Parte III:

Algoritmo di Branch-and-Bound

Divide et Impera

Sia

$$z^* = \max \{c^T x : x \in S\} \qquad (1)$$

un problema di ottimizzazione combinatoria difficile da risolvere.

Domanda: E' possibile decomporre il problema (1) in una collezione di sottoproblemi tali che

- ogni nuovo sottoproblema sia facile da risolvere, e
- le informazioni ottenute risolvendo la collezione di sottoproblemi ci guidino nella ricerca della soluzione ottima del problema (1)?

Divide et Impera

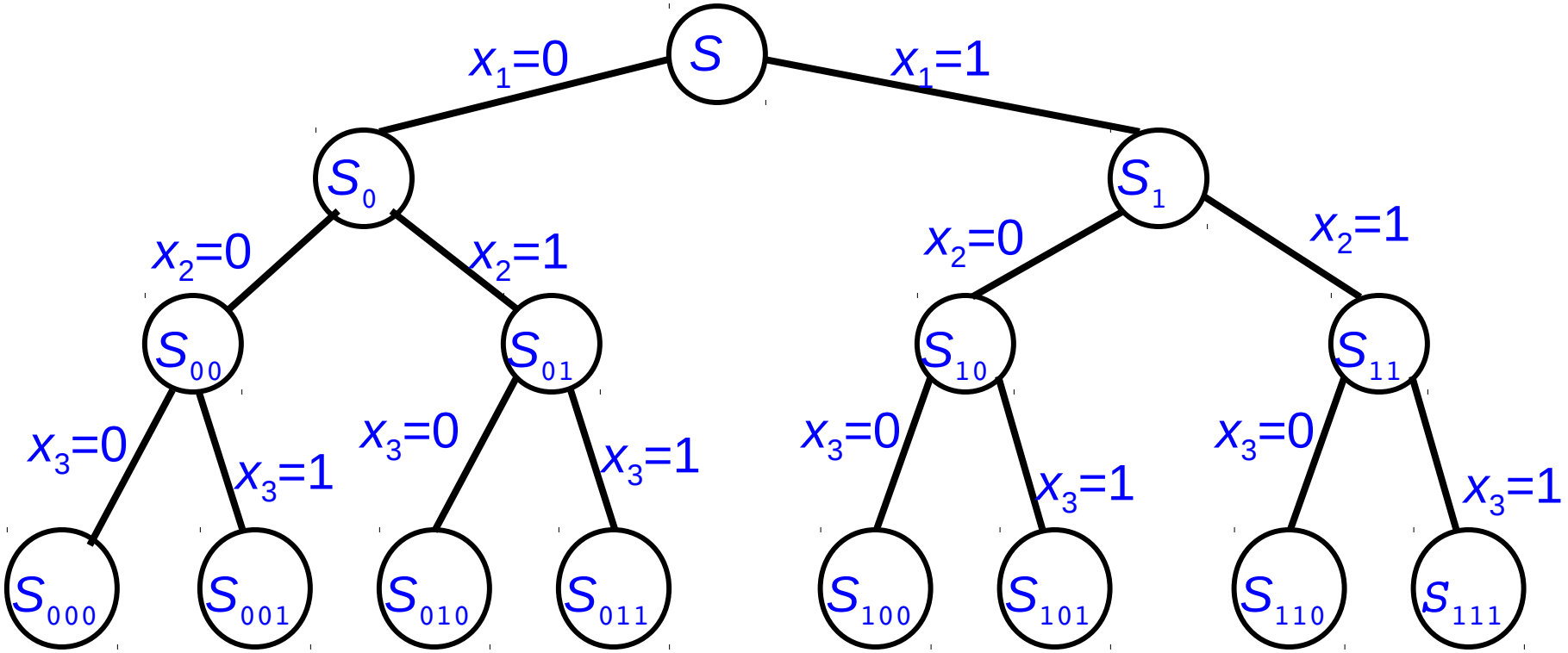
Proposizione: Sia $S = S_1 \cup S_2 \cup \ldots \cup S_K$ una decomposizione della regione ammissibile S in K sottoinsiemi, e sia $z^h = \max \{c^T x : x \in S_h\}$, per $h = 1,\ldots,K$. Allora $z^* = \max_h z^h$.

Osservazione: Non abbiamo richiesto che la decomposizione sia una partizione (ovvero, che $S_i \cap S_j = \emptyset$), ma non abbiamo nemmeno escluso che possa esserlo!

Una tipica rappresentazione dell'approccio divide et impera è tramite un albero di enumerazione.

Un esempio

Se $S \subseteq \{0,1\}^3$, possiamo costruire il seguente albero di enumerazione



Le foglie dell'albero corrispondono esattamente ai punti che andremmo a esaminare se potessimo fare un'enumerazione completa.

Divide et Impera

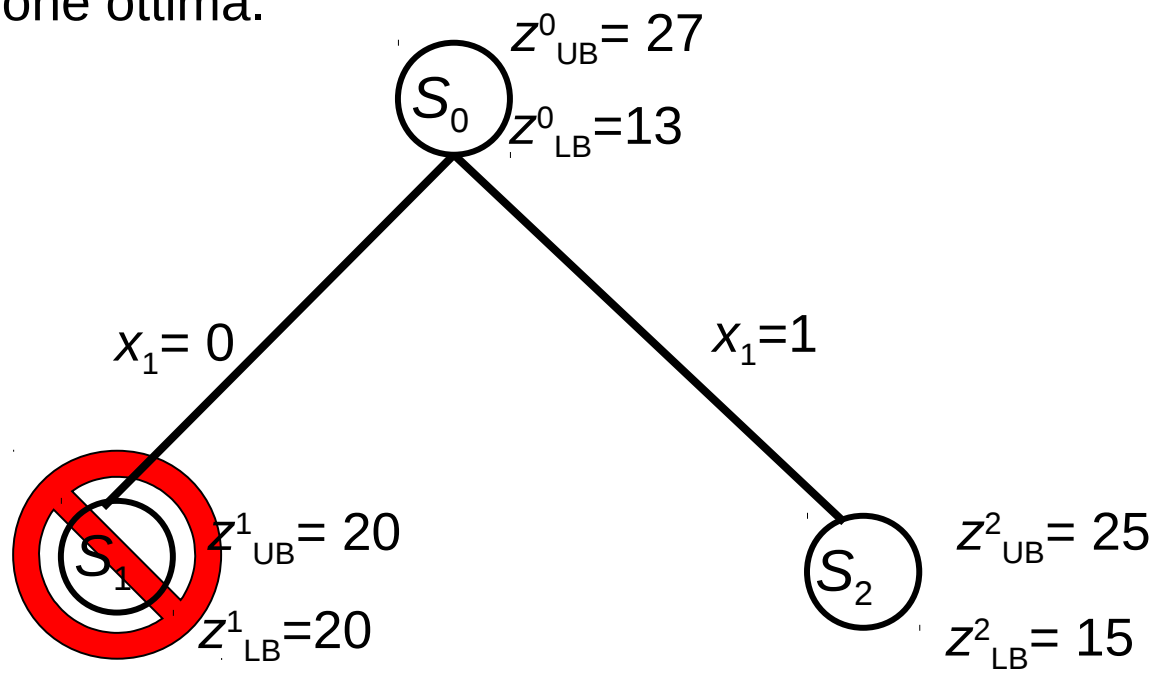
Osservazione: Costruendo in questo modo l'albero di enumerazione stiamo semplicemente enumerando TUTTI gli insiemi ammissibili!!!!

Per la maggior parte dei problemi è impossibile effettuare una enumerazione completa.

Poiché per un problema di ottimizzazione combinatoria conosciamo i bound "primali" e i bound "duali", l'idea è quella di utilizzare tali bound per enumerare gli insiemi ammissibili in modo più efficace.

Potatura per ottimalità

Le informazioni su upper e lower bound per il valore ottimo z^* permettono di decidere quali nodi dell'albero ha senso continuare ad esaminare per trovare la soluzione ottima.



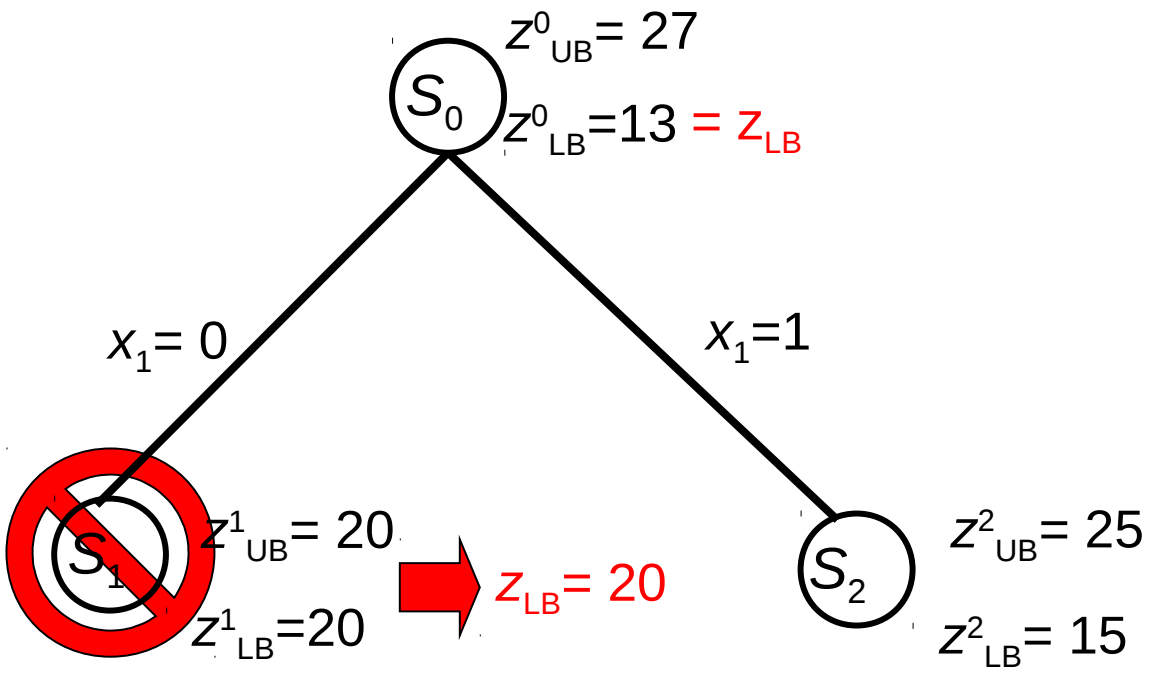
Poiché $z^1_{LB} = z^1_{UB} = 20$, sicuramente $z^1 = 20$. Pertanto, non c'è bisogno di esplorare ulteriormente il nodo S_1 che può essere *potato per ottimalità*.

