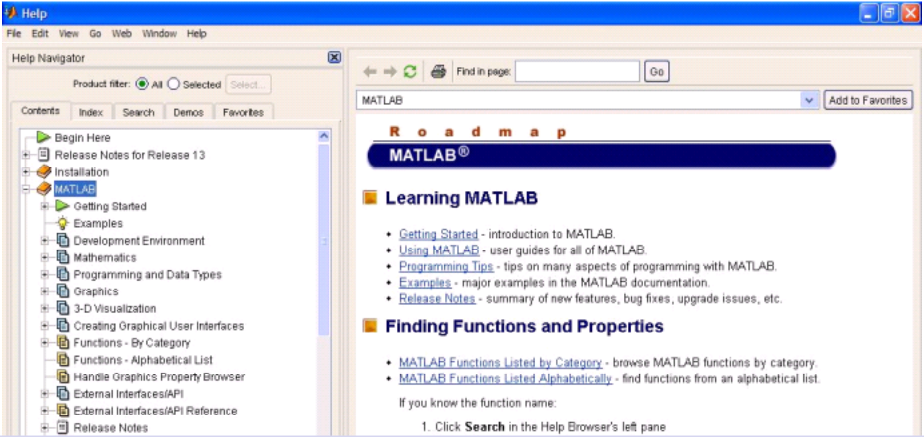
In essa si possono distinguere vari parti:

- la *barra dei menu* É la riga in alto contenente e comprende i nomi di 6 menu, ognuno contenente i comandi per fornire istruzioni al programma;
  File Edit View Web Window Help
- la *barra degli strumenti* rappresenta operazioni di MATLAB. Fare clic equivale ad aprire il menu e selezionare la corrispondente opzione. Le prime sette icone da sinistra corrispondono alle opzioni *New File, Open File, Cut, Copy, Paste, Undo e Redo*. L'ottava icona avvia Simulink; la nona icona (quella con il punto interrogativo) permette di accedere alla guida di MATLAB. Il riquadro a destra della barra degli strumenti indica la cartella di lavoro corrente (*Current Directory*).
- *Command Window*(la finestra dei comandi) in cui si digitano i nomi dei comandi o le istruzioni da eseguire.

Barre

- *Command History* (la cronologia dei comandi) visualizza tutti i comandi che sono stati digitati durante la sessione di lavoro corrente.
- *Launch Pad*(strumenti di MATLAB) contiene un grafo ad albero i cui nodi rappresentano tutte le cartelle e i file di MATLAB. Se vicino ad un nodo c'é un $+$, significa che esso contiene cartelle e file che non sono visualizzati. Se si fa clic sul $+$ vengono visualizzate le cartelle ed i file contenuti, e il $+$ viene trasformato in - (facendo clic sul meno si torna alla situazione precedente).
- *Workspace* mostra i nomi e i valori di tutte le variabili utilizzate nella sessione di lavoro corrente.

Per avere la documentazione di MATLAB c'é la guida interna.
Per accedere alla guida é sufficiente fare clic sull'icona con il punto interrogativo, o selezionare l'opzione *Help* dal menu *View*.
Sullo schermo apparirá il browser illustrato nella figura.



Quando nella finestra dei comandi compare il prompt di MATLAB ($>>$), il programma é pronto a ricevere nuove istruzioni.
Se il cursore non si trova dopo il prompt, si puó utilizzare il mouse per spostarlo.
Quando invece sono in corso operazioni il prompt scompare.
Per annullare una operazione, premere contemporaneamente i tasti `Ctrl` e `c`.
Digitando `1/700` dopo il prompt e premendo *Invio* si ottiene

```
ans =
    0.0014
```

MATLAB assegna la risposta ad una variabile temporanea chiamata `ans`.
Per default il risultato viene visualizzato con 4 cifre decimali.
Il comando `format` permette di modificare il formato di uscita secondo la seguente tabella:

Comando	**Descrizione**
`format short`	4 cifre decimali
`format long`	14 cifre decimali
`format short e`	4 cifre decimali piú l'esponente
`format long e`	15 cifre decimali piú l'esponente
`format rat`	approssimazione razionale

Table: Formati numerici

Formati

Se digitiamo

```
>> format long
>> 1/700
```

otteniamo

```
ans =
    0.00142857142857
```

In modo analogo

```
>> format short e
>> 1/700
```

restituisce

```
ans =
    1.4286e-003
```

```
>> format long e
>> 1/700
ans =
    1.428571428571429e-003
```

Mentre,

```
>> format rat
>> 1/700
```

visualizza

```
ans =
    1/700
```

Esempio

MATLAB puó essere usato come calcolatrice come descritto nella seguente tabella:

Simbolo	Operazione	Formato di MATLAB
ˆ	elevazione a potenza	âb
*	Moltiplicazione	a*b
/	Divisione	a/b
+	Addizione	a+b
-	Sottrazione	a-b

Table: Operazione aritmetiche

Esempio

Se si commette un errore durante la digitazione , si ritorna ai comandi digitati in precedenza con la freccia in alto ($\uparrow$) e si utilizza la freccia in basso ($\downarrow$) per far scorrere al contrario la lista dei comandi.

Per spostare il cursore a sinistra o a destra all'interno della riga corrente é sufficiente premere i tasti ($\leftarrow$) o ($\rightarrow$)

Quando si trova il comando in cui si é commesso un errore, si modifica utilizzando il tasto *Canc* per cancellare il carattere che si trova davanti al cursore o quello *Backspace* per cancellare il carattere che si trova dietro il cursore.

Il punto e virgola posto alla fine di un comando indica a MATLAB di non visualizzare i risultati dell'istruzione sullo schermo.

Variabili e Costanti

I programmi in genere registrano dati in memoria.

Le zone di memoria in cui vengono registrati i dati si chiamano *variabili*.
Per assegnare il valore ad una variabile MATLAB usa il segno uguale (=).

Per esempio il comando $x = 2$, permette di registrare nella variabile x il valore 2.

Per registrare nella variabile x il risultato dell'addizione tra 3 e il contenuto della variabile y si scrive:

```
>> x=y+3
```

Esempio

Il comando $x = 5$ é diverso dal comando $5 = x$ che genera un messaggio di errore.

Il nome di una variabile deve iniziare con una lettera, che puó essere seguita da una qualunque combinazione di lettere e cifre, ma non piú lungo di 32 caratteri.

MATLAB distingue tra caratteri maiuscoli e minuscoli: `A` e `a` identificano variabili diverse.

Per conoscere il valore corrente di una variabile é sufficiente digitare il suo nome e premere *Invio*.

Esempio

Per sapere se la variabile x é giá stata definita, basta digitare

```
>>exist('x')
```

Se MATLAB restituisce il valore 1, la variabile esiste; se restituisce il valore 0, la variabile non esiste.
I nomi ed i valori di tutte le variabili utilizzate si trovano nel *Workspace*.
In alternativa,il comando

```
>> who
```

elenca i nomi di tutte le variabili, ma non indica i loro valori.

MATLAB conserva l'ultimo valore di una variabile finché la sessione di lavoro corrente é aperta o finché la variabile non é eliminata espressamente.

Esempio

Il comando

$$>> \texttt{clear}$$

rimuove tutte la variabili dall'area di lavoro. Il comando

$$>> \texttt{clear var1 var2 ...}$$

consente di eliminare le variabili var1 var2 . . ..
Prima di uscire da MATLAB é possibile salvare la sessione di lavoro con il comando

$$>> \texttt{save}$$

Questo comando salva tutte le variabili nel file matlab.mat.

