Integer linear programs

Suppose that we have a *linear program*,

$$\text{[LP]} \qquad z^{LP} = \min \left\{ c^T x : Ax \geq b,\ x \geq 0 \right\},$$

where A is an $m \times n$-matrix, c is an n-vector, b is an m-vector, and x is an n-vector of variables. Adding the restriction that the variables must take integer values we get an *integer linear program*,

$$\text{[IP]} \qquad z^{IP} = \min \left\{ c^T x : Ax \geq b,\ x \geq 0,\ x \text{ integer} \right\}.$$

If all variables are restricted to the values 0 or 1, we have a *binary linear program*,

$$\text{[BP]} \qquad z^{BP} = \min \left\{ c^T x : Ax \geq b,\ x \in \{0,1\} \right\}.$$

Since integer linear programs look very much like linear programs, linear programming theory and practice are fundamental in the understanding and solution of integer linear programs. However, the simple idea of "rounding" the values of a solution to [LP] is almost never sufficient to find an optimal, or even good or just feasible, solution to [IP]. Consider the following [IP]:

$$\begin{array}{lrrcl} \min & -1.00x_1 & -0.64x_2 & & \\ \text{s.t.} & 50x_1 & +31x_2 & \leq & 250 \\ & 3x_1 & -2x_2 & \geq & -4 \\ & & x_1, x_2 & \geq & 0 \text{ and integer} \end{array}$$

The linear programming solution $x_{LP} = (\frac{376}{193}, \frac{950}{193}) \approx (1.95, 4.92)$ is far from the optimal integer solution $x_{IP} = (5,0)$, and neither rounding x_{LP} to $(2,5)$ nor truncating it to $(1,4)$ is even feasible. The feasible solution $x = (2,4)$, that is the "closest" one to x_{LP}, has objective value -4.56, which is far from optimal. The objective values at x_{LP} and x_{IP} are $-\frac{984}{193} \approx -5.10$ and -5, respectively.

Considering a binary linear program, an optimal solution to the corresponding linear program (where the constraints $x \in \{0,1\}$ are relaxed to $x \in [0,1]$) may very well be $(\frac{1}{2}, \ldots, \frac{1}{2})$, which tells absolutely nothing about an optimal solution that we did not know before.

A general procedure for solving integer linear programs to optimality is the branch–and–bound concept.

Branch–and–bound–algorithms

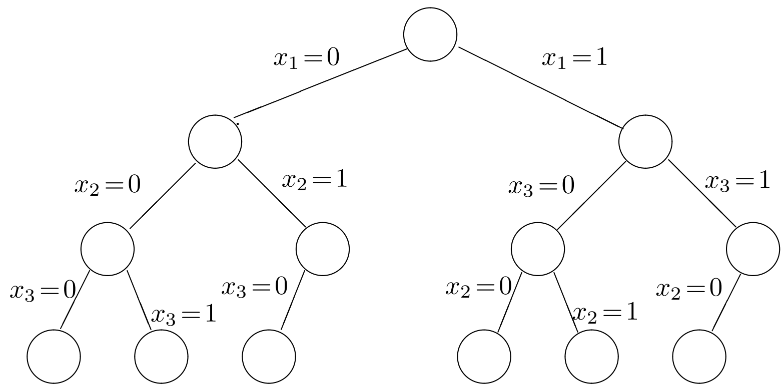
Branch–and–bound (B&B) is a general principle for finding optimal solutions to integer linear programs. It can be adapted to different types of models and it may also be combined with other (e.g., non-optimizing) algorithms. (Synonyms in the literature are, e.g., tree search and implicit enumeration.) Consider the integer linear program

$$\text{[IP]} \qquad \begin{array}{rcll} z^* & = & \min & c^T x \\ & & \text{s.t.} & x \in X, \end{array}$$

where, e.g., $X = \{x \in \Re^n \mid Ax \geq b,\ x \geq 0,\ x \text{ integer}\}$.

The basic idea of B&B is to enumerate all feasible solutions through the successive partition of X into a family of subsets. The enumeration is organized in a tree, using graph search, and is made implicit by utilizing estimates of z^*; these estimates are received from solutions to *relaxations* of [IP]. B&B algorithms have an exponential worst case–complexity, since it may occur that all feasible solutions are explicitly enumerated.

Example: Partition of a set of binary variables via branching: $X = \{\{0,1\}^3 \mid x_2 + x_3 \leq 1\}$. (In each branching one of the variables is fixed to 0 in one branch and to 1 in the other.)



Four concepts must be specificed to define a complete B&B–algorithm:

- Relaxation: a simplification of [IP] through the removal of constraints.
  - **Purpose:** to receive simple (i.e., polynomially solvable) subproblems, and optimistic estimates of z^*.
- Branching strategy: rules for partitioning of a subset of X.
  - **Purpose:** to exclude the optimal solution to a relaxation of [IP] if this solution is not feasible in [IP]; to shrink the subsets until only feasible solutions are left.
- Tree search strategy: defines the order in which the nodes in the B&B–tree are created and searched.
  - **Purpose:** to quickly find good feasible solutions; to restrict the size of the tree.
- Criteria for cutting off nodes: rules to decide whether or not a subset of X should be partitioned further.
  - **Purpose:** to avoid the search through branches that cannot contain a candidate to an optimal solution.

