Branch & Bound (II)

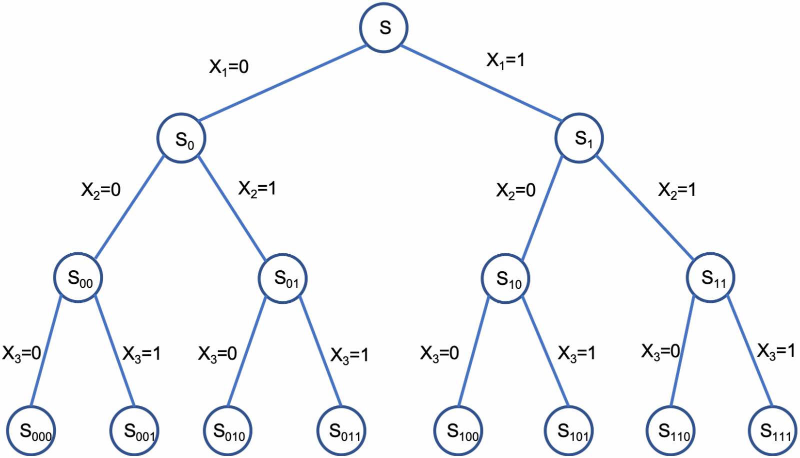
Lorenzo Castelli, Università degli Studi di Trieste.

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

B&B for binary problems

When addressing a binary problem, the natural choice is to branch on the binary variables, i.e., considering at each level of the tree a variable x_j and branching on it: $x_j = 0$ and $x_j = 1$.

Binary tree


S
X1=0
X1=1
S0
S1
X2=0
X2=1
X2=0
X2=1
S00
S01
S10
S11
X3=0
X3=1
X3=0
X3=1
X3=0
X3=1
X3=0
X3=1
S000
S001
S010
S011
S100
S101
S110
S111

B&B for the knapsack problem

The Knapsack problem is formulated as

$$
\begin{aligned}
z = \max \sum_{j=1}^{n} p_j x_j& \\
\sum_{j=1}^{n} w_j x_j &\leq W \\
x_j \in \{0, 1\} &\text{ for } j = 1, \ldots, n.
\end{aligned}
$$

We always assume $\sum_{j=1}^{n} w_j > W$. Otherwise the solution is trivial.

Dantzig's upper bound

1. Variables are ordered so that $p_1/w_1 \geq \cdots \geq p_n/w_n$.
2. Set

$$x_j = 1 \text{ for } j = 1, \ldots, r-1$$
$$x_r = \frac{W - \sum_{j=1}^{r-1} w_j}{w_r}$$
$$x_j = 0 \text{ for } j = r+1, \ldots, n$$

3. where r is such that $\sum_{j=1}^{r-1} w_j \leq W$ and $\sum_{j=1}^{r} w_j > W$.
4. If $\overline{W} = W - \sum_{j=1}^{r-1} w_j$ then $UB = \lfloor \sum_{j=1}^{r-1} p_j + p_r \frac{\overline{W}}{w_r} \rfloor$
5. We can prove it always holds that $z^* \leq UB$.

Knapsack problem - example

$$
\begin{aligned}
z = \max & 6x_1 + 8x_2 + 7x_3 + 5x_4 \\
& 3x_1 + 2x_2 + 5x_3 + 5x_4 \leq 8 \\
& x_j \in \{0,1\} \text{ for } j = 1, \ldots, 4
\end{aligned}
$$

We first re-arrange the variables so that $p_1/w_1 \geq p_2/w_2 \geq p_3/w_3 \geq p_4/w_4$

$$
\begin{aligned}
z = \max & 8x_1 + 6x_2 + 7x_3 + 5x_4 \\
& 2x_1 + 3x_2 + 5x_3 + 5x_4 \leq 8 \\
& x_j \in \{0,1\} \text{ for } j = 1, \ldots, 4
\end{aligned}
$$