Esempio

Se si desidera salvare la sessione di lavoro con un nome diverso si digita

$$>> \texttt{save nomefile}$$

Tutte le variabili saranno salvate nel file nomefile.mat. Il comando

$$>> \texttt{load nomefile}$$

ricarica tutte le variabili memorizzate in nomefile.mat mantenendo inalterato il nome con cui erano state memorizzate.
Il comando

$$>> \texttt{clc}$$

cancella il contenuto della finestra dei comandi, lasciando inalterate le variabili.

Vettori e matrici

Per creare un vettore riga basta digitare gli elementi all'interno di una coppia di parentesi quadre separandoli con uno spazio o una virgola.

Ad esempio, il comando per creare il vettore $a = (2, 4, 10)$ é:

$>>$ `a=[2 4 10]`

oppure

$>>$ `a=[2, 4, 10]`

Per creare un vettore colonna si possono digitare gli elementi separati dal punto e virgola.

Ad esempio, il vettore $b = \begin{pmatrix} 1 \\ 5 \\ 7 \end{pmatrix}$ é definito dal comando:

$>>$ `b=[1; 5; 7]`

E' possibile creare un vettore riga o colonna “accodando” due vettori:

```
>> x=[2, 4, 20];
>> y=[9, 6, 3];
```

allora `z=[x, y]` da come risultato:

```
z=
    2     4     20     9     6     3
```

Per selezionare le componenti di indici $a, b, c, \dots$ del vettore x si scrive `x([a, b, c, ...])`, ad esempio:

```
>> x=[4, 6, 1, 9, 3];
>> z=x([1, 2, 4])
z=
    4     6     9
```

Per generare un vettore x con elementi intervallati regolarmente da a a b con incremento pari a q si scrive `x=[a:q:b]`:

```
>> x=[0:2:8]
x=
     0     2     4     6     8
```

Se viene omesso l'incremento q, allora MATLAB lo pone uguale a 1:

```
>> x=4:6
x=
     4     5     6
```

Sono permessi anche incrementi negativi, ad esempio:

```
>> x=7:-2:3
x=
     7     5     3
```

Esempio

Per avere un vettore x con m componenti intervallate regolarmente da a a b si usa il comando `x=linspace(a,b,m)`, ad esempio:

```
>> x=linspace(1,7,4)
x=
    1     3     5     7
```

Il comando `length(x)` fornisce il numero di componenti del vettore x.
La somma e la sottrazione di due vettori riga (o colonna) della stessa lunghezza si effettuano con il + ed il −, ad esempio:

```
>> x=[3, 4, 8, 1];
>> y=[-2, 9, 1, 0];
>> x+y
ans=
    1     13     9     1
>> x-y
ans=
    5     -5     7     1
```

Prodotto

Il prodotto tra uno scalare ed un vettore si effettua con il $*$:

```
>> x=[3, -4, 8, 1];
>> 2*x
ans=
    6      -8      16      2
```

Anche il prodotto scalare tra due vettori si effettua con il $*$:

```
>> x=[3, 4, 8, 1];
>> y=[-2; 9; 1; 0];
>> x*y
ans=
    38
```

Per creare una matrice si inseriscono gli elementi per riga separati da spazi o virgole e per passare alla riga successiva si usa il punto e virgola.

Per inserire la matrice $A = \begin{pmatrix} 1 & 2 & 3 & 6 \\ 3 & -5 & -8 & 12 \\ 2 & 0 & 1 & 9 \end{pmatrix}$, si digita:

```
>> A=[1, 2, 3, 6; 3, -5, -8, 12; 2, 0, 1, 9]
```

Per creare una matrice ottenuta dalla matrice A aggiungendo il vettore colonna $b = \begin{pmatrix} 3 \\ -5 \\ 0 \end{pmatrix}$ si scrive `[A, b]`, mentre per aggiungere la riga $c = (6, 5, 8, 3)$ si digita `[A;c]`.

