Methods and Models for Combinatorial Optimization

Solution Methods for Integer Linear Programming

L. De Giovanni M. Di Summa G. Zambelli

Contents

1 Preliminary definitions

An *integer linear programming problem* is a problem of the form

$$\begin{array}{c} z_I = \max c^T x \\ Ax \leq b \\ x \geq 0 \\ x_i \in \mathbb{Z}, \qquad i \in I, \end{array} \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $I \subseteq \{1, \ldots, n\}$ is the index set of the *integer variables*. Variables x_i with $i \notin I$ are the *continuous variables*. If the problem has both integer and continuous variables, then it is a *mixed integer linear programming problem*, while if all variables are integer it is a *pure integer linear programming problem*.

The set

$$X = \{x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ for every } i \in I\}$$

is the *feasible region* of the problem.

$$\begin{array}{c} z_L = \max c^T x \\ Ax \leq b \\ x \geq 0 \end{array} \tag{2}$$

is called the *linear relaxation* (or *continuous relaxation*) of (1).

Note the following easy fact:

$$z_I \leq z_L. \tag{3}$$

This is because if x^I is the optimal solution of (1) and x^L is the optimal solution of (2), then x^I satisfies the constraints of (2), and thus $z_I = c^T x^I \leq c^T x^L = z_L$.

In the following we illustrate the basics of two methods that are widely used by integer linear programming solvers: the Branch-and-Bound method and the cutting plane method.

2 Branch-and-Bound for integer linear programming

The most common method for the solution of integer linear programming problems is called *Branch-and-Bound*. This method exploits the following simple observation:

> *Given a partition of the feasible region X into subsets $X_1, \ldots, X_p$, define $z_I^{(k)} = \max\{c^T x \mid x \in X_k\}$ for $k = 1, \ldots, p$. Then*
>
> $$z_I = \max_{k=1,\ldots,p} z_I^{(k)}.$$

The Branch-and-Bound method proceeds by partitioning X into smaller subsets and solving the problem $\max c^T x$ on every subset. This is done recursively, by further dividing the feasible regions of the subproblems in subsets. If this recursion were carried out completely, in the end we would enumerate all integer solutions of the problem. In this case, at least two issues would arise: first, if the problem has infinitely many feasible solutions the complete enumeration is not possible; second, even assuming that the feasible region contains a finite number of points, this number might be extremely large and thus the enumeration would require an unpractical amount of time.

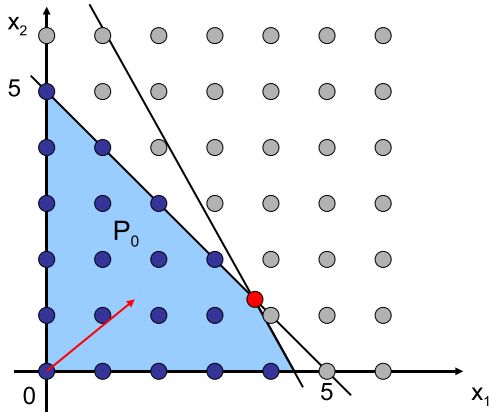
The Branch-and-Bound algorithm aims at exploring only the "promising" areas of the feasible region, by storing upper and lower bounds for the optimal value within a certain area and using these bounds to decide that certain subproblems do not need to be solved.

In the following we illustrate the method with an example. The general principles, which are valid for a wider class of combinatorial optimization problems, will be examined later.

Example Consider the following problem (P_0):

$$
\begin{array}{lrcll}
z_I^0 = \max & 5x_1 + \frac{17}{4}x_2 & & & \\
& x_1 + x_2 & \leq & 5 & \\
& 10x_1 + 6x_2 & \leq & 45 & (P_0) \\
& x_1, x_2 & \geq & 0 & \\
& x_1, x_2 & \in & \mathbb{Z} &
\end{array}
$$

The feasible region of (P_0) (blue points) and the feasible region of its linear relaxation (light blue quadrilateral) are represented in the next picture. (The arrow indicates the direction of optimization.)



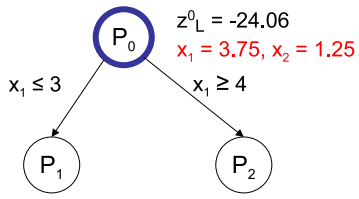
After solving the linear relaxation of (P_0), we obtain the optimal solution $x_1 = 3.75$, $x_2 = 1.25$ (red point), whose objective value is $z_L^0 = 24.06$.

We have thus obtained an upper bound for the optimal value z_I^0 of (P_0), namely $z_I^0 \leq 24.06$. Now, since x_1 must take an integer value in (P_0), the optimal solution has to satisfy either the condition $x_1 \leq 3$ or $x_1 \geq 4$. It follows that the optimal solution of (P_0) will be the better of the optimal solutions of the subproblems (P_1) and (P_2) defined as follows:

$$
\begin{array}{lrcl}
z_I^1 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& x_1 & \leq & 3 \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
\qquad (P_1)
$$

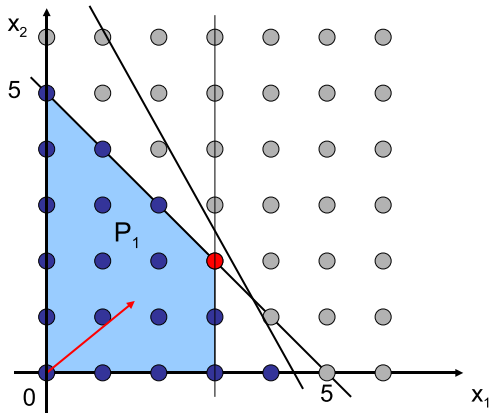
$$
\begin{array}{lrcl}
z_I^2 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& x_1 & \geq & 4 \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
\qquad (P_2)
$$

This operation is called a *branching on variable* x_1. Note that the solution $(3.75, 1.25)$ does not belong to the linear relaxation of (P_1) or (P_2). We can represent the subproblems and the corresponding bounds by means of a tree, called the Branch-and-Bound tree.



The leaves of the tree are the *active problems* (in our case, problems (P_1) and (P_2)).

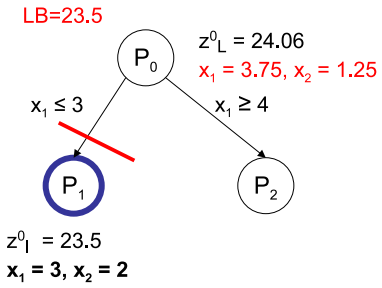
Consider problem (P_1), which is represented below.



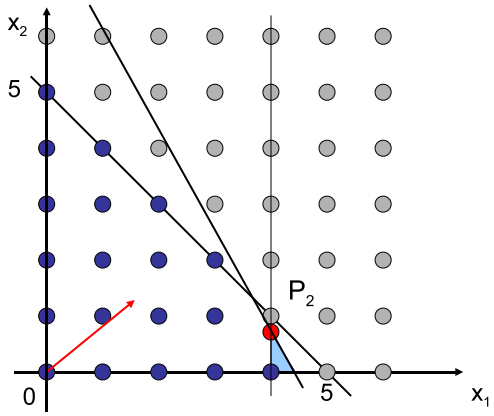
The optimal solution of the linear relaxation of (P_1) is $x_1 = 3$, $x_2 = 2$, with objective value $z_L^1 = 23.5$. Since this solution is integer, $(3, 2)$ is also the optimal integer solution of (P_1). For this reason, there is no need to branch on node (P_1), which can be pruned. We say that (P_1) is *pruned by optimality*. Also note that the optimal solution of (P_0) will necessarily have objective value $z_I^0 \geq z_I^1 = 23.5$. Therefore $LB = 23.5$ is a lower bound

for the optimal value, and $(3, 2)$ is called the *incumbent solution*, i.e., the best integer solution found so far.

The Branch-and-Bound tree is now the following.



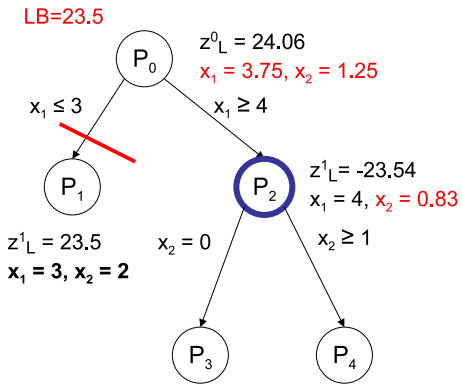
The only non-pruned leaf is (P_2), which therefore is the only active problem, and is represented below.



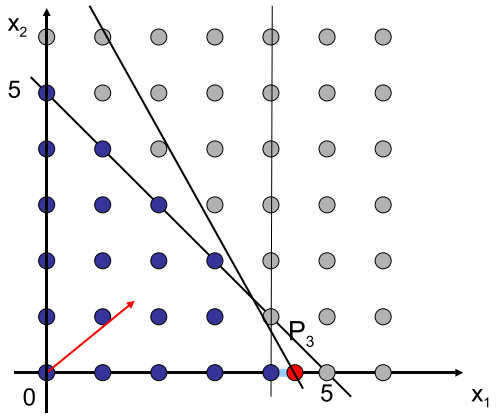
The optimal solution of the linear relaxation of (P_2) is $x_1 = 4$, $x_2 = 0.83$, with objective value $z_L^2 = 23.54$. Then $z_I^2 \leq 23.54$ and therefore 23.54 is an upper-bound for the optimal value of (P_2). Note that $LB = 23.5 < 23.54$, thus (P_1) might have a better solution than the incumbent solution. Since the value of x_2 is 0.83, which is not an integer, we branch on x_2, obtaining the subproblems (P_3) and (P_4) shown below.

$$
\begin{array}{lrcll}
z_I^2 = \max & 5x_1 + \frac{17}{4}x_2 & & & \\
& x_1 + x_2 & \leq & 5 & \\
& 10x_1 + 6x_2 & \leq & 45 & \\
& x_1 & \geq & 4 & (P_3) \\
& x_2 & \leq & 0 & \\
& x_1, x_2 & \geq & 0 & \\
& x_1, x_2 & \in & \mathbb{Z} &
\end{array}
\qquad
\begin{array}{lrcll}
z_I^2 = \max & 5x_1 + \frac{17}{4}x_2 & & & \\
& x_1 + x_2 & \leq & 5 & \\
& 10x_1 + 6x_2 & \leq & 45 & \\
& x_1 & \geq & 4 & (P_4) \\
& x_2 & \geq & 1 & \\
& x_1, x_2 & \geq & 0 & \\
& x_1, x_2 & \in & \mathbb{Z} &
\end{array}
$$

The Branch-and-Bound tree is now the following.