Potatura per ottimalità

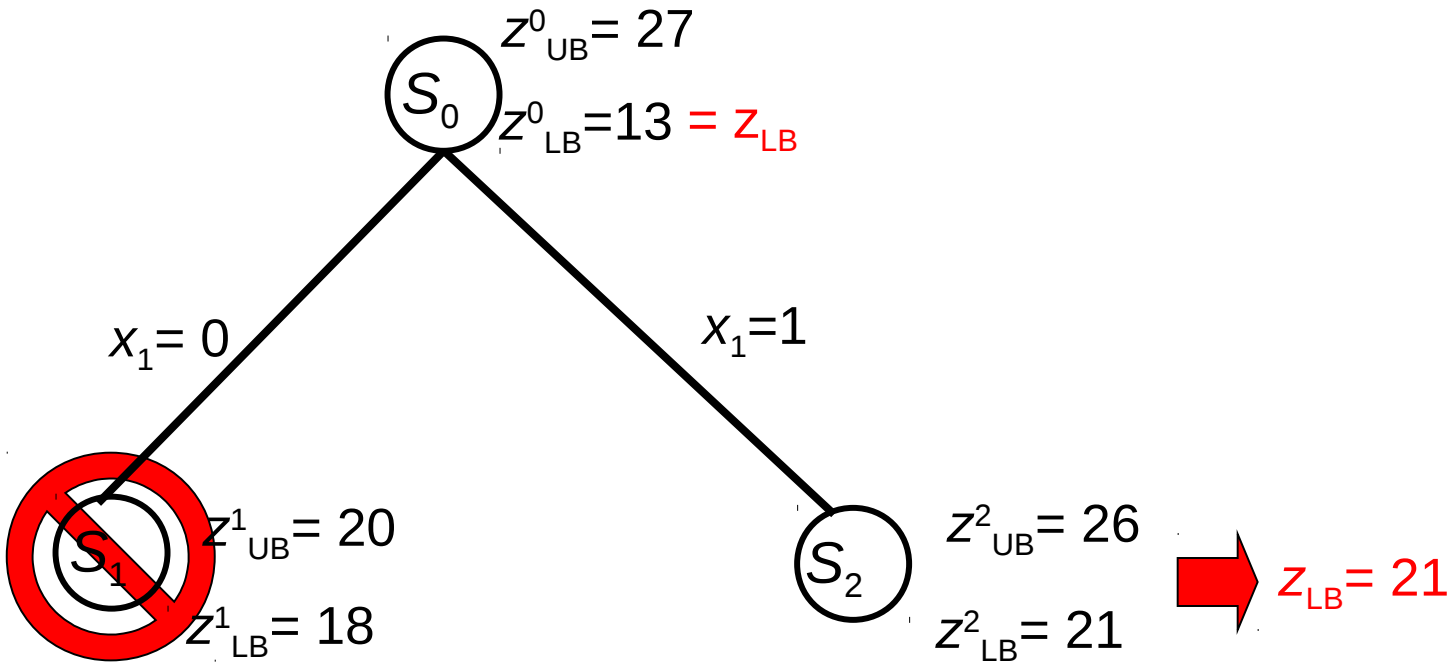
Ad ogni passo teniamo traccia di un lower bound globale z_{LB}.

Per il nodo S_0, $z_{LB} = z^0_{LB}$ (alternativamente, $z_{LB} = -\infty$).

Per ciascun nodo di ogni livello dell'albero, aggiorniamo il valore di z_{LB} se in corrispondenza di quel nodo abbiamo calcolato un lower bound MIGLIORE (ossia maggiore) di quello corrente.

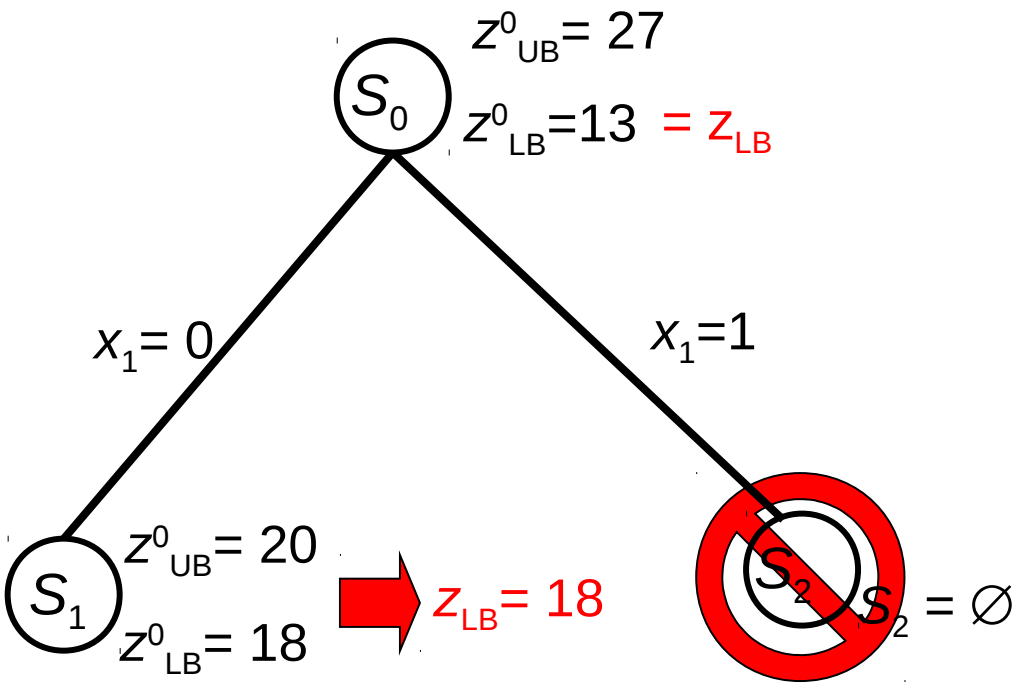


Potatura per bound



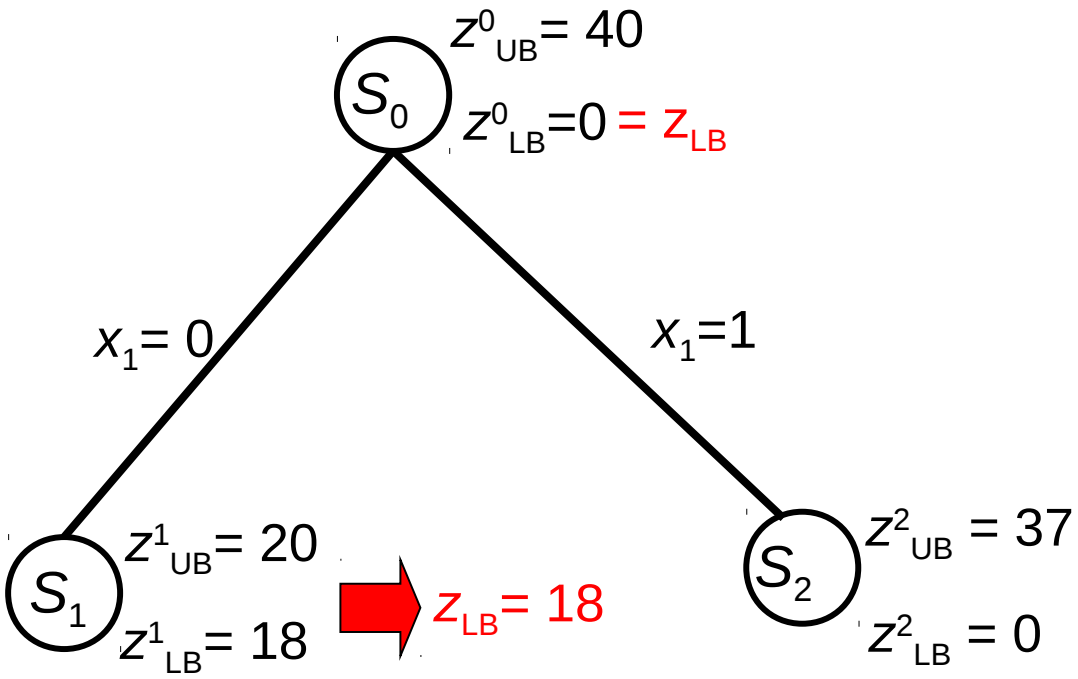
Poiché z_{LB} = 21, la soluzione ottima del problema iniziale ha valore almeno pari a 21. Dal momento che z^1_{UB} = 20, sicuramente non è possibile ottenere l'ottimo esplorando il nodo S_1. Pertanto, S_1 può essere *potato per bound*.