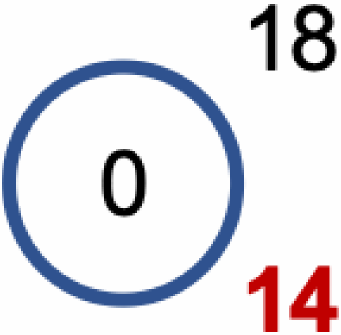
Root node - 0

We first calculate the upper bound UB_0

- Object **1** can enter, hence $x_1 = 1$ and $\overline{W} = 6$
- Object **2** can enter, hence $x_2 = 1$ and $\overline{W} = 3$
- Object **3** cannot enter, hence $r = 3$ and $x_3 = 3/5$
- $x_4 = 0$
- $UB_0 = \lfloor 8 + 6 + 7 * (3/5) \rfloor = \lfloor 14 + 21/5 \rfloor = 18$

A lower bound is easily identified by setting $x_1 = x_2 = 1, x_3 = x_4 = 0$, hence $LB = 14$

Root node - 0



Branching on x_3

$x_3 = 0$

$$z = \max 8x_1 + 6x_2 + 5x_4$$
$$2x_1 + 3x_2 + 5x_4 \leq 8$$
$$x \in \{0, 1\}^4$$

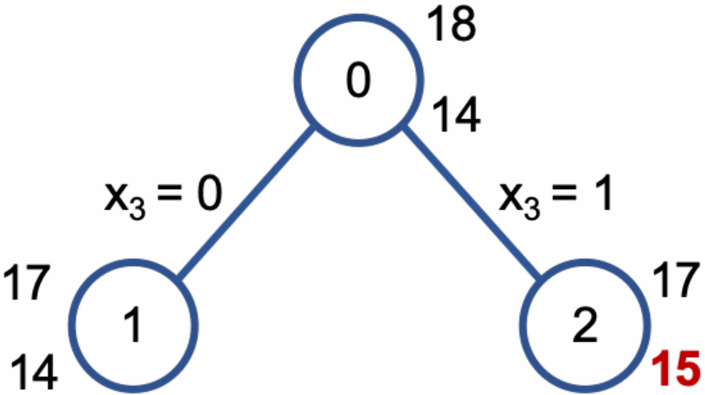
- Object 1 can enter, hence $x_1 = 1$ and $\overline{W} = 6$
- Object 2 can enter, hence $x_2 = 1$ and $\overline{W} = 3$
- Object 4 cannot enter, hence $r = 4$ and $x_4 = 3/5$
- $UB_1 = \lfloor 8 + 6 + 5 * (3/5) \rfloor = \lfloor 14 + 3 \rfloor = 17$
- $LB_1 = 14$ due to $(1, 1, 0, 0)$

$x_3 = 1$

$$z = \max 8x_1 + 6x_2 + 5x_4 + 7$$
$$2x_1 + 3x_2 + 5x_4 \leq 3$$
$$x \in \{0, 1\}^4$$

- Object 1 can enter, hence $x_1 = 1$ and $\overline{W} = 1$
- Object 2 cannot enter, hence $r = 2$ and $x_2 = 1/3$
- $x_4 = 0$
- $UB_2 = \lfloor 8 + 7 + 6 * (1/3) \rfloor = \lfloor 15 + 2 \rfloor = 17$
- $LB_2 = 15$ due to $(1, 0, 1, 0)$
- Hence $LB = 15$

Branching on x_3



$x_3 = 0$, branching on x_4

$x_3 = 0, x_4 = 0$

$$
\begin{aligned}
z = \max & \ 8x_1 + 6x_2 \\
& 2x_1 + 3x_2 \leq 8 \\
& x \in \{0,1\}^4
\end{aligned}
$$

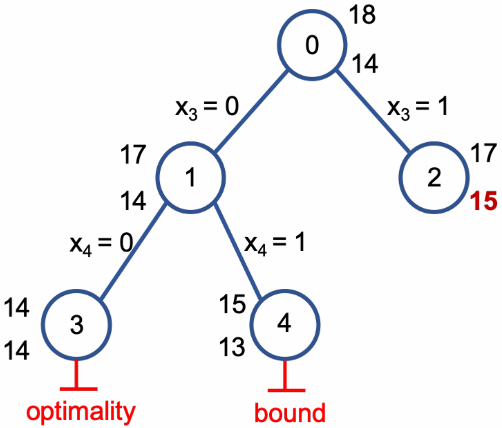
- Object 1 can enter, hence $x_1 = 1$ and $\overline{W} = 6$
- Object 2 can enter, hence $x_2 = 1$ and $\overline{W} = 3$
- $UB_3 = \lfloor 8 + 6 \rfloor = 14$
- $LB_3 = 14$ due to $(1, 1, 0, 0)$
- STOP. Pruned by optimality.

