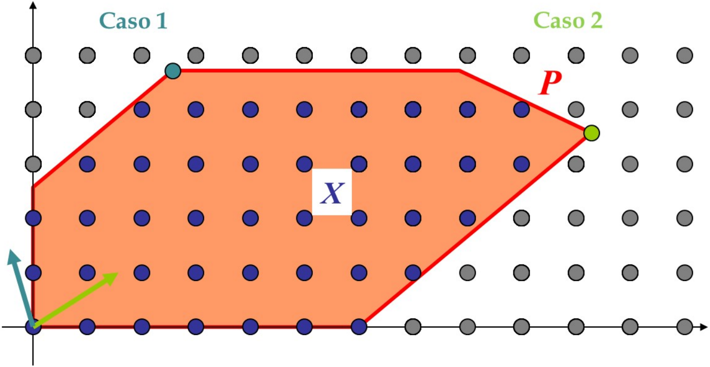
Branch-and-bound per problemi di programmazione lineare intera

Luigi De Giovanni

Dipartimento di Matematica, Università di Padova

Programmazione Lineare Intera

$$
\begin{array}{ll}
\min / \max & c^T x \\
\text{s.t.} & Ax = b \\
& x \in \mathbb{Z}_+^n
\end{array}
$$



Branch-and-Bound (B&B) per PLI: Fatto 1

Problema di PLI(M)

$$
\begin{aligned}
z_I = \max/\min\ & c^T x \\
Ax &\leq b \\
x &\geq 0 \\
x_i &\in \mathbb{Z}, \qquad i \in I.
\end{aligned}
\tag{1}
$$

Rilassamento continuo (o lineare)

$$
\begin{aligned}
z_L = \max/\min\ & c^T x \\
Ax &\leq b \\
x &\geq 0
\end{aligned}
\tag{2}
$$

Problema max:	$z_L \geq z_I$	z_L è un *Upper Bound (UB)*!
Problema min:	$z_L \leq z_I$	z_L è un *Lower Bound (UB)*!

In generale, z_L è **stima ottimistica** *di z_I*

Branch-and-Bound (B&B) per PLI: Fatto 2

Principio divide et impera

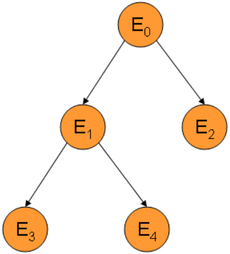
$z = \text{opt}\{f(x) : x \in X\}$

$X = \bigcup_{i=1}^{n} X_i = X$

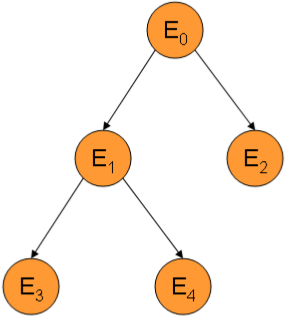
$z^{(k)} = \text{opt}\{f(x) : x \in X_k\}$

$$z = \text{opt}\ \{\ z^{(k)},\ k = 1, \ldots, n\ \}$$

Applicazione ricorsiva tramite **branching**: la regione ammissibile di un sottoproblema E_i è ulteriormente suddivisa, generando una struttura ad *albero*



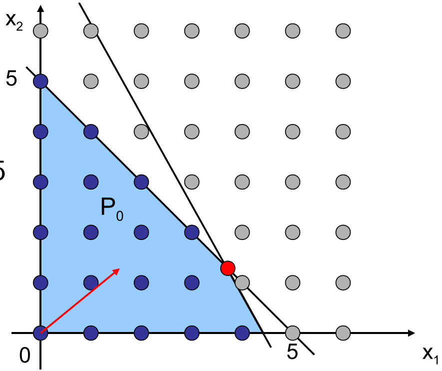
Branching: regole base



- $E_0 = X$
- $\boxed{E_i = \bigcup_{j \text{ figlio di } i} E_j}$
- preferibilmente $E_j \cap E_k = \emptyset, \ \forall\, j, k$ figli di i

Esempio: applicazione dei *fatti* al problema P_0

$$
(P_0) \qquad \begin{array}{rrcl}
z_I^0 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
$$



Rilassamento lineare: $x_1 = 3.75$, $x_2 = 1.75$, con valore $z_L^0 = 24.06$

Branch da P_0 su variabile frazionaria $x_1 = 3.75$

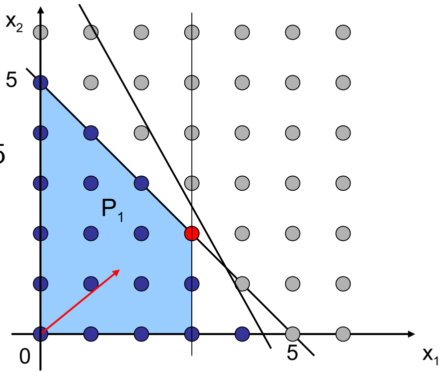
$$
\begin{array}{rrcl}
z_I^1 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& x_1 & \leq & 3 \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
\qquad (P_1)
$$

$$
\begin{array}{rrcl}
z_I^2 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& x_1 & \geq & 4 \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
\qquad (P_2)
$$

- Non perdo soluzioni intere: $E_1 \cup E_2 = E_0$, $z_I = \max\{z_I^1, z_I^2\}$
- z_L^0 esclusa!

