BRANCH-AND-BOUND ALGORITHMS FOR (MAX) IP

1. Generalities

A branch-and-bound algorithm for a given IP solves it by generating a family of related LPs.

Branching: This family is arranged in a *tree structure*. Each problem is represented by a *node* in the tree. A problem (parent node) may *branch* into a set of subproblems (children nodes) by subdividing the feasible region via the imposition of additional, mutually exclusive constraints, each represented by an *arc* in the tree. A node is said to be *fathomed* if no further branching from it can yield the solution of the original IP; otherwise it is said to be *active*. If a node has an integer-feasible solution, then this solution is a candidate solution of the IP.

Bounding: If the LP corresponding to a node is feasible, then its value is a lower bound for the value of the IP. The *current lower bound* is the minumum of the set of values of the nodes created so far that have an integer-feasible solution; if this set is empty, then the current lower bound is $-\infty$.

2. Step by step description

(1) Set CLB= $-\infty$. Form the first LPR. This is the first new node.
(2) For each new node: solve the LP and
   (a) if the LP is infeasible or unbounded, the node is fathomed;
   (b) if the solution is IF, the node is fathomed;
      - if the value of the node is > CLB, set CLB = value of the node, and declare the node to be the current candidate,
      - look at all active nodes, and prune those (if any) with value $\leq$ CLB.
   (c) if the optimal solution of the LP is not IF, then
      - if its value is $\leq$ CLB, then the node is inferior: prune it;
      - if its value > CLB, declare the node to be active.
(3) If there are no active nodes, STOP: if there is a current candidate it is optimal, and if there is no current candidate, the IP has no optimal solution (it's either infeasible or unbounded).
(4) If there are active nodes,
   - choose an active node with the highest value
   - choose an integer variable whose value is non-integer in the current LP solution, say $x_i = b_i$ where $b_i \notin \mathbb{Z}$
   - form two new nodes, one by adding the constraint $x_i \leq [b_i]$ to the current LP, and one by adding the constraint $x_i \geq [b_i] + 1$ to the current LP.
   - remove the parent node from the list of active nodes
   - go to 2.