Potatura per inammissibilità



Poiché il sottoproblema associato al nodo S_2 è inammissibile, non è possibile ottenere l'ottimo esplorando il nodo S_2. Pertanto S_2 può essere *potato per inammissibilità*.

Nessuna potatura



In questo caso non possiamo fare nessuna osservazione che permetta di potare uno dei due nodi in esame e quindi è necessario esplorare sia S_1 che S_2.

Esempio

P $\max\ 4x_1 - x_2$

$7x_1 - 2x_2 \leq 14$

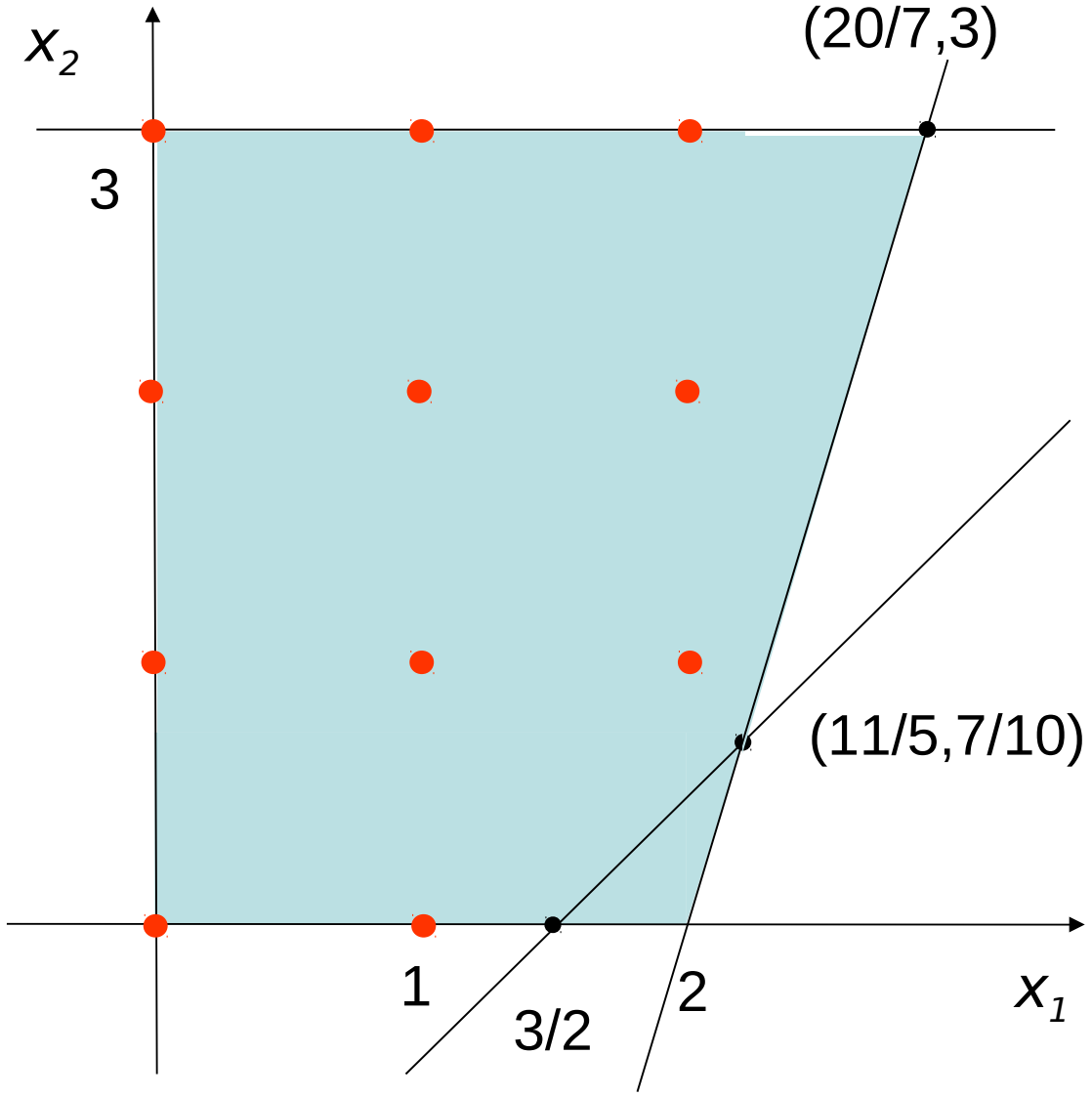
$x_2 \leq 3$

$2x_1 - 2x_2 \leq 3$

$x_1 \geq 0$

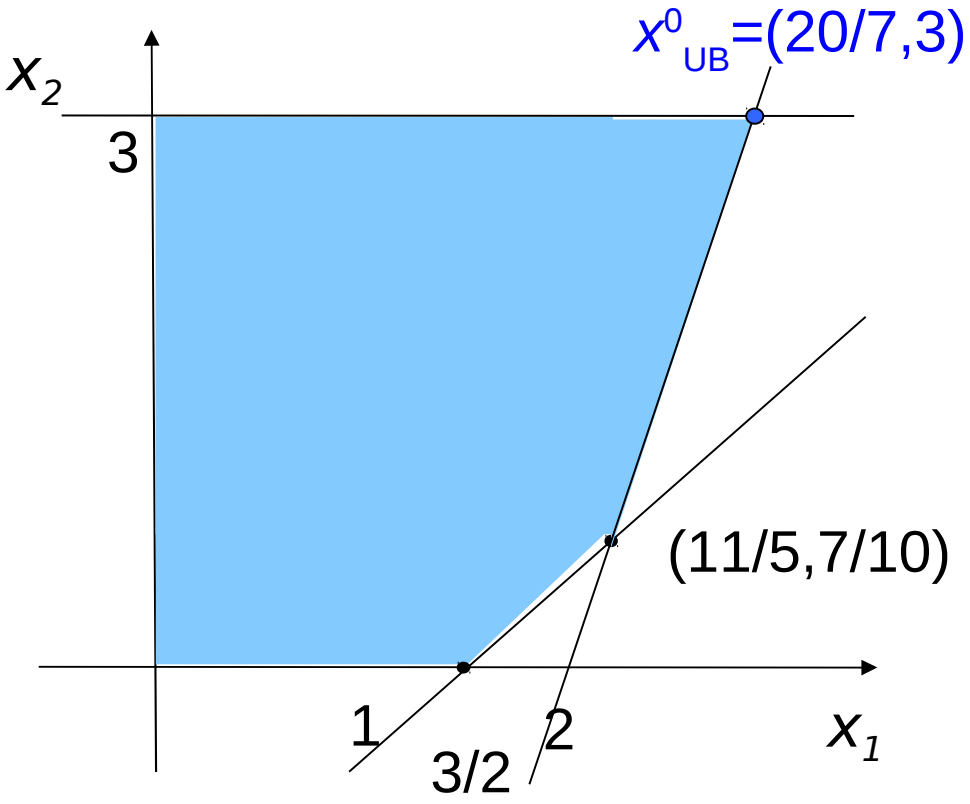
$x_2 \geq 0$

$x \in Z^2$



Esempio

Problema al nodo radice S_0: Rilassamento lineare di P.



Valore dell'UB al nodo radice: $z^0_{UB} = 59/7$.

Poiché non abbiamo una soluzione ammissibile, fissiamo $z^0_{LB} = -\infty$

Esempio

$z^0_{UB}=59/7$

$z^0_{LB}=-\infty=z_{LB}$

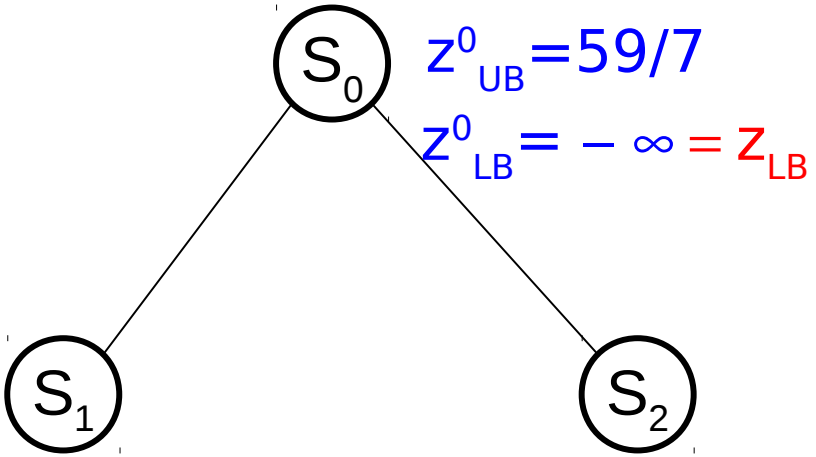
Consideriamo la variabile frazionaria x_1 del vettore x^0_{UB} e generiamo i sottoproblemi di S_0 applicando la seguente regola di branching:

$$S_1=S_o\cap\left\{x:x_1\leq\left\lfloor\hat{x}_1\right\rfloor\right\} \qquad S_2=S_o\cap\left\{x:x_1\geq\left\lceil\hat{x}_1\right\rceil\right\}$$