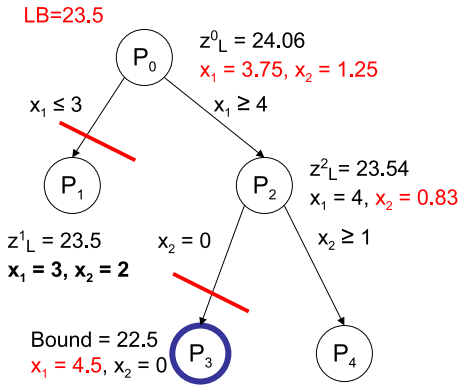


The active nodes are (P_3) and (P_4). If we solve the linear relaxation of (P_3) (see the picture below),

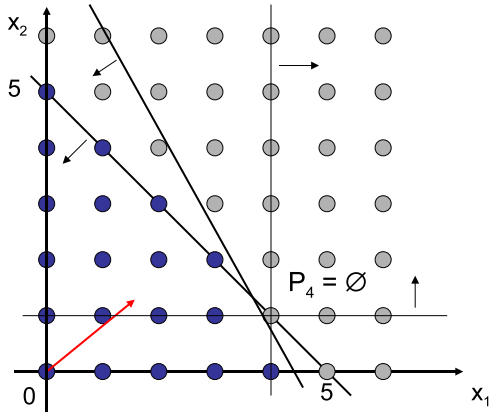


we find the optimal solution $x_1 = 4.5$, $x_2 = 0$, with objective value $z_L^3 = 22.5$. Then the value of the optimal integer solution of (P_3) must satisfy $z_I^3 \leq 22.5$, but since we have already found an integer solution with value 23.5 (which is a lower bound), we do not need to further explore the feasible region of (P_3), as we are sure that it cannot contain any integer solution with value larger than 22.5 (and $23.5 > 22.5$). We can then *prune node* (P_3) *by bound*.

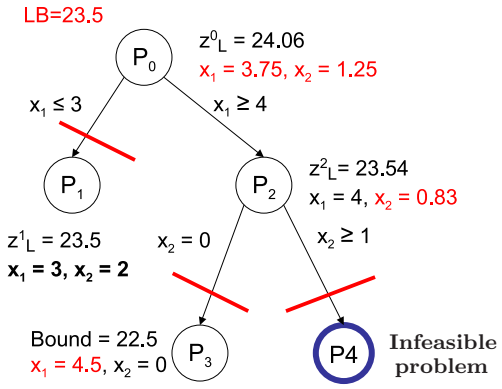
The current Branch-and-Bound tree, shown below, now contains a single active problem: (P_4).



By solving the linear relaxation of (P_4), we determine that there is no feasible solution in the linear relaxation, and therefore (P_4) contains no integer solution.



Node (P_4) can then be *pruned by infeasibility*. The Branch-and-Bound tree, shown below, does not have any active node and therefore the incumbent solution is the best integer solution of the problem; in other words, $(3, 2)$ is an optimal solution of (P_0).



Branch-and-Bound for (mixed) integer linear programming We now describe formally the Branch-and-Bound method. We want to solve problem (P_0):

$$\begin{aligned} z_I = \max\, & c^T x \\ & Ax \leq b \\ & x \geq 0 \\ & x_i \in \mathbb{Z}, \qquad i \in I. \end{aligned}$$

where I is, as usual, the index set of the integer variables.

The algorithm will store a lower bound LB for the optimal value z_I as well as the *incumbent solution*, i.e., the best integer solution x^* for (P_0) found so far (thus $x_i^* \in \mathbb{Z}$ for every $i \in I$, and $c^T x^* = LB$). A Branch-and-Bound tree $\mathcal{T}$ will be constructed, whose non-pruned leaves will be the *active nodes*. We denote by ℓ the maximum index of a node (P_i) in the Branch-and-Bound tree.

Branch-and-Bound method

Initialization: $\mathcal{T} := \{(P_0)\}$, $\ell := 0$, $LB := -\infty$, x^* not defined.

1. If there is an active node in $\mathcal{T}$, select an active node (P_k); otherwise return the optimal solution x^* and STOP.
2. Solve the linear relaxation of (P_k), thus determining either an optimal solution $x^{(k)}$ of value $z_L^{(k)}$, or the infeasibility of the problem.
   (a) If the linear relaxation of (P_k) is infeasible, prune (P_k) in $\mathcal{T}$ (*pruning by infeasibility*);
   (b) If $z_L^{(k)} \leq LB$, then (P_k) cannot have better solutions than the incumbent solution x^*; then prune (P_k) in $\mathcal{T}$ (*pruning by bound*);
   (c) If $x_i^{(k)} \in \mathbb{Z}$ for every $i \in I$, then $x^{(k)}$ is an optimal solution of (P_k) (and feasible for (P_0)), therefore
      - If $c^T x^{(k)} > LB$ (always true if (b) does not hold), set $x^* := x^{(k)}$ and $LB := c^T x^{(k)}$;
      - Prune (P_k) in $\mathcal{T}$ (*pruning by optimality*);
3. If none of cases (a), (b), (c) holds, then select an index $h \in I$ such that $x_h^{(k)} \notin \mathbb{Z}$, branch on variable x_h, and construct the following two children of (P_k) in $\mathcal{T}$:

$$(P_{\ell+1}) := (P_\ell) \cap \{x_h \leq \lfloor x_h^{(k)} \rfloor\} \quad , \quad (P_{\ell+2}) := (P_\ell) \cap \{x_h \geq \lceil x_h^{(k)} \rceil\}$$

   Make $(P_{\ell+1})$ and $(P_{\ell+2})$ active and (P_k) non-active. Set $\ell := \ell + 2$ and go to 1.

In the above algorithm and in the following, given a number a we denote by $\lfloor a \rfloor$ e $\lceil a \rceil$ the rounding-down and rounding-up of a to the closest integer, respectively.

There are many fundamental details to take care of in order to make a Branch-and-Bound method efficient. Here we examine only the following aspects.

Solution of the linear relaxation of every node The linear relaxation of any node corresponds to the linear relaxation of the parent node plus a single constraint. If the relaxation of the parent node has been solved with the simplex method, we know an optimal basic solution of the relaxation of the parent node. By using a variant of the simplex method called "dual simplex method", one can efficiently obtain an optimal solution for the same problem with a new constraint added. This feature allows for a fast exploration of the nodes of the Branch-and-Bound tree in (mixed) integer linear programming problems.

Selection of an active node Step 1 of the algorithm requires to select a node from the list of active nodes. The number of nodes that will be opened overall depends on how this list is handled; in particular, this depends on the criteria used to select an active node. In fact, there are two conflicting targets to keep in mind when choosing an active node:

- finding a (good) feasible integer solution as soon as possible. This brings at least two advantages: an integer solution provides a lower bound for the optimal value of the problem, and having a good lower bound increases the chances of pruning some nodes by bound. Furthermore, in the event that one needs to stop the algorithm before its natural termination, we have at least found a (good) feasible solution for the problem, though maybe not the optimal one;
- exploring a small number of nodes.

The above targets suggest the following strategies:

Depth-First-Search This corresponds to a LIFO (Last In First Out) strategy on the list of active nodes. With this strategy, new constraints are added one after the other at every branching and therefore it is likely to find an integer solution soon. Another advantage is that a small number of nodes are active and thus a small amount of memory is needed. The main drawback is that it is possible that "non-promising nodes" are explored, i.e., nodes that do not contain (good) integer solutions.

Best-Node In order to avoid visiting nodes not containing good integer solutions, one can select an active node such that its upper bound z_L^k is as large as possible, i.e., $z_L^k = \max_t z_L^t$, where the maximum is taken over the index set of active nodes. A drawback of this strategy is that many nodes have to be kept active and therefore a large amount of memory is needed.

In practice, a hybrid approach consists in starting with the Depth-First-Search strategy in order to find an integer solution as soon as possible (so that pruning by bound becomes possible), and then using the Best-Node strategy. Furthermore, integer linear programming solvers usually make use of heuristics to find an integer solution quickly before starting the Branch-and-Bound method.

Evaluation of feasible solutions In order to prune nodes by bound, good quality feasible solutions are needed. For this reason, when designing a Branch-and-Bound algorithm we have to decide how and when feasible solutions should be computed. There are several options, among which we mention the following:

- waiting for the enumeration to generate a node whose linear relaxation has an integer optimal solution;
- implementing a heuristic algorithm that finds a good integer solution before starting the exploration;
- exploiting (several times during the algorithm, with frequency depending on the specific problem) the information obtained during the exploration of the tree to construct better and better feasible solutions (e.g., by rounding the solution of the linear relaxation in a suitable way, so that a feasible integer solution is obtained).

In any case, the trade-off between the quality of the incumbent solution and the computational effort needed to obtain it has to be taken into account.