Il comando `eye(n)` genera la matrice identitá di ordine n:

```
>> eye(3)
ans=
    1    0    0
    0    1    0
    0    0    1
```

Il comando `ones(m,n)` genera una matrice di ordine $m \times n$ i cui elementi sono tutti uguali a 1:

```
>> ones(3,2)
ans=
    1    1
    1    1
    1    1
```

`zeros(m,n)` genera una matrice nulla $m \times n$:

```
>> zeros(3,4)
ans=
    0     0     0     0
    0     0     0     0
    0     0     0     0
```

Il comando `diag` ha due modi di funzionamento: se l'argomento é una matrice quadrata, fornisce gli elementi sulla diagonale; se l'argomento é un vettore, genera una matrice diagonale i cui elementi diagonali sono gli elementi del vettore.

Ad esempio:

```
>> diag([6, 4, 9; 5, 8, 2; 7, 5, 1])
ans=
    6
    8
    1
```

e

```
>> diag([3, 5, 1])
ans=
    3    0    0
    0    5    0
    0    0    1
```

Per sapere la dimensione di una matrice si usa il comando `size`:

```
>> A=[3, 5, 1; 6, 2, 8];
>> size(A)
ans=
    2    3
```

Matrici

La somma e la differenze tra matrici si effettuano rispettivamente con il segno + ed il segno − come tra due vettori:

```
>> A=[-7, 16; 4, 9];
>> B=[6, -5; 12, -2];
>> A+B
ans=
     -1   11
     16    7
```

Per fare il prodotto tra uno scalare ed una matrice basta usare il segno $*$, ad esempio:

```
>> A=[-7, 16; 4, 9];
>> 2*A
ans=
     -14   32
       8   18
```

Il prodotto riga per colonna tra una matrice A di ordine $m \times n$ e d una matrice B di ordine $n \times p$ si effettua anch'esso con il segno $*$, ad esempio:

```
>> A=[-7, 16; 4, 9];
>> B=[6, -5; 12, -2];
>> A*B
ans=
     150     3
     132   -38
```

Operatori relazionali

MATLAB dispone di 6 operatori relazionali che consentono di confrontare variabili e vettori:

Simbolo	Descrizione
==	uguale a
~=	diverso da
<	minore di
<=	minore o uguale a
>	maggiore di
>=	maggiore o uguale a

Table: Operatori relazionali

Esempio

Il risultato di un confronto puó essere 0 (falso) oppure 1 (vero).
Tale risultato puó essere assegnato ad una variabile. Ad esempio, se $x = 2$ e $y = 3$, allora il comando `z=x<y` assegna alla variabile z il risultato del confronto $x < y$. In questo caso si ottiene:

```
z=1
```

Gli operatori relazionali permettono anche di confrontare, elemento per elemento, due vettori aventi lo stesso numero di componenti. Ad esempio, supponiamo che:

```
>> x=[6, 3, 9, 11];
>> y=[14, 2, 9, 13];
```

il comando `z=x<y` da come risultato

```
z=
    1    0    0    1
```

mentre il comando `z=x<=y` da come risultato

```
z=
    1    0    1    1
```

Nell'*Optimization toolbox* di MATLAB, la funzione `linprog` risolve un problema di PL della forma:

$$\begin{cases} \min \; c^{\mathsf{T}} x \\ A\,x \leq b \\ D\,x = e \\ l \leq x \leq u \end{cases} \tag{15}$$

dove c, x, b, e, l, u sono vettori e A, D sono matrici. Se, ad esempio, non ci sono vincoli di uguaglianza, si pongono `D=[ ]` ed `e=[ ]`.

Esempio

La sintassi della funzione é la seguente:

```
[x, v] = linprog(c, A, b, D, e, l, u)
```

dove gli input

```
c, A, b, D, e, l, u
```

definiscono il problema da risolvere, mentre gli output sono:

- `x` É una soluzione ottima del problema (15);
- `v` É il valore ottimo del problema (15).