$$S_1=S_o\cap\left\{x:x_1\leq\lfloor 20/7\rfloor=2\right\}$$

$$S_2=S_o\cap\left\{x:x_1\geq\lceil 20/7\rceil=3\right\}$$

Esempio



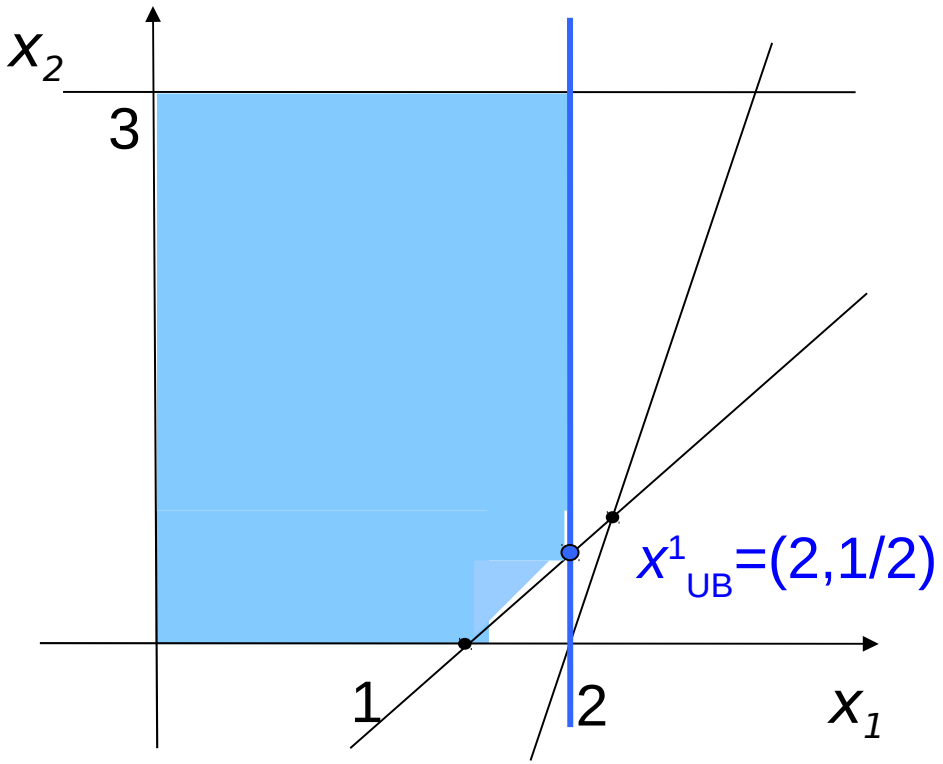
$$S_1 = S_0 \cap \{x : x_1 \leq 2\} \qquad S_2 = S_0 \cap \{x : x_1 \geq 3\}$$

Memorizziamo S_1 e S_2 in una lista di sottoproblemi che devono essere esplorati: $L = \{S_1, S_2\}$

Scegliamo di analizzare il sottoproblema S_1.

Esempio

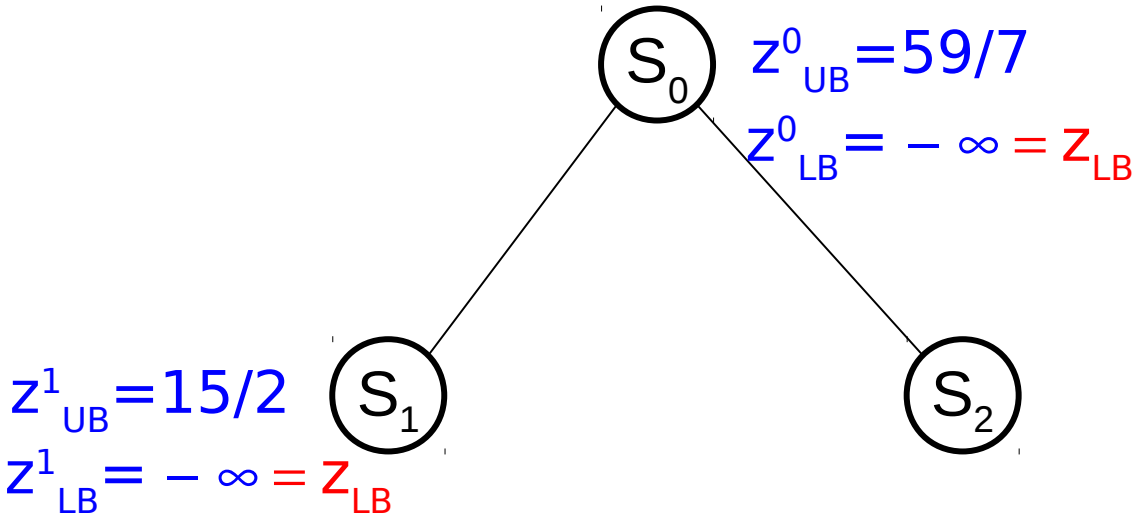
Risolviamo il sottoproblema al nodo S_1.



$$\max\ 4x_1 - x_2$$
$$7x_1 - 2x_2 \le 14$$
$$x_2 \le 3$$
$$2x_1 - 2x_2 \le 3$$
$$x_1 \le 2$$
$$x_1 \ge 0$$
$$x_2 \ge 0$$

Valore dell'UB per il sottoproblema S_1: $z^1_{UB} = 15/2$.

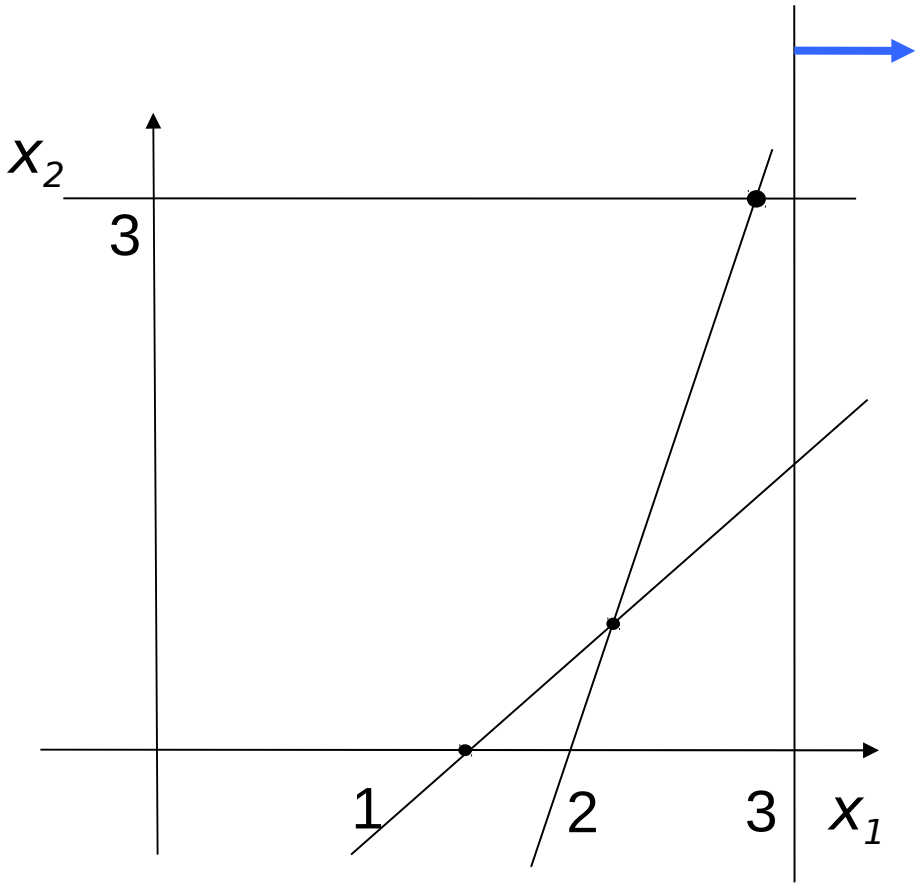
Esempio



Poiché non possiamo applicare nessun criterio di potatura, il nodo S_1 dovrà essere esplorato.

Esempio

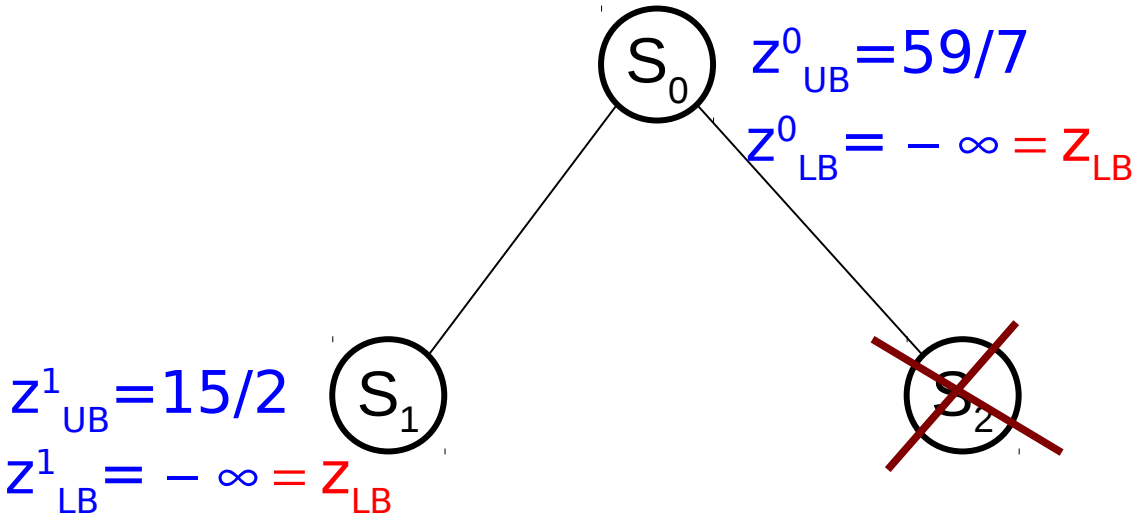
Risolviamo il sottoproblema al nodo S_2.



$$\max\ 4x_1 - x_2$$
$$7x_1 - 2x_2 \le 14$$
$$x_2 \le 3$$
$$2x_1 - 2x_2 \le 3$$
$$x_1 \le 2$$
$$x_1 \ge 3$$
$$x_1 \ge 0$$
$$x_2 \ge 0$$

Il sottoproblema al nodo S_2 è inammissibile!