Stopping criteria The Branch-and-Bound method naturally stops when there are no active nodes left. In this case, the incumbent solution is an optimal integer solution. However, one can stop the algorithm when a given time limit or memory limit has been reached, but in this case the incumbent solution (if any has been found) is not guaranteed to be optimal. Nonetheless, it is possible to estimate the quality of this solution. Indeed, at any time during the construction of the Branch-and-Bound tree we know a lower bound LB (given by the value of the incumbent solution), but also an upper bound UB, given by the maximum of all values $z_L^{(k)}$ of the active nodes: this value is an optimistic estimation of the integer optimal value z_I (meaning that $z_I \leq UB$). If the algorithm is stopped before its natural termination, the difference between the value of the incumbent solution LB and the bound UB is an estimation of the quality of the incumbent solution. For this reason, a possible stopping criterion might be to terminate the algorithm when the difference between these two bounds is smaller than a given value (fixed in advance).

Choice of the branching variable There are several options for the choice of the branching variable, but a common one is to select the variable with the *most fractional* value, i.e., the variable whose fractional part is the closest to 0.5. In other words, if we define $f_i = x_i^{(k)} - \lfloor x_i^{(k)} \rfloor$, we choose $h \in I$ such that

$$h = \arg\min_{i \in I} \{\min\{f_i, 1 - f_i\}\}.$$

2.1 Branch-and-Bound: general principles

The Branch-and-Bound method for mixed integer linear programming described above is an application of a more general scheme, valid for every combinatorial optimization problem.

A **combinatorial optimization problem** is defined as follows:

$$\begin{array}{rl} \min / \max & f(x) \\ \text{s.t.} & x \in X \end{array}$$

where X is a *FINITE* set of points and $f(x)$ is a generic objective function. Examples of combinatorial optimization problems are the following:

- shortest path problem:
  $X = \{\text{all possible paths from } s \text{ to } t\}$, $f(x)$: cost of path $x \in X$;
- graph coloring:
  $X = \{\text{all feasible combinations of color assignments to the nodes of the graph}\}$, $f(x)$: number of colors used in the combination $x \in X$;
- linear programming:
  $X = \{\text{all feasible basic solutions}\}$, $f(x) = c^T x$.

A general method for finding an optimal solution among all the solutions in X is based on an enumeration scheme that exploit the *finiteness* of the space of feasible solutions. This scheme can be seen as a *universal algorithm for combinatorial optimization*:

1. generate all possible solutions x;
2. for every such x, verify whether $x \in X$;
3. if so, calculate $f(x)$;
4. choose a feasible x achieving the best possible value $f(x)$.

The above scheme is extremely simple, but has two clear disadvantages. First, calculating $f(x)$ is not always an easy task (for instance, there are problems in which for each x a simulation is needed to evaluate $f(x)$); second, *the cardinality of X might be* ***extremely*** *large*. This leads us to two questions:

1. how to *generate* the space of (feasible) solutions;
2. how to efficiently *explore* this space.

The branch-and-bound algorithm takes into account these issues when implementing the enumeration scheme seen above.

How to generate solutions: the branching operation We make the following observation:

> *Given a combinatorial optimization problem* $z = \max / \min\{f(x) : x \in X\}$ *and given a subdivision of the feasible region* X *into sets* $X_1, \ldots, X_p$ *such that* $\bigcup_{i=1}^{p} X_i = X$*, let* $z^{(k)} = \max / \min\{f(x) : x \in X_k\}$ *for* $k = 1, \ldots, p$*. Then the value of the optimal solution is* $z = \max / \min_{k=1,\ldots,p} z^{(k)}$.

We can then apply the principle of *divide et impera*: we split X into smaller subsets and we solve the problem on each subset. This is done recursively by in turn dividing the feasible region of the subproblems into smaller subsets. If this recursion were carried out completely, we would eventually enumerate all feasible solutions. The successive subdivisions of X can be represented by means of a tree, called the *tree of feasible solutions* (see Figure 1).

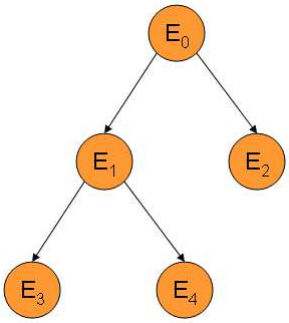


Figure 1: How to generate solutions: the *branching* operation.

Let P_0 be a given combinatorial optimization problem, with solution set $E_0 = X$. E_0 is the root of the tree and, more generally, E_i is the solution set associated with node i. The branching operation of a node E_i generates some *child* nodes. This subdivision must be such that

$$E_i = \bigcup_{j \text{ child of } i} E_j$$

i.e., every solution of the parent node must be in (at least) one of its children. In other words, during the construction of the tree no solution can be lost (otherwise the above observation would not hold). A desirable, but not necessary, condition is the disjointness of the child nodes (in this case the subdivision would be a partition of the parent node):

$$E_j \cap E_k = \emptyset, \ \forall\ j, k \text{ children of } i.$$

The subdivisions would terminate when every node contains only one solution. Therefore, if E_f is a leaf node, one has $|E_f| = 1$.

The construction of child nodes to subdivide a parent node is called *BRANCHING*: a set of level h is subdivided into t sets of level $h+1$.

Example 1 *Consider a combinatorial optimization problem with n binary variables $x_i \in \{0,1\}$, $i = 1, \ldots, n$.*

In this case, given a problem P_i and the corresponding set of feasible solutions E_i, we can easily obtain two subproblems and two subsets of E_i by fixing one of the binary variables to 0 for one subproblem and to 1 for the other subproblem. With this binary *branching rule* (in which every node is subdivided into two child nodes), we obtain a tree as in Figure 2.

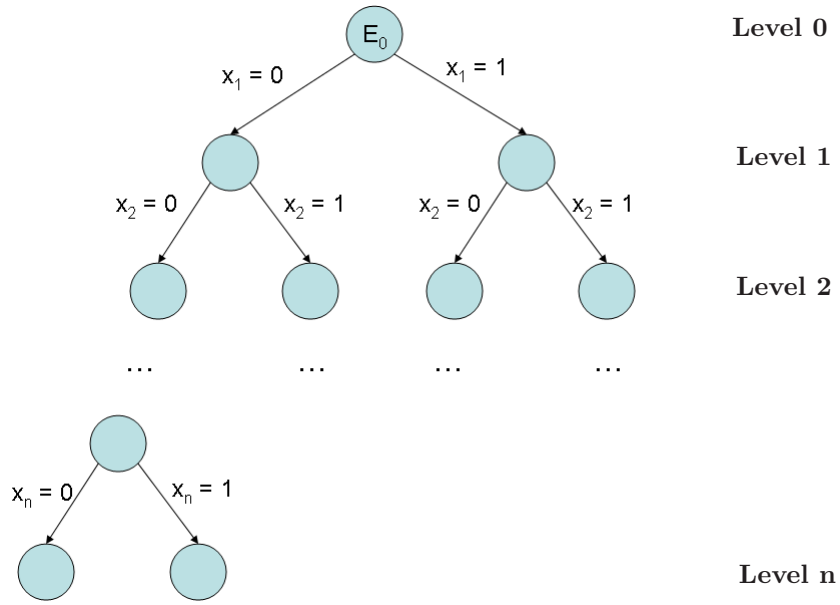


Figure 2: Binary branching.

At every level, one of the variables is fixed to 0 or 1. Then a node of level h contains all the feasible solutions with variables $x_1, \ldots, x_h$ fixed at some precise value, and therefore all nodes are disjoint. The leaves are at level n, where all variables have been fixed: every leaf represents one of the possible binary strings of length n, i.e., one specific solution of the problem. Note that the number of leaves is 2^n and the number of levels is $n+1$ (including the root, whose level is 0).

How to efficiently explore: the bounding operation In general, the number of leaves yielded by the branching operations is exponentially large. For this reason, the complete construction of the tree of feasible solutions (which corresponds to enumerating all feasible solutions) cannot be carried out in practice. We then need a method that allows us to explore only "good" areas of the feasible region, ensuring that the optimal solution does not lie in one of the other areas. To do so, for every node we consider a *BOUND*, i.e., an *optimistic* estimation of the value that the objective value can take in

the region represented by that node. For instance, given a *minimization* problem with binary variables, suppose that we know a feasible solution of value 10. With the initial feasible region we associate node E_0 and then develop the first level of the branching tree by fixing variable x_1 to 0 and 1, thus obtaining two child nodes E_1 and E_2 (see Figure 3).

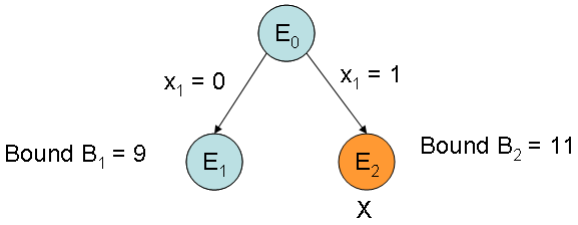


Figure 3: Use of bounds.

Let $z^{(1)}$ be the optimal value of the objective function in E_1 and $z^{(2)}$ be the optimal value of the objective function in E_2. As already observed, the optimal value of the objective function for the initial problem is

$$z = \min\{z^{(1)}, z^{(2)}\}.$$

Suppose that, without enumerating all possible solutions, we have somehow established that every solution such that $x_1 = 0$ has objective value at least 9. This means that the objective function value over E_1 is at least 9. Then 9 is an optimistic estimation of the objective function in E_1, i.e., $z^{(1)} \geq 9$. Similarly, suppose that we have a *lower bound* of 11 for E_2: no solution satisfying $x_1 = 1$ (solutions in E_1) has objective value smaller than 11: in other words, $z^{(2)} \geq 11$. Our first observation says that $z = \min\{z^{(1)}, z^{(2)}\}$. Furthermore, by using the information of the known feasible solution, we have $z \leq 10$; since $z^{(2)} \geq 11 \geq 10$, it is not possible to find a solution with objective value larger than 10 in E_2. Therefore

> *E_2 does not contain the optimal solution and thus we do not need to develop and explore the subtree rooted at E_2.*

A similar argument does not hold for node E_1: one of the solutions in this node *might* have objective value smaller than 10 and therefore this node might contain an optimal solution.

