“Sapienza” Università di Roma - Dipartimento di Ingegneria Informatica, Automatica e Gestionale

Algoritmo di Branch & Bound

Docente: Renato Bruni

bruni@dis.uniroma1.it

Corso di: Ottimizzazione Combinatoria

Vogliamo Risolvere PLI (o PL01)

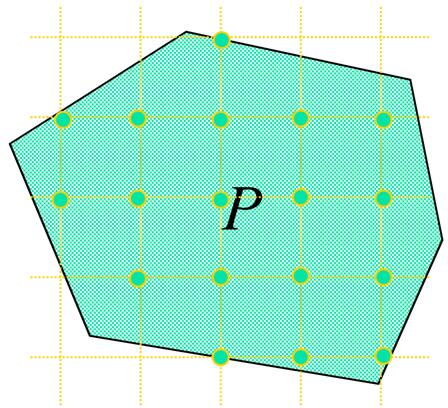
Dato un problema di Programmazione Lineare **Intera** (o **Binaria**),

vogliamo trovare **numericamente**, nell'insieme ammissibile S, **l'ottimo** x^*

$$\begin{cases} \min \; c^{\mathrm{T}}x \\ x \in S \end{cases}$$

Come visto, S è spesso espresso come l'insieme di punti interi (o binari) all'interno di un poliedro P (**formulazione scelta**, $\boldsymbol{S = P \cap Z^n}$)

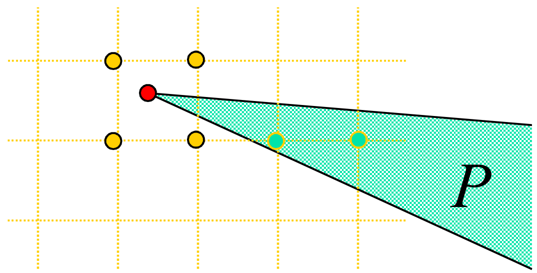
$$\begin{cases} \min \; c^{\mathrm{T}}x \\ x \in P \quad (\text{es. } Ax \geq b) \\ x \in Z^n \quad (\text{o } \; x \in \{0,1\}^n) \end{cases}$$



Possibili Approcci

- Dato che generalmente il numero di punti ammissibili è finito, si potrebbe pensare ad una **enumerazione**
- Ma, per problemi appena realistici questi punti sono un numero **enorme**: l'enumerazione completa richiederebbe tempi improponibili, come visto nella prima lezione
- Serve eventualmente un **altro** tipo di enumerazione
- Oppure si potrebbe pensare di migliorare la formulazione del problema, o di approssimare per eccesso e per difetto la soluzione fino a far incontrare le due approssimazioni (gap = 0 → ottimo)

- Attenzione: **arrotondando** all'intero più vicino una soluzione del problema di PL non ho nessuna garanzia né di ottimalità né di ammissibilità (soprattutto per problemi binari)



Idee di Base del Branch & Bound

Approccio di soluzione basato sull'enumerazione **implicita**. Idee di base:

- **Partizionare** l'insieme ammissibile in sottoinsiemi più facili P_i (sottoproblemi)

$$P_i \cap P_j = \emptyset \quad i \neq j$$

$$\bigcup_i P_i = P$$

- Procurarsi una soluzione ammissibile (**ottimo corrente**) ad esempio risolvendo il problema su alcuni sottoinsiemi, o tramite un euristica
- Continuare a risolvere il problema sui restanti sottoinsiemi
  - **scartando** quelli dove quanto di meglio potrei ottenere (dato dal **lower bound**) non è migliore di quanto già ho (ottimo corrente)
  - **analizzando** gli **altri**: aggiornando l'ottimo corrente nel caso di soluzioni ammissibili migliori o partizionando ulteriormente i sottoinsiemi non abbastanza facili

Come Fare Ciò?

Per mettere in pratica queste idee servono delle tecniche per effettuare:

- **Bounding**: tecniche per valutare **quanto di meglio** potrei ottenere su un sottoinsieme P_i

  Vogliamo approssimare per difetto la soluzione ottima del sottoproblema: trovare **lower bound** $\leq c^T x^*$ (limitatamente a P_i)

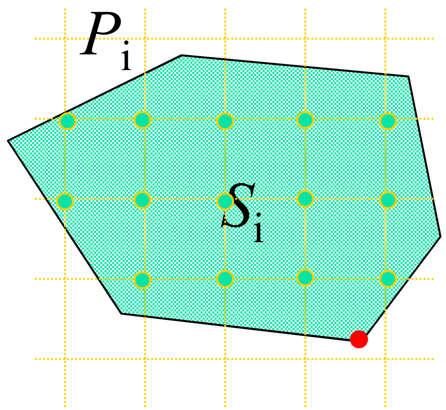
  Cerco compromesso tra velocità di calcolo e accuratezza del bound: tanti più sottoproblemi posso eliminare, tanto più velocizzo la soluzione del problema complessivo

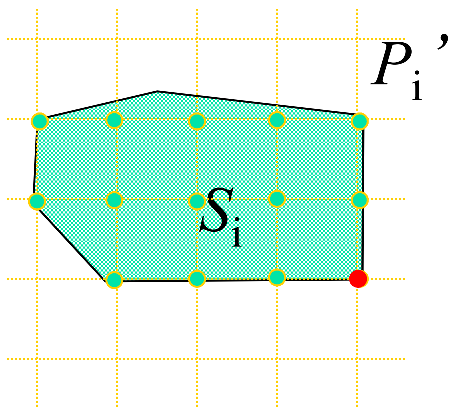
- **Branching**: tecniche per generare i sottoproblemi P_i

  Vogliamo generare sottoproblemi abbastanza facili ma non troppo numerosi

Bounding 1

- **Rilassamento Lineare** (più usato): elimino i vincoli di interezza del sottoproblema P_i Ho problemi di PL, risolubili facilmente ad es. col simplesso
- Il minimo di $c^T x$ scegliendo tra tutti i punti (interi o meno) **sarà** $\leq$ del minimo della stessa funzione scegliendo solo tra i punti interi
- L'accuratezza, cioè la distanza dall'ottimo intero del sottoproblema, dipenderà dalla **qualità** della formulazione P_i del sottoproblema S_i
- In alcuni casi fortunati potrei trovare direttamente l'ottimo intero di S_i