Esempio

Supponiamo di dover risolvere il problema di PL:

$$\begin{cases} \max\ x_1 + 2\,x_2 + 3\,x_3 \\ 3\,x_1 + 4\,x_3 \leq 5 \\ 5\,x_1 + x_2 + 6\,x_3 = 7 \\ 8\,x_1 + 9\,x_3 \geq 2 \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 4 \\ x_3 \geq 0 \end{cases} \tag{16}$$

Lo trasformiamo nella forma (15):

$$\begin{cases} -\min\ -x_1 - 2\,x_2 - 3\,x_3 \\ 3\,x_1 + 4\,x_3 \leq 5 \\ -8\,x_1 - 9\,x_3 \leq -2 \\ 5\,x_1 + x_2 + 6\,x_3 = 7 \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 4 \\ 0 \leq x_3 \leq +\infty \end{cases}$$

Esempio

e scriviamo:

```
>> c = [-1; -2; -3];
>> A = [3, 0, 4; -8, 0, -9];
>> b = [5; -2];
>> D = [5, 1, 6];
>> e = 7;
>> l = [0;0;0];
>> u = [5;4;inf];

>> [x, v] = linprog(c, A, b, D, e, l, u)
```

si ottengono:

```
x=
    0.0000
    4.0000
    0.5000

v=
    -9.5000
```

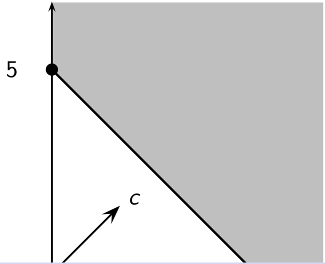
Esempio

Per risolvere problemi di PL la funzione `linprog` utilizza, di *default*, un metodo a punti interni invece del simplesso.

Quindi potrebbe non fornire come soluzione ottima un vertice. Ad esempio il problema

$$\begin{cases} \min \; x_1 + x_2 \\ x_1 + x_2 \geq 5 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases} \tag{17}$$

ammette infinite soluzioni ottime (il segmento di estremi $(0,5)$ e $(5,0)$).



Esempio

Se poniamo

```
>> c = [1; 1];
>> A = [-1, -1];
>> b = -5;
>> l = [0;0];
```

il comando

```
>> x = linprog(c, A, b, [ ], [ ], l, [ ])
```

fornisce la soluzione ottima

```
x=
    2.5000
    2.5000
```

che non é un vertice del poliedro del problema (17).

Per risolvere il problema con l'algoritmo del simplesso é necessario cambiare le opzioni della funzione `linprog` con il seguente comando:

```
>> options = optimoptions('linprog','Algorithm','dual-simplex');
```

Ora il comando

```
>> x = linprog(c, A, b, [ ], [ ], l, [ ], [ ], options)
```

fornisce la soluzione ottima

```
x=
    5
    0
```

che é un vertice del problema (17).

Una fabbrica di detersivi produce due tipi di saponi che passano attraverso 4 fasi di lavorazione: le ore necessarie per ogni fase di lavorazione per quintale di prodotto sono riportate nella tabella che segue, in cui compaiono anche le ore mensili a disposizione per ciascuna fase.

	Fase a	Fase b	Fase c	Fase d
Sapone A	1.5	1.5	3	2.5
Sapone B	2.5	2	3	4
Ore mensili disponibili	155	200	240	400

Il guadagno netto é di 2100 euro per quintale di sapone A e 3400 euro per quintale di sapone B. Quanti quintali dei saponi A e B bisogna produrre per massimizzare il guadagno?