In general, if a feasible solution $\bar{x}$ of value $f(\bar{x}) = \bar{f}$ is known, the availability of a *bound* associated with the nodes of the branching tree allows us to "prune" (i.e., not develop) subtrees that are guaranteed to contain no optimal solution, i.e., the subtrees rooted at nodes whose bound is not better than $\bar{f}$.

Note that, even though the leaf nodes of the subtree rooted at E_2 have not been constructed, we know that none of them contains a solution of value better than 11, and since

we have a feasible solution of value 10, this information is sufficient to exclude them as possible optimal solutions: we have *implicitly* explored the subtree. Therefore, the bounding operation allows us to perform an *implicit enumeration* of all the feasible solutions of a combinatorial optimization problem.

The Branch-and-Bound method: general scheme From the above observations, it is possible to define the Branch-and-Bound method for the solution of combinatorial optimization problems. This is an algorithm that enumerates explicitly or implicitly all the feasible solutions, based on the following operations:

- *branching operation*: construction of the tree of feasible solutions;
- availability of a feasible solution of value $\bar{f}$;
- *bounding operation*: optimistic estimation of the objective function for the solutions represented by each node (*bound*), in order to avoid the complete development of all subtrees (implicit enumeration of the solutions represented by nodes whose bound is not better than $\bar{f}$).

The Branch-and-Bound method can be summarized as follows. Given a combinatorial optimization problem $z = \min / \max\{f(x) : x \in X\}$, define:

- P_0: the initial optimization problem;
- L: the list of open nodes: every node is represented as a pair (P_i, B_i), where P_i is the subproblem and B_i is the corresponding bound;
- $\bar{z}$: the value of the best feasible solution found so far;
- $\bar{x}$: the best feasible solution found so far (incumbent solution).

Branch-and-Bound method

0.	*Initialization*:	Compute an optimistic estimation B_0 of the objective function and set $L = \{(P_0, B_0)\}$, $\bar{x} = \emptyset$, $\bar{z} = +\infty(\min)[-\infty(\max)]$.
1.	*Stopping criterion*:	If $L = \emptyset$, then STOP: $\bar{x}$ is an optimal solution. If time limit exceeded, or number of nodes exceeded, or number of open nodes $\|L\|$ exceeded, etc. STOP: $\bar{x}$ is a feasible solution (not necessarily optimal).
2.	*Node selection*:	Select $(P_i, B_i) \in L$ for the branching operation.
3.	*Branching*:	Subdivide P_i into t subproblems $P_{ij}, j = 1, \ldots, t$ such that $\bigcup_j P_{ij} = P_i$.
4.	*Bounding*:	Compute an optimistic estimation B_{ij} (corresponding to a —perhaps not feasible— solution x_{ij}^R) for each subproblem P_{ij}.
5.	*Pruning*:	If P_{ij} is not feasible, go to 1. If B_{ij} is not better than $\bar{z}$, go to 1. If x_{ij}^R is feasible and better than $\bar{z}$, set $\bar{z} \leftarrow B_{ij}$, $\bar{x} \leftarrow x_{ij}^R$; delete from L all nodes k with L_k not better than $\bar{z}$; go to 1. Otherwise, add (P_{ij}, B_{ij}) to L and go to 1.

The above is a generic solution scheme for combinatorial optimization problems. To implement a Branch-and-Bound algorithm for a specific problem, as we did for mixed integer linear programming problems, the following elements should be determined:

(1) Branching rules: how to construct the tree of feasible solutions.

(2) Bound computation: how to estimate node bounds.

(3) Pruning rule: how to close (i.e., prune) nodes.

(4) Exploration rules: defining priorities in the exploration of the open nodes.

(5) How to compute the value of one or more feasible solutions (these solutions will be compared with bounds in order to close nodes).

(6) Stopping criteria: conditions for the termination of the algorithm.

The strategy for each of the above points depends on the specific problem.

3 Alternative formulations

Consider an integer linear programming problem:

$$\begin{aligned} z_I = \max\, & c^T x \\ Ax &\leq b \\ x &\geq 0 \\ x_i &\in \mathbb{Z}, \qquad i \in I \end{aligned} \tag{4}$$

where $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$ and $I \subseteq \{1, \ldots, n\}$ is the index set of the integer variables.[1] Let

$$X = \{x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ for every } i \in I\}$$

be the feasible region of the problem, and let

$$\begin{aligned} z_L = \max\, & c^T x \\ Ax &\leq b \\ x &\geq 0 \end{aligned} \tag{5}$$

be the linear relaxation of (4).

Notice that the linear relaxation is not unique. Given a matrix $A' \in \mathbb{R}^{m' \times n}$ and a vector $b' \in \mathbb{R}^{m'}$, we say that

$$\begin{aligned} A'x &\leq b' \\ x &\geq 0 \end{aligned}$$

is a *formulation* for X if

$$X = \{x \in \mathbb{R}^n \mid A'x \leq b',\, x \geq 0,\, x_i \in \mathbb{Z} \text{ for every } i \in I\}.$$

In this case, the linear programming problem

$$\begin{aligned} z'_L = \max\, & c^T x \\ A'x &\leq b' \\ x &\geq 0 \end{aligned} \tag{6}$$

is a linear relaxation of (4), as well. It is clear that X can have infinitely many possible formulations, and therefore there are infinitely many possible linear relaxations for (4).

[1] From now on all coefficients will be rational, as this condition is essential to ensure some of the properties illustrated in this section, which do not always hold if the coefficients are irrational. (After all, we are interested in implementing algorithms, and therefore the rationality assumption does not pose any practical restriction.

3.1 Good formulations

Given two formulations for X,

$$Ax \leq b,\ x \geq 0$$

and

$$A'x \leq b,\ x \geq 0,$$

we say that the first formulation is better than the second one if

$$\{x \in \mathbb{R}^n \mid Ax \leq b,\ x \geq 0\} \subseteq \{x \in \mathbb{R}^n \mid A'x \leq b',\ x \geq 0\}.$$

This notion is justified by the fact that, if $Ax \leq b,\ x \geq 0$ is a better formulation than $A'x \leq b',\ x \geq 0$, then

$$z_I \leq z_L \leq z'_L,$$

and therefore the linear relaxation given by $Ax \leq b,\ x \geq 0$ yields a tighter bound on the optimal value of the integer problem than the bound given by the linear relaxation $A'x \leq b',\ x \geq 0$.

Recall that having good bounds is essential for visiting only a small number of nodes of the Branch-and-Bound tree.

Example 2 *Facility location.*
There are n possible locations where facilities can be opened to provide some service to m customers. If facility i, $i = 1, \ldots, n$, is opened, a fixed cost f_i must be paid. The cost for serving customer j with facility i is c_{ij} for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Every customer must be served from exactly one facility (but the same facility can serve more customers). The problem is to decide which facilities should be opened and to assign each customer to a facility at the minimum total cost.

We have the following decision variables:

$$y_i = \begin{cases} 1 & \text{if facility } i \text{ is opened,} \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, n.$$

$$x_{ij} = \begin{cases} 1 & \text{if facility } i \text{ serves customer } j, \\ 0 & \text{otherwise} \end{cases} \qquad i = 1, \ldots, n,\ j = 1, \ldots, m.$$

The total cost is

$$\sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

which is the objective function to minimize.

The fact that each customer should be served by exactly one facility can be modeled by the constraints

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, m.$$

Finally, customer j can be served by facility i only if i will be opened, and therefore we have to model the following conditions:

$$y_i = 0 \Rightarrow x_{ij} = 0, \quad i = 1, \ldots, n,\ j = 1, \ldots, m.$$

This can be modeled by linear constraints in at least two ways:

Method 1:

$$x_{ij} \leq y_i, \quad i = 1, \ldots, n,\ j = 1, \ldots, m.$$

Thus, if $y_i = 0$, then $x_{ij} = 0$ for all j, while if $y_i = 1$, the above constraints do not pose any limitation on the x_{ij} variables.

Method 2:

$$\sum_{j=1}^{m} x_{ij} \leq m y_i, \quad i = 1, \ldots, n.$$

Note that, if $y_i = 0$, then $x_{ij} = 0$ for all j, while if $y_i = 1$ then the constraint becomes $\sum_{j=1}^{m} x_{ij} \leq m$, which does not pose any restriction on the x_{ij} variables, as the sum of m binary variables is always $\leq m$.

The constraints in Method 1 are called *non-aggregated constraints*, those in Method 2 are called *aggregated constraints*. We then have two possible formulations for the problem.