Bounding 2

Sono possibili anche **altre** tecniche per individuare un lower bound:

- Rilassamento di altri vincoli difficili, ottenendo un sottoproblema più facile con insieme ammissibile più grande

  Aumentando il numero di punti tra cui scegliere, il minimo di $c^T x$ non potrà che **diminuire o restare uguale**

- Modifica della funzione obiettivo in modo che la nuova funzione obiettivo sia $\leq c^T x$ sul sottoinsieme P_i (= ci dia un lower bound) ma renda il sottoproblema più facile

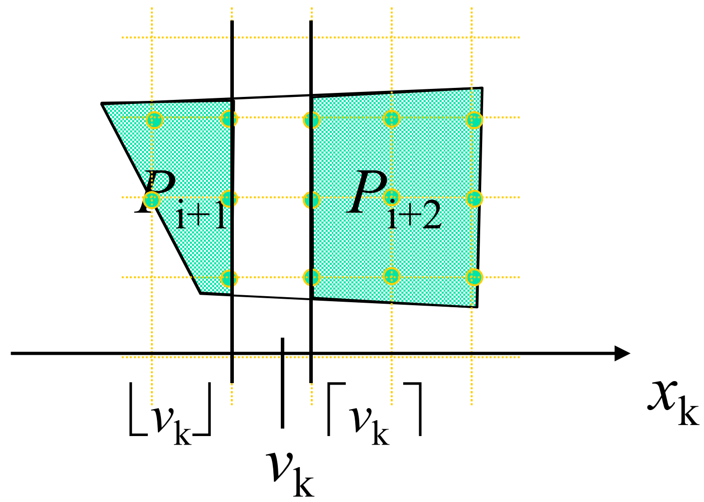
Branching

- **Binario** agli interi più vicini (più usato): se risolvendo il rilassamento lineare trovo una soluzione x che ha componenti non intere (dette **frazionarie**), ad es. x_k con valore v_k

P_i era un sottoinsieme non abbastanza facile → lo **partiziono** in P_{i+1} e P_{i+2}

$$P_{i+1} = P_i \cap \{x:\ x_k \le \lfloor v_k \rfloor\}$$

$$P_{i+2} = P_i \cap \{x:\ x_k \ge \lceil v_k \rceil\}$$



- Così elimino una striscia di P_i che però **non contiene** soluzioni intere: tagliando in questo modo avrò prima o poi soluzioni intere ai rilassamenti
- Per problemi binari semplicemente fisso la variabile a 0 e a 1

Assemblando le Parti

- Siamo adesso in grado di costruire uno **schema** complessivo di Branch & Bound per problemi di minimo del tipo descritto

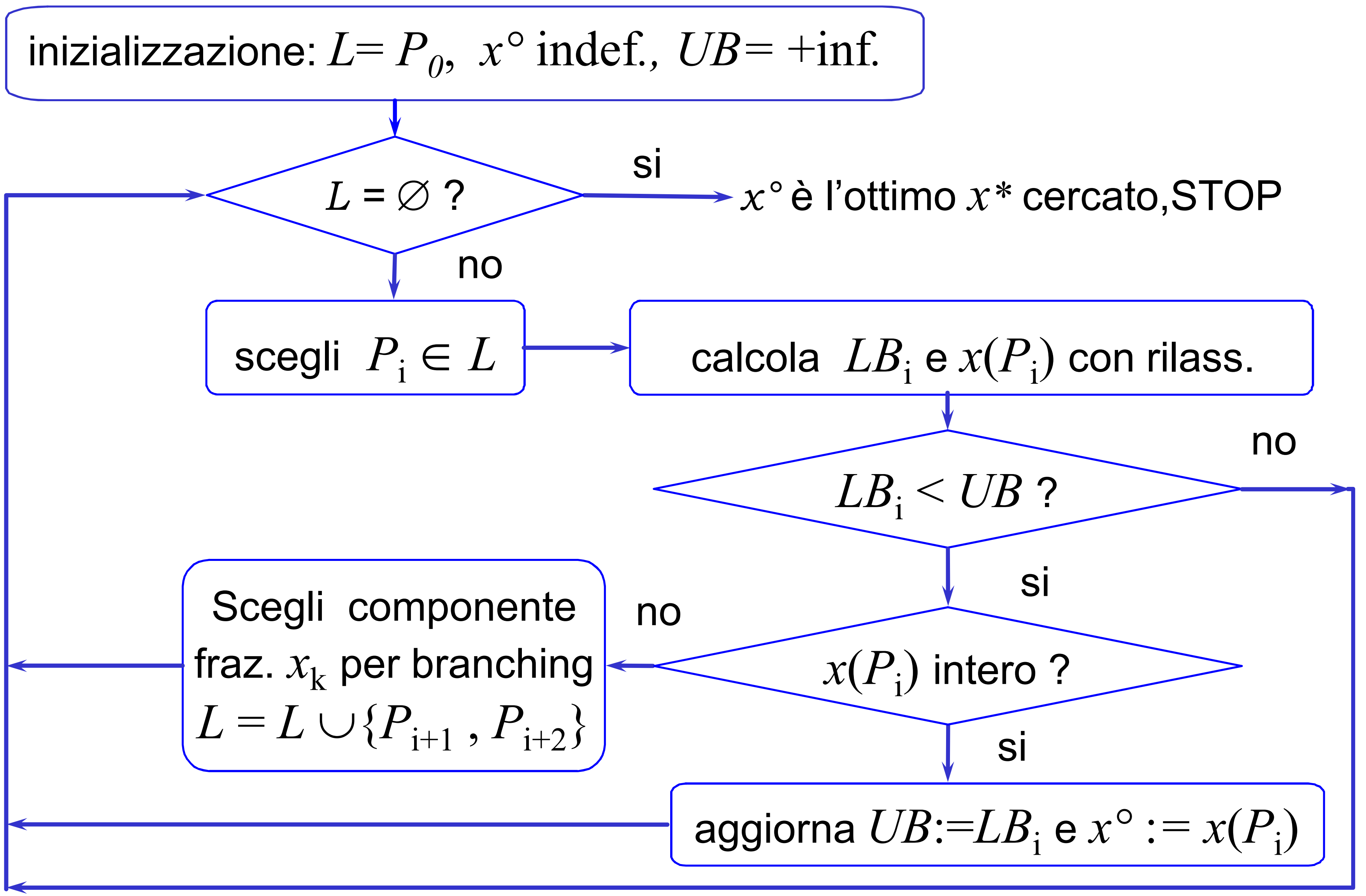
$$\begin{cases} \min \ c^T x \\ x \in P \\ x \in Z^n \quad (\text{o } x \in \{0,1\}^n) \end{cases}$$