Esempio

Se indichiamo con x_1 il numero di quintali prodotti del sapone A e con x_2 quelli del sapone B, il problema si puó formulare come segue:

$$\begin{cases} \max\ 2100\ x_1 + 3400\ x_2 \\ 1.5\ x_1 + 2.5\ x_2 \leq 155 \\ 1.5\ x_1 + 2\ x_2 \leq 200 \\ 3\ x_1 + 3\ x_2 \leq 240 \\ 2.5\ x_1 + 4\ x_2 \leq 400 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}$$

Per risolverlo con la funzione `linprog` dobbiamo trasformarlo:

$$\begin{cases} -\min\ -2100\ x_1 - 3400\ x_2 \\ 1.5\ x_1 + 2.5\ x_2 \leq 155 \\ 1.5\ x_1 + 2\ x_2 \leq 200 \\ 3\ x_1 + 3\ x_2 \leq 240 \\ 2.5\ x_1 + 4\ x_2 \leq 400 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}$$

Esempio

COMANDI DI MATLAB

funzione obiettivo	`c=[-2100;-3400]`

vincoli	`A=[1.5, 2.5; 1.5, 2; 3, 3; 2.5, 4]`
	`b=[155; 200; 240; 400]`
	`lb=[0; 0]`

Comando risolutivo	`[x,fval]=linprog(c,A,b,[ ],[ ],lb,[ ])`

SOLUZIONI

Soluzione ottima	(45, 35)
Valore ottimo	213500

ANALISI DI SENSIBILITÀ

Supponendo di portare a 168 le ore mensili a disposizione per la fase a, determinare la nuova strategia di produzione ottima, il nuovo guadagno ed il costo per ogni ora lavorativa aggiuntiva per la fase a affinché la nuova strategia sia conveniente rispetto a quella vecchia.

Nuova soluzione ottima	(32,48)
Nuovo valore ottimo	230400
Costo per ogni ora aggiuntiva	< 1300

Un'azienda deve produrre almeno 500 litri di un cocktail utilizzando tre tipi di succhi di frutta S_1, S_2 e S_3. La disponibilitá ed il costo dei diversi tipi di succhi di frutta sono indicati nella seguente tabella:

Tipo di succo	Disponibilitá max in litri	Costo in euro per litro
S_1	560	1.5
S_2	260	1
S_3	950	4

La dose di miscelatura per il cocktail é: non piú del 25 % di S_1, non meno del 50 % di S_2 e non meno del 40 % di S_3.

Si vuole determinare la combinazione dei tre tipi di succhi di frutta che minimizzi la spesa.

Esempio

Indicando con x_1 il numero di litri del succo S_1, con x_2 quelli di S_2 e con x_3 quelli di S_3, si ha il seguente modello:

$$\begin{cases} \min\ 1.5\ x_1 + x_2 + 4\ x_3 \\ x_1 \leq 0.25(x_1 + x_2 + x_3) \\ x_2 \geq 0.5(x_1 + x_2 + x_3) \\ x_3 \geq 0.4(x_1 + x_2 + x_3) \\ x_1 + x_2 + x_3 \geq 500 \\ 0 \leq x_1 \leq 560 \\ 0 \leq x_2 \leq 260 \\ 0 \leq x_3 \leq 950 \end{cases}$$

Esempio

COMANDI DI MATLAB

funzione obiettivo	`c=[1.5; 1; 4]`

vincoli	`A=[3, -1, -1; 1, -1, 1; 2, 2, -3; -1, -1, -1]`
	`b=[0; 0; 0; -500]`
	`lb=[0; 0; 0]`
	`ub=[560; 260; 950]`

Comando risolutivo	`[x,fval]=linprog(c,A,b,[ ],[ ],lb,ub)`

SOLUZIONI

Soluzione ottima	(40, 260, 200)
Valore ottimo	1120

ANALISI DI SENSIBILITÀ

Supponendo che la massima disponibilitá del secondo succo di frutta aumenti di 40 litri, diventando quindi di 300 litri, determinare la nuova soluzione ottima e la nuova spesa minima.