With the constraints of Method 1, the model is:

$$\begin{array}{ll} \min \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} & \\ \sum_{i=1}^{n} x_{ij} = 1, & j = 1, \ldots, m; \\ x_{ij} \leq y_i, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq x_{ij} \leq 1, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq y_i \leq 1, & i = 1, \ldots, n; \\ x_{ij} \in \mathbb{Z},\ y_i \in \mathbb{Z} & i = 1, \ldots, n,\ j = 1, \ldots, m. \end{array} \tag{7}$$

With the constraints of Method 2, the model is:

$$\begin{array}{ll} \min \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} & \\ \sum_{i=1}^{n} x_{ij} = 1, & j = 1, \ldots, m; \\ \sum_{j=1}^{m} x_{ij} \leq m y_i, & i = 1, \ldots, n; \\ 0 \leq x_{ij} \leq 1, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq y_i \leq 1, & i = 1, \ldots, n; \\ x_{ij} \in \mathbb{Z},\ y_i \in \mathbb{Z} & i = 1, \ldots, n,\ j = 1, \ldots, m. \end{array} \tag{8}$$

The first formulation has more constraints than the second one (there are mn non-aggregated constraints and only n aggregated constraints). We now verify that the first formulation is better than the second one. To see this, let P_1 be the set of points (x, y) that satisfy

$$\begin{array}{ll} \sum_{i=1}^{n} x_{ij} = 1, & j = 1, \ldots, m; \\ x_{ij} \leq y_i, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq x_{ij} \leq 1, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq y_i \leq 1, & i = 1, \ldots, n; \end{array}$$

and let P_2 be the set of points (x, y) that satisfy

$$\begin{array}{ll} \sum_{i=1}^{n} x_{ij} = 1, & j = 1, \ldots, m; \\ \sum_{j=1}^{m} x_{ij} \leq m y_i, & i = 1, \ldots, n; \\ 0 \leq x_{ij} \leq 1, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq y_i \leq 1, & i = 1, \ldots, n. \end{array}$$

We show that $P_1 \subsetneq P_2$.

First of all we verify that $P_1 \subseteq P_2$. To do so, we show that if $(x, y) \in P_1$ then $(x, y) \in P_2$. If $(x, y) \in P_1$, then $x_{ij} \leq y_i$ for all $i = 1, \ldots, n$, $j = 1, \ldots, m$. Then for every fixed $i \in \{1, \ldots, n\}$, the sum of the m inequalities $x_{ij} \leq y_i$ for $j = 1, \ldots, m$ gives $\sum_{j=1}^{m} x_{ij} \leq m y_i$, and therefore $(x, y) \in P_2$.

Finally, in order to show that $P_1 \neq P_2$, it is sufficient to find a point in $P_2 \setminus P_1$. Take $n = 2$, $m = 4$, and consider the point given by

$$x_{11} = 1,\ x_{12} = 1,\ x_{13} = 0,\ x_{14} = 0;$$

$$x_{21} = 0,\ x_{22} = 0,\ x_{23} = 1,\ x_{24} = 1;$$

$$y_1 = \frac{1}{2},\ y_2 = \frac{1}{2}.$$

This point satisfy the aggregated constraints but violates the non-aggregated constraints, because $1 = x_{11} \not\leq y_1 = \frac{1}{2}$. ■

Example 3 *Multi-period production.*
A company is planning the production over a time horizon of n periods. For each period $t = 1, \ldots, n$, the company knows an estimation of the demand d_t. If production takes place in period t, then the company pays a fixed cost f_t (this cost is independent of the amount produced) and a cost c_t for each unit produced. Moreover, part of the amount produced can be stored in stock at the cost of h_t for each unit of product stored from period t to period $t + 1$, for $t = 1, \ldots, n - 1$.

The company wants to decide how much to produce in each period and how much to store in stock at the end of each period, so as to satisfy the demand and minimize the total cost.

For every period $t = 1, \dots, n$, let x_t be a variable indicating the amount produced in period t. Let y_t be a binary variable taking value 1 if production takes place in period t, 0 otherwise. Finally, let s_t be the amount stored in stock at the end of period t. For ease of notation, we define $s_0 = 0$.

The total cost is

$$\sum_{t=1}^{n} c_t x_t + \sum_{t=1}^{n} f_t y_t + \sum_{t=1}^{n-1} h_t s_t,$$

which is the objective function to minimize.

In every period t, the available amount of product is $s_{t-1} + x_t$, of which d_t unit are used to satisfy the demand in period t and the rest is stored in stock. Then

$$s_{t-1} + x_t = d_t + s_t, \quad t = 1, \dots, n.$$

Of course,

$$s_t \geq 0,\ t = 1, \dots, n-1; \quad x_t \geq 0,\ t = 1, \dots, n.$$

Finally, we have to ensure that if production takes place in period t then $y_t = 1$. This can be enforced as follows:

$$x_t \leq M y_y \quad t = 1, \dots, n,$$

where M is an upper bound on the maximum value that x_t can take. For instance, one can take $M = \sum_{t=1}^{n} d_t$. We also have to impose

$$0 \leq y_t \leq 1 \quad t = 1, \dots, n$$

$$y_t \in \mathbb{Z} \quad t = 1, \dots, n.$$

The optimal solution of the continuous relaxation of this formulation can have fractional components. As an example, assume that the storage cost are very high compared to the other costs. Then in the optimal solution we have $s_t = 0$ for every $t = 1, \dots, n-1$. In this situation, the demand is satisfied at the minimum cost if $x_t = d_t$ for every $t = 1, \dots, n$. Now, with integrality restriction we would have $y_t = 1$ for every $t = 1, \dots, n$. However, in the continuous relaxation we will have $y_t = d_t/M < 1$.

It is possible to write a better formulation by using additional variables. For every period $i = 1, \dots, n$ and every period $j = i, \dots, n$, let w_{ij} be the amount produced in period i used to satisfy (part of) the demand in period j.

Then the total amount sold in period t is given by $\sum_{i=1}^{t} w_{it}$, and therefore we have to impose

$$\sum_{i=1}^{t} w_{it} \geq d_t \quad t = 1, \dots, n.$$

It is also clear that

$$x_t = \sum_{j=t}^{n} w_{tj} \quad t = 1, \ldots, n$$

and

$$s_t = \sum_{i=1}^{t} \sum_{j=t+1}^{n} w_{ij} \quad t = 1, \ldots, n-1.$$

Finally,

$$w_{ij} \leq d_j y_i, \quad i = 1, \ldots, n,\ j = i, \ldots, n$$

and $y_t \in \{0, 1\}$ for $t = 1, \ldots, n$, to force y_t to take value 1 whenever $x_t > 0$.

Let P_1 be the set of points (x, y, s) that satisfy

$$\begin{array}{rl} s_{t-1} + x_t = d_t + s_t & t = 1, \ldots, n; \\ x_t \leq M y_y & t = 1, \ldots, n; \\ x_t \geq 0 & t = 1, \ldots, n; \\ s_t \geq 0 & t = 1, \ldots, n-1; \\ 0 \leq y_t \leq 1 & t = 1, \ldots, n \end{array}$$

and let P_2 be the set of points (x, y, s) for which there exists some w such that (x, y, s, w) satisfies

$$\begin{array}{rl} \sum_{i=1}^{t} w_{it} \geq d_t & t = 1, \ldots, n; \\ x_t = \sum_{j=t}^{n} w_{tj} & t = 1, \ldots, n; \\ s_t = \sum_{i=1}^{t} \sum_{j=t+1}^{n} w_{ij} & t = 1, \ldots, n-1; \\ w_{ij} \leq d_j y_i, & i = 1, \ldots, n, j = i, \ldots, n; \\ w_{ij} \geq 0 & i = 1, \ldots, n, j = i, \ldots, n; \\ 0 \leq y_t \leq 1 & t = 1, \ldots, n. \end{array}$$

One can verify that every point in P_2 satisfies also the constraints of P_1, thus $P_2 \subseteq P_1$. On the other hand, the point $s_t = 0$ $(t = 1, \ldots, n-1)$, $x_t = d_t$ $(t = 1, \ldots, n)$, $y_t = d_t/M$ $(t = 1, \ldots, n)$ is in P_1 but not in P_2. To see this, note that the only w satisfying $x_t = \sum_{j=t}^{n} w_{tj}$ $(t = 1, \ldots, n)$ is given by $w_{tt} = d_t$ for $t = 1, \ldots, n$ and $w_{ij} = 0$ for $1 \leq i < j \leq n$. However, this point violates the constraint $w_{tt} \leq d_t y_t$. This shows that $P_2 \subsetneq P_1$. ■

3.2 Ideal formulation

We define as the *ideal formulation for* X the formulation for X whose continuous relaxation is as small as possible (with respect to set inclusion). In the following we formalize this notion.

Definition 1 *A set* $P \subseteq \mathbb{R}^n$ *is a* polyhedron *if there exists a system of linear inequalities* $Cx \leq d$, $x \geq 0$ *(where* $C \in \mathbb{R}^{m \times n}$ *and* $d \in \mathbb{R}^m$*) such that* $P = \{x \,|\, Cx \leq d,\ x \geq 0\}$.

We can then say that a polyhedron P is a formulation for the set X if

$$X = \{x \in P \,|\, x_i \in \mathbb{Z}\ \forall i \in I\}.$$

Given tow polyhedra P and P', both being a formulation for X, we say that P is a better formulation than P' if $P \subset P'$.

Recall that a set $C \subseteq \mathbb{R}^n$ is *convex* if, for every pair of points $x, y \in C$, the line segment joining x and y is fully contained in C (Figure 4). It is easy to verify that every polyhedron is a convex set.

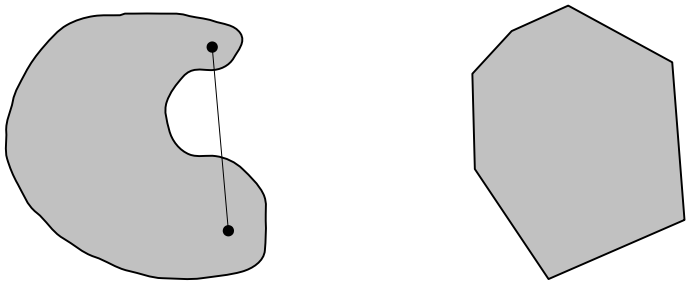


Figure 4: A convex set and a non-convex set.