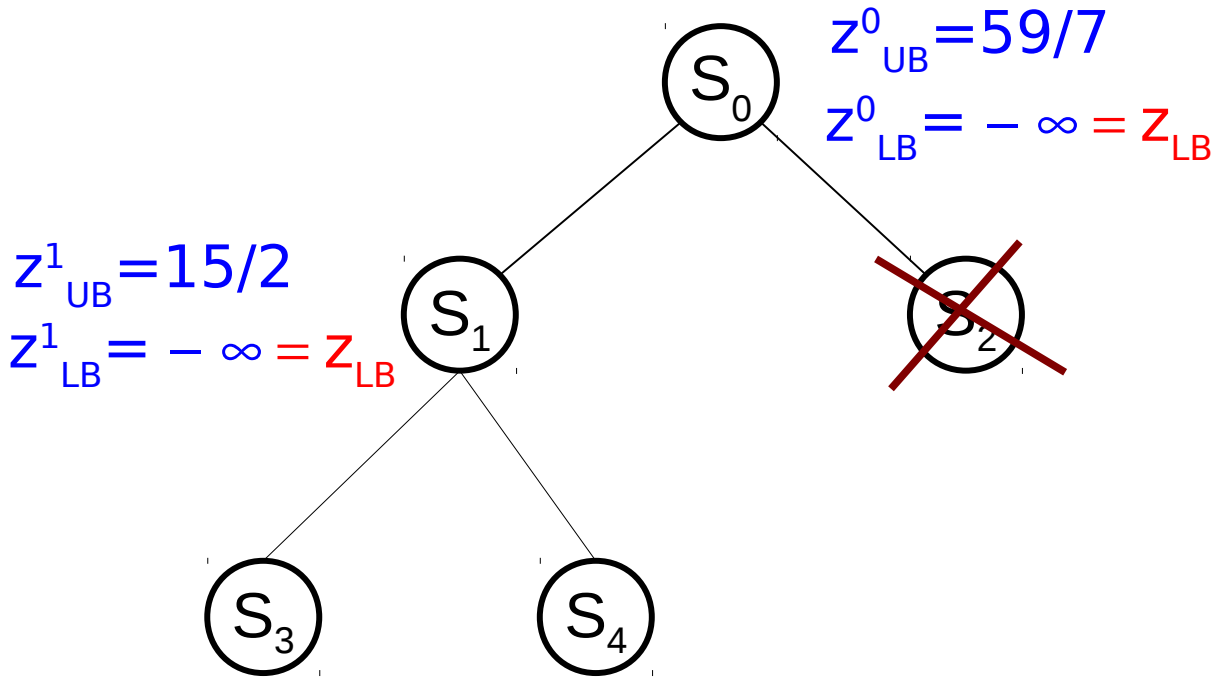
Esempio



Il nodo S_2 può essere chiuso per inammissibilità.

Esploriamo il nodo S_1

Esempio

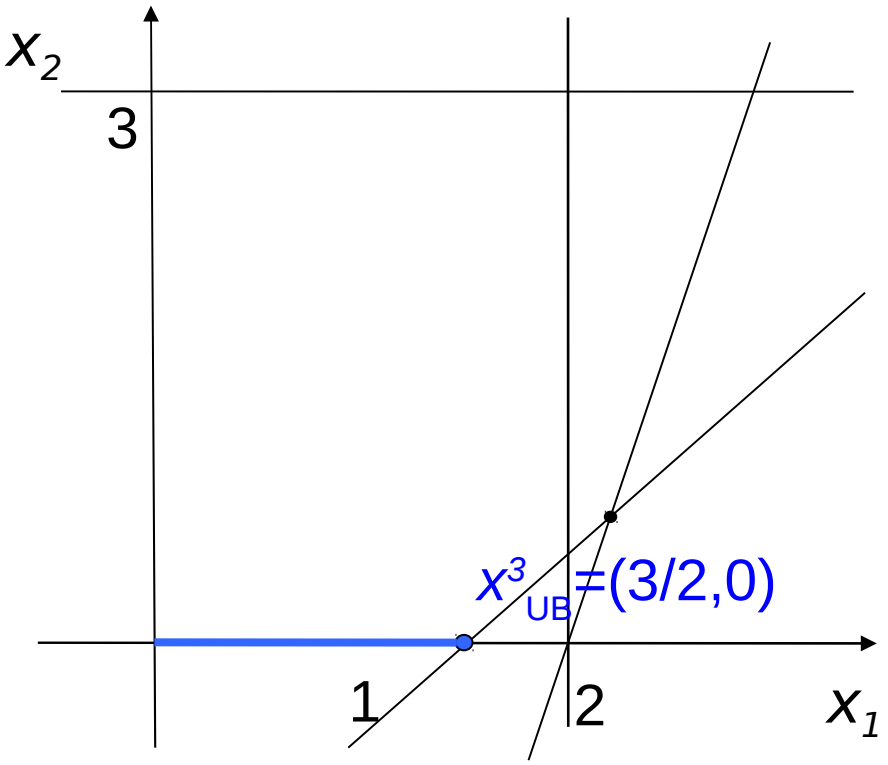


$$S_3 = S_1 \cap \{x : x_2 \leq 0\} \qquad S_4 = S_1 \cap \{x : x_2 \geq 1\}$$

Memorizziamo S_3 e S_4 nella lista di sottoproblemi che devono essere esplorati: $L = \{S_3, S_4\}$. Scegliamo di analizzare il sottoproblema S_3.

Esempio

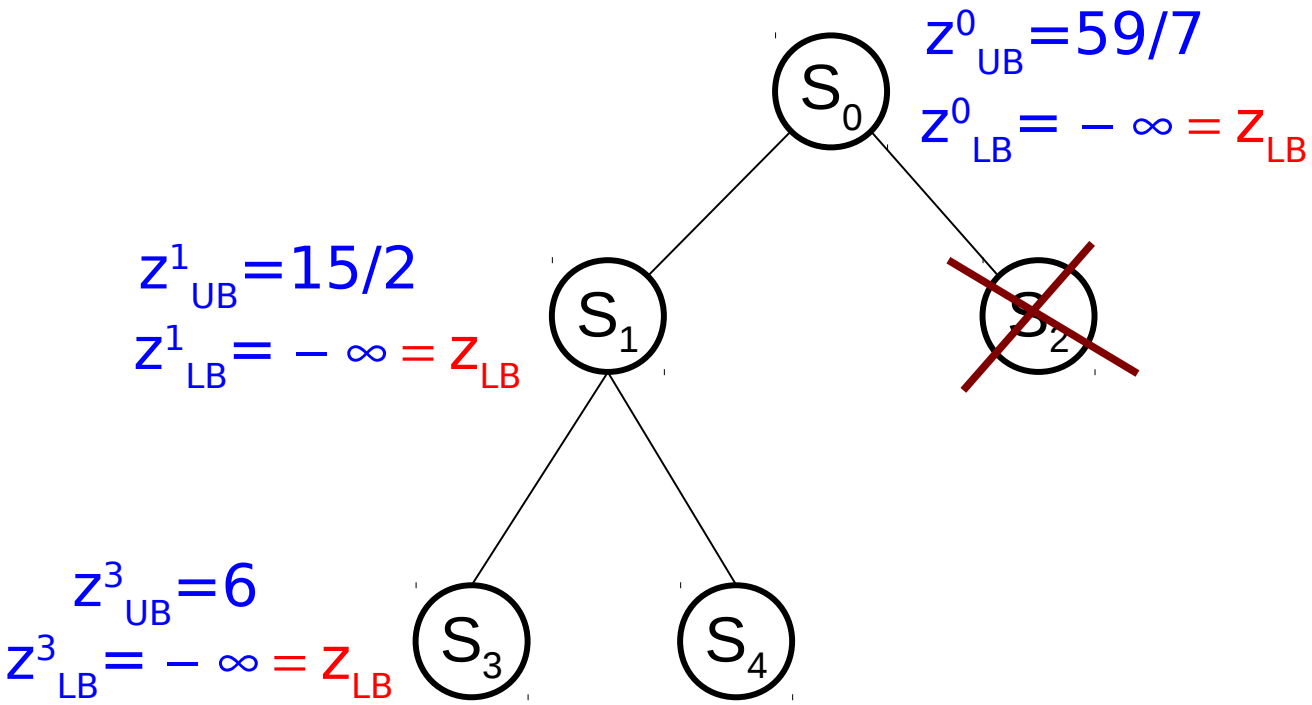
Risolviamo il sottoproblema al nodo S_3.



$$\max\ 4x_1 - x_2$$
$$7x_1 - 2x_2 \le 14$$
$$2x_1 - 2x_2 \le 3$$
$$x_1 \le 2$$
$$x_2 \le 0$$
$$x_1 \ge 0$$
$$x_2 \ge 0$$

Valore dell'UB per il sottoproblema S_3: $z^3_{UB} = 6$.