Problema P_1

$$
(P_1) \quad
\begin{array}{rrcl}
z_I^1 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& \color{red}{x_1} & \color{red}{\leq} & \color{red}{3} \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
$$



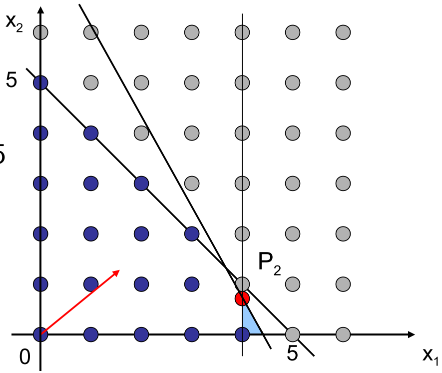
Rilassamento lineare: $x_1 = 3$, $x_2 = 2$, con valore $z_L^1 = 23.5$

Soluzione intera (rilassamento ammissibile): nodo potato per **S.A.**

aggiornamento incumbent, $\bar{z} = 23.5$

Problema P_2

$$
(P_2) \quad \begin{array}{rrcl}
z_I^1 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
 & x_1 + x_2 & \leq & 5 \\
 & 10x_1 + 6x_2 & \leq & 45 \\
 & \color{red}{x_1} & \color{red}{\geq} & \color{red}{4} \\
 & x_1, x_2 & \geq & 0 \\
 & x_1, x_2 & \in & \mathbb{Z}
\end{array}
$$



Rilassamento lineare: $x_1 = 4$, $x_2 = 0.83$, con valore $z_L^2 = 23.54$

Branch da P_2 su variabile frazionaria $x_2 = 0.83$

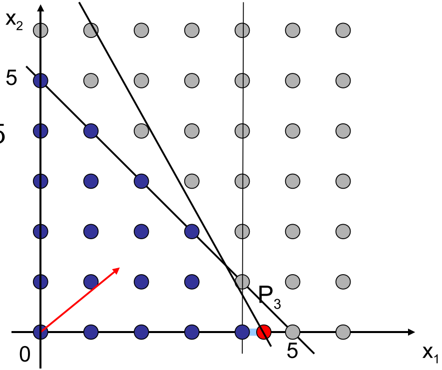
$$
\begin{array}{rrcl}
z_I^1 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& {\color{red}x_1} & {\color{red}\geq} & {\color{red}4} \\
& {\color{red}x_2} & {\color{red}\leq} & {\color{red}0} \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
\qquad (P_3)
$$

$$
\begin{array}{rrcl}
z_I^2 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& {\color{red}x_1} & {\color{red}\geq} & {\color{red}4} \\
& {\color{red}x_2} & {\color{red}\geq} & {\color{red}1} \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
\qquad (P_4)
$$

$$E_3 \cup E_4 = E_2$$

Problema P_3

$$
(P_2) \qquad
\begin{array}{rrcl}
z_I^1 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
 & x_1 + x_2 & \leq & 5 \\
 & 10x_1 + 6x_2 & \leq & 45 \\
 & \color{red}{x_1} & \color{red}{\geq} & \color{red}{4} \\
 & \color{red}{x_2} & \color{red}{\leq} & \color{red}{0} \\
 & x_1, x_2 & \geq & 0 \\
 & x_1, x_2 & \in & \mathbb{Z}
\end{array}
$$

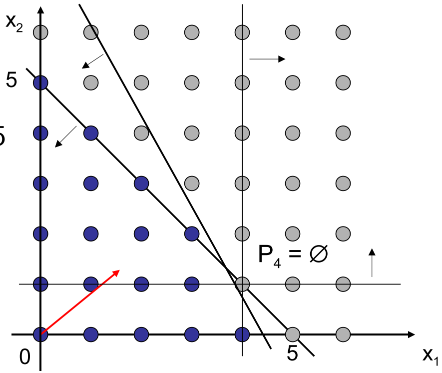


Rilassamento lineare: $x_1 = 4.5$, $x_2 = 0$, con valore $z_L^3 = 22.5$

$z_L^3 \leq \bar{z}$: nodo potato per **N.M.**

Problema P_4

$$
(P_2)\quad
\begin{array}{rrcl}
z_I^1 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
 & x_1 + x_2 & \leq & 5 \\
 & 10x_1 + 6x_2 & \leq & 45 \\
 & {\color{red}x_1} & {\color{red}\geq} & {\color{red}4} \\
 & {\color{red}x_2} & {\color{red}\geq} & {\color{red}1} \\
 & x_1, x_2 & \geq & 0 \\
 & x_1, x_2 & \in & \mathbb{Z}
\end{array}
$$