Given any set $X \subseteq \mathbb{R}^n$ (in our case X is the set of feasible solutions of an integer linear programming problem), we denote by $\text{conv}(X)$ the *convex hull* of X, i.e., the smallest convex set containing X (Figure 5). In other words, $\text{conv}(X)$ is the unique convex set in $\mathbb{R}^n$ such that $X \subseteq \text{conv}(X)$ and $\text{conv}(X) \subseteq C$ for every convex set C containing X.

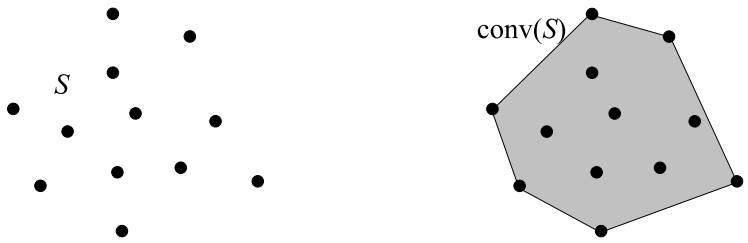


Figure 5: A set and its convex hull.

Given a formulation $P = \{x \,|\, Cx \leq d,\ x \geq 0\}$ for X, since P is a convex set containing X, we have that

$$X \subseteq \text{conv}(X) \subseteq P.$$

Then $\text{conv}(X)$ is contained in the feasible region of the continuous relaxation of every formulation of X.

The following is a fundamental result in integer linear programming. It shows that there exists a formulation for X whose continuous relaxation is precisely $\text{conv}(X)$.

Theorem 1 (Fundamental Theorem of Integer Linear Programming) *Given $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$, let $X = \{x \in \mathbb{R}^n \mid Ax \leq b,\ x \geq 0,\ x_i \in \mathbb{Z}$ for every $i \in I\}$. Then* $\mathrm{conv}(X)$ *is a polyhedron.*
In other words, there exist a matrix $\tilde{A} \in \mathbb{Q}^{\tilde{m} \times n}$ and a vector $\tilde{b} \in \mathbb{Q}^{\tilde{m}}$ such that

$$\mathrm{conv}(X) = \{x \in \mathbb{R}^n \mid \tilde{A}x \leq \tilde{b},\ x \geq 0\}.$$

Given $\tilde{A} \in \mathbb{Q}^{\tilde{m} \times n}$ and $\tilde{b} \in \mathbb{Q}^{\tilde{m}}$ such that $\mathrm{conv}(X) = \{x \in \mathbb{R}^n \mid \tilde{A}x \leq \tilde{b},\ x \geq 0\}$, we say that $\tilde{A}x \leq \tilde{b},\ x \geq 0$ is the ***ideal formulation*** for X. The previous theorem says that such a formulation always exists.

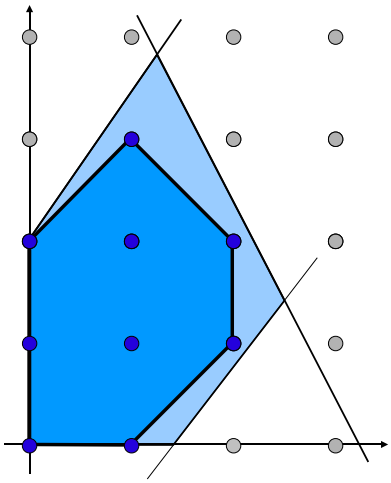


Figure 6: A formulation for a set of integer points and the ideal formulation.

We now consider again our integer linear programming problem:

$$\begin{aligned} z_I = \max\ & c^T x \\ & x \in X \end{aligned} \tag{9}$$

where $X = \{x \in \mathbb{R}^n \mid Ax \leq b,\ x \geq 0,\ x_i \in \mathbb{Z}$ for every $i \in I\}$, for some given matrix $A \in \mathbb{Q}^{m \times n}$ and vector $b \in \mathbb{Q}^m$.

Let $\tilde{A}x \leq \tilde{b}$ the ideal formulation for X, and consider the following linear programming problem:

$$\begin{aligned} \tilde{z} = \max\ & c^T x \\ & \tilde{A}x \leq \tilde{b} \\ & x \geq 0. \end{aligned} \tag{10}$$

We need to extend the notion of basic solution to a system of the above type, which is not in standard form. By adding slack variables, we obtain

$$\begin{aligned} \tilde{z} = \max\ & c^T x \\ & \tilde{A}x + Is = \tilde{b} \\ & x \geq 0,\ s \geq 0. \end{aligned} \tag{11}$$

From the theory of linear programming, we know that there exists an optimal solution (x^*, s^*) for (11) that is a basic solution of the system. By construction, $s^* = \tilde{b} - \tilde{A}x^*$. If (x^*, s^*) is a basic solution of $\tilde{A}x + Is = \tilde{b}$, $x, s \geq 0$, then we say that x^* is a basic solution for the system $\tilde{A}x \leq \tilde{b}$, $x \geq 0$. Therefore there exists an optimal solution of (10) that is a basic solution of the system $\tilde{A}x \leq \tilde{b}$, $x \geq 0$.

Theorem 2 *Let $X = \{x \in \mathbb{R}^n \,|\, Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ for every } i \in I\}$. The system $\tilde{A}x \leq \tilde{b}$, $x \geq 0$ is the ideal formulation of X if and only if all its basic solutions are elements of X. In particular, $z_I = \tilde{z}$ for every cost vector $c \in \mathbb{R}^n$.*

The above theorem implies that solving (10) is equivalent to solving (9). This is because, given a basic optimal solution x^* for (10) (thus $\tilde{z} = c^T x^*$), by the theorem $x^* \in X$, and therefore x^* is feasible for the integer linear programming problem (9), implying that $\tilde{z} = c^T x^* \leq z_I$; but (10) is a continuous relaxation of (9), thus the inequality $z_I \leq \tilde{z}$ also holds; then $\tilde{z} = z_I$, and thus x^* is optimal also for (9).

Therefore, *in principle*, solving an integer linear programming problem is equivalent to solving a linear programming problem (with no integer variables) in which the constraints define the ideal formulation. However, there are two major problems which make integer linear programming harder than linear programming:

- the ideal formulation, in general, is not known, and it can be very difficult to find it;
- even in the cases in which the ideal formulation is known, it is often described by a huge number of constraints, and therefore problem (10) cannot be solved directly with standard linear programming algorithms (such as the simplex method).

Example 4 *Maximum weight matching.*
Let $G = (V, E)$ be an undirected graph. A *matching* in G is a set of edges $M \subseteq E$ such that no two edges in M share a common vertex. In other words, $M \subseteq E$ is a matching if every node of G is the endpoint of at most one edge in M.

The maximum weight matching problem is the following: given an undirected graph $G = (V, E)$ and weights on its edges $w_e, e \in E$, find a matching M in G whose edges have maximum total weight $\sum_{e \in M} w_e$.

This can be formulated as an integer linear programming problem. For every edge $e \in E$, let x_e be a binary variable such that

$$x_e = \begin{cases} 1 & \text{if } e \text{ is in the matching,} \\ 0 & \text{otherwise,} \end{cases} \quad e \in E.$$

Let $M = \{e \in \,|\, x_e = 1\}$. The weight of M is given by

$$\sum_{e \in E} w_e x_e.$$

In order for M to be a matching, we have to impose that for every node $v \in V$ there is at most one edge $e \in E$, with v as one of its endpoints, satisfying $x_e = 1$. This can be modeled with the following linear constraint:

$$\sum_{u \in V \text{ s.t. } uv \in E} x_{uv} \leq 1, \quad v \in V.$$

Then the maximum weight matching problem can be formulated as the following integer linear programming problem:

$$\begin{array}{ll} \max \sum_{e \in E} w_e x_e & \\ \sum_{u \in V \text{ s.t. } uv \in E} x_{uv} \leq 1, & v \in V \\ 0 \leq x_e \leq 1, & e \in E \\ x_e \in \mathbb{Z}, & e \in E. \end{array} \tag{12}$$

This formulation is not ideal, in general. For instance, let G be a "triangle", i.e., $V = \{a, b, c\}$ and $E = \{ab, ac, bc\}$. Let $w_{ab} = w_{ac} = w_{bc} = 1$. Every matching consisting of one edge is a maximum weight matching of weight 1. However, setting $x^*_{ab} = x^*_{ac} = x^*_{bc} = \frac{1}{2}$ gives a feasible solution for the linear relaxation of (12) with weight 1.5. One can check that this is a basic solution, and thus, by Theorem 2, the formulation is not ideal.

We can find a better formulation by adding "ad-hoc" inequalities, which will be obtained by exploiting the structure of the problem.

Let $U \subseteq V$ be a subset of nodes with $|U|$ odd. Given any matching M in G, every node in U is the endpoint of at most one edge in M, and every edge in M has at most two endpoints in U. Then the number of edges in M with both endpoints in U is at most $|U|/2$. Since $|U|$ is odd, and the number of edges in M with both endpoints in U is integer, M contains at most $(|U| - 1)/2$ edged with both endpoints in U. This means that every integer point x satisfying the constraints of (12) must also satisfy

$$\sum_{\substack{u,v \in U \\ uv \in E}} x_{uv} \leq \frac{|U| - 1}{2} \quad \text{for every } U \subseteq V \text{ such that } |U| \text{ is odd.}$$

These inequalities are called *odd-cut inequalities*.

In the example of the triangle, V itself has odd cardinality, thus one can take $U = V$ and write the odd-cut inequality

$$x_{ab} + x_{ac} + x_{bc} \leq 1$$

(as $(|V| - 1)/2 = 1$). The point x^* violates this inequality, as $x^*_{ab} + x^*_{ac} + x^*_{bc} = \frac{3}{2} > 1$.

