Insegnamento di algoritmi avanzati

Branch & Bound

Roberto Posenato

ver. 0.5, 01/10/2009

Sommario

Introduzione

In questa lezione si introducono i concetti fondamentali circa la tecnica *branch & bound*.

In due parole...

La tecnica del branch & bound può essere considerata un'evoluzione del backtracking adatta per risolvere problemi di ottimizzazione.

Dettagli

Schema generale

Tecnica del branch & bound:

- Si applica quando il grafo implicito è un albero e rappresenta soluzioni parziali o ammissibili per un problema di ottimizzazione.
- Si assume che ad ogni nodo si possa calcolare un limite superiore e uno inferiore al valore delle soluzioni ammissibili raggiungibili dal nodo.
- Raggiunto un nodo (branching), per ciascun nodo-figlio:
  - si determina il limite (bound) al valore delle possibili soluzioni che si possono determinare proseguendo per il figlio.
  - se il limite indica che ogni soluzione possibile non sarà migliore di quella già determinata, non si esplora l'albero lungo quel figlio (prune).

Dettagli

Pseudocodifica

Assunzioni per poter rappresentare un semplice schema di un algoritmo branch & bound per un problema di minimizzazione:

- la funzione obiettivo $m()$ ha valori non negativi;
- l'albero implicito è costituito da nodi che rappresentano elementi della soluzione e ha grado $\leq m$ e altezza n;
- è disponibile una funzione lower bound $\texttt{LB}(s[1, \ldots, j, \ldots])$ che, in base al cammino nell'albero rappresentato dal vettore s, restituisce il valore minimo delle soluzioni ammissibili che si possono generare a partire da s.

Dettagli

Pseudocodifica

Procedura `branchBound(`i`)`

```
1:  if (i < n) then // i è "l'indice" del nodo corrente
2:      Determina i nodi figli di i;
3:      foreach (nodo figlio j) do // l'ordine può essere importante!
4:          s[i + 1] = j;                    // Soluzione parziale nota = s[1, ..., i]
5:          if (LB(s) < MINCOST) then branchBound(i + 1);
6:      endfch
7:  else
8:      if (m(s) < MINCOST) then
9:          MINSOL = s;                      // MINSOL = soluzione ottimale
10:         MINCOST = m(s);                  // MINCOST = valore ottimale
11:     endif
12: endif
```

Dettagli

Pseudocodifica

Complessità

Nel caso peggiore:

Il ciclo **foreach** viene eseguito m volte per ogni chiamata, che sono n al più.

Se $\texttt{LB}(s[1, \ldots, j, \ldots])$ ha costo $O(f(n))$, allora complessità totale è

$$O(m^n f(n)^n)$$

Dettagli

Aspetti cruciali

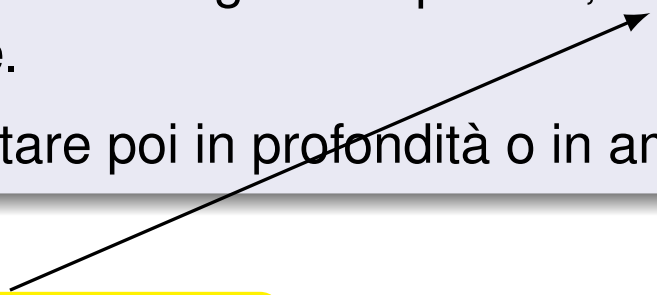
In pratica, il tempo di computazione è influenzato da diversi fattori:

- *Calcolo lower (upper) bound*
  Più la funzione di bound è precisa, meno parti di albero si devono esplorare. Un buon bound può però richiedere un tempo di calcolo non trascurabile.
- *Ordine nodi figli da visitare*.
  L'ordine più usato per visitare i figli è il LIFO perché richiede meno spazio. Ci sono altri ordini. . . che possono portare ad esplorare meno rami.

Dettagli
Considerazioni

Scelta ordine di esplorazione dei nodi figli

- È possibile che i nodi-figli con bound *migliore* permettano di determinare soluzioni o ulteriori bound migliori e, quindi, tagliare ulteriormente altri rami prima che questi vengano visitati.
- Quindi, una volta selezionati i nodi-figli da esplorare, si inizia da quello con il bound migliore.
- Rimane da decidere se visitare poi in profondità o in ampiezza.



Dettagli
Considerazioni

Scelta ordine di esplorazione dei nodi

- Se si usa il calcolo del bound in modo ancora più generale, il branch & bound può essere visto come ulteriore metodo di ricerca in un albero.
- I nodi vengono visitati secondo l'ordine stabilito dalla funzione bound sui valori contenuti nei sottoalberi.