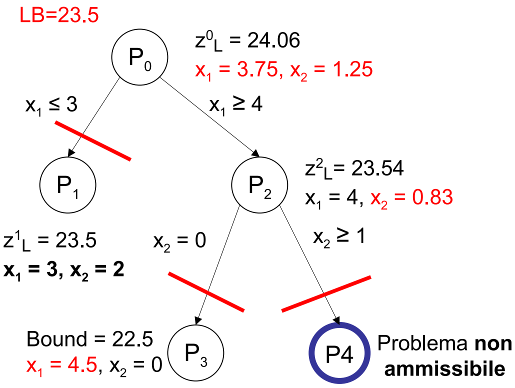


Rilassamento lineare: non ammissibile.

Anche (P_4) non è ammissibile: nodo potato per **N.A.**

Tutti i nodi fathomed: $\bar{x} = (2, 3)$ con $\bar{z} = 23.5$ ottima!

Albero di branch-and-bound



Branch-and-bound: idea base

- **Branch**: costruzione dell'albero delle soluzioni (enumerazione ricorsiva)
  - Nel caso peggiore, genera tutte le "foglie", corrispondenti alle singole soluzione intere nella regione ammissibile (i vincoli determinano univocamente ciascun valore)
- **Soluzione ammissibile** (incumbent solution): valore possibile, ma non dimostrabilmente ottimo
- **Bound**: valutazione ottimistica della funzione obiettivo per le soluzioni associate ad un nodo (sottoalbero)
- **Fathom**: se il bound di un nodo non è migliore dell'incumbent, il relativo sottoalbero si può potare

Enumerazione implicita dello spazio delle soluzioni

Metodo del **Branch-and-Bound (B&B)** per PLIM

Inizializzazione: Risolvi rilassamento ottenendo x_0^R e stima ottimistica B_0
e poni $L = \{(P_0, B_0)\}$, $\bar{x} = \emptyset$, $\bar{z} = +\infty(\min)[-\infty(\max)]$

`Repeat`:

Criterio di Stop: Se $L = \emptyset$, allora `stop`: $\bar{x}$ è la soluzione ottima.

Selezione nodo: Seleziona ed estrai $(P_i, B_i) \in L$ per effettuare il branch

Branching: Dividi P_i in $P_{|L|+1}$ $(x_{ik} \leq \lfloor x_{ik}^R \rfloor)$ e $P_{|L|+2}$ $(x_{ik} \geq \lceil \hat{x}_{ik}^R \rceil)$

`For each` sottoproblema $P_j, j = |L|+1 ... |L|+2$:

Bounding: Risolvi rilassamento di P_j ottenendo stima ottimistica B_j
e soluzione x_j^R oppure inammissibilità

Fathoming: `If` P_j non è ammissibile: `continue`
`elseif` B_j non è migliore di $\bar{z}$: `continue`
`elseif` x_j^R è intera:
`if` x_j^R anche migliore di $\bar{z}$:
aggiorna $\bar{z} \leftarrow B_j$, $\bar{x} \leftarrow x_{ij}^R$
elimina da L tutti i nodi k con L_k non migliore di $\bar{z}$
`continue` (x_{ij}^R è ottima per P_j)

Ricorsione: `else` aggiungi (P_j, B_j) a L (B_j è più promettente di $\bar{z}$)

Esempio

Risolvere con il metodo del Branch-and-bound:

$$\begin{array}{ll} \max & 3\,x_1 - 8\,x_2 + 3\,x_3 - 8\,x_4 + 13\,x_5 \\ s.t. & -2\,x_1 + 7\,x_2 + 4\,x_3 + 1.5\,x_4 + 9\,x_5 \leq 16 \\ & -6\,x_1 + 5\,x_2 + 5\,x_3 + 7.2\,x_4 - 3\,x_5 \geq 7 \\ & x_1, \ldots, x_5 \in Z^+ \end{array}$$