This proves that the following is a better formulation for our problem:

$$\begin{array}{ll} \min \sum_{e \in E} w_e x_e & \\ \sum_{u \in V \, t.c. \, uv \in E} x_{uv} = 1, & v \in V \\ \sum_{u,v \in U, t.c. \, uv \in E} x_{uv} \leq \frac{|U|-1}{2} & U \subseteq V, \ |U| \text{ odd}, \\ 0 \leq x_e \leq 1, & e \in E \\ x_e \in \mathbb{Z}, & e \in E. \end{array} \tag{13}$$

Indeed, it is possible to show that (13) is the ideal formulation for the maximum weight matching problem, and therefore it would be sufficient to solve its continuous relaxation to find the optimum of the integer problem. However, the number of constraints is exponential (there are $2^{|V|-1}$ subsets of V with odd cardinality) and it is therefore practically impossible to solve the relaxation. (Even for a graph with just 40 nodes, there are more than 500 billion odd-cut inequalities). A better strategy is to solve a sequence of linear relaxations, starting from problem (12) and at every iteration adding one or more odd-cut inequalities that exclude the current optimal solution, until an optimal solution of the integer problem is found. This idea is discussed in the next section (in a more general framework).

4 The cutting plane method

The idea behind the cutting plane method is to solve a sequence of linear relaxations that approximate better and better the convex hull of the feasible region around the optimal solution.

More formally, suppose that we want to solve the integer linear programming problem (P_I)

$$\begin{array}{l} \max c^T x \\ x \in X \end{array} \qquad (P_I)$$

where $X = \{x \in \mathbb{R}^n \,|\, Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ for every } i \in I\}$, for some given matrix $A \in \mathbb{Q}^{m \times n}$ and vector $b \in \mathbb{Q}^m$.

We say that a linear inequality $\alpha^T x \leq \beta$, where $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$, is *valid* for X if $\alpha^T x \leq \beta$ is satisfied by every $x \in X$. Note that if $A'x \leq b'$, $x \geq 0$, is a formulation for X, then also the system obtained by adding a valid inequality $\alpha^T x \leq \beta$ to the system $A'x \leq b'$ is a formulation for X.

Given a point $x^* \notin \text{conv}(X)$, we say that a valid inequality for X $\alpha^T x \leq \beta$ *cuts off* (or *separates*) x^* if $\alpha x^* > \beta$. Such an inequality is also called a *cut* or *cutting plane*. If $A'x \leq b'$, $x \geq 0$ is a formulation for X, x^* is a point satisfying $A'x^* \leq b'$, $x^* \geq 0$, and $\alpha^T x \leq \beta$ is a cutting plane separating x^*, then also the system $A'x \leq b'$, $\alpha^T x \leq \beta$, $x \geq 0$ is a formulation for X.

Cutting plane method

Start with the linear relaxation $\max\{c^T x \,|\, Ax \leq b,\, x \geq 0\}$.

1. Solve the current linear relaxation, and let x^* be a basic optimal solution;
2. If $x^* \in X$, then x^* is optimal for (P_I); STOP.
3. Otherwise, find an inequality $\alpha^T x \leq \beta$ that is valid for X and cuts off x^*;
4. Add the inequality $\alpha^T x \leq \beta$ to the current linear relaxation and go to 1.

It is clear that the above method is a general framework for tackling integer linear programming problems, but in order to implement it one needs an automatic technique to find valid inequalities that cut off the current solution. Below, we give a possible technique to do this.

4.1 Gomory cuts

Gomory cutting plane method can be applied only to pure integer linear programming problems (although there are extensions to the mixed integer case). Thus we consider the problem

$$\begin{array}{c} z_I = \min c^T x \\ Ax = b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{array} \tag{14}$$

where $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$. Define $X = \{x \,|\, Ax = b,\, x \geq 0,\, x \in Z^n\}$.

Solve the continuous relaxation with the simplex method, thus obtaining the problem in tableau form with respect to an optimal basis B (below, N is the set of indices of the non-basic variables):

$$\begin{array}{lrcll} \min & z & & & \\ & -z & + \sum_{j \in N} \bar{c}_j x_j & = & -z_B \\ & x_{\beta[i]} & + \sum_{j \in N} \bar{a}_{ij} x_j & = & \bar{b}_i, \quad i = 1, \ldots, m \\ & & x & \geq & 0. \end{array}$$

Since B is an optimal basis, the reduced costs are non-negative: $\bar{c}_j \geq 0$ for every $j \in N$. The optimal basic solution x^* is given by

$$\begin{array}{rcll} x^*_{\beta[i]} & = & \bar{b}_i, & i = 1, \ldots, m; \\ x^*_j & = & 0, & j \in N; \end{array}$$

therefore $x^* \in \mathbb{Z}^n$ if and only if $\bar{b}_i \in \mathbb{Z}$ for all $i = 1, \ldots, m$.

If this is not the case, let $h \in \{1, \ldots, m\}$ be an index such that $\bar{b}_h \notin \mathbb{Z}$.

Every vector x satisfying $Ax = b$, $x \geq 0$, also satisfies

$$x_{\beta[h]} + \sum_{j \in N} \lfloor \bar{a}_{hj} \rfloor x_j \leq \bar{b}_h$$

because

$$\bar{b}_h = x_{\beta[h]} + \sum_{j \in N} \bar{a}_{hj} x_j \geq x_{\beta[h]} + \sum_{j \in N} \lfloor \bar{a}_{hj} \rfloor x_j,$$

where the inequalities follows from the fact that $x_j \geq 0$ and $\bar{a}_{hj} \geq \lfloor \bar{a}_{hj} \rfloor$ for every j.

Now, since all variables are constrained to take an integer value, every feasible solution satisfies

$$x_{\beta[h]} + \sum_{j \in N} \lfloor \bar{a}_{hj} \rfloor x_j \leq \lfloor \bar{b}_h \rfloor \tag{15}$$

because $x_{\beta[h]} + \sum_{j \in N} \lfloor \bar{a}_{hj} \rfloor x_j$ is an integer number.

Inequality (15) is a *Gomory cut*. The above discussion shows that the Gomory cut (15) is a valid inequality for X. Moreover, we now verify that it cuts off the current optimal solution x^*:

$$x^*_{\beta[h]} + \sum_{j \in N} \lfloor \bar{a}_{hj} \rfloor x^*_j = x^*_{\beta[h]} = \bar{b}_h > \lfloor \bar{b}_h \rfloor,$$

where the inequality $\bar{b}_h > \lfloor \bar{b}_h \rfloor$ holds because $\bar{b}_h$ is not integer.

It is convenient to rewrite the Gomory cut (15) in an equivalent form. By adding a slack variable s, (15) becomes

$$x_{\beta[h]} + \sum_{j \in N} \lfloor \bar{a}_{hj} \rfloor x_j + s = \lfloor \bar{b}_h \rfloor, \quad s \geq 0.$$

Since all coefficients in the above equations are integer, if x has integer entries then s is an integer as well. We can then require s to be an integer variable.

If from the above equation we subtract the tableau equation

$$x_{\beta[h]} + \sum_{j \in F} \bar{a}_{hj} x_j = \bar{b}_h,$$

we obtain

$$\sum_{j \in N} (\lfloor \bar{a}_{hj} \rfloor - a_{hj}) x_j + s = \lfloor \bar{b}_h \rfloor - b_h.$$

This form of the cut is known as *Gomory fractional cut* (or *Gomory cut in fractional form*).

If we add this constraint to the previous optimal tableau, we obtain

$$\begin{array}{lllllll}
\min & z & & & & & \\
 & -z & & + \sum_{j \in N} \bar{c}_j x_j & & = & -z_B \\
 & & x_{\beta[i]} & + \sum_{j \in N} \bar{a}_{ij} x_j & & = & \bar{b}_i, \qquad i = 1, \ldots, m \\
 & & & \sum_{j \in N} (\lfloor \bar{a}_{hj} \rfloor - a_{hj}) x_j & + s & = & \lfloor \bar{b}_h \rfloor - b_h \\
 & & & x, & s & \geq & 0.
\end{array}$$

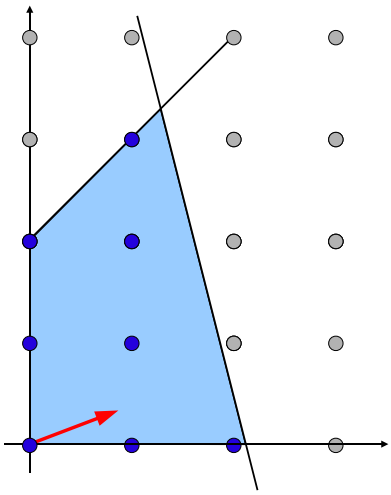
This problem is already in tableau form with respect to the basic variables $x_{\beta[1]}, \ldots, x_{\beta[m]}, s$ (i.e., the same basis as before, plus variable s); moreover, this basis is feasible for the dual, as all reduced costs are non-negative (they coincide with the previous reduced costs).

Also note that $\bar{b}_i \geq 0$ for $i = 1, \ldots, m$, while the right-hand side of the constraint in which s appears is $\lfloor \bar{b}_h \rfloor - b_h < 0$. We can then solve this new linear relaxation with the dual simplex method; s will leave the basis at the first iteration.