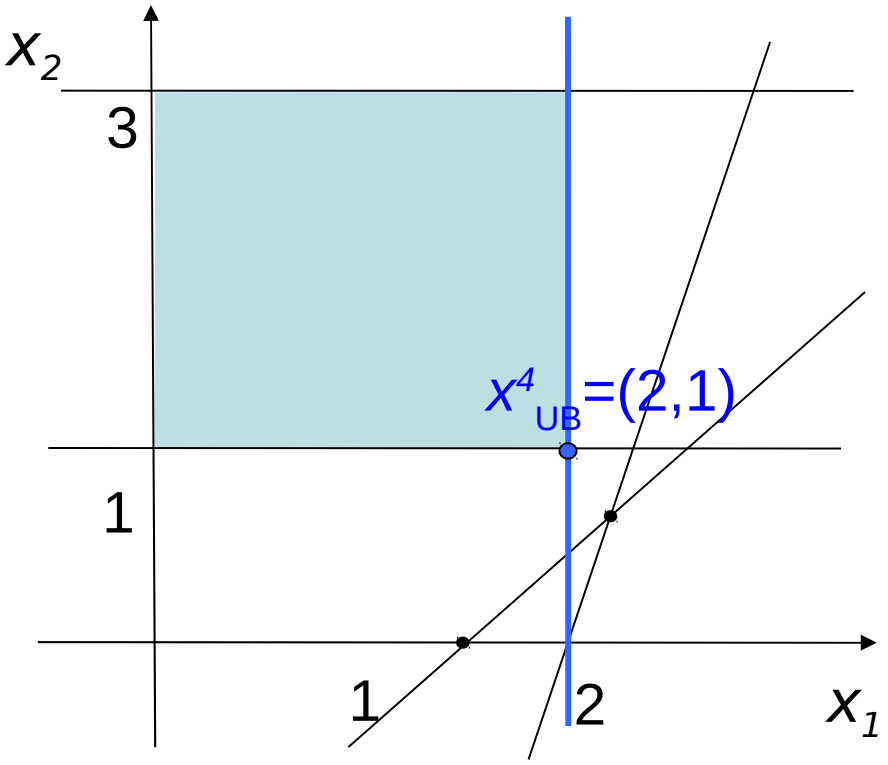
Esempio



Poiché in questo momento non possiamo applicare nessun criterio di potatura, il nodo S_3 dovrà essere esplorato.

Esempio

Risolviamo il sottoproblema al nodo S_4.

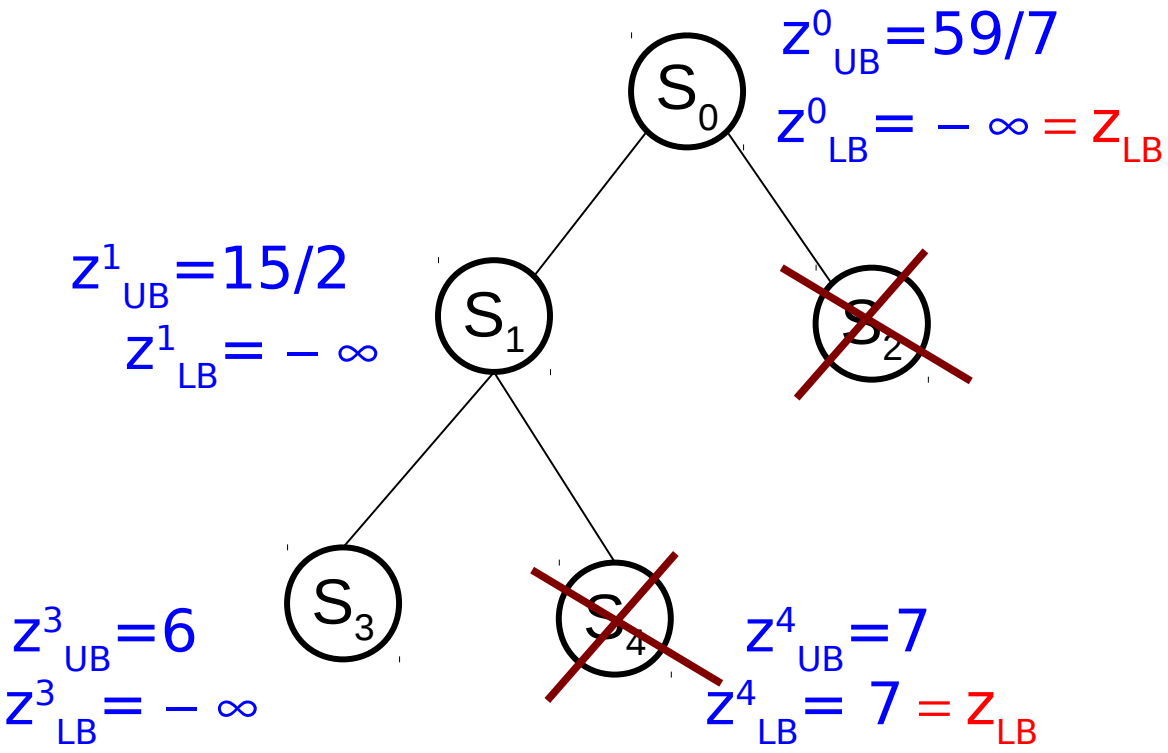


$$\max\ 4x_1 - x_2$$
$$7x_1 - 2x_2 \le 14$$
$$x_2 \le 3$$
$$2x_1 - 2x_2 \le 3$$
$$x_1 \le 2$$
$$x_2 \ge 1$$
$$x_1, x_2 \ge 0$$

Valore dell'UB per il sottoproblema S_4: $z^4_{UB} = 7$.

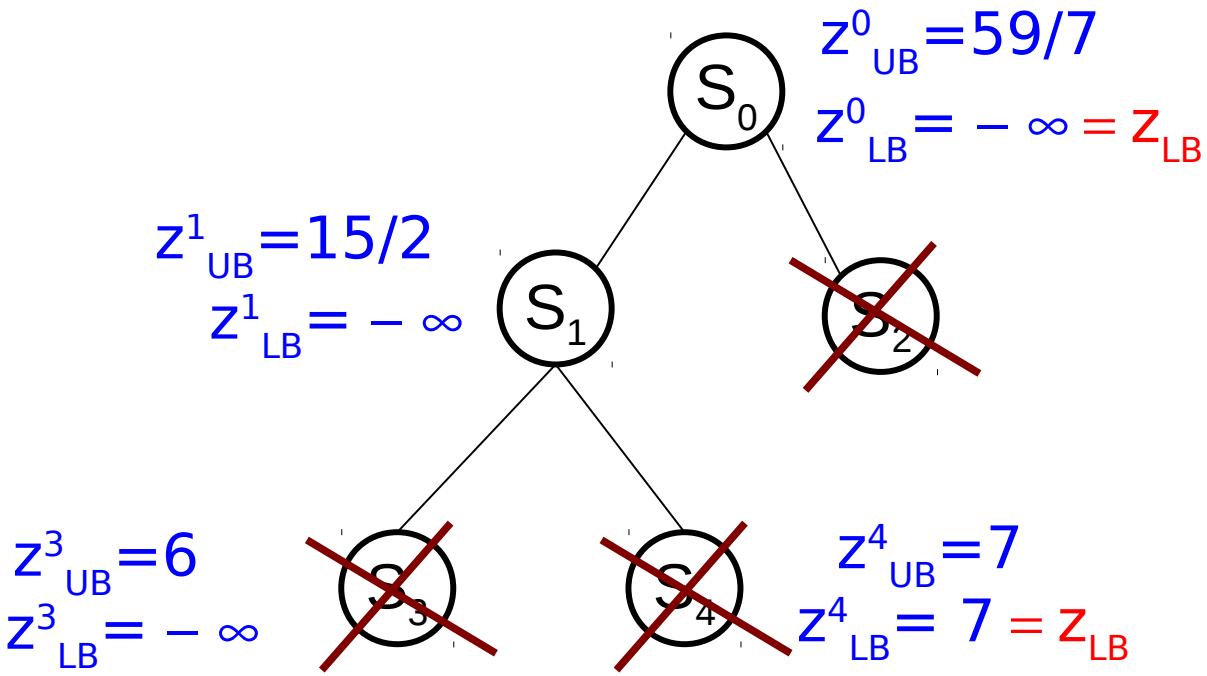
Osserviamo che la soluzione trovata è intera! Possiamo aggiornare il valore del lower bound globale!