Strategia best-first

Per implementare questa strategia è sufficiente usare un coda con priorità per gestire i nodi da esplorare:

Tipo ricerca	Struttura dati principale
breadth-first	coda
depth-first	pila
best-first	coda con priorità (heap)

Esempi di applicazione

Problema dell'assegnamento

PROBLEMA ASSEGNAMENTO

DESCRIZIONE: Un insieme di n agenti, che devono essere assegnati a n task: 1 agente per 1 task.
Una funzione *costo* c_{ij} che indica costo agente i per svolgere task j.

QUESITO: Determinare l'assegnamento dei task tale che il costo totale sia minimo.

Esempi di applicazione

Problema dell'assegnamento

Esempio 1 (Istanza del problema dell'assegnamento)

4 agenti: a, b, c, d. 4 task: $1, 2, 3, 4$.

Funzione costo:

c_{ij}	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Un semplice upper bound per la soluzione è la somma della diagonale: $11 + 15 + 19 + 28 = 73$

Un semplice lower bound è prendere il minimo x ogni colonna: $11 + 12 + 13 + 22 = 58$

La soluzione dovrà essere in $[58 \ldots 73]$.

Qual è la soluzione ottima?

Esempi di applicazione

Problema dell'assegnamento

Per risolvere l'esempio con la tecnica branch & bound:

- Si deve determinare la soluzione in modo incrementale visitando l'albero implicito:
  - Si determina un range iniziale per la soluzione ottimale;
  - Si inizia con un assegnamento vuoto (radice);
  - Ad ogni livello, si determina l'assegnamento per un nuovo agente (altezza albero = n).
- Ad ogni livello, si considerano tutti i nodi che rappresentano un assegnamento ammissibile per l'agente associato al livello.

Esempi di applicazione

Problema dell'assegnamento

- Per ogni nodo si calcola il lower bound per le soluzioni che si possono ottenere se si proseguisse per quel nodo.
- Il lower bound indica se il nodo è da considerare:
  - se è inferiore all'upper bound corrente, si considera;
  - altrimenti si scarta per sempre.
- Fra tutti i nodi da considerare, si prosegue il cammino in depth-first search a partire dal nodo lower bound inferiore.
- Quando si giunge una soluzione finale, si aggiorna l'upper bound e si scartano tutti i nodi dell'albero che superano tale bound.

Esempi di applicazione

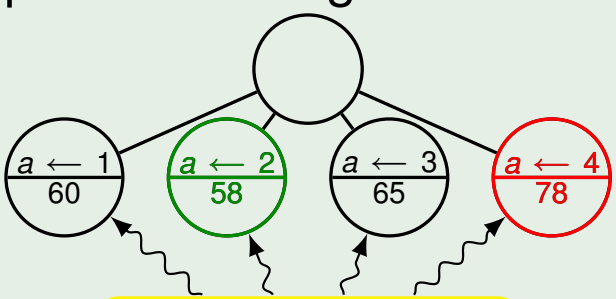
Problema dell'assegnamento

Istanza del problema dell'assegnamento (2)

Funzione costo:

c_{ij}	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Se si parte da a, ci sono 4 possibili assegnamenti:



costo di 'a' + costo minimo in ciascuna delle colonne rimanenti private della riga 'a'.

- Considerato che la soluzione $a \leftarrow 4$ porterà a una soluzione con costo ≥ 78, oltre l'upper 73 già determinato, viene scartata.
- Il nodo $a \leftarrow 2$ ha il miglior lower bound: sembra quindi il più promettente da esplorare (non vi è certezza!)

Esempi di applicazione

Problema dell'assegnamento

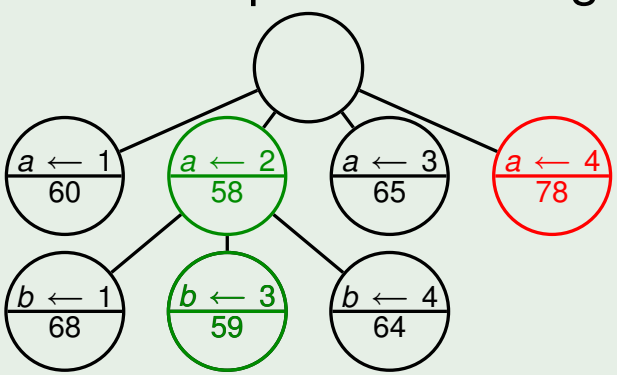
Istanza del problema dell'assegnamento (3)

Funzione costo:

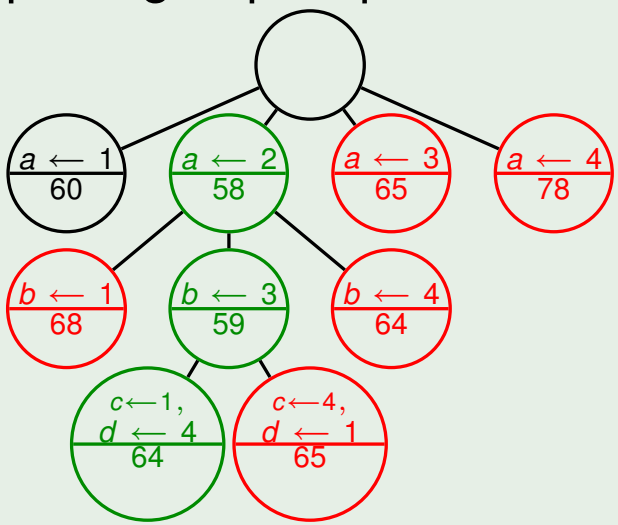
c_{ij}	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Livello 2 = attività per b.

Ci sono 3 possibili assegnamenti:



Miglior lower bound = $b \leftarrow 3$: si prosegue per questo ramo.



Task per $c \Rightarrow$ task per d.

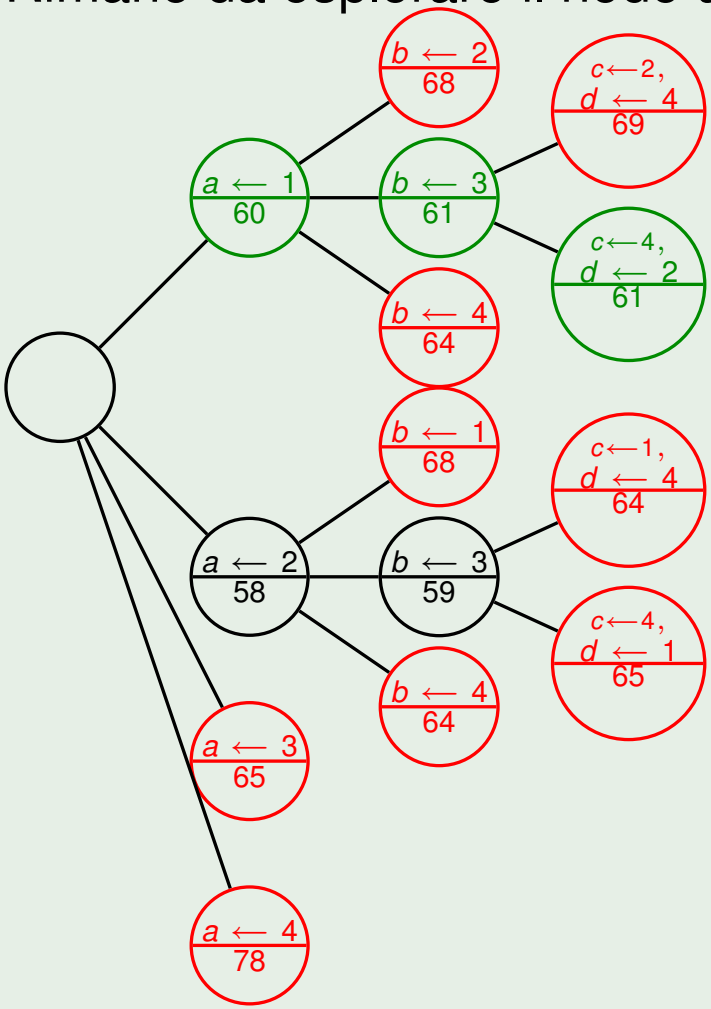
Nuovo upper bound elimina molti rami dell'albero.

Esempi di applicazione

Problema dell'assegnamento

Istanza del problema dell'assegnamento (4)

Rimane da esplorare il nodo $a \leftarrow 1$.



- La soluzione ottimale è $a \rightarrow 1, b \rightarrow 3, c \rightarrow 4, d \rightarrow 2$, con valore 61.
- Sono stati valutati 15 nodi su 40 possibili (la radice non conta!).
- Delle $4! = 24$ soluzioni possibili, solo 6 sono state esaminate: 4 dell'albero e 2 iniziali.

Esempi di applicazione

Problema dello zaino

PROBLEMA ZAINO

DESCRIZIONE: Un insieme di n di oggetti caratterizzati ciascuno da un valore v_i e da un peso p_i, entrambi interi positivi; un intero positivo P, portata massima dello zaino.

QUESITO: Determinare un sottoinsieme $X = \{x_1, x_2, \ldots, x_n\}$, con $x_i = 0$ o 1, degli oggetti tale che il valore totale $\sum_{i=1}^{n} x_i v_i$ sia massimo e il peso totale $\sum_{i=1}^{n} x_i p_i \leq P$.