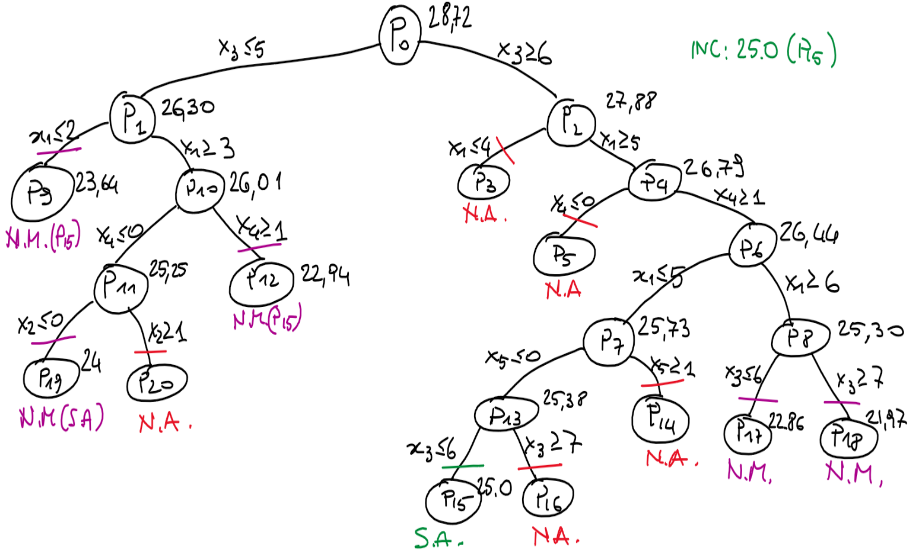
- Branching: binario su variabile "meno frazionaria"
- Bound: rilassamento continuo (usare AMPL!)
- Fathoming: standard
- Esplorazione: a piacere (Best Bound First)
- Stop: lista nodi aperti vuota

Esempio: soluzione

Nodi numerati nell'ordine di esplorazione (BBF)

Esempio: soluzione

Nodi numerati nell'ordine di esplorazione (BBF)



Esempio - miglioramenti

Si consideri il problema dell'Esempio. Quale sarebbe lo sviluppo dell'albero di B&B con le seguenti varianti:

- Variante A: provare a generare, ad ogni nodo, una soluzione ammissibile approssimando la soluzione frazionaria ottenuta con il rilassamento continuo

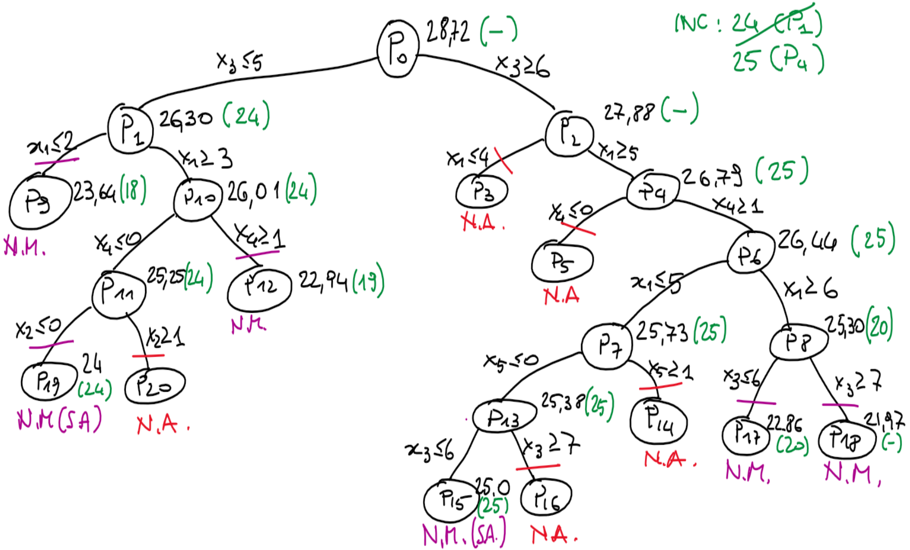
- Variante B: migliorare il bound osservando che tutti i coefficienti e tutte le variabili della funzione obiettivo, **nello specifico problema in esame**, sono interi, quindi il valore della fuzione obiettivo è intero.

Esempio con Variante A: soluzione

"n.d.": non disponibile (la soluzione arrotondata non è ammissibile)

Esempio con Variante A: soluzione

"n.d.": non disponibile (la soluzione arrotondata non è ammissibile)

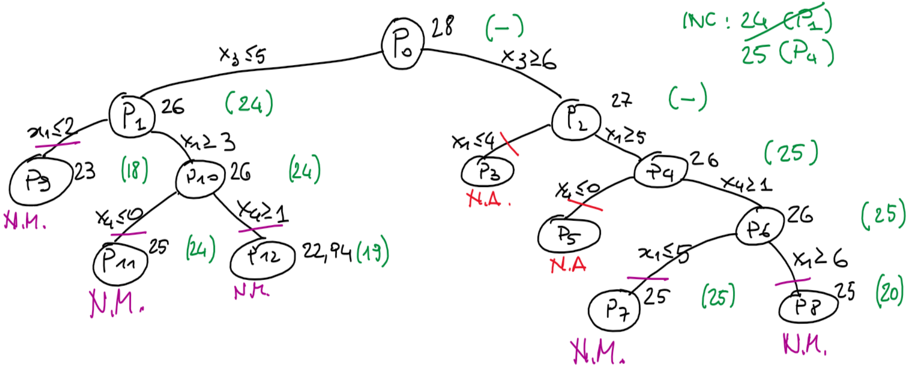


Esempio con Varianti A e B: soluzione

"n.d.": non disponibile (la soluzione arrotondata non è ammissibile)