Esempio



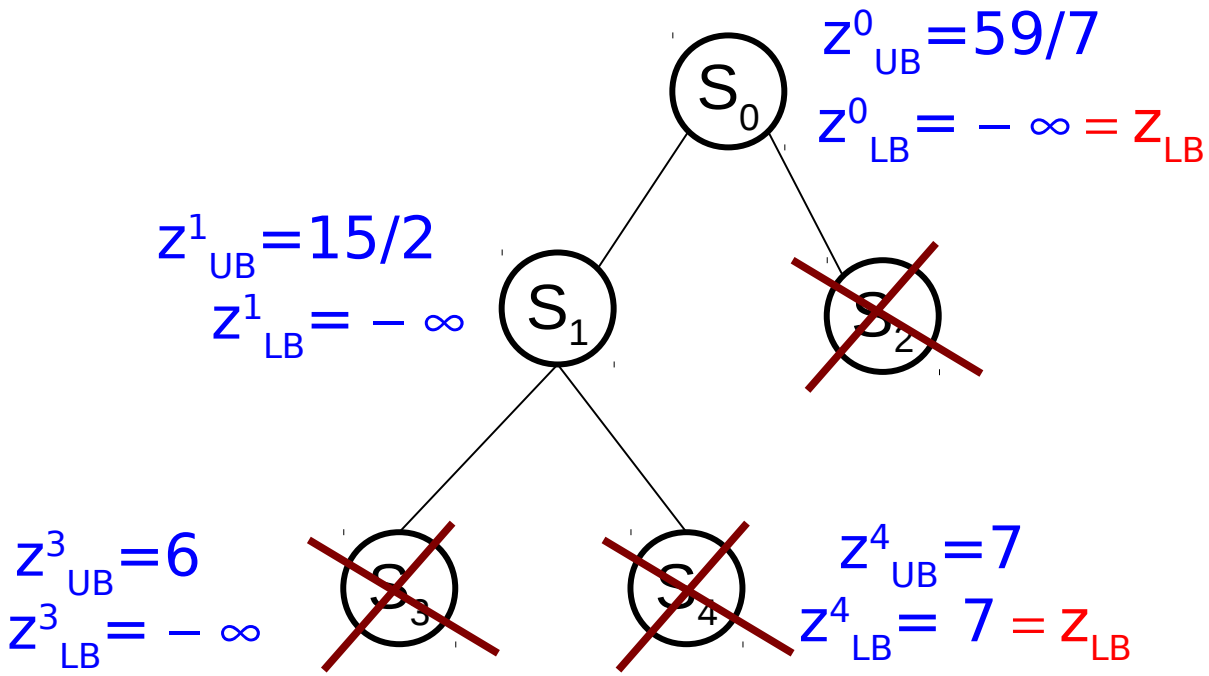
Il nodo S_4 può essere chiuso per ottimalità.

Esempio



In seguito all'aggiornamento del valore del lower bound possiamo chiudere anche il nodo S_3 per bound. Infatti: $z^3_{UB} = 6 < z^4_{LB}$.

Esempio



Poiché non ci sono altri nodi da esplorare nella lista *L*, l'algoritmo di Branch & Bound termina con la soluzione ottima intera $x^* = (2, 1)$ di valore $z^* = 7$.

Procedura di Branch-and-Bound

Tramite l'albero di enumerazione, enumeriamo un insieme di soluzioni "implicitamente" perché alcuni rami dell'albero vengono potati per:

- ottimalità, oppure
- bound, oppure
- inammissibilità.

Nella procedura di Branch & Bound è importante

- Scegliere la strategia per il calcolo del bound duale.
- Scegliere la variabile su cui applicare la procedura di branching.
- Scegliere la strategia di esplorazione dell'albero di enumerazione.

Procedura di Branch-and-Bound

Scelta della strategia di esplorazione dell'albero di enumerazione

VISITA DEL NODO CON MAX z_{UB}

(Vantaggio: limita il numero di nodi visitati

Svantaggio: si impiega più tempo per generare soluzioni ammissibili.)

VISITA IN PROFONDITA'

(Vantaggi: genera rapidamente una soluzione ammissibile, limita la memoria necessaria per memorizzare l'albero delle soluzioni

Svantaggio: rischio di esplorare completamente sotto-alberi con soluzioni scadenti)

Procedura di Branch-and-Bound

Siano

- P_0 = problema di ottimizzazione iniziale associato al nodo S_0.
- L = lista dei sottoproblemi che devono essere risolti
- P_i = formulazione del sottoproblema associato al nodo S_i.

L'algoritmo di Branch-and-Bound può essere schematizzato come segue.

Procedura di Branch-and-Bound

Step 0. **Inizializzazione**: $L = S_0$; $z_{LB} = -\infty(\max)$; $x^* = 0$.

Step 1. **Criterio di arresto**: Se $L = \varnothing$, allora STOP: x^* è la soluzione ottima.

Step 2. **Selezione del nodo**: Seleziona il sottoproblema P_i (nodo S_i) da esplorare dalla lista L.

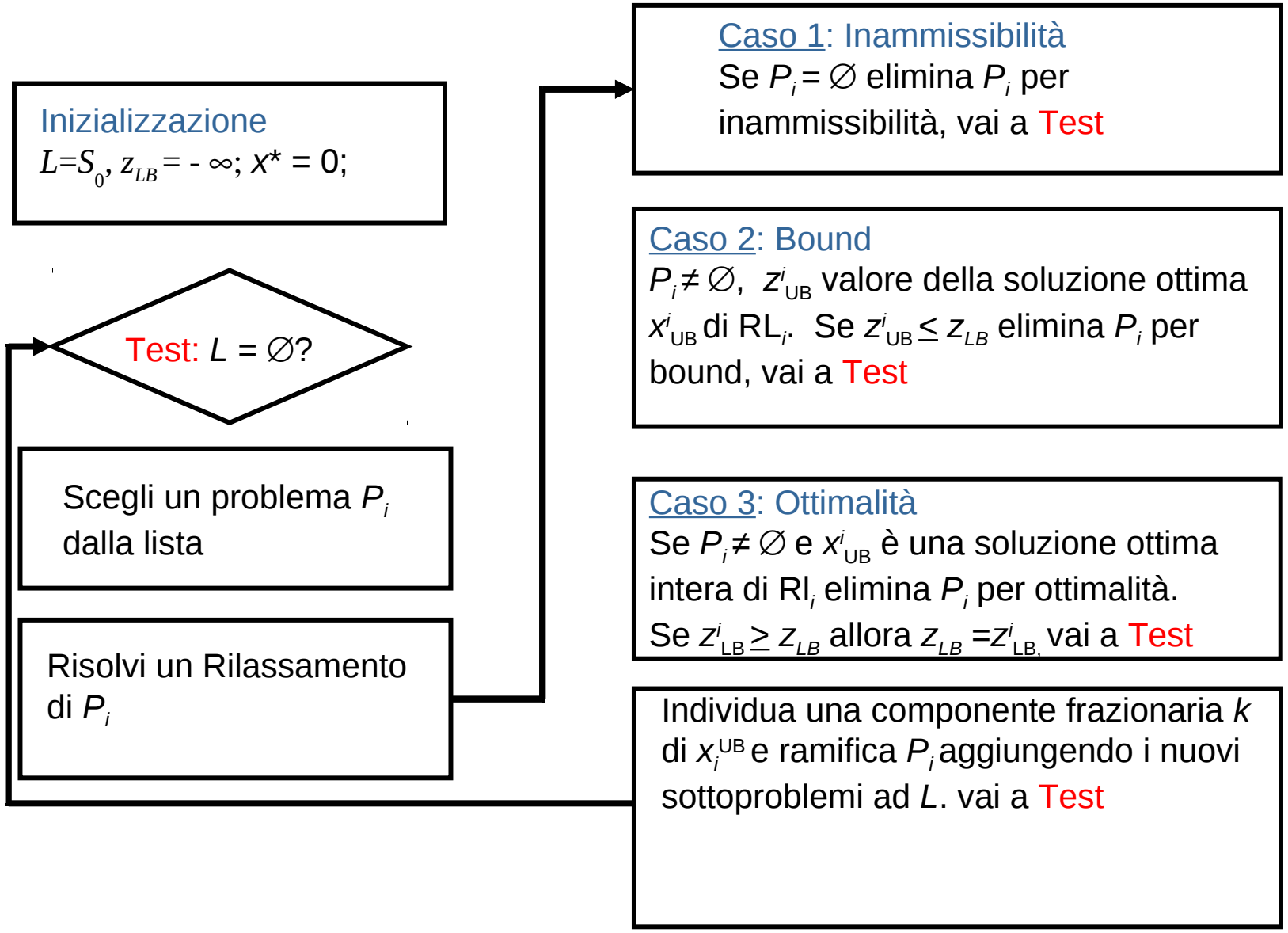
Step 3. **Bounding**: Risolvi un rilassamento di P_i.

Step 4. **Fathoming**:

- Se P_i non è ammissibile, elimina P_i da L e vai allo Step 1.
- Se $z^i_{UB} \leq z_{LB}$, elimina P_i da L e vai allo Step 1.
- Se x^i_{UB} è intera, elimina P_i da L. Se, inoltre, $z^i_{LB} > z_{LB}$, aggiorna z_{LB} e vai allo Step 1.

Step 5. **Branching**: Individua una componente frazionaria di x^i_{UB} e genera i sottoproblemi di P_i da aggiungere alla lista L. *count* = *count*+1. Vai allo Step 1.