Senza perdere in generalità, si assume che i tipi siano ordinati in modo che $\frac{v_i}{p_i} \geq \frac{v_{i+1}}{p_{i+1}}$.

Esempi di applicazione

Problema dello zaino

Osservazione

Data una soluzione parziale determinata fino alla k-esima componente $x_1, \ldots, x_k$, con $k < n$, un upper bound per la soluzione finale è

$$\underbrace{\sum_{i=1}^{k} x_i v_i}_{\text{valore corrente}} + \underbrace{\left(P - \sum_{i=1}^{k} x_i p_i\right)}_{\text{peso rimanente ammissibile}} \cdot \frac{v_{k+1}}{p_{k+1}}$$

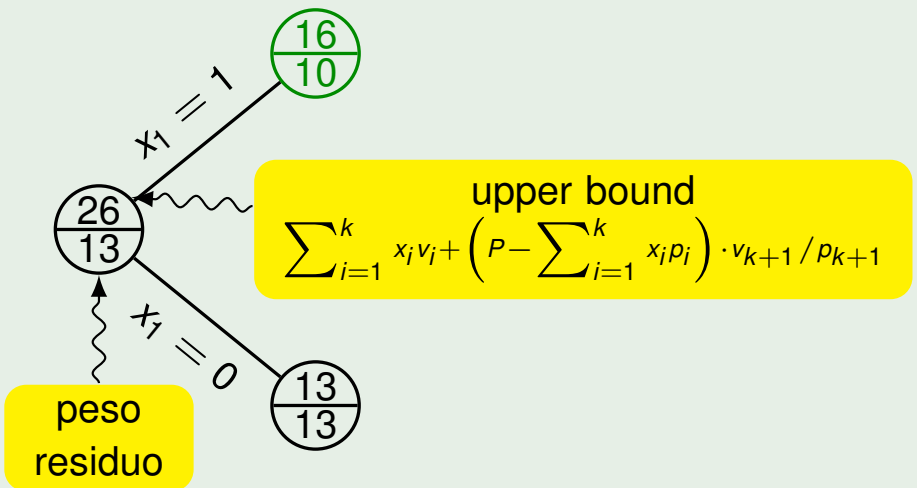
Esempi di applicazione

Problema dello zaino

Esempio 2 (Istanza del Problema dello zaino)

$v = [6, 13, 4, 3]$, $p = [3, 13, 5, 5]$ e $P = 13$. $\Rightarrow \frac{v_i}{p_i} = [2, 1, 0.8, 0.6]$

Al livello $i - 1$ consideriamo se inserire o non inserire l'elemento x_i $(i = 1, \ldots, 4)$:



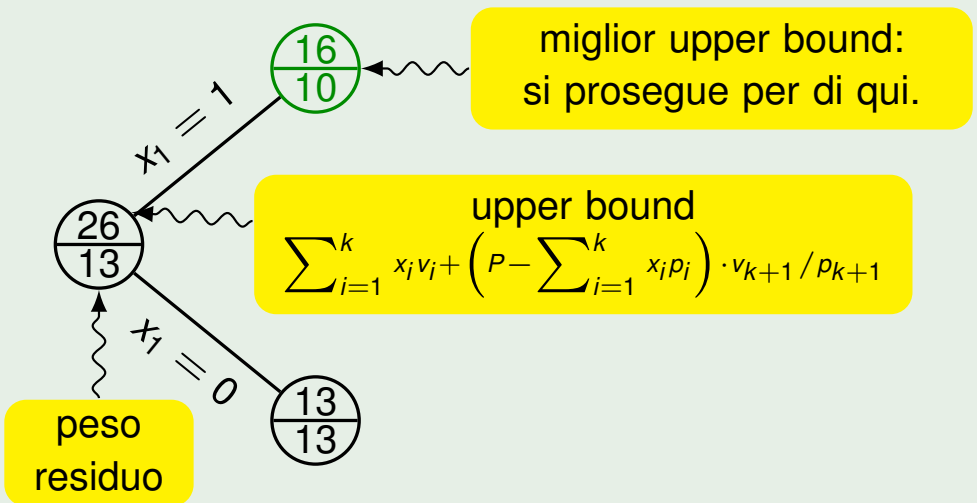
Esempi di applicazione

Problema dello zaino

Esempio 2 (Istanza del Problema dello zaino)

$v = [6, 13, 4, 3]$, $p = [3, 13, 5, 5]$ e $P = 13$. $\Rightarrow \frac{v_i}{p_i} = [2, 1, 0.8, 0.6]$

Al livello $i - 1$ consideriamo se inserire o non inserire l'elemento x_i $(i = 1, \ldots, 4)$:



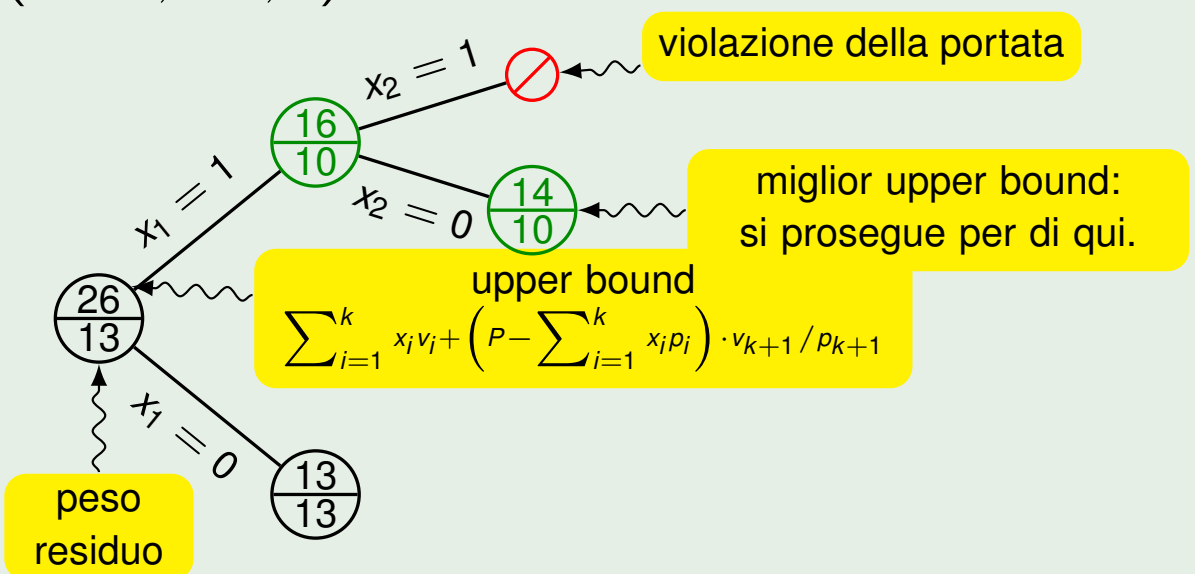
Esempi di applicazione

Problema dello zaino

Esempio 2 (Istanza del Problema dello zaino)

$v = [6, 13, 4, 3]$, $p = [3, 13, 5, 5]$ e $P = 13$. $\Rightarrow \frac{v_i}{p_i} = [2, 1, 0.8, 0.6]$

Al livello $i - 1$ consideriamo se inserire o non inserire l'elemento x_i $(i = 1, \ldots, 4)$:



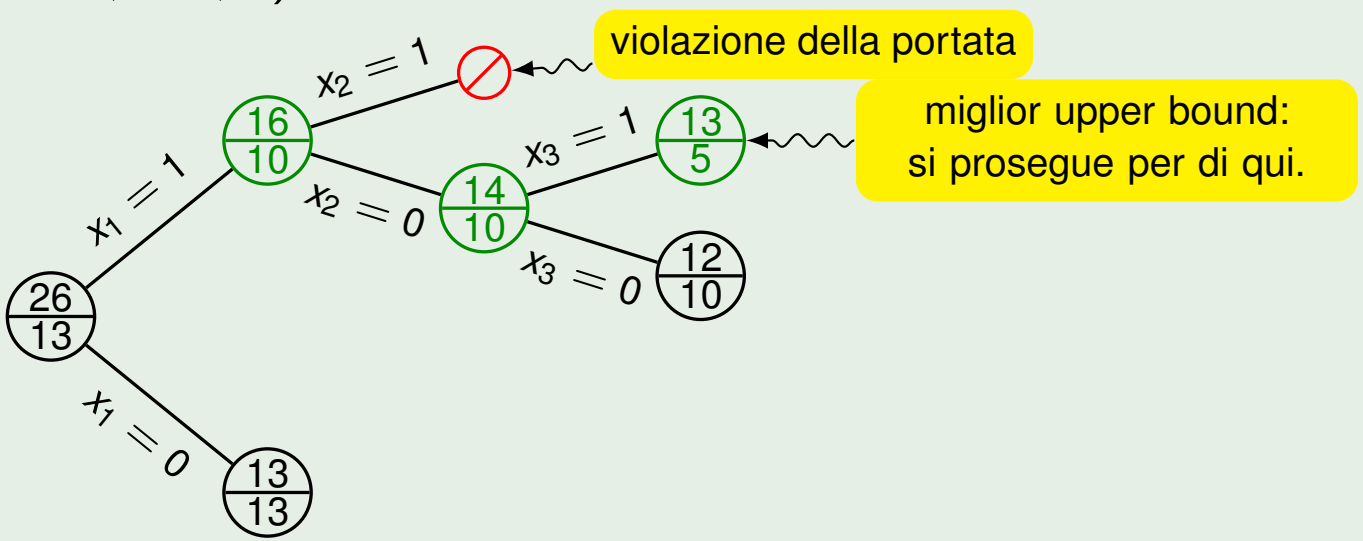
Esempi di applicazione

Problema dello zaino

Esempio 2 (Istanza del Problema dello zaino)