Indichiamo con L la **lista dei sottoproblemi** P_i da risolvere (problemi aperti);

con x^o l'**ottimo corrente**, con UB (**upper bound**) il valore $c^T x^o \geq c^T x^*$;

con LB_i e $x(P_i)$ rispettivamente il **lower bound** trovato per il **sottoproblema** P_i e la **soluzione** ad esso corrispondente

Schema del Branch & Bound per min



Ulteriori Aspetti

- Scelta del sottoproblema $P_i \in L$
  - quello con minimo LB (**best bound**: più promettenti)
  - LIFO (last in first out)
  - FIFO (first in first out)
- Scelta della variabile di branching x_k
  - variabile più intera
  - variabile più frazionaria
  - ordine predefinito
- Precisione numerica nei confronti e tolleranza interi
- Tutte le scelte **influenzano** l’evoluzione dell’algoritmo, quindi i tempi di calcolo
- Purtroppo **non esiste** una scelta che sia sempre la migliore per tutti i problemi

Osservazioni

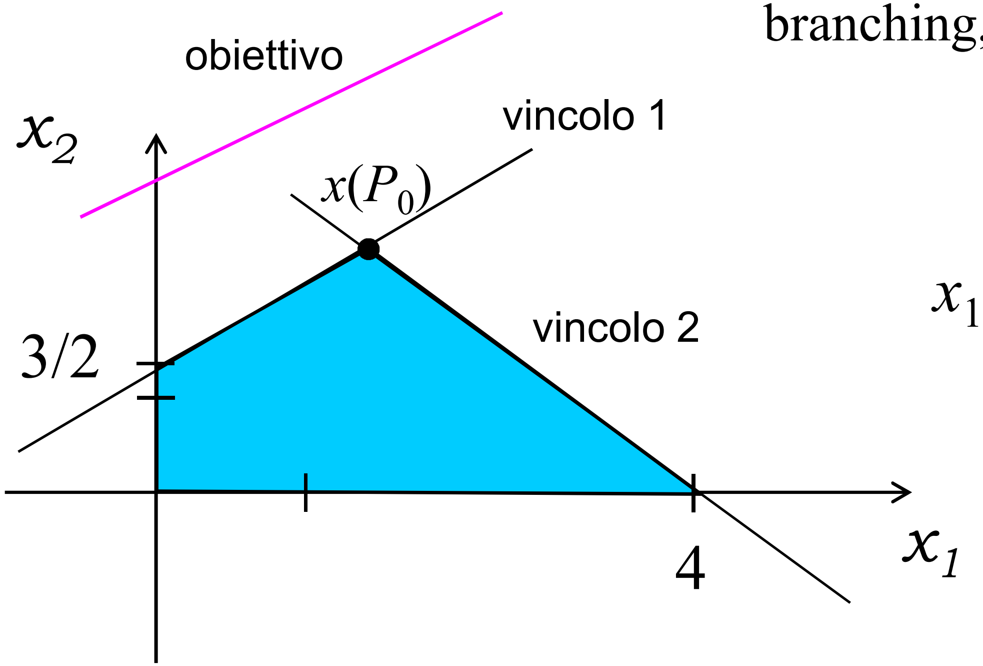
- È un algoritmo **esatto**, cioè garantisce, dato tempo sufficiente e a meno di imprecisioni numeriche (sempre presenti su macchine reali) di trovare l'ottimo se esso esiste
- Per problemi di max è tutto speculare: dai ril. lin. ottengo UB_i ; l'ottimo corrente è un LB, che inizializzo a $-\inf$; il confronto è $UB_i > LB$
- L'evoluzione dell'algoritmo è rappresentabile come la visita di un **albero** (detto albero di branching, vedremo in seguito su un esempio)
- Implica la risoluzione di un **gran numero** di rilassamenti lineari, infatti la PLI è più complessa della PL
- Se ho **formulazioni buone** dei vari problemi ho LB migliori: posso allora cercare di migliorare queste formulazioni (detto Branch & Cut)
- Se ho una formulazione iniziale **così buona** (**ottima**) che la soluzione del primo rilassamento è già **intera**, ho la soluzione ottima intera senza alcun branching (cioè velocemente)

Esempio 1

$$P_0 \quad \begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \leq 9 \\ x_1 + x_2 \leq 4 \\ x_1 , x_2 \geq 0 \\ x_1 , x_2 \in Z \end{cases} \quad \Rightarrow \quad x(P_0) \; \begin{array}{l} x_1 = 3/2 \\ x_2 = 5/2 \end{array}$$

$$UB_0 = 7/2$$

soluzione frazionaria, non ho ottimo corrente, devo fare branching, ad esempio su x_1