$x_3 = 0, x_4 = 1$

$$
\begin{aligned}
z = \max & \ 8x_1 + 6x_2 + 5 \\
& 2x_1 + 3x_2 \leq 3 \\
& x \in \{0,1\}^4
\end{aligned}
$$

- Object 1 can enter, hence $x_1 = 1$ and $\overline{W} = 1$
- Object 2 cannot enter, hence $r = 2$ and $x_2 = 1/3$
- $UB_4 = \lfloor 8 + 5 + 6 * (1/3) \rfloor = \lfloor 13 + 2 \rfloor = 15$
- $LB_4 = 13$ due to $(1, 0, 0, 1)$
- STOP. Pruned by bound: $UB_4 = LB$

$x_3 = 0$, branching on x_4



$x_3 = 1$, branching on x_2

$x_3 = 1, x_2 = 0$

$$\begin{aligned} z = \max & \ 8x_1 + 5x_4 + 7 \\ & 2x_1 + 5x_4 \leq 3 \\ & x \in \{0,1\}^4 \end{aligned}$$

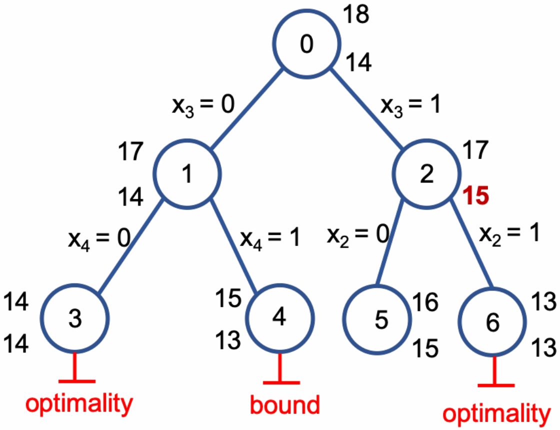
- Object 1 can enter, hence $x_1 = 1$ and $\overline{W} = 1$
- Object 4 cannot enter, hence $r = 4$ and $x_4 = 1/5$
- $UB_5 = \lfloor 8 + 7 + 5 * (1/5) \rfloor = \lfloor 15 + 1 \rfloor = 16$
- $LB_5 = 15$ due to $(1, 0, 1, 0)$

$x_3 = 1, x_2 = 1$

$$\begin{aligned} z = \max & \ 8x_1 + 5x_4 + 13 \\ & 2x_1 + 5x_4 \leq 0 \\ & x \in \{0,1\}^4 \end{aligned}$$

- Object 1 cannot enter, $r = 1$ and $x_1 = 0$
- $x_4 = 0$
- $UB_6 = \lfloor 13 + 8 * (0) \rfloor = 13$
- $LB_6 = 13$ due to $(0, 1, 1, 0)$
- STOP. Pruned by optimality.

$x_3 = 1$, branching on x_2



$x_3 = 1, x_2 = 0$, branching on x_4

$x_3 = 1, x_2 = 0, x_4 = 0$

$$z = \max 8x_1 + 7$$
$$2x_1 \leq 3$$
$$x \in \{0,1\}^4$$

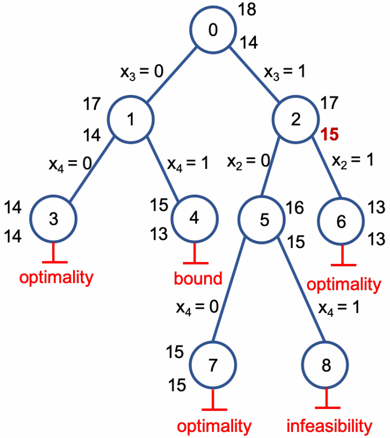
- Object 1 can enter, hence $x_1 = 1$ and $\overline{W} = 1$
- $UB_7 = \lfloor 8 + 7 \rfloor = 15$
- $LB_7 = 15$ due to $(1, 0, 1, 0)$
- STOP. Pruned by optimality.