Esempio con Varianti A e B: soluzione

"n.d.": non disponibile (la soluzione arrotondata non è ammissibile)



Esempio "PLI generico": osservazioni

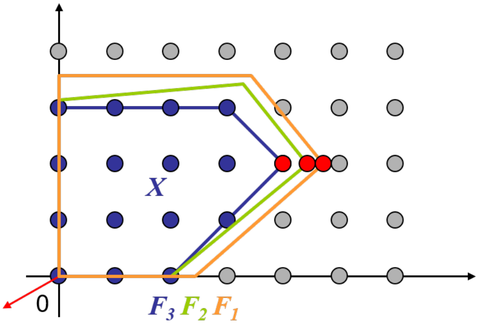
- variante A: si ottengono diverse soluzioni ammissibili, che permettono rapidamente di aggiornare l'incumbent prima al valore 24 (nodo P_1) e poi al valore 25 (noto P_4). Seguendo l'esplorazione BBF, alcuni nodi (ad esempio P_9) vengono chiusi prima rispetto al precedente albero (con un piccolo risparmio di memoria utilizzata), tuttavia non si risparmia in termini i nodi complessivamente valutati.

- variante A+B: in questo caso, tutti i bound ottenuti con il rilassamento continuo possono essere ulteriormente approssimati all'intero **inferiore** (problema di massimo), permettendo, grazie all'incumbent ottenuta con l'arrotondamento al non P_4, di chiudere subito anche i nodi P_7, P_8 e P_{11} come non miglioranti.

B&B per PLI: scelte progettuali (cenni)

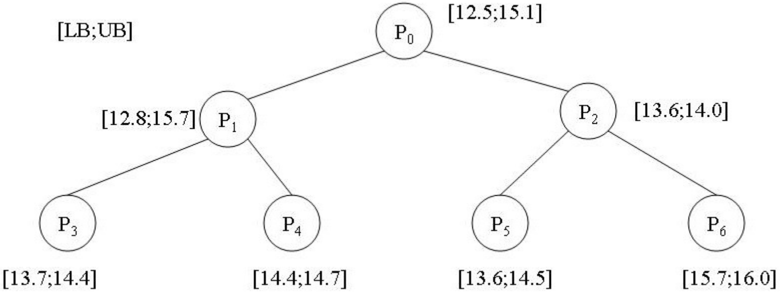
- **Bound** con rilassamento continuo
  - sfruttare proprietà del problema allo studio, formulazioni più stringenti*
- **Branch** binario su una variabile frazionaria
  - scelgo, e.g., "più" frazionaria, "più" intera, *diving* etc.)
  - possibile branching t-ario se pochi valori alternativi
- Fathoming standard
  - [N.M.] Assenza di soluzione migliorante (B_i non migliora $f(\bar{x})$)
  - [S.A.] Valutazione ottimistica (rilassaè anche di soluzione ammissibile
  - [N.A.] Sottoproblema (rilassamento) non ammissibile
- Strategie di esplorazione: *Depth First*, *Best Bound First*, *Mista*, diving etc.
- Valutazione di soluzioni ammissibili
  - euristiche (o meta-euristiche) ad-hoc prima del branch-and-bound
  - *rounding heuristic* sulla soluzione frazionaria ad ogni nodo (o sotto particolari condizioni)
- Arresto standard (tutti nodi fathomed, $\bar{x}$ ottima), oppure max time ($\bar{x}$ potrebbe non essere ottima), *optimality gap* entro soglia etc.

*Esempio: bound e formulazioni alternative per PLIM



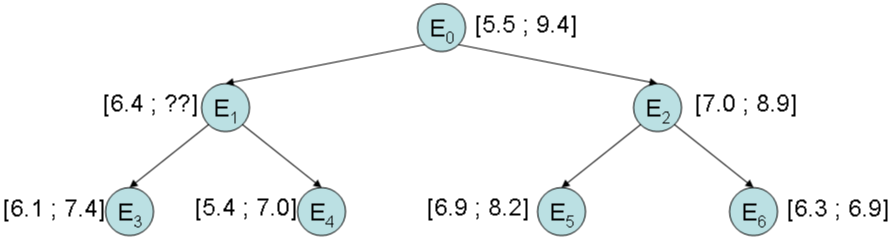
- F_2 è *migliore* di F_1: fornisce bound più stringenti (più vicini all'ottimo): UB più bassi (per problemi di max) o LB più alti (per problemi di min)
- F_3 è la *formulazione ideale*: permette di risolvere il problema al nodo radice (senza branching)

Esercizio



- min o max?
- nodi da chiudere?
- intervallo ottimo?
- best bound first?
- LB e UB per chiudere...