Branch-and-Bound



Esempio

Consideriamo il problema di knapsack

$$\max\ 30x_1 + 36x_2 + 15x_3 + 11x_4 + 5x_5 + 3x_6$$
$$9x_1 + 12x_2 + 6x_3 + 5x_4 + 3x_5 + 2x_6 \leq 17 \qquad (1)$$
$$x \in \{0,1\}^6$$

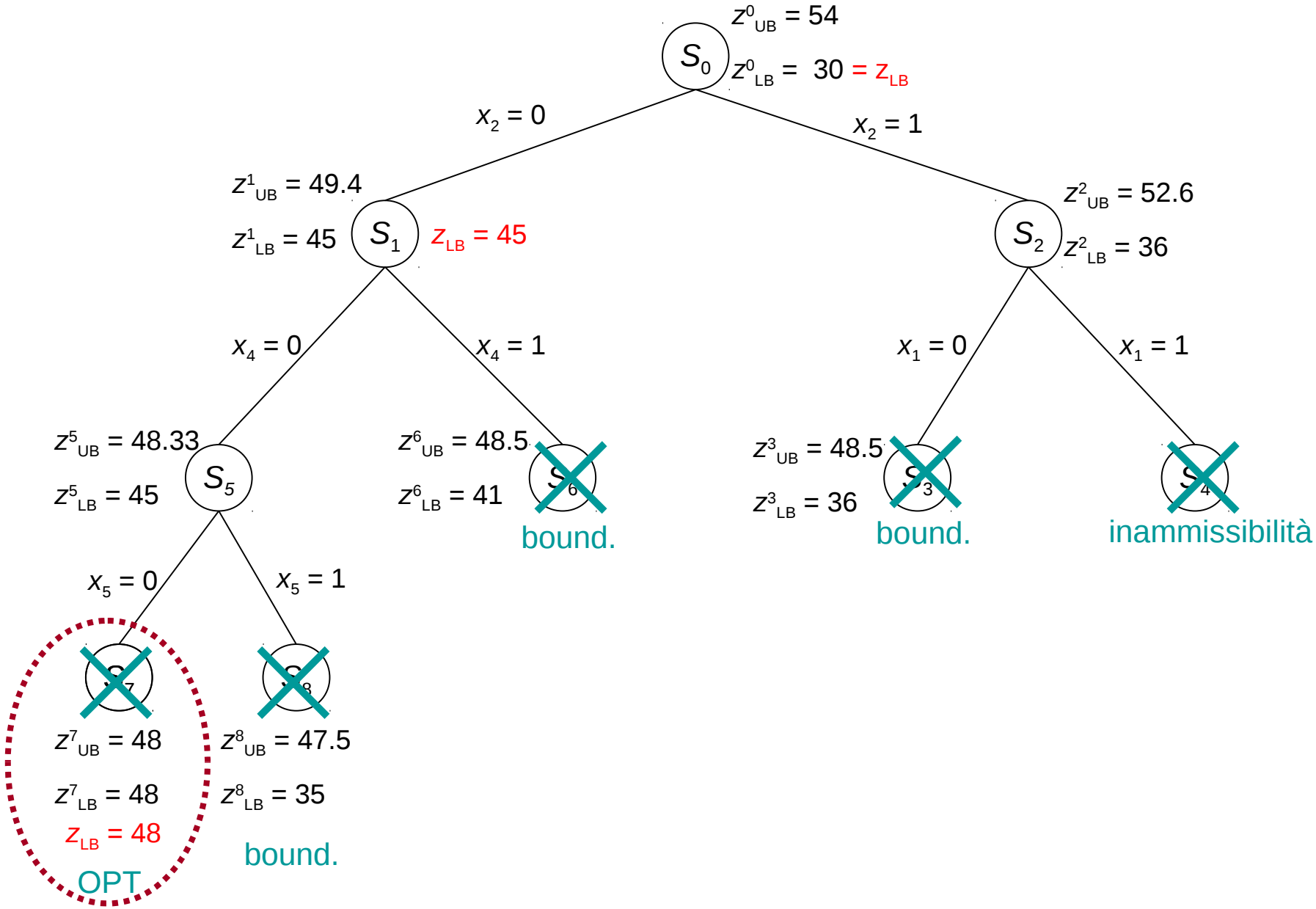
La soluzione ottima del Knapsack continuo è $x^0_{UB} = (1, 2/3, 0, 0, 0, 0)$ di valore 54 (z^0_{UB})

Inizializziamo il valore del lower bound uguale al valore di una soluzione ammissibile del problema $x^0_{LB} = (1, 0, 0, 0, 0, 0)$ di valore 30 (z^0_{LB}).

Inizializziamo il valore del lower bound globale $z_{LB} = 30$.

Osservazione: Poiché la funzione obiettivo ha coefficienti interi, possiamo scrivere la condizione di potatura per bound (caso 2) come $\lfloor z^i_{UB} \rfloor \leq z_{LB}$.

Esercizio 1


$z^0_{UB} = 54$
S_0
$z^0_{LB} = 30 = z_{LB}$
$x_2 = 0$
$x_2 = 1$
$z^1_{UB} = 49.4$
$z^1_{LB} = 45$
S_1
$z_{LB} = 45$
$z^2_{UB} = 52.6$
S_2
$z^2_{LB} = 36$
$x_4 = 0$
$x_4 = 1$
$x_1 = 0$
$x_1 = 1$
$z^5_{UB} = 48.33$
$z^5_{LB} = 45$
S_5
$z^6_{UB} = 48.5$
$z^6_{LB} = 41$
S_6
bound.
$z^3_{UB} = 48.5$
$z^3_{LB} = 36$
S_3
bound.
S_4
inammissibilità
$x_5 = 0$
$x_5 = 1$
S_7
S_8
$z^7_{UB} = 48$
$z^7_{LB} = 48$
$z_{LB} = 48$
OPT
$z^8_{UB} = 47.5$
$z^8_{LB} = 35$
bound.

Algoritmo di programmazione dinamica

Programmazione Dinamica

Schema di principio: Sia *P* un problema di OC. La Programmazione Dinamica (PD):

- risolve un sottoproblema di *P* all'ottimo.
- iterativamente "estende" la soluzione a *P.*

Proprietà fondamentale (optimal substructure)

Sia KP (N, b) un problema di knapsack con soluzione ottima x^*.

Consideriamo il problema di knapsack KP($N \setminus r$, $b - a_r$) che si ottiene da KP eliminando un qualsiasi oggetto r e diminuendo la capacità dello zaino della quantità a_r.

La soluzione ottima di KP(r, $b-a_r$) si ottiene da x^* semplicemente eliminando la variabile x_r.

Esempio

Consideriamo il seguente problema di knapsack:

$$\max 6x_1 + 10x_2 + 12x_3$$

$$x_1 + 2x_2 + 3x_3 \leq 5 \qquad (1)$$

$$x \in \{0,1\}^3$$

La soluzione ottima è $x^{*(1)} = (0, 1, 1)$, di valore 22.

Consideriamo il problema che si ottiene eliminando l'oggetto 2 e diminuendo la capacità b del corrispondente ingombro a_2:

$$\max 6x_1 + 12x_3$$

$$x_1 + 3x_3 \leq 5 - 2 = 3 \qquad (2)$$

$$x \in \{0,1\}^2$$

La soluzione ottima è $x^{*(2)}=(0, 1)$, di valore 12.

Osservazione: La soluzione $x^{*(2)}= (0, 1)$ è una "sottosoluzione" di $x^{*(1)} = (0, 1, 1)$, ovvero si ottiene da $x^{*(1)}$ semplicemente eliminando la variabile x_2.

Notazione

Siano r e d due interi tali che $1 \leq r \leq n$, $0 \leq d \leq b$.

Sia KP (r, d) il problema di knapsack:

$$\max \sum_{j=1}^{r} p_j x_j$$
$$\sum_{j=1}^{r} a_j x_j \leq d$$
$$x \in \{0, 1\}^r$$

KP (r, d) è un sottoproblema di KP, avente soluzione ottima di valore $z_r(d)$.

Programmazione dinamica

Con questo formalismo, il valore della soluzione ottima di KP vale $z_n(b)$.

Calcolo di $z_n(b)$:

1. Calcolo $z_r(d)$ per $r = \{1, \ldots, n\}$.
2. Per ogni r, calcolo $z_r(d)$ per $d = \{0, \ldots, b\}$.

Osservazione:

$z_r(d)$ si calcola in modo ricorsivo se conosco i valori $z_{r-1}(d)$ per $d = \{0, \ldots, b\}$.

Formula ricorsiva

Condizione iniziale di ricorsione:

$$z_1(d) = \begin{cases} 0 & per \quad d < a_1 \\ p_1 & per \quad d \geq a_1 \end{cases}$$

Questa condizione implica:

- Se $z_1(d) = 0 \Rightarrow x_1 = 0$
- Se $z_1(d) = p_1 \Rightarrow x_1 = 1$

Formula ricorsiva

Formula di ricorsione:

$$z_r(d) = \begin{cases} z_{r-1}(d) & \text{se } d < a_r \\ \max\{z_{r-1}(d),\ z_{r-1}(d-a_r)+p_r\} & \text{se } d \geq a_r \end{cases}$$

La formula implica

- Se $z_r(d) = z_{r-1}(d) \Rightarrow x_r = 0$ [l'oggetto r NON è stato scelto]
- Se $z_r(d) = z_{r-1}(d-a_r)+c_r \Rightarrow x_r = 1$ [l'oggetto r E' stato scelto]

Algoritmo e complessità

$DP\text{-}KP(n, b, a, c)$
$for \quad d = 0 \quad to \quad a_1 - 1$
$\quad z_1(d) = 0;$
$for \quad d = a_1 \quad to \quad b$
$\quad z_1(d) = p_1;$
$for \quad m = 2 \quad to \quad n$
$\quad for \quad d = 0 \quad to \quad a_m - 1$
$\quad z_m(d) = z_{m-1}(d)$
$\quad for \quad d = a_m \quad to \quad b$
$\quad if \; (z_{m-1}(d) > z_{m-1}(d - a_m) + p_m)$
$\quad then \; \max = z_{m-1}(d);$
$\quad else \; \max = z_{m-1}(\hat{b} - a_m) + p_m;$
$\quad z_m(d) = \max;$
$return \quad z_n(b)$

Osservazione: La complessità dell'algoritmo è O(*nb*). Poiché dipende dall'intero *b* si dice che l'algoritmo ha complessità pseudo polinomiale.