$x_3 = 1, x_2 = 0, x_4 = 1$

$$z = \max 8x_1 + 12$$
$$2x_1 \leq -2 \textit{ INFEASIBLE!!}$$
$$x \in \{0,1\}^4$$

- STOP. Pruned by infeasibility.

$x_3 = 1, x_2 = 0$, branching on x_4



Knapsack problem - example

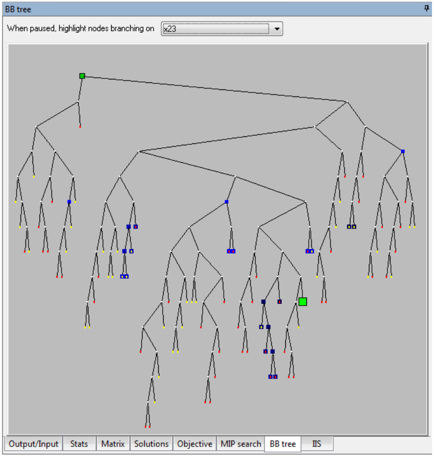
- The optimal solution is $(\mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{0})$ and $z^* = \mathbf{15}$
- This solution was found on node $\mathbf{2}$ but six other nodes had to be visited before confirming that $(\mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{0})$ is indeed the optimal solution.

Node selection

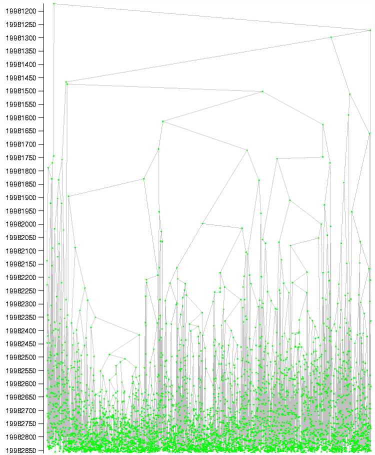
Given a list $\mathcal{L}$ of active subproblems (or active nodes), the question is to decide which node should be examined in detail next. There are two basic options

- A priori rules that determine, in advance, the order in which the tree will be developed.
- Adaptive rules that choose a node using information (bounds, etc.) about the status of the active nodes.