Nuova soluzione ottima	(0,300,200)
Nuova spesa minima	1100

Esempio

Un'impresa produce un bene in 2 stabilimenti, situati a Pontedera e a Rosignano. La produzione viene immagazzinata in 2 depositi a Pisa e a Livorno e poi distribuita alla rete di vendita al dettaglio.
I dati riguardano il costo unitario di trasporto, la capacitá produttiva massima settimanale dei 2 stabilimenti e le statistiche di vendita settimanale di ognuno dei 2 depositi.

	Pisa	Livorno	Capacitá produttiva massima
Pontedera	10	30	105
Rosignano	35	6	80
Vendita	110	46	

Esempio

Indichiamo con x_1 la quantitá di merce spedita da Pontedera a Pisa, con x_2 quella spedita da Pontedera a Livorno, con x_3 quella da Rosignano a Pisa e con x_4 quella da Rosignano a Livorno. Il modello é:

$$\begin{cases} \min\ 10\ x_1 + 30\ x_2 + 35\ x_3 + 6\ x_4 \\ x_1 + x_2 \leq 105 \\ x_3 + x_4 \leq 80 \\ x_1 + x_3 = 110 \\ x_2 + x_4 = 46 \\ x \geq 0 \end{cases}$$

Esempio

COMANDI DI MATLAB

funzione obiettivo	`c=[10; 30; 35; 6]`

vincoli	`A=[1, 1, 0, 0; 0, 0, 1, 1]`
	`b=[105; 80]`
	`Aeq=[1, 0, 1, 0; 0, 1, 0, 1]`
	`beq=[110; 46]`
	`lb=[0; 0; 0; 0]`

Comando risolutivo	`[x,fval]=linprog(c,A,b,Aeq,beq,lb,[ ])`

SOLUZIONI

Soluzione ottima	(105, 0, 5, 46)
Valore ottimo	1501

ANALISI DI SENSIBILITÀ

Supponendo che la capacitá produttiva dello stabilimento di Pontedera diventi 110, determinare la nuova soluzione ottima e la nuova spesa minima.

Nuova soluzione ottima	(110, 0, 0, 46)
Nuova spesa minima	1376

Esempio

Un'industria siderurgica ha tre stabilimenti che necessitano di 50, 70 e 60 tonnellate di acciaio a settimana.
L'acciaio puó essere acquistato da due fornitori. Il primo puó fornire al massimo 30 tonnellate a settimana a ciascun stabilimento, mentre il secondo puó fornire al massimo 40 tonnellate a settimana a ciascun stabilimento.
Il primo fornitore non puó fornire piú di 100 tonnellate a settimana e deve fornire non meno di 25 tonnellate a settimana al terzo stabilimento. La seguente tabella indica i costi unitari di trasporto (euro/ton) dai fornitori agli stabilimenti.

Fornitori	Stabilimenti		
	1	2	3
1	2	3	5
2	3	3.6	3.2

Determinare come si deve rifornire l'industria per minimizzare il costo di trasporto.

Esempio

Indichiamo con x_1, x_2, x_3 le tonnellate di acciaio spedite rispettivamente dal primo fornitore ai tre stabilimenti e con x_4, x_5, x_6 le tonnellate di acciaio spedite rispettivamente dal secondo fornitore ai tre stabilimenti. Il modello é:

$$\begin{cases} \min\ 2\ x_1 + 3\ x_2 + 5\ x_3 + 3\ x_4 + 3.6\ x_5 + 3.2\ x_6 \\ x_1 + x_4 = 50 \\ x_2 + x_5 = 70 \\ x_3 + x_6 = 60 \\ x_1 + x_2 + x_3 \leq 100 \\ 0 \leq x_1 \leq 30 \\ 0 \leq x_2 \leq 30 \\ 25 \leq x_3 \leq 30 \\ 0 \leq x_4 \leq 40 \\ 0 \leq x_5 \leq 40 \\ 0 \leq x_6 \leq 40 \end{cases}$$