Esercizio



- min o max? valore '??'?
- intervallo ottimo?
- nodi da chiudere?
- best bound first?
- LB e UB per chiudere...

B&B: algoritmo *generale* di ottimizzazione combinatoria

- **Regole di branching**: strategia per costituire sottoproblemi sempre più semplici (al limite una soluzione: converge!)
  – $E_i : \cup_i E_i = E$ (must!) [e $E_i \cap E_j = \emptyset$ opzionale]
- **Bound**: *lower* bound (min, LB) o *upper* bound (max, UB).
  – Valutazione **ottimistica**...: $LB \leq f(E_i)$ $\quad UB \geq f(E_i)$
  – ...ma non troppo! **efficienza** computazionale .vs. **qualità** bound
- **Regole di fathoming**: evito di esplorare nodo se
  – **[N.M.]** Assenza di soluzione migliorante (B_i non migliora $f(\bar{x})$)
  – **[S.A.]** Valutazione ottimistica è anche di soluzione ammissibile
  – **[N.A.]** Sottoproblema non ammissibile ($E_i = \emptyset$)
- **Strategie di esplorazione**: *Depth First*, *Best Bound First*, *Mista*
- **Valutazione di soluzioni ammissibili**: opzionale!
  – sforzo computazionale .vs. possibilità di potare nodi
- **Criteri di arresto**: tutti i nodi *fathomed* per garanzia di ottimalità (oppure criteri *euristici* con garanzia di performance)

B&B: algoritmo *generale* di ottimizzazione combinatoria

- **Regole di branching**: strategia per costituire sottoproblemi sempre più semplici (al limite una soluzione: converge!)
  – $E_i : \cup_i E_i = E$ (must!) [e $E_i \cap E_j = \emptyset$ opzionale]
- **Bound**: *lower* bound (min, LB) o *upper* bound (max, UB).
  – Valutazione **ottimistica**...: $LB \leq f(E_i) \quad UB \geq f(E_i)$
  – ...ma non troppo! **efficienza** computazionale .vs. **qualità** bound
- **Regole di fathoming**: evito di esplorare nodo se
  – **[N.M.]** Assenza di soluzione migliorante (B_i non migliora $f(\bar{x})$)
  – **[S.A.]** Valutazione ottimistica è anche di soluzione ammissibile
  – **[N.A.]** Sottoproblema non ammissibile ($E_i = \emptyset$)
- **Strategie di esplorazione**: *Depth First*, *Best Bound First*, *Mista*
- **Valutazione di soluzioni ammissibili**: opzionale!
  – sforzo computazionale .vs. possibilità di potare nodi
- **Criteri di arresto**: tutti i nodi *fathomed* per garanzia di ottimalità (oppure criteri *euristici* con garanzia di performance)

B&B: algoritmo *generale* di ottimizzazione combinatoria

- **Regole di branching**: strategia per costituire sottoproblemi sempre più semplici (al limite una soluzione: converge!)
  – $E_i : \cup_i E_i = E$ (must!) [e $E_i \cap E_j = \emptyset$ opzionale]
- **Bound**: *lower* bound (min, LB) o *upper* bound (max, UB).
  – Valutazione **ottimistica**...: $LB \leq f(E_i) \quad UB \geq f(E_i)$
  – ...ma non troppo! **efficienza** computazionale .vs. **qualità** bound
- **Regole di fathoming**: evito di esplorare nodo se
  – **[N.M.]** Assenza di soluzione migliorante (B_i non migliora $f(\bar{x})$)
  – **[S.A.]** Valutazione ottimistica è anche di soluzione ammissibile
  – **[N.A.]** Sottoproblema non ammissibile ($E_i = \emptyset$)
- **Strategie di esplorazione**: *Depth First*, *Best Bound First*, *Mista*
- **Valutazione di soluzioni ammissibili**: opzionale!
  – sforzo computazionale .vs. possibilità di potare nodi
- **Criteri di arresto**: tutti i nodi *fathomed* per garanzia di ottimalità (oppure criteri *euristici* con garanzia di performance)

B&B: algoritmo *generale* di ottimizzazione combinatoria

- **Regole di branching**: strategia per costituire sottoproblemi sempre più semplici (al limite una soluzione: converge!)
  – $E_i : \cup_i E_i = E$ (must!) [e $E_i \cap E_j = \emptyset$ opzionale]
- **Bound**: *lower* bound (min, LB) o *upper* bound (max, UB).
  – Valutazione **ottimistica**...: $LB \leq f(E_i) \quad UB \geq f(E_i)$
  – ...ma non troppo! **efficienza** computazionale .vs. **qualità** bound
- **Regole di fathoming**: evito di esplorare nodo se
  – **[N.M.]** Assenza di soluzione migliorante (B_i non migliora $f(\bar{x})$)
  – **[S.A.]** Valutazione ottimistica è anche di soluzione ammissibile
  – **[N.A.]** Sottoproblema non ammissibile ($E_i = \emptyset$)
- **Strategie di esplorazione**: *Depth First*, *Best Bound First*, *Mista*
- **Valutazione di soluzioni ammissibili**: opzionale!
  – sforzo computazionale .vs. possibilità di potare nodi
- **Criteri di arresto**: tutti i nodi *fathomed* per garanzia di ottimalità (oppure criteri *euristici* con garanzia di performance)