$$x_1 \begin{cases} x_1 \leq 1 & P_1 \\ x_1 \geq 2 & P_2 \end{cases}$$

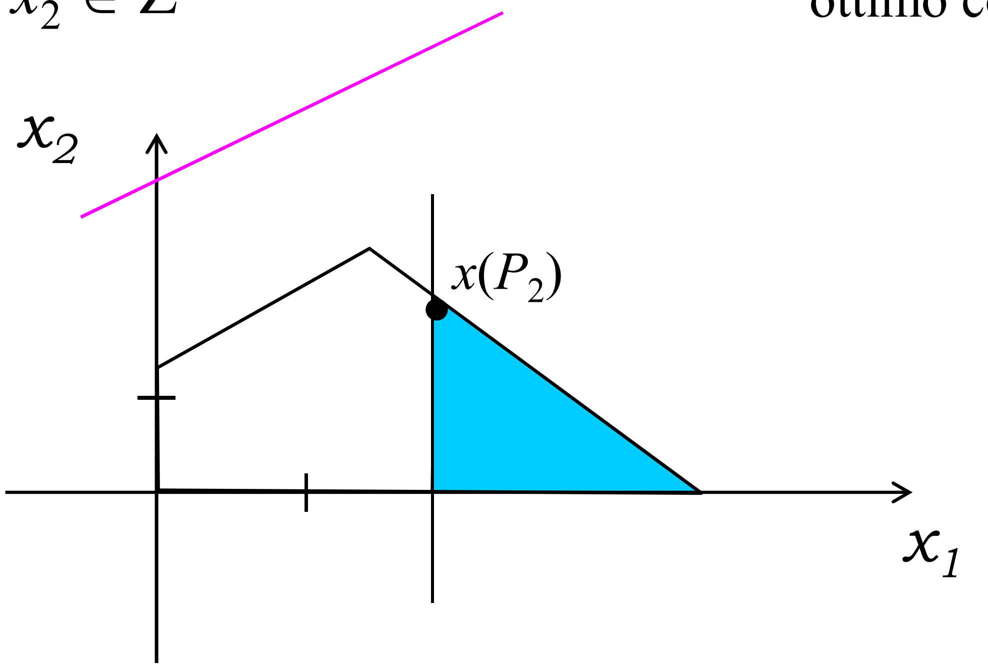
Esempio 1

$$P_2 \quad \begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \leq 9 \\ x_1 + x_2 \leq 4 \\ x_1 \geq 2 \\ x_1 \,,\, x_2 \geq 0 \\ x_1 \,,\, x_2 \in Z \end{cases}$$

$$\Rightarrow \quad x(P_2) \quad \begin{matrix} x_1 = 2 \\ x_2 = 2 \end{matrix}$$

$$UB_2 = 2$$

soluzione intera, aggiorno ottimo corrente, no branching



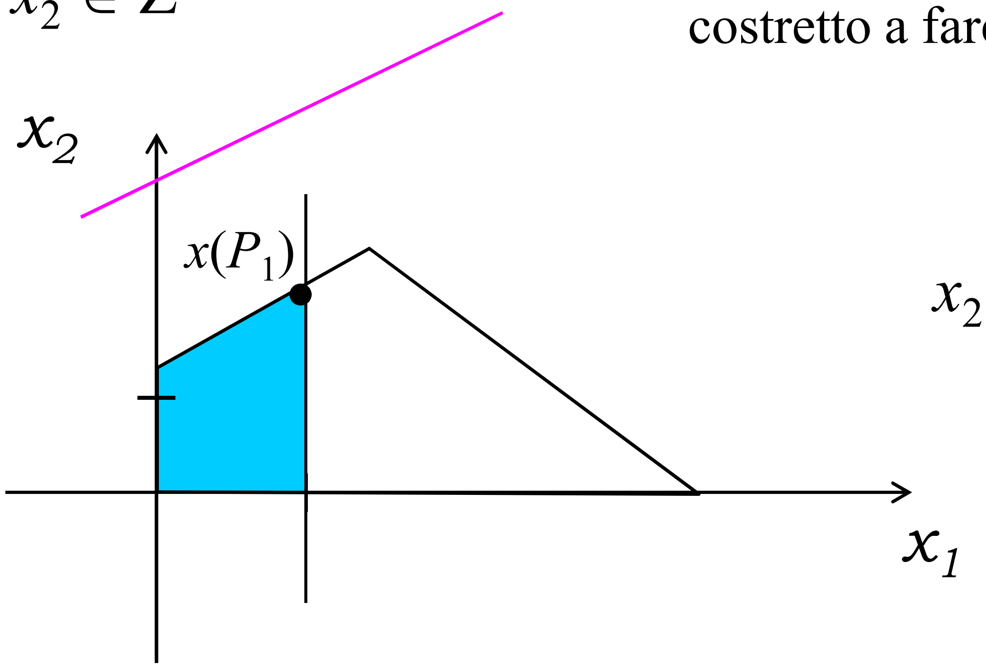
Esempio 1

$$
P_1 \quad \begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \le 9 \\ x_1 + x_2 \le 4 \\ x_1 \le 1 \\ x_1 , x_2 \ge 0 \\ x_1 , x_2 \in Z \end{cases}
$$

$x(P_1)$ $\quad x_1 = 1$, $x_2 = 13/6$

$UB_1 = 10/3$

$UB_1 >$ valore ottimo corrente, la soluzione è frazionaria, sono costretto a fare branching



