

Algoritmi branch-and-bound

- Algoritmi per problemi di ottimizzazione combinatoria di tipo \mathcal{NP} -difficile:

F = insieme soluzioni ammissibili (supposto finito): $y \in F$;

$z(y)$: funzione obiettivo (da massimizzare);

(z, F) = problema di determinare la soluzione $y^* \in F$ tale che

$$z(y^*) \geq z(y) \quad \forall y \in F$$

- Algoritmi di tipo branch-and-bound:

“ingredienti”:

- albero decisionale;
- strategie di esplorazione;
- rilassamenti;
- dominanze;
- tecniche di riduzione;
- algoritmi euristici.

Metodo branch-and-bound

- $P^0 = (z, F(P^0))$ (problema da risolvere);

$$Z(P^0) = \max\{z(y) : y \in F(P^0)\}.$$

- *Branch-and-bound*: suddivisione di P^0 in n_0 sottoproblemi

$$P^1, P^2, \dots, P^{n_0}$$

tali da “rappresentare globalmente” P^0 ;

ottenuta mediante suddivisione di $F(P^0)$ in sottoinsiemi

$$F(P^1), F(P^2), \dots, F(P^{n_0})$$

$$\text{tali che } \bigcup_{k=1}^{n_0} F(P^k) = F(P^0)$$

$$(Z(P^k) = \max\{z(y) : y \in F(P^k)\} \quad (k = 1, \dots, n_0))$$

Si ottiene: $Z(P^0) = \max\{Z(P^1), Z(P^2), \dots, Z(P^{n_0})\}$.

- (risolvere P^0) \Leftrightarrow (risolvere P^k ($k = 1, \dots, n_0$))



1) determinare la soluzione ottima di P^k , oppure

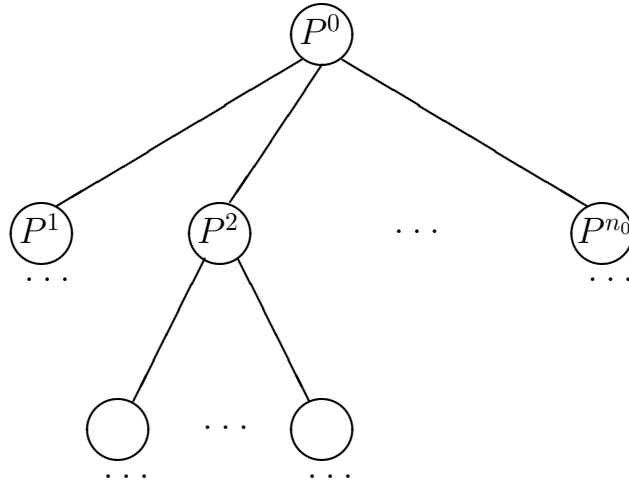
2) dimostrare che $F(P^k) = \emptyset$, oppure

3) dimostrare che $Z(P^k) \leq Z$ (Z = valore della miglior soluzione ammissibile di P^0 ottenuta in precedenza).

- Preferibilmente, $F(P^i) \cap F(P^j) = \emptyset \quad \forall P^i, P^j : i \neq j$
 (“partizione” di $F(P^0)$ in n_0 sottoinsiemi).

- Suddivisione ripetuta per ogni sottoproblema non risolto.

Albero decisionale



foglie = sottoproblemi da risolvere.

- Diverse strategie per la scelta del sottoproblema da affrontare.
- Per ogni P^k , calcolo di un valore $U(P^k)$ (*upper bound*) tale che

$$U(P^k) \geq Z(P^k)$$

- \Rightarrow **3)** se $U(P^k) \leq Z$, P^k non viene considerato.

- Per ottenere un upper bound di $P^k = (z, F(P^k))$:

rilassamento di P^k : problema $R(P^k) = (z_r, F_r(P^k))$ tale che:

a) $F_r(P^k) \supseteq F(P^k)$,

b) $z_r(y) \geq z(y) \quad \forall y \in F(P^k)$.

- $U(P^k) = Z(R(P^k)) = \max\{z_r(y) : y \in F_r(P^k)\}$.

- Proprietà: $Z(R(P^k))$ il più possibile prossimo a $Z(P^k)$;

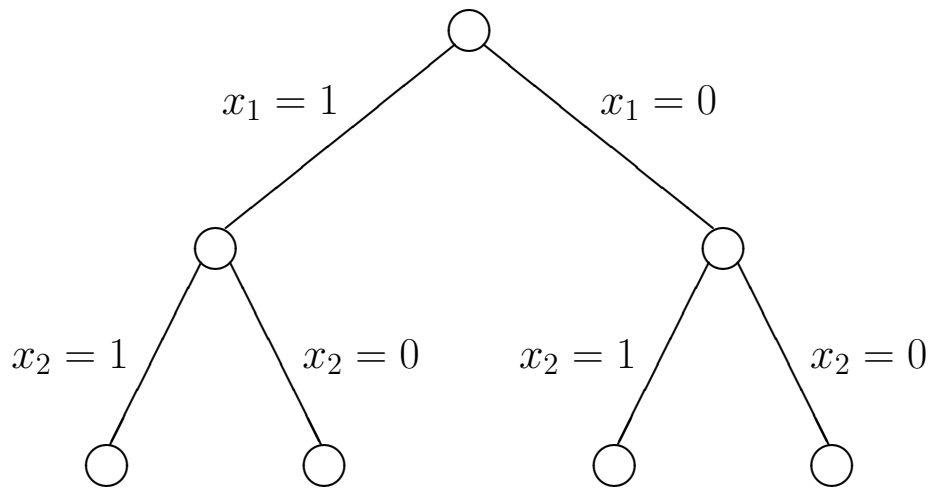
$R(P^k)$ sufficientemente “facile” da risolvere.

Algoritmo branch-and-bound per KP01

- È conveniente ordinare gli elementi secondo valori non crescenti del profitto per unità di peso:

$$\frac{p_j}{w_j} \geq \frac{p_{j+1}}{w_{j+1}} \quad \forall j$$

- Albero decisionale binario:



...

- Strategia di esplorazione:
 - al primo livello $x_1 = 1, x_1 = 0$;
 - se esiste nodo attivo generato da $(x_j = 1)$ si ramifica da questo, altrimenti dall'ultimo nodo attivo generato da $(x_j = 0)$.

⇓

L'insieme dei nodi attivi contiene sempre
al più un nodo generato da $(x_j = 1)$
uno o più nodi generati da $(x_j = 0)$

- Upper bound \Leftarrow rilassamento continuo:

$$x_j \in \{0, 1\} \quad (j = 1, \dots, n) \quad \text{sostituito da}$$

$$0 \leq x_j \leq 1 \quad (j = 1, \dots, n).$$

- Soluzione del rilassamento continuo (Dantzig, 1957):

- si inseriscono con continuità gli elementi migliori frazionando il primo che non entra interamente, cioè:

- $s := \min\{i : \sum_{j=1}^i w_j > c\}$ (*elemento critico*);

- $\bar{c} := c - \sum_{j=1}^{s-1} w_j$;

$$x_j = 1, j = 1, \dots, s-1; x_s = \bar{c}/w_s; x_j = 0, j = s+1, \dots, n$$

- $U := \left\lfloor \sum_{j=1}^{s-1} p_j + \bar{c} \frac{p_s}{w_s} \right\rfloor$