B&B: algoritmo *generale* di ottimizzazione combinatoria

- **Regole di branching**: strategia per costituire sottoproblemi sempre più semplici (al limite una soluzione: converge!)
  – $E_i : \cup_i E_i = E$ (must!) [e $E_i \cap E_j = \emptyset$ opzionale]
- **Bound**: *lower* bound (min, LB) o *upper* bound (max, UB).
  – Valutazione **ottimistica**...: $LB \leq f(E_i) \quad UB \geq f(E_i)$
  – ...ma non troppo! **efficienza** computazionale .vs. **qualità** bound
- **Regole di fathoming**: evito di esplorare nodo se
  – **[N.M.]** Assenza di soluzione migliorante (B_i non migliora $f(\bar{x})$)
  – **[S.A.]** Valutazione ottimistica è anche di soluzione ammissibile
  – **[N.A.]** Sottoproblema non ammissibile ($E_i = \emptyset$)
- **Strategie di esplorazione**: *Depth First*, *Best Bound First*, *Mista*
- **Valutazione di soluzioni ammissibili**: opzionale!
  – sforzo computazionale .vs. possibilità di potare nodi
- **Criteri di arresto**: tutti i nodi *fathomed* per garanzia di ottimalità (oppure criteri *euristici* con garanzia di performance)

B&B: algoritmo *generale* di ottimizzazione combinatoria

- **Regole di branching**: strategia per costituire sottoproblemi sempre più semplici (al limite una soluzione: converge!)
  – $E_i : \cup_i E_i = E$ (must!) [e $E_i \cap E_j = \emptyset$ opzionale]
- **Bound**: *lower* bound (min, LB) o *upper* bound (max, UB).
  – Valutazione **ottimistica**...: $LB \leq f(E_i) \quad UB \geq f(E_i)$
  – ...ma non troppo! **efficienza** computazionale .vs. **qualità** bound
- **Regole di fathoming**: evito di esplorare nodo se
  – **[N.M.]** Assenza di soluzione migliorante (B_i non migliora $f(\bar{x})$)
  – **[S.A.]** Valutazione ottimistica è anche di soluzione ammissibile
  – **[N.A.]** Sottoproblema non ammissibile ($E_i = \emptyset$)
- **Strategie di esplorazione**: *Depth First*, *Best Bound First*, *Mista*
- **Valutazione di soluzioni ammissibili**: opzionale!
  – sforzo computazionale .vs. possibilità di potare nodi
- **Criteri di arresto**: tutti i nodi *fathomed* per garanzia di ottimalità (oppure criteri *euristici* con garanzia di performance)

Esempio (dummy): scelta ottima di appalti

Una grossa azienda di costruzioni edili deve decidere la combinazione ottimale degli appalti da accettare per la costruzione degli edifici A, B e C.
I profitti attesi per i tre edifici sono di 3, 5 e 7 milioni di euro rispettivamente.
L'azienda dispone di 4 ruspe speciali e gli edifici richiedono risp. 3, 2 e 3 ruspe.
È possibile inoltre affittare fino a due altre ruspe speciali per la durata dei lavori, al costo di un milione di euro a ruspa.

Decisioni:

- accettare appalto i, $i \in \{A, B, C\}$. Possibili decisioni: sì/no.
- numero di ruspe da affittare. Possibili decisioni: 0, 1 o 2.

Possibili combinazioni: $2 \times 2 \times 2 \times 3 = 24$

Branch: scegliere una decisione (nell'ordine A-B-C-num.ruspe) e creare un sottoproblema per ogni valore

Bound: somma profitti di tutti gli appalti possibili meno costo ruspe "fissate" (valutazione imprecisa ma ottimistica e veloce, senza ragionamenti su ruspe "necessarie")

Esempio (dummy): scelta ottima di appalti

Una grossa azienda di costruzioni edili deve decidere la combinazione ottimale degli appalti da accettare per la costruzione degli edifici A, B e C.
I profitti attesi per i tre edifici sono di 3, 5 e 7 milioni di euro rispettivamente.
L'azienda dispone di 4 ruspe speciali e gli edifici richiedono risp. 3, 2 e 3 ruspe.
È possibile inoltre affittare fino a due altre ruspe speciali per la durata dei lavori, al costo di un milione di euro a ruspa.