x_2

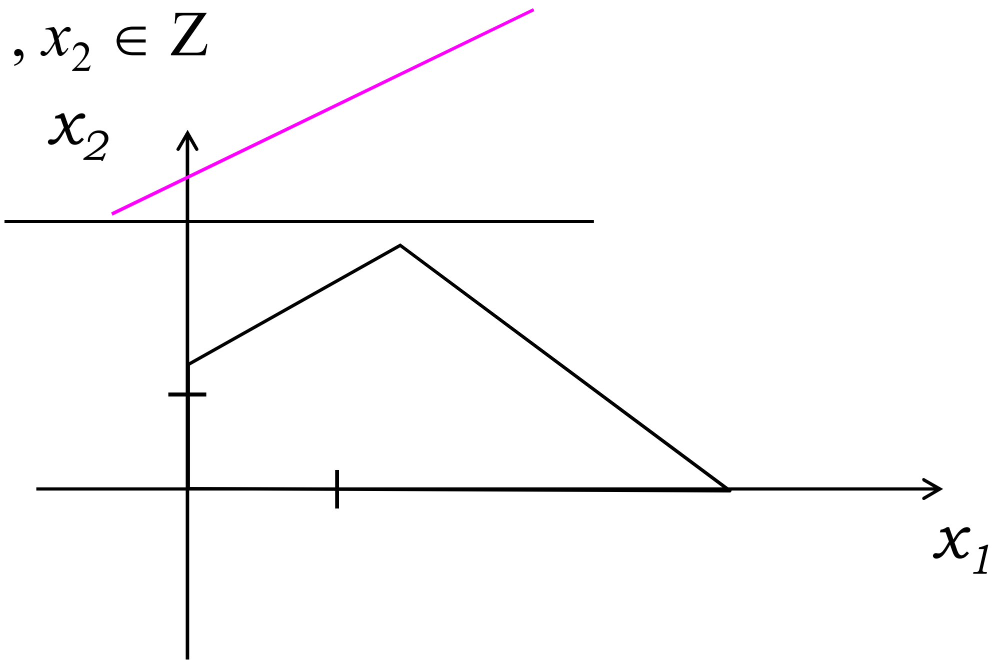
$x_2 \le 2 \quad P_3$

$x_2 \ge 3 \quad P_4$

Esempio 1

$$
\begin{cases}
\max -x_1 + 2x_2 \\
-4x_1 + 6x_2 \leq 9 \\
x_1 + x_2 \leq 4 \\
x_1 \leq 1 \\
x_2 \geq 3 \\
x_1 , x_2 \geq 0 \\
x_1 , x_2 \in Z
\end{cases}
\qquad P_4 \Rightarrow
$$

problema inammissibile, lo chiudo

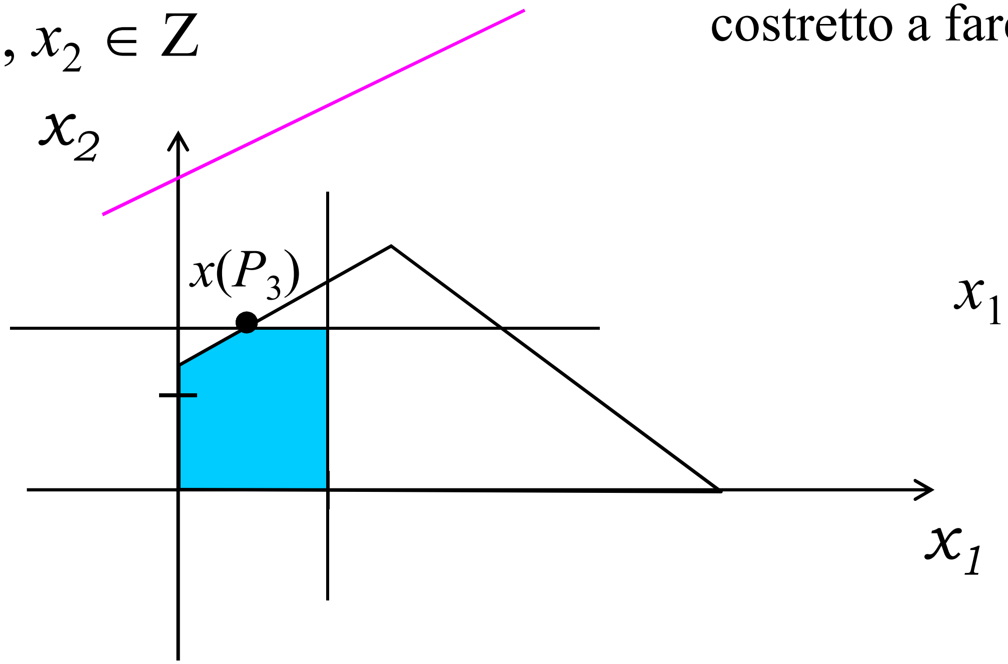


Esempio 1

$$P_3 \left\{ \begin{array}{l} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \leq 9 \\ x_1 + x_2 \leq 4 \\ x_1 \leq 1 \\ x_2 \leq 2 \\ x_1 , x_2 \geq 0 \\ x_1 , x_2 \in Z \end{array} \right. \Rightarrow x(P_3) \begin{array}{l} x_1 = 3/4 \\ x_2 = 2 \end{array}$$

$$UB_3 = 13/4$$

$UB_3 >$ valore ottimo corrente, la soluzione è frazionaria, sono costretto a fare branching



$$x_1 \begin{cases} x_1 \leq 0 & P_5 \\ x_1 \geq 1 & P_6 \end{cases}$$

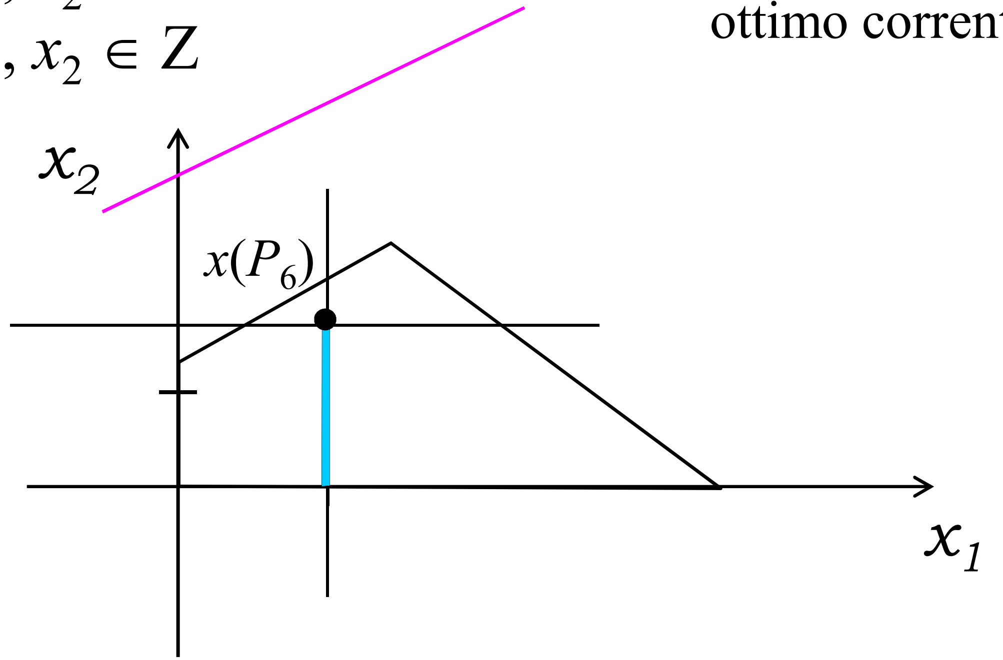
Esempio 1

$$P_6 \quad \begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \leq 9 \\ x_1 + x_2 \leq 4 \\ x_1 \leq 1 \\ x_2 \leq 2 \\ x_1 \geq 1 \\ x_1 , x_2 \geq 0 \\ x_1 , x_2 \in Z \end{cases}$$