COMANDI DI MATLAB

funzione obiettivo	c=[2; 3; 5; 3; 3.6; 3.2]
vincoli	A=[1, 1, 1, 0, 0, 0] b=100 Aeq=[1, 0, 0, 1, 0, 0; 0, 1, 0, 0, 1, 0; 0, 0, 1, 0, beq=[50; 70; 60] lb=[0; 0; 25; 0; 0; 0] ub=[30; 30; 30; 40; 40; 40]
Comando risolutivo	[x,fval]=linprog(c,A,b,Aeq,beq,lb,ub)

SOLUZIONI

Soluzione ottima	(30, 30, 25, 20, 40, 35)
Valore ottimo	591

ANALISI DI SENSIBILITÀ

Supponendo che il primo fornitore possa spedire al massimo 35 tonnellate a settimana ai tre stabilimenti, determinare la nuova soluzione ottima e la nuova spesa minima.

Nuova soluzione ottima	(35, 35, 25, 15, 35, 35)
Nuova spesa minima	583

Esempio

Uno stabilimento produce tre diversi tipi di pitture per l'edilizia: una economica, una normale ed una di extra qualitá. Ogni pittura viene lavorata da tre linee di produzione A, B e C. I tempi (in minuti) necessari alla lavorazione di ogni quintale, la disponibilitá delle linee di produzione ed i profitti dei tre tipi di pittura sono indicate in tabella:

	Economica	Normale	Extra	Disponibilitá
A	20	30	62	480
B	31	42	51	480
C	16	81	10	300
Profitto	100	150	220	

La quantitá della pittura extra deve essere non piú del 20% del totale, mentre quella economica deve essere non meno del 40% del totale. Determinare le quantitá dei tre diversi tipi di pittura in modo da massimizzare il profitto.

Esempio

Sia con x_1 la quantitá prodotta di pittura economica, con x_2 quella di pittura normale e con x_3 quella di pittura extra. Il modello di PL é il seguente:

$$\begin{cases} \max\ 100\ x_1 + 150\ x_2 + 220\ x_3 \\ 20\ x_1 + 30\ x_2 + 62\ x_3 \leq 480 \\ 31\ x_1 + 42\ x_2 + 51\ x_3 \leq 480 \\ 16\ x_1 + 81\ x_2 + 10\ x_3 \leq 300 \\ x_3 \leq 0.2(x_1 + x_2 + x_3) \\ x_1 \geq 0.4(x_1 + x_2 + x_3) \\ x \geq 0 \end{cases}$$

che equivale a:

$$\begin{cases} -\min\ -100\ x_1 - 150\ x_2 - 220\ x_3 \\ 20\ x_1 + 30\ x_2 + 62\ x_3 \leq 480 \\ 31\ x_1 + 42\ x_2 + 51\ x_3 \leq 480 \\ 16\ x_1 + 81\ x_2 + 10\ x_3 \leq 300 \\ -0.2\ x_1 - 0.2\ x_2 + 0.8\ x_3 \leq 0 \\ -0.6\ x_1 + 0.4\ x_2 + 0.4\ x_3 \leq 0 \\ x \geq 0 \end{cases}$$

COMANDI DI MATLAB

funzione obiettivo	`c=[-100; -150; -220]`

vincoli	`A=[20,30,62;31,42,51;16,81,10; -0.2,-0.2,0.8;-0.6,0.4,0.4]`
	`b=[480; 480; 300; 0; 0]`
	`lb=[0; 0; 0]`

Comando risolutivo	`[x,fval]=linprog(c,A,b,[ ],[ ],lb,[ ])`

SOLUZIONI

Soluzione ottima	(8.95, 1.60, 2.64)
Valore ottimo	1718

ANALISI DI SENSIBILITÀ

Supponendo che i minuti a disposizione della linea di produzione C diventino 360, determinare la nuova soluzione ottima ed il nuovo profitto ottimo.

Nuova soluzione ottima	(7.71, 2.60, 2.57)
Nuovo profitto ottimo	1729