Decisioni:

- accettare appalto i, $i \in \{A, B, C\}$. Possibili decisioni: sì/no.
- numero di ruspe da affittare. Possibili decisioni: 0, 1 o 2.

Possibili combinazioni: $2 \times 2 \times 2 \times 3 = 24$

Branch: scegliere una decisione (nell'ordine A-B-C-num.ruspe) e creare un sottoproblema per ogni valore

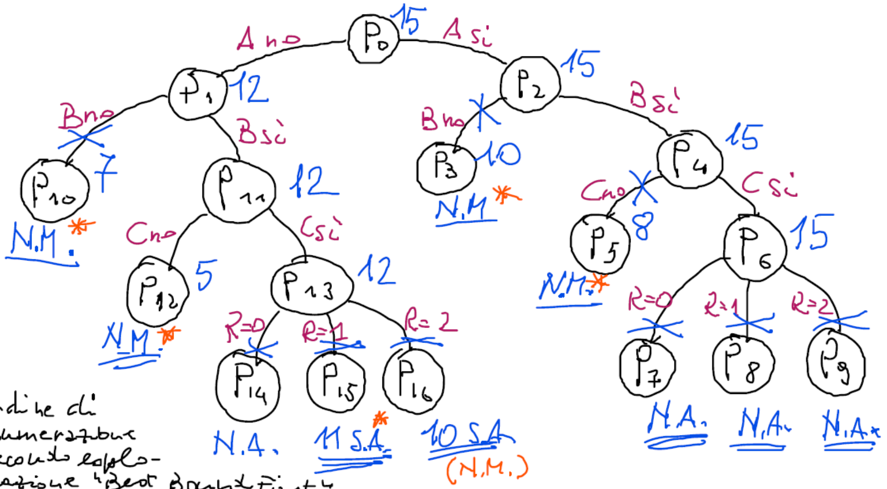
Bound: somma profitti di tutti gli appalti possibili meno costo ruspe "fissate" (valutazione imprecisa ma ottimistica e veloce, senza ragionamenti su ruspe "necessarie")

Esempio: albero di branch-and-bound

A: 3 M\$, 3 ruspe B: 5 M\$, 2 ruspe C: 7 M\$, 3 ruspe

Esempio: albero di branch-and-bound

A: 3 M$, 3 ruspe B: 5 M$, 2 ruspe C: 7 M$, 3 ruspe



Ordine di numerazione secondo esplorazione "Best Bound First"

Sol. ottima in P15!

Esempio: regola alternativa per bound migliore

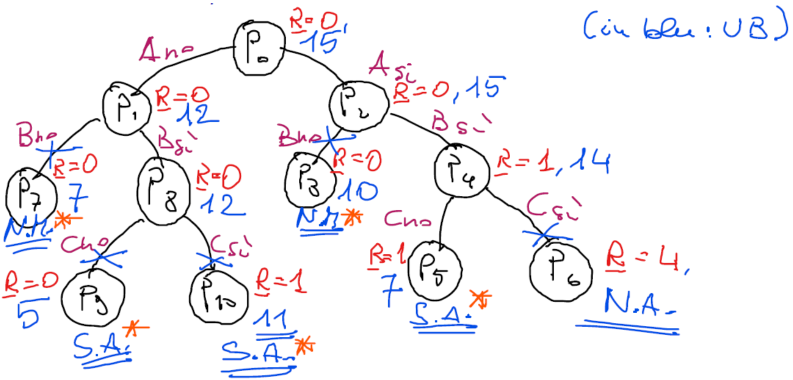
Bound: sommare i profitti di tutti gli appalti possibili *e valutare una stima per difetto R delle ruspe necessarie (sulla base degli appalti fissati)*

(A: 3 M$, 3 ruspe B: 5 M$, 2 ruspe C: 7 M$, 3 ruspe)

Esempio: regola alternativa per bound migliore

Bound: sommare i profitti di tutti gli appalti possibili *e valutare una stima per difetto R delle ruspe necessarie (sulla base degli appalti fissati)*

(A: 3 M\$, 3 ruspe   B: 5 M\$, 2 ruspe   C: 7 M\$, 3 ruspe)



→ la stima di R porta non solo a un U.B., ma anche a una soluzione ammissibile.

Esempio: algoritmo generale per path-finding

Vogliamo trovare un cammino minimo "generalizzato" con costi che variano nel tempo o dipendono dal cammino parziale, presenza dinamica di ostacoli **etc.**

- Branching: ad ogni passo, esplora le diverse direzioni ammesse
- Bounding: verifica ammissibilità; costo parziale[1] + cammino minimo con costi statici (LB su ogni arco) e senza vincoli
- Eventuale soluzione ammissibile: euristiche rapide di completamento (e.g., scelta greedy del prossimo arco)

[1] per cammino minimo standard e costi ≥ 0, il costo del cammino parziale è un lower bound. La strategia Best-Bound-First riproduce Dijkstra.