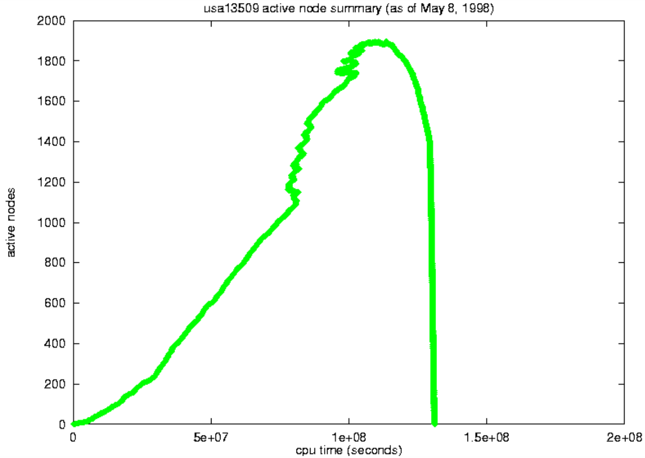
B&B - A very small tree



B&B - A large tree



B&B - Active nodes



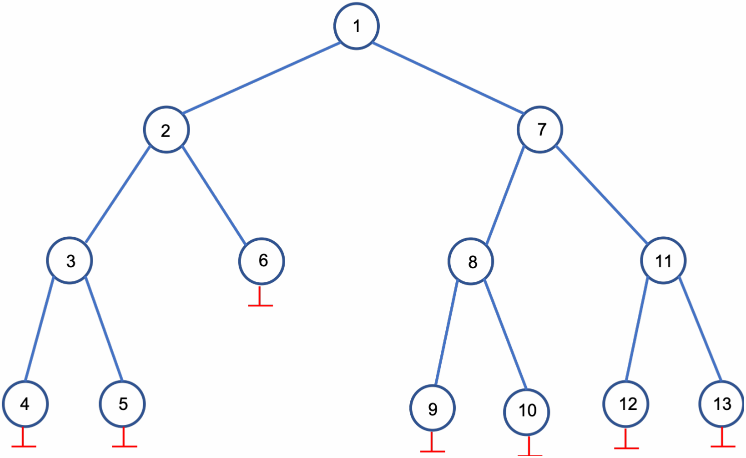
A priori rules

A widely used a priori rule is depth-first search plus backtracking.

- In depth-first search, if the current node is not pruned, the next node considered is one of its two sons.
- Backtracking means that when a node is pruned, we go back on the path from this node toward the root until we find the first node (if any) that has a son that has not yet been considered.

It is a completely a priori rule if we fix a rule for choosing branching variables and specify that, for instance, the left son is considered before the right son.

Depth-first search plus backtracking



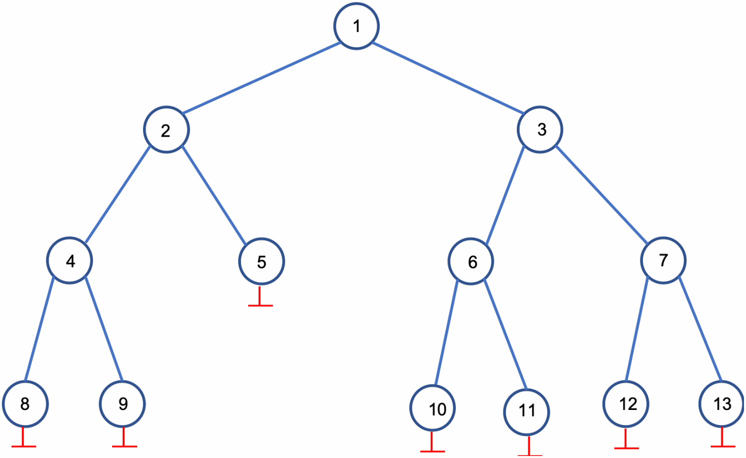
A priori rules

Definition

The level of a node in an enumeration tree is the number of edges in the unique path between it and the root.

In breadth-first search, all the nodes at a given level are considered before any nodes at the next lower level.

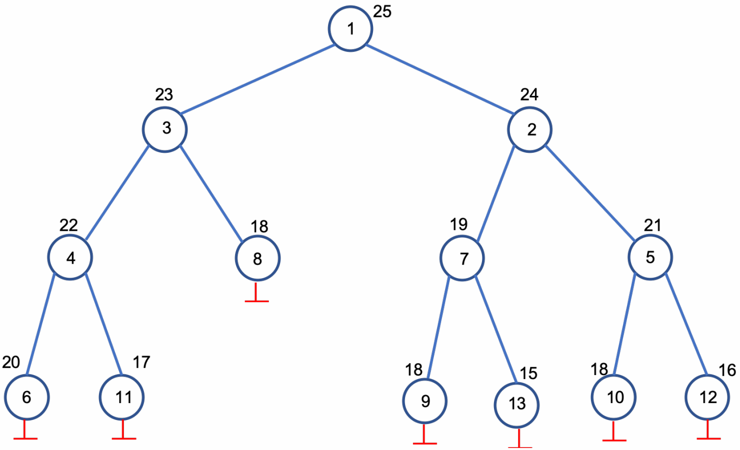
Breadth-first search



Adaptive rules

Best-first search: From all active nodes choose the one that has the largest upper bound. Thus if $\mathcal{L}$ is the set of active nodes, select an $i \in \mathcal{L}$ that maximises $\bar{z}^i$.