$\Rightarrow$ $x(P_6)$ $\begin{matrix} x_1 = 1 \\ x_2 = 2 \end{matrix}$

$UB_6 = 3$

$UB_6 >$ valore ottimo corrente, la soluzione è intera, aggiorno ottimo corrente



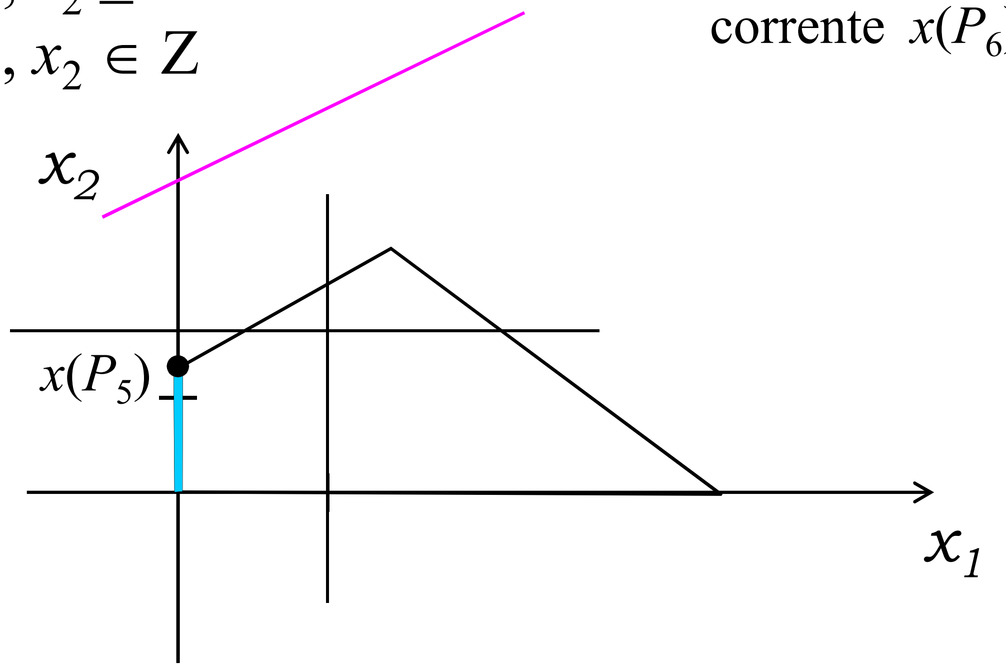
Esempio 1

$$P_5 \quad \begin{cases} \max -x_1 + 2x_2 \\ -4x_1 + 6x_2 \le 9 \\ x_1 + x_2 \le 4 \\ x_1 \le 1 \\ x_2 \le 2 \\ x_1 \le 0 \\ x_1\ ,\ x_2 \ge 0 \\ x_1\ ,\ x_2 \in Z \end{cases}$$