Example 5 *Solve the following problem with Gomory cutting plane method.*

$$\begin{array}{rcrcl} \min\ z & = & -11x_1 - 4.2x_2 & & \\ & & -x_1 + x_2 & \leq & 2 \\ & & 8x_1 + 2x_2 & \leq & 17 \\ & & x_1, x_2 & \geq & 0 \text{ integer.} \end{array} \tag{16}$$

The feasible region of the linear relaxation is shown in the picture:



We add slack variables x_3 and x_4 to put the system in standard form:

$$\begin{array}{rcrcl} -z - 11x_1 - 4.2x_2 & = & 0 \\ -x_1 + x_2 + x_3 & = & 2 \\ 8x_1 + 2x_2 + x_4 & = & 17 \\ x_1, x_2, x_3, x_4 & \geq & 0 \text{ integer.} \end{array}$$

Note that we can impose that x_3 and x_4 are integer because since all coefficients are integer, whenever x_1 and x_2 are integer also x_3 and x_4 are integer. Thus we have a pure integer programming problem and we can apply Gomory cutting plane method.
If we solve the linear relaxation, we obtain the following tableau:

$$\begin{array}{rrrrcl} -z & & +1.16x_3 & +1.52x_4 & = & 28.16 \\ & x_2 & +0.8x_3 & +0.1x_4 & = & 3.3 \\ x_1 & & -0.2x_3 & +0.1x_4 & = & 1.3 \end{array}$$

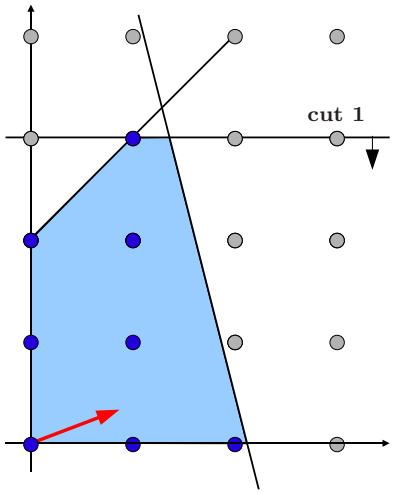
The corresponding basic solution is $x_3 = x_4 = 0$, $x_1 = 1.3$, $x_2 = 3.3$ (the upper vertex of the quadrilateral in the above picture), with objective function value $z = -28.16$. Since the values of x_1 and x_2 are not integer, this is not a feasible solution of (16). We can derive, e.g., a Gomory cut from the equation $x_2 + 0.8x_3 + 0.1x_4 = 3.3$, thus obtaining

$$x_2 \leq 3.$$

If this constraint is added to the original linear relaxation, we obtain a better formulation:

$$\begin{aligned} \min z = 11x_1 + 4.2x_2 & \\ -x_1 + x_2 &\leq 2 \\ 8x_1 + 2x_2 &\leq 17 \\ x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

whose feasible region is the following:



In order to solve this new problem, we first write the cut in fractional form with a slack variable x_5:

$$-0.8x_3 - 0.1x_4 + x_5 = -0.3.$$

We then add the constraint to the previous tableau:

$$\begin{array}{rrrrrrr} -z & & +1.16x_3 & +1.52x_4 & & = & 28.16 \\ & x_2 & +0.8x_3 & +0.1x_4 & & = & 3.3 \\ x_1 & & -0.2x_3 & +0.1x_4 & & = & 1.3 \\ & & -0.8x_3 & -0.1x_4 & +x_5 & = & -0.3 \end{array}$$

If the dual simplex method is applied, x_5 leaves the basis and x_3 enters, as $\min\left\{\frac{1.16}{0.8}, \frac{1.52}{0.1}\right\} = \frac{1.16}{0.8}$. After this single iteration, we obtain the following optimal tableau:

$$\begin{array}{rrrrrr} -z & & +1.375x_4 & +1.45x_5 & = & 27.725 \\ & x_2 & & +x_5 & = & 3 \\ x_1 & & +0.125x_4 & -0.25x_5 & = & 1.375 \\ & x_3 & +0.125x_4 & -1.25x_5 & = & 0.375 \end{array}$$

The corresponding basic solution is $x_1 = 1.375$, $x_2 = 3$ $x_3 = 0.375$ (upper-right vertex in the previous picture), with objective value $z = 27.725$.

From the equation $x_3 + 0.125x_4 - 1.25x_5 = 0.375$ of the tableau, we obtain the Gomory cut

$$x_3 - 2x_5 \leq 0.$$

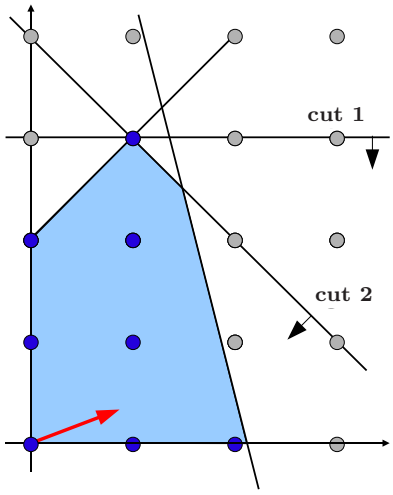
Since $x_3 = 2 + x_1 - x_2$ and $x_5 = 3 - x_2$, in the original space of variables (x_1, x_2) the above inequality can be rewritten as

$$x_1 + x_2 \leq 4.$$

If this constraint is added to the original problem, we obtain the new linear relaxation

$$\begin{aligned} \min z = 11x_1 + 4.2x_2 & \\ -x_1 + x_2 &\leq 2 \\ 8x_1 + 2x_2 &\leq 17 \\ x_2 &\leq 3 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

shown in the picture.



Written in fractional form, the cut is

$$-0.125x_4 - 0.75x_5 + x_6 = -0.375.$$

If we add this constraint to the tableau, we obtain

$$\begin{array}{rrrrrrr} -z & & +1.375x_4 & +1.45x_5 & & = & 27.725 \\ & x_2 & & +x_5 & & = & 3 \\ x_1 & & +0.125x_4 & -0.25x_5 & & = & 1.375 \\ & x_3 & +0.125x_4 & -1.25x_5 & & = & 0.375 \\ & & -0.125x_4 & -0.75x_5 & +x_6 & = & -0.375 \end{array}$$

Now x_6 leaves the basis and x_5 enters:

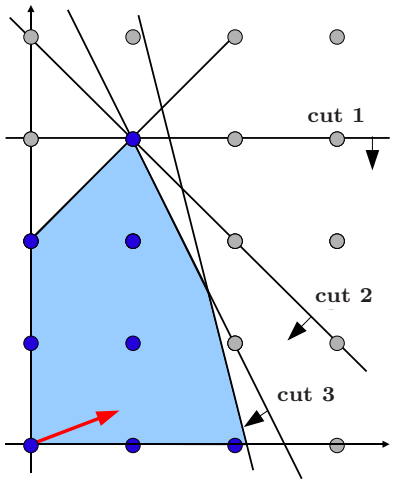
$$\begin{array}{rrrrrrrrr} -z & & & & +17/15x_4 & & +29/15x_6 & = & 27 \\ & & x_2 & & -1/6x_4 & & +4/3x_6 & = & 2.5 \\ & x_1 & & & +1/6x_4 & & -1/3x_6 & = & 1.5 \\ & & & x_3 & & & +x_6 & = & 0 \\ & & & & 1/6x_4 & +x_5 & -4/3x_6 & = & 0.5 \end{array}$$

This tableau is already optimal, with optimal solution $x_1 = 1.5$, $x_2 = 2.5$ (upper-right vertex in the previous picture) and objective value $z = 27$.

From the equation $1/6x_4 + x_5 - 4/3x_6 = 0.5$ of the tableau, we derive the Gomory cut $x_5 - 2x_6 \leq 0$. Since $x_5 = 3 - x_2$ and $x_6 = 4 - x_1 - x_2$, in the original space of variables (x_1, x_2) the cut reads $2x_1 + x_2 \leq 5$. The new continuous relaxation is

$$\begin{array}{rrcl} \min z = & 11x_1 + 4.2x_2 & & \\ & -x_1 + x_2 & \leq & 2 \\ & 8x_1 + 2x_2 & \leq & 17 \\ & x_2 & \leq & 3 \\ & x_1 + x_2 & \leq & 4 \\ & 2x_1 + x_2 & \leq & 5 \\ & x_1, x_2 & \geq & 0, \end{array}$$

shown in the picture.



In fractional form, the cut is

$$-1/6x_4 - 2/3x_6 + x_7 = -0.5.$$

We add this constraint to the tableau:

$$\begin{array}{rrrrrrrrrrrr}
-z & & & & +17/15x_4 & & & & +29/15x_6 & & = & 27 \\
 & & x_2 & & -1/6x_4 & & & & +4/3x_6 & & = & 2.5 \\
 & x_1 & & & +1/6x_4 & & & & -1/3x_6 & & = & 1.5 \\
 & & & x_3 & & & & & +x_6 & & = & 0 \\
 & & & & 1/6x_4 & & +x_5 & & -4/3x_6 & & = & 0.5 \\
 & & & & -1/6x_4 & & & & -2/3x_6 & +x_7 & = & -0.5
\end{array}$$

In this case, two iterations of the dual simplex method are needed to obtain an optimal tableau: first x_7 leaves the basis and x_6 enters, then x_3 leaves and x_4 enters the basis. We obtain:

$$\begin{array}{rrrrrrrrrrrr}
-z & & & & +13/15x_3 & & & & & +76/15x_7 & = & 23,6 \\
 & & x_2 & & +2/3x_3 & & & & & +1/3x_7 & = & 3 \\
 & x_1 & & & -1/3x_3 & & & & & +1/3x_7 & = & 1 \\
 & & & & 4/3x_3 & +x_4 & & & & -10/3x_7 & = & 3 \\
 & & & & -2/3x_3 & & +x_5 & & & -1/3x_7 & = & 0 \\
 & & & & -1/3x_4 & & & & x_6 & -2/3x_7 & = & 0
\end{array}$$

The corresponding optimal solution is $x_1 = 1$, $x_2 = 3$, with objective value $z = 23.6$. Since this solution has integer components, this is an optimal solution for the initial integer linear programming problem.