- **Relaxation**

  A relaxation of [IP] consists of removing som of its constrants, and by that actually enlarge the feasible set, or, letting $X^R \supset X$,

$$\text{[R-IP]} \qquad z^R = \min \ c^T x \quad \text{s.t.} \quad x \in X^R.$$

However, the main purpose of relaxation is to receive a more easily solveable so called subproblem. Examples of relaxations of [IP] are

- removal of integrality requirements $\Rightarrow$ linear continuous subproblems
- removal of complicating continuous constraints $\Rightarrow$ simpler subproblems
- Lagrangean relaxation of complicating continuous constraints $\Rightarrow$ simpler subproblems and possibility to get better optimistic estimates of z^*

If an optimal solution to a relaxation of [IP] problem is not feasible in the original [IP], then we try to fulfil the relaxed constraints through a branching in the B&B–tree.

The optimal value of a subproblem is an optimistic estimate of the value of any feasible solution that can be achieved in the corresponding branch in the tree, since each branching brings more restrictions.

The simplification of [IP] through the relaxation may not be too severe, because then one would probably have to make very many branchings to receive solutions to the relaxations that are feasible also in the original [IP], i.e., the tree would grow too large and deep.

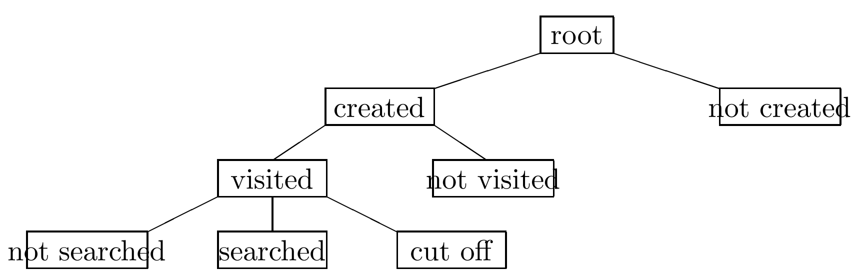
- **Branching strategy**

  A branching is used to strengthen a relaxation. The branching should correspond to a partition of the present subset of X^R, such that each feasible solution to [IP] can be found in exactly one branch of the tree (a proper partition).

  The branching must imply that the optimal solution to the subproblem, which is infeasible in the original problem, is cut off; however, no solution that is feasible in the original problem is allowed to be cut off.

- **Tree search strategy**

  The B&B-tree is not given in advance, but it is created during the execution of the algorithm. To construct the tree and search through its nodes, graph search is used.



Branching priority: indicates whether the subsequent nodes in the tree are to be created simultaneously or not, and the order in which they are created.

- Deep–first–order: only one of the successors of each node is created.
- Breadth–first–order: all the successors of a node are created simultaneously.

Visiting priority: indicates which of the active (i.e., created but not visited nor searched) nodes that is no be visited next. Hybrids exist, but mainly the following two are used:

- Deep–first–order (for both branching and visist): Yields quickly feasible solution-s; deep and narrow trees.
- Best–first–order: choose the active node that corresponds to the currently best optimistic estimate of z^*. This is the branch most likely to contain an optimal solution. Demands large memory; yields the smallest tree.

- **Criteria for cutting off nodes**

  If the solution x^P to a relaxed problem is not feasible in [IP], but has an objective value z^P that is lower (for a minimization problem) than the objective value $\overline{z}$ (pessimistic estimate) for the best known feasible solution to [IP], then the node P *may not* be cut off.

  Cutting rules: cut off a node P in the B&B–tree if the corresponding subproblem

  – contains no feasible solution,

  – has an optimal solution that is feasible in the original [IP] $\Rightarrow x^P$ is a a candidate for an optimal solution (a pessimistic estimate), or

  – has an optimal objective value, $z^P \geq \overline{z}$, where $\overline{z}$ is the objective value of the best known feasible solution to [IP].

  For none of these cases a feasible solution to [IP], having a lower objective value (for a minimization problem) than the best known (pessimistic estimate), can be found after a further branching at node P.

  To find pessimistic estimates of z^* (corresponding to feasible solutions to [IP]) heuristics are often used.

Solve the integer linear program below using the Branch & Bound–algorithm

$$
\begin{array}{llrcrcll}
\max & z = & 5x_1 & + & 3x_2 & & & \\
\text{då} & & 2x_1 & - & x_2 & \geq & 2 & (1) \\
& & -x_1 & + & 3x_2 & \geq & 3 & (2) \\
& & 5x_1 & + & 6x_2 & \leq & 60 & (3) \\
& & & & x_1, x_2 & \geq & 0,\ \text{integer} &
\end{array}
$$