$x(P_5)$ $\quad x_1 = 0$, $x_2 = 3/2$

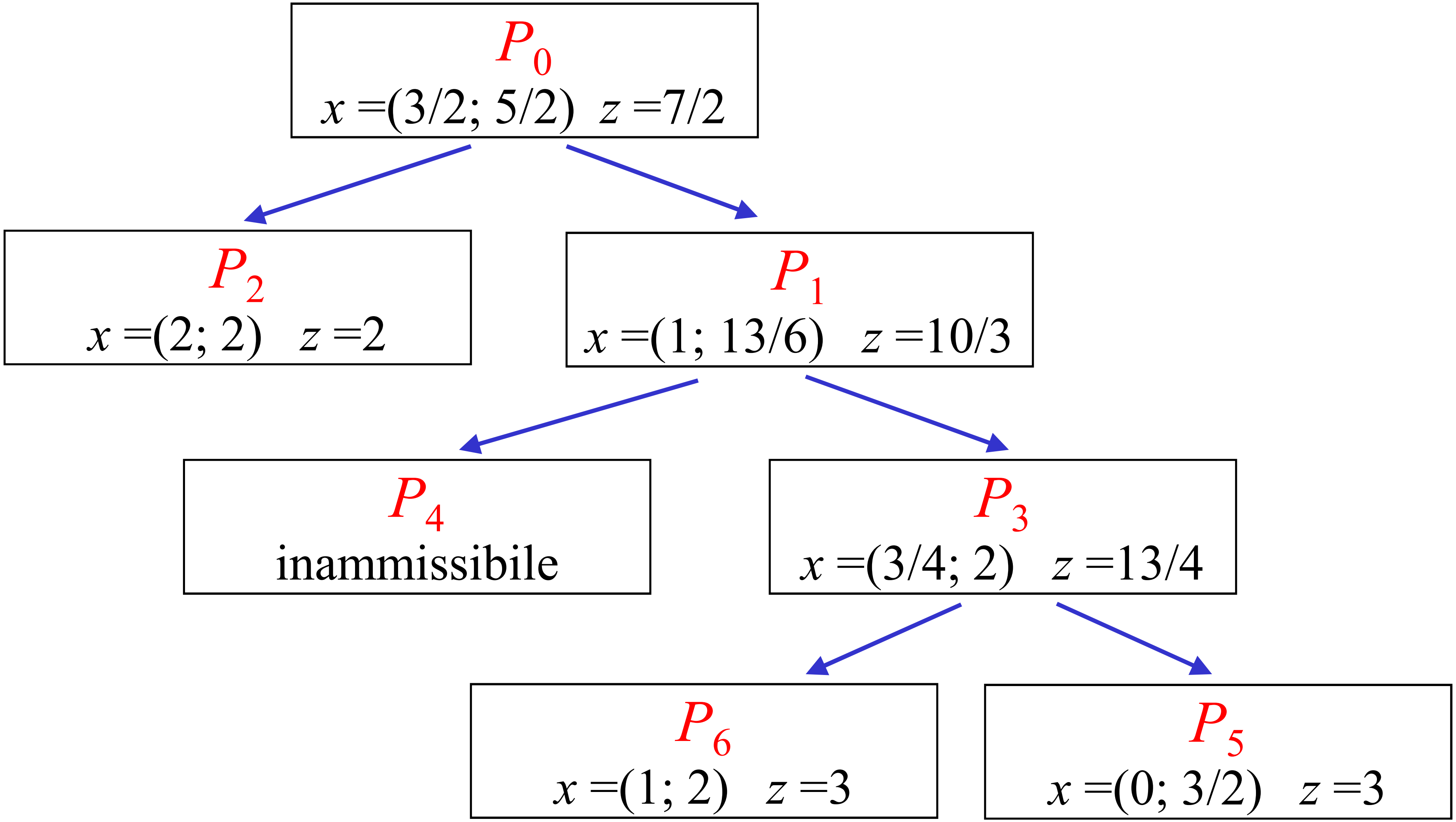
$UB_5 = 3$

$UB_5 \le$ valore ottimo corrente, allora posso chiudere il problema e l'ottimo corrente $x(P_6)$ è l'ottimo complessivo



Albero di Branching

L'evoluzione dell'algoritmo in questo esempio 1 si può rappresentare così



Esempio 2

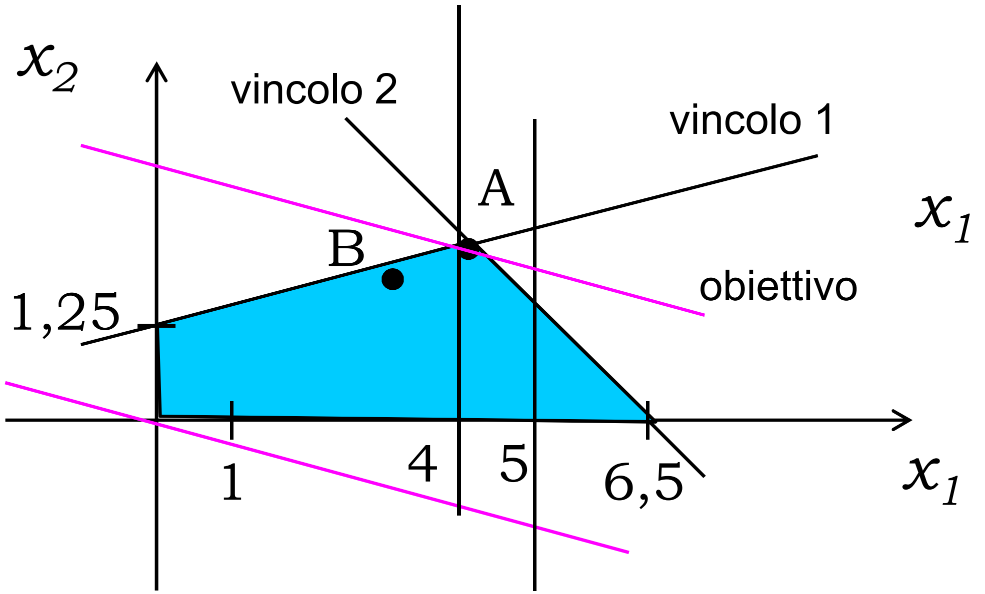
$$\max\ x_1 + 4x_2$$

$$P_0 \begin{cases} -x_1 + 4x_2 \le 5 \\ 2x_1 + 2x_2 \le 13 \\ x_1 \ge 0 \quad x_2 \ge 0 \\ \cancel{x \in Z^2} \end{cases}$$

e disponiamo anche di una soluzione ammissibile B

$\Rightarrow$ $B \begin{cases} \hat{x}_1 = 3 \\ \hat{x}_2 = 2 \end{cases}$ $LB = 11$

$\Rightarrow$ $A \begin{cases} \hat{x}_1 = 4{,}2 \\ \hat{x}_2 = 2{,}3 \end{cases}$ $UB_0 = 13{,}4$



x_1

$x_1 \le 4$

P_1

$x_1 \ge 5$

P_2

Esempio 2

$$\max\ x_1 + 4x_2$$

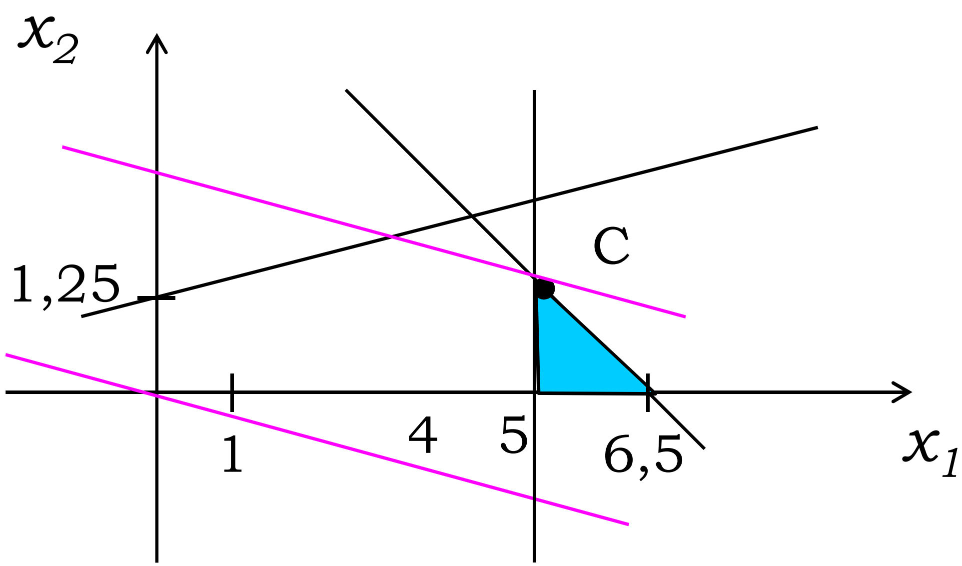
$$P_2 \begin{cases} -x_1 + 4x_2 \le 5 \\ 2x_1 + 2x_2 \le 13 \\ x_1 \ge 0 \quad x_2 \ge 0 \\ x_1 \ge 5 \\ \cancel{x \in Z^2} \end{cases}$$

$LB = 11$

$$\Rightarrow \quad C \begin{cases} \hat{x}_1 = 5 \\ \hat{x}_2 = 1{,}5 \end{cases}$$

$$\boxed{UB_2 = 11}$$

Chiudo il sottoproblema perché non migliore del LB



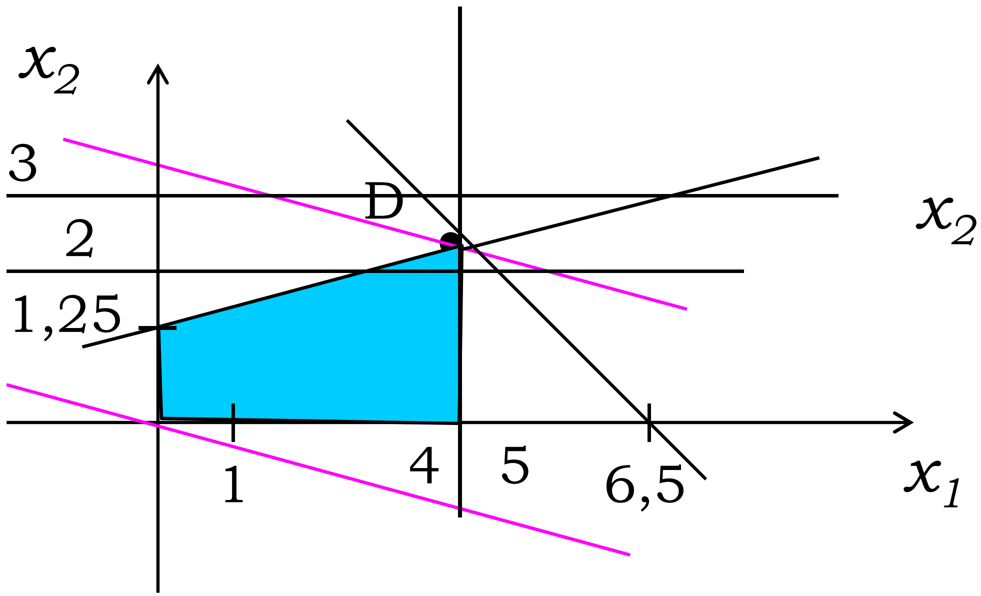
Esempio 2

$$\max \; x_1 + 4x_2$$

$$P_1 \begin{cases} -x_1 + 4x_2 \le 5 \\ 2x_1 + 2x_2 \le 13 \\ x_1 \ge 0 \quad x_2 \ge 0 \\ x_1 \le 4 \\ \cancel{x \in Z^2} \end{cases}$$

$LB = 11$

$\Rightarrow \quad D \begin{cases} \hat{x}_1 = 4 \\ \hat{x}_2 = 2{,}25 \end{cases} \qquad UB_1 = 13$



x_2

$x_2 \le 2$

P_3

$x_2 \ge 3$

P_4 $\boxed{\text{vuoto}}$

Esempio 2

$$\max\ x_1 + 4x_2$$

$$P_3 \begin{cases} -x_1 + 4x_2 \le 5 \\ 2x_1 + 2x_2 \le 13 \\ x_1 \ge 0 \quad x_2 \ge 0 \\ x_1 \le 4 \\ x_2 \le 2 \\ \cancel{x \in Z^2} \end{cases}$$

$LB = 11$

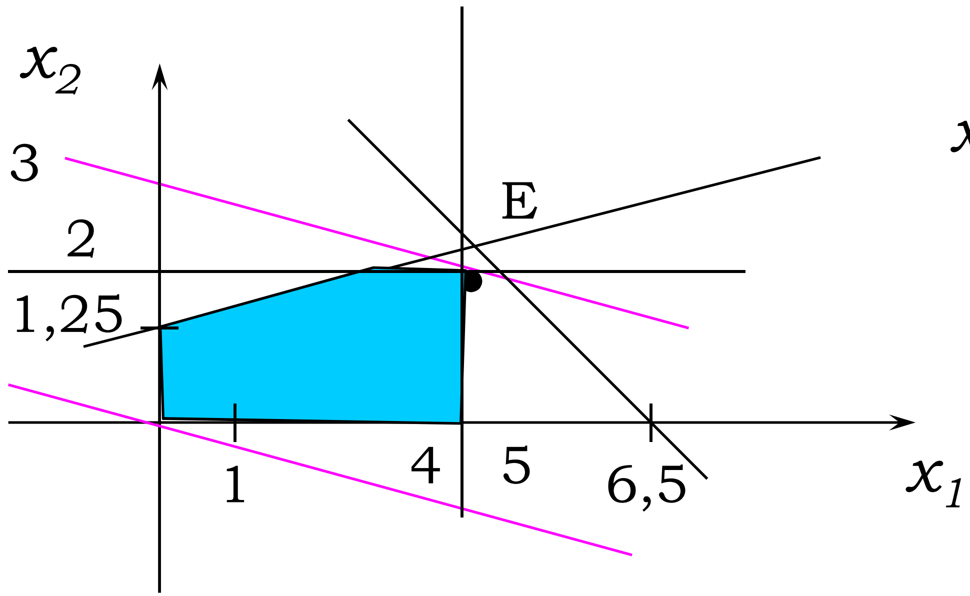
$\Rightarrow \quad E \begin{cases} \hat{x}_1 = 4 \\ \hat{x}_2 = 2 \end{cases}$

$UB_3 = 12$

x intera

aggiorna ***LB***

x **ottima**



Albero di Branching

L'evoluzione dell'algoritmo in questo esempio 2 si può rappresentare così