Best-first search



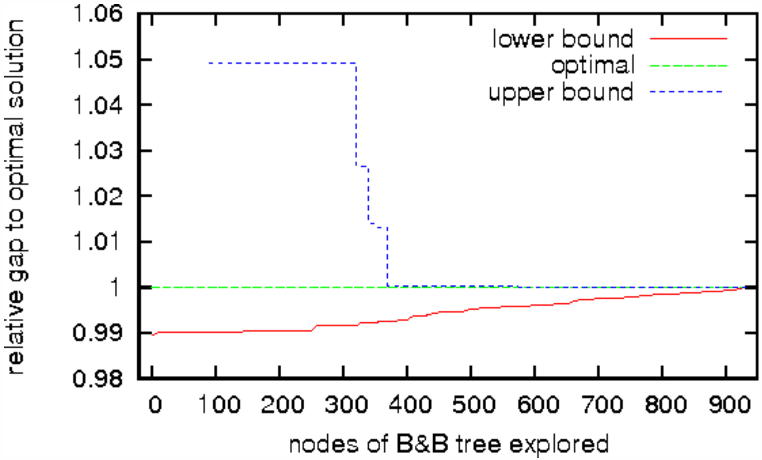
Best-first search

- $\boldsymbol{UB_1 = 25}$. Not pruned. Its sons have $\boldsymbol{UB = 24}$ and $\boldsymbol{UB = 23}$, $\boldsymbol{\mathcal{L} = \{23, 24\}}$
- $\boldsymbol{UB_2 = 24}$. Not pruned. Its sons have $\boldsymbol{UB = 21}$ and $\boldsymbol{UB = 19}$, $\boldsymbol{\mathcal{L} = \{19, 21, 23\}}$
- $\boldsymbol{UB_3 = 23}$. Not pruned. Its sons have $\boldsymbol{UB = 22}$ and $\boldsymbol{UB = 18}$, $\boldsymbol{\mathcal{L} = \{18, 19, 21, 22\}}$
- $\boldsymbol{UB_4 = 22}$. Not pruned. Its sons have $\boldsymbol{UB = 20}$ and $\boldsymbol{UB = 17}$, $\boldsymbol{\mathcal{L} = \{17, 18, 19, 20, 21\}}$
- $\boldsymbol{UB_5 = 21}$. Not pruned. Its sons have $\boldsymbol{UB = 18}$ and $\boldsymbol{UB = 16}$, $\boldsymbol{\mathcal{L} = \{16, 17, 18, 18, 19, 20\}}$
- $\boldsymbol{UB_6 = 20}$. Pruned. $\boldsymbol{\mathcal{L} = \{16, 17, 18, 18, 19\}}$
- $\boldsymbol{UB_7 = 19}$. Not pruned. Its sons have $\boldsymbol{UB = 18}$ and $\boldsymbol{UB = 15}$,$\boldsymbol{\mathcal{L} = \{15, 16, 17, 18, 18, 18\}}$
- $\boldsymbol{UB_8 = UB_9 = UB_{10} = 18}$. All pruned. $\boldsymbol{\mathcal{L} = \{15, 16, 17\}}$
- $\boldsymbol{UB_{11} = 17}$. Pruned. $\boldsymbol{\mathcal{L} = \{15, 16\}}$
- $\boldsymbol{UB_{12} = 16}$. Pruned. $\boldsymbol{\mathcal{L} = \{15\}}$
- $\boldsymbol{UB_{13} = 15}$. Pruned. $\boldsymbol{\mathcal{L} = \emptyset}$

Stopping criteria

- Ideally, the B&B algorithm stops when the optimal solution is found, i.e., when the list of active nodes is empty (i.e., $\mathcal{L} = \emptyset$)
- Practically, the tree can be so large that it is not possible to reach the condition $\mathcal{L} = \emptyset$.
- Some stopping criteria are
  - Time Run the algorithm for a predefined amount of time
  - Number of nodes The algorithm visits a predefined amount of nodes only.
  - Number of solutions found The algorithm stops when a predefined number of integer solutions is reached
  - Gap The algorithm stops at node t if $\overline{z}^t - \underline{z} \leq K$
  - Relative gap The algorithm stops at node t if $\frac{\overline{z}^t - \underline{z}}{\underline{z}} \leq \epsilon(\%)$

Relative gap - example 1



Relative gap - example 2