- Es: $n = 5$

$$p' = (12, 12, 7, 6, 2)$$

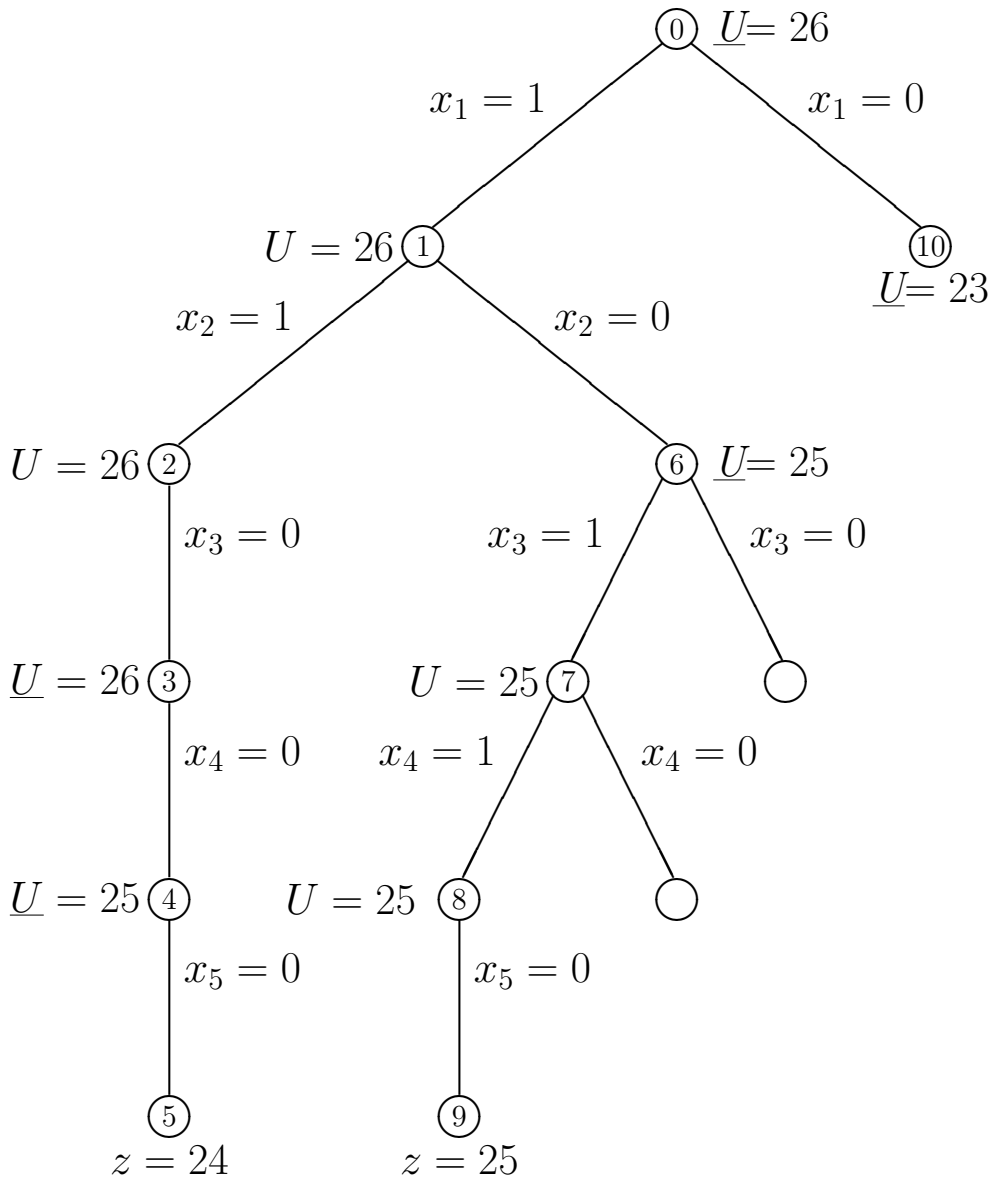
$$w' = (4, 5, 3, 3, 2)$$

$$c = 10$$

$$s = 3, \bar{c} = 1$$

$$U = \left\lfloor 12 + 12 + 1 \cdot \frac{7}{3} \right\rfloor = 26$$

Es: $n = 5$
 $p' = (12, 12, 7, 6, 2)$
 $w' = (4, 5, 3, 3, 2)$
 $c = 10$



Soluzione ottima: nodo 9: $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1, x_5 = 0; z = 25$

Rilassamenti (Problema di massimizzazione)

- Rilassamento: qualunque metodo che
 - a) ampli la regione ammissibile
 - e/o
 - b) aumenti il valore della funzione obiettivo in corrispondenza della regione ammissibile originale.
- Alcuni rilassamenti *classici* possono essere applicati al modello matematico del problema.

- Dato un problema (P) di PLI binaria:

$$Z(P) = \max \sum_{j=1}^n c_j x_j \quad (1)$$

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad (i = 1, \dots, m) \quad (2)$$

$$\sum_{j=1}^n d_{kj} x_j = e_k \quad (k = 1, \dots, l) \quad (3)$$

$$x_j \in \{0, 1\} \quad (j = 1, \dots, n) \quad (4)$$

- Il problema rilassato $R(P)$ ha un valore ottimo $Z(R(P))$ “migliore” di $Z(P)$, cioè $Z(R(P)) \geq Z(P)$ (si dice che $Z(R(P))$ è un limite superiore o *Upper Bound* di $Z(P)$).
- Criteri: $Z(R(P))$ il più *prossimo* a $Z(P)$;
 $R(P)$ sufficientemente *facile* da risolvere.
- Due tipi distinti di vincoli per evidenziare diverse proprietà.
- I rilassamenti classici si estendono facilmente a PLI generale.