$v = [6, 13, 4, 3]$, $p = [3, 13, 5, 5]$ e $P = 13$. $\Rightarrow \frac{v_i}{p_i} = [2, 1, 0.8, 0.6]$

Al livello $i - 1$ consideriamo se inserire o non inserire l'elemento x_i $(i = 1, \ldots, 4)$:



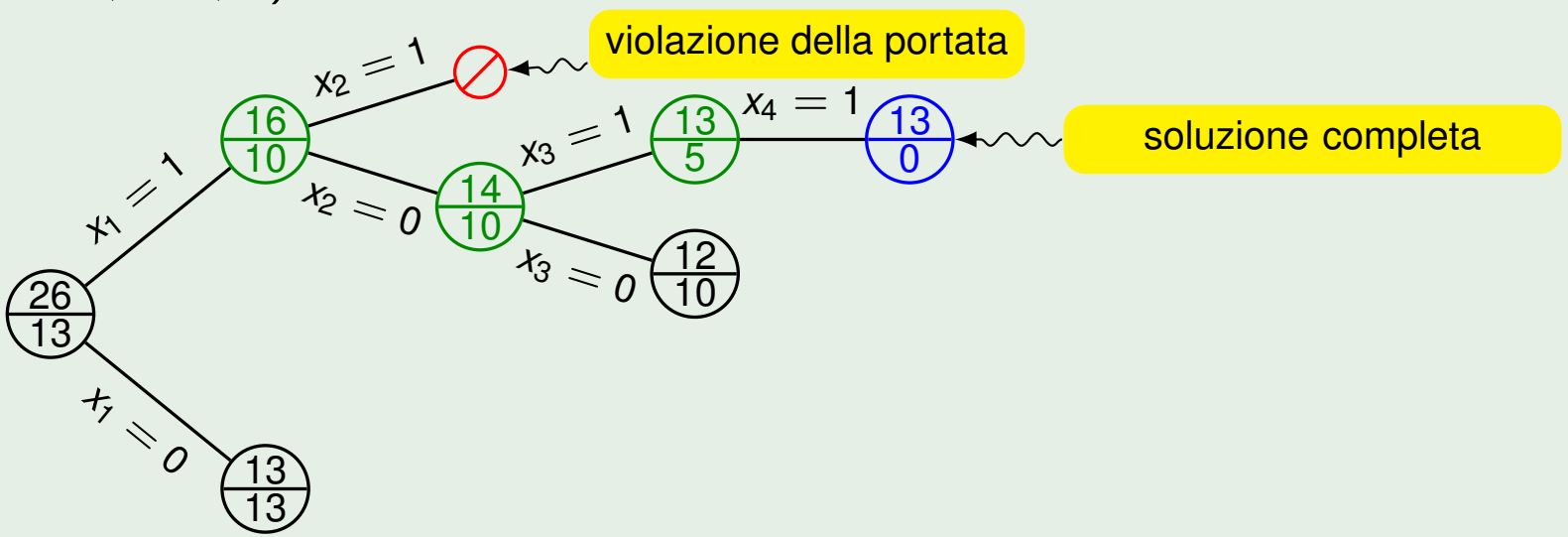
Esempi di applicazione

Problema dello zaino

Esempio 2 (Istanza del Problema dello zaino)

$v = [6, 13, 4, 3]$, $p = [3, 13, 5, 5]$ e $P = 13$. $\Rightarrow \frac{v_i}{p_i} = [2, 1, 0.8, 0.6]$

Al livello $i - 1$ consideriamo se inserire o non inserire l'elemento x_i $(i = 1, \ldots, 4)$:



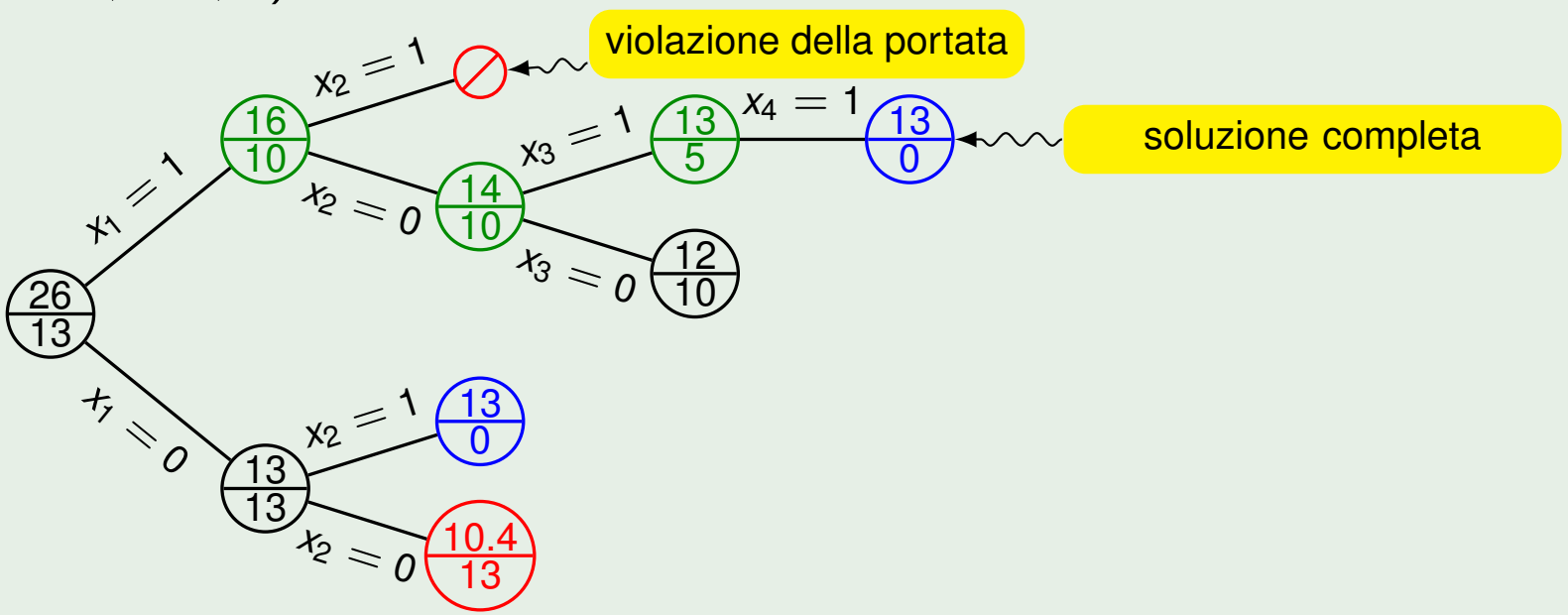
Esempi di applicazione

Problema dello zaino

Esempio 2 (Istanza del Problema dello zaino)

$v = [6, 13, 4, 3]$, $p = [3, 13, 5, 5]$ e $P = 13$. $\Rightarrow \frac{v_i}{p_i} = [2, 1, 0.8, 0.6]$

Al livello $i - 1$ consideriamo se inserire o non inserire l'elemento x_i $(i = 1, \ldots, 4)$:



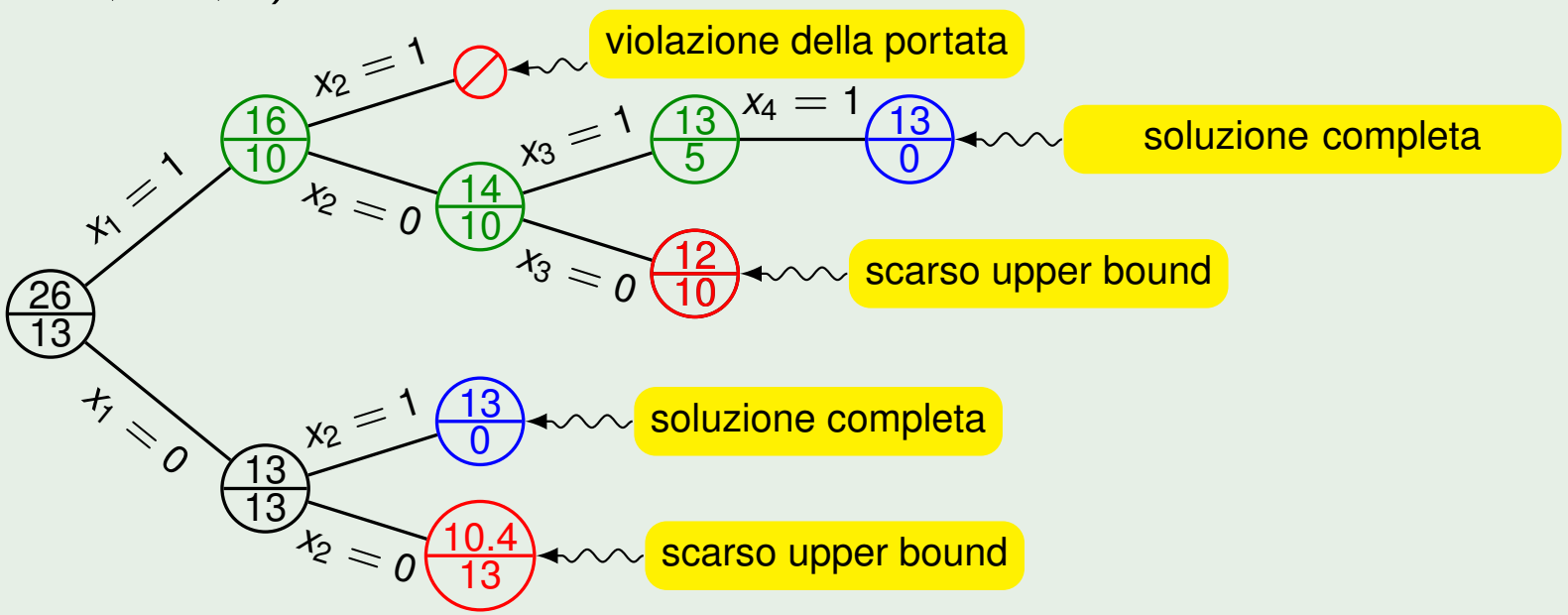
Esempi di applicazione

Problema dello zaino

Esempio 2 (Istanza del Problema dello zaino)

$v = [6, 13, 4, 3]$, $p = [3, 13, 5, 5]$ e $P = 13$. $\Rightarrow \frac{v_i}{p_i} = [2, 1, 0.8, 0.6]$

Al livello $i - 1$ consideriamo se inserire o non inserire l'elemento x_i $(i = 1, \ldots, 4)$:



Esempi di applicazione

Calcolo di LB

PROBLEMA COMMESSO VIAGGIATORE (TRAVELLING SALESMAN PROBLEM - TSP)

DESCRIZIONE: Un grafo non diretto completo di n città e una funzione $d : E \longrightarrow \mathbf{N}$, distanza tra le città.

QUESITO: Determinare un cammino che visiti tutte le città una e una sola volta partendo e terminando nella medesima città e che sia di lunghezza minima.

- La soluzione è un circuito hamiltoniano di lunghezza minima.
- La soluzione finale può essere costruita incrementalmente aggiungendo una città alla volta.
- Una soluzione parziale $s[1, \ldots, i]$ di i città è quindi un cammino hamiltoniano di lunghezza minima.

Esempi di applicazione

Calcolo di LB

- Data una soluzione parziale di $s[1, \ldots, i]$, questa può essere espansa se si aggiunge una città non ancora presente nel cammino (questo indica quali sono i nodi figli).
- Definizioni preliminari:
  - $A = \min_h \{d[s[1], h] \mid h \notin s\}$, lower bound (l.b.) per distanza minima dall'inizio circuito.
  - $B = \min_h \{d[s[i], h] \mid h \notin s\}$, l.b. per distanza minima dalla fine.
  - $c[1] = 0, c[i] = c[i-1] + d[s[i-1], s[i]]$, lunghezza di s.
  - $D[h] = \min_{p,q} \{d[p, h] + d[h, q] \mid h \neq p \neq q\} \forall h, p, q \notin s$, l.b. per attraversare 1 delle rimanenti $n - i$ città.
- Una funzione `LB(s)` è data da

$$\texttt{LB(s)} = \begin{cases} c[i] + \left\lceil \frac{A+B+\sum_{h \notin s} D[h]}{2} \right\rceil & i < n, \\ c[i] + d[s[i], s[1]] & i == n. \end{cases}$$

Esercizi

Assegnamento

Esercizio 1

Si implementi l'algoritmo descritto in questa lezione per il problema dell'assegnamento.

Esercizi

Zaino

Esercizio 2

Si implementi l'algoritmo descritto in questa lezione per il problema dello zaino.
Si consideri l'istanza in cui $P = 12$ e gli elementi sono caratterizzati dalle seguenti coppie (peso, valore):
$(6, 25), (2, 8), (5, 19), (7, 26), (3, 10), (4, 10), (1, 1)$.
Si calcoli l'albero implicito di esecuzione.

Esercizi

MINIMO RICOPRIMENTO DI VERTICI

Esercizio 3

Si consideri il problema del MINIMO RICOPRIMENTO DI VERTICI.

- Si determini una funzione upper bound e una lower bound per le soluzioni parziali e si determini inoltre la sua complessità computazionale.
- Si determini un algoritmo branch & bound per il problema.
- Si scriva l'albero implicito determinato dall'algoritmo quando applicato al seguente grafo: