CHAPTER 9
SOLVING MIXED INTEGER PROBLEMS

There are a few things that I want you to remember from exploring this chapter:

1. That if integers are part of your formulation you may easily end up dealing with combinatorial explosion.
2. That you can use special algorithms to deal with this:
    a. For MILP: branch and cut and Gomory cuts for example
    b. For MINLPs: Outer approximation, branch and reduce

Introduction

In the previous chapter we studied how to formulate integer and mixed integer programming problems, that is problems in which discrete variables are present. Discrete optimization problems suffer from what is known as the curse of dimensionality. This curse comes, conceptually, from the absence of optimality conditions for this type of problem (note that we cannot define derivatives for discrete sets, so the optimality conditions previously developed do not apply). Thus, when searching for an optimum one needs to explore systematically all the discrete options available. If an exhaustive enumeration of all the options for a discrete problem are considered, very quickly the problems of interest become intractable. For example, for the knapsack problem that we discussed we have that:

- If n=34 different objects than the problem of enumerating all the solutions takes 1 second
- If n=40 different objects than the problem of enumerating all the solutions takes 1 minute
- If n=45 different objects than the problem of enumerating all the solutions takes 1 hour
- If n=50 different objects than the problem of enumerating all the solutions takes 1 day
- If n=58 different objects than the problem of enumerating all the solutions takes 1 year
- If n=69 different objects than the problem of enumerating all the solutions takes 2583 years
- If n=78 different objects than the problem of enumerating all the solutions takes 1.500.000 years

This numbers may slightly change with the evolution of computers, but note that to completely enumerate the solutions for 78 objects in reasonable time (let's say less than two days) one needs improvements in computational speed of 6 orders of magnitude. This is way beyond what one expects even in the rosier scenarios, and points toward the need to develop efficient algorithms.

The branch and bound method

The branch and bound method is a powerful algorithm that can be used to solve MILP problems. There are two fundamental principles on which this algorithm relies:

- The first intuition is that an original problem defined over a large feasible set F, can be partitioned into a collection of smaller subsets ($F = F_1 \cup \dots \cup F_n$). For each of these subsets it is possible to solve a subproblem $P_k = \min_{x \in F_k} f(x)$. We also have $l(P_k)$ is a lower bound of subproblem P_k. Now if we have i such that

$$f(x_i^*) \leq f(x_k^*), k = 1, \dots, n$$

$$f(x_i^*) \leq l(P_k)$$

Then x_i^* is an optimal solution of the optimization problem P. Note that

$$\underset{problem\ optimal}{f(x_i^*)} \leq \underset{Lower\ bound\ of\ subproblem}{l(P_k)} \leq \underset{Optimal\ of\ subproblem}{f(x_k^*)}$$

Intuitively there are two hidden messages here

  - You can find the optimal of the original problem by finding the optimal of all the subproblems.
  - Actually, in many instances you do not need to solve all the subproblems it is enough to have a lower bound to exclude that part of the feasible region. Consider the following example, the whose square represents the feasible region, but we partition it into four subproblems. The first thing we do is to solve the first subproblem P_1, from there we extract a lower bound, and let's say that we also extract a candidate solution, x^* such that $f(x^*) = 20$. Note that based on this information it is possible to say that in this case there is no need to solve subproblems P_2, P_3, and P_4.

P_1 $f(x^*) = 20$ $l(P_1) = 20$	P_2 $l(P_2) = 40$
P_3 $l(P_3) = 100$	P_4 $l(P_4) = 21$

- The second critical intuition to develop the algorithm is that in the case of integer programming, it is possible to find a lower bound on the variables by relaxing the integrality constraint. In other words, if I assume that all my integer variables are real instead of integers, and solve the problem then the solution to this problem (which is called the relaxed problem) provides a lower bound for the problem.

Armed with these intuitions, it is now possible to propose an algorithm to solve a Mixed Integer Linear programming problem as follows:

1. Solve LP Relaxation (Ignore Integer Constraints):
   - Solve the problem without integer constraints using the Simplex method.
     - If the LP solution is integer, return it as the optimal solution.
2. Branching (Choose a Fractional Variable):

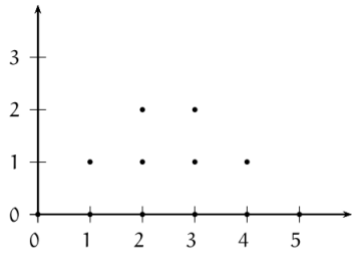
- Identify a decision variable x_k that is fractional.
- Create two new subproblems:
    i. **Left Branch:** Add constraint $x_k \leq \lfloor x_k^* \rfloor$
    ii. **Right Branch:** Add constraint $x_k \geq \lceil x_k^* \rceil$

3. Bounding (Pruning Subproblems):
    - Solve the LP relaxation for each branch.
    - If a subproblem's solution is worse than the current best integer solution, discard it (prune).
    - If the solution is integer, update the best-known solution.
4. Repeat Steps 2-3 for Remaining Subproblems:
    - Continue branching until all subproblems are either solved to integer optimality or pruned.
5. Return the Best Integer Solution Found.

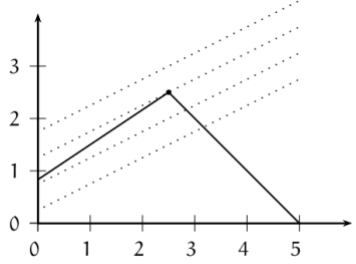
Example: Let's consider the following optimization problem:

$$\begin{aligned} & \min x_1 - 2x_2 \\ & -4x_1 + 6x_2 \leq 5 \\ s.t. \quad & x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in N \end{aligned}$$

This is an integer problem, and the feasible region and its relaxation are presented below:

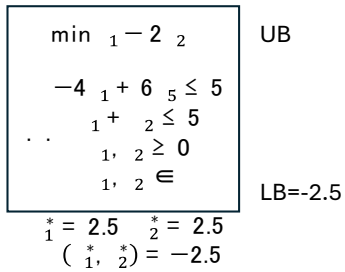


(a) Feasible set of P_0

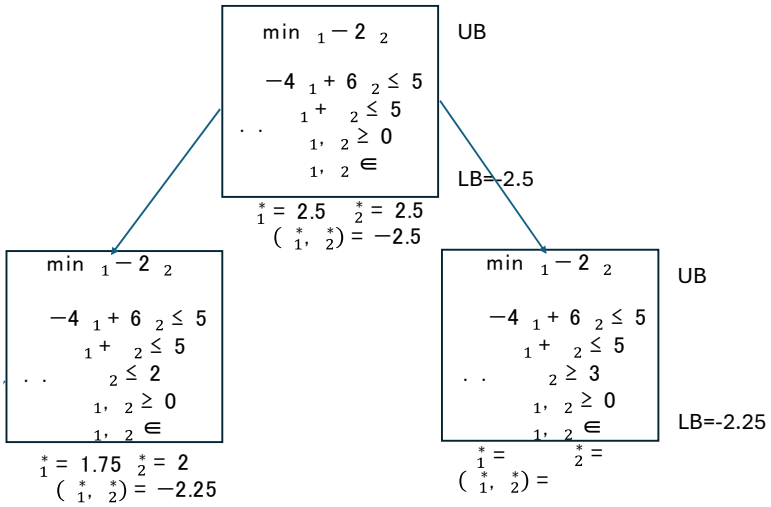


(b) Feasible set and level curves of $R(P_0)$

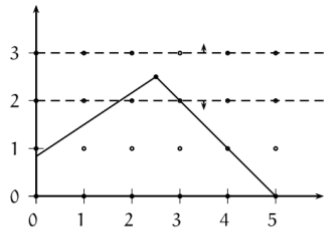
If we follow the proposed algorithm, the first thing that we would need to do is to solve the original problem, while relaxing the integrality constraint:



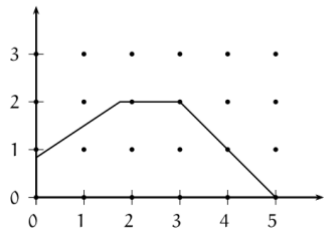
Note that at this stage there is no upper bound, an upper bound is given by a feasible solution (that is, an integer solution). To proceed, what we do is branching, so we choose one of the variables that is not integer, and we create two subproblems (we are going to choose variable x_2), and solve the relaxed version of the resulting two subproblems.



The feasible set for these subproblems and the additional constraints are shown in the figure:

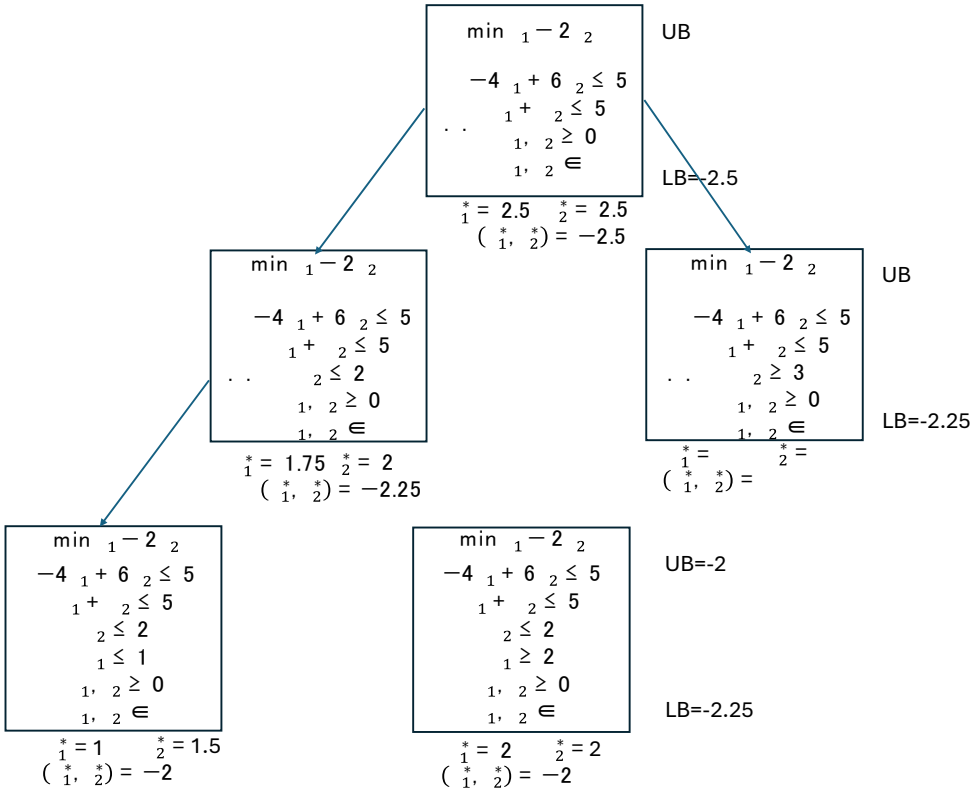


(a) Additional branching constraints



(b) Feasible set of problem P_1

The optimal point of subproblem 1 is not optimal because it is not integer, thus, we need to branch one more time around variable x_1. We then introduce two new subproblems such that $x_1 \leq 1$ and $x_1 \geq 2$.



While solving these two new subproblems we find two things (1) one of them is actually feasible, this will constitute an upper bound to our problem. (2) the other one does not provide a solution, but it has a feature, it provides a lower bound that is identical to the solution of the problem on the right, this means one thing, that imposing additional constraints can only worsen the solution, this means that we do not need to explore this branch anymore, and we can stop here and use the found solution as an optimal.

Example: Let's consider this other example in which we have a problem that is formulated in terms of binaries.

$$\max Z = 86y_1 + 4y_2 + 40y_3$$
$$s.t. \quad \begin{matrix} 744y_1 + 76y_2 + 42y_3 \leq 875 \\ 67y_1 + 27y_2 + 53y_3 \\ y_1, y_2, y_3 \in \{0,1\} \end{matrix}$$

The relaxed version of this problem is:

$$\max 86y_1 + 4y_2 + 40y_3$$
$$s.t. \quad \begin{matrix} 744y_1 + 76y_2 + 42y_3 \leq 875 \\ 67y_1 + 27y_2 + 53y_3 \\ 0 \leq y_1, y_2, y_3 \leq 1 \end{matrix}$$

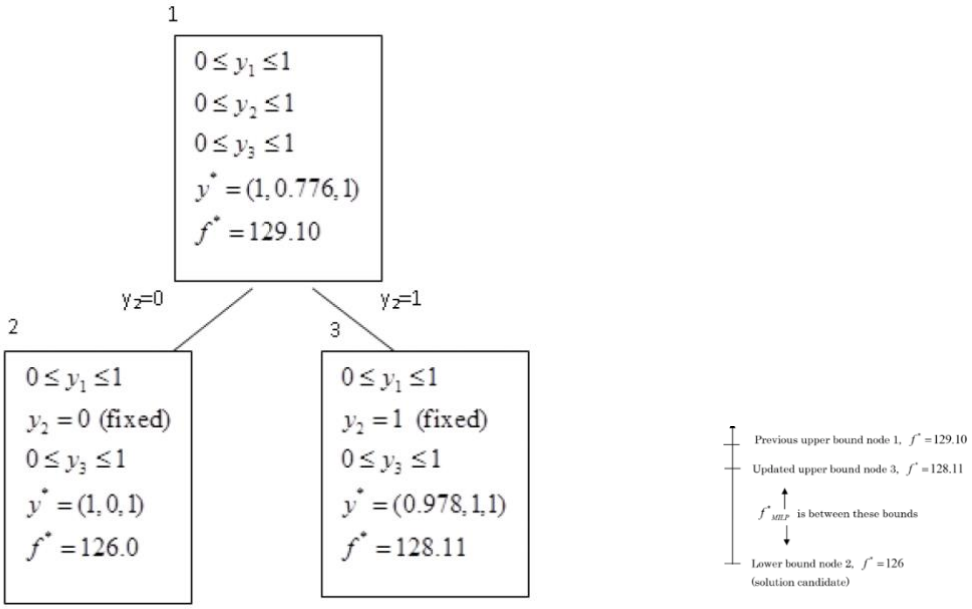
UB=129.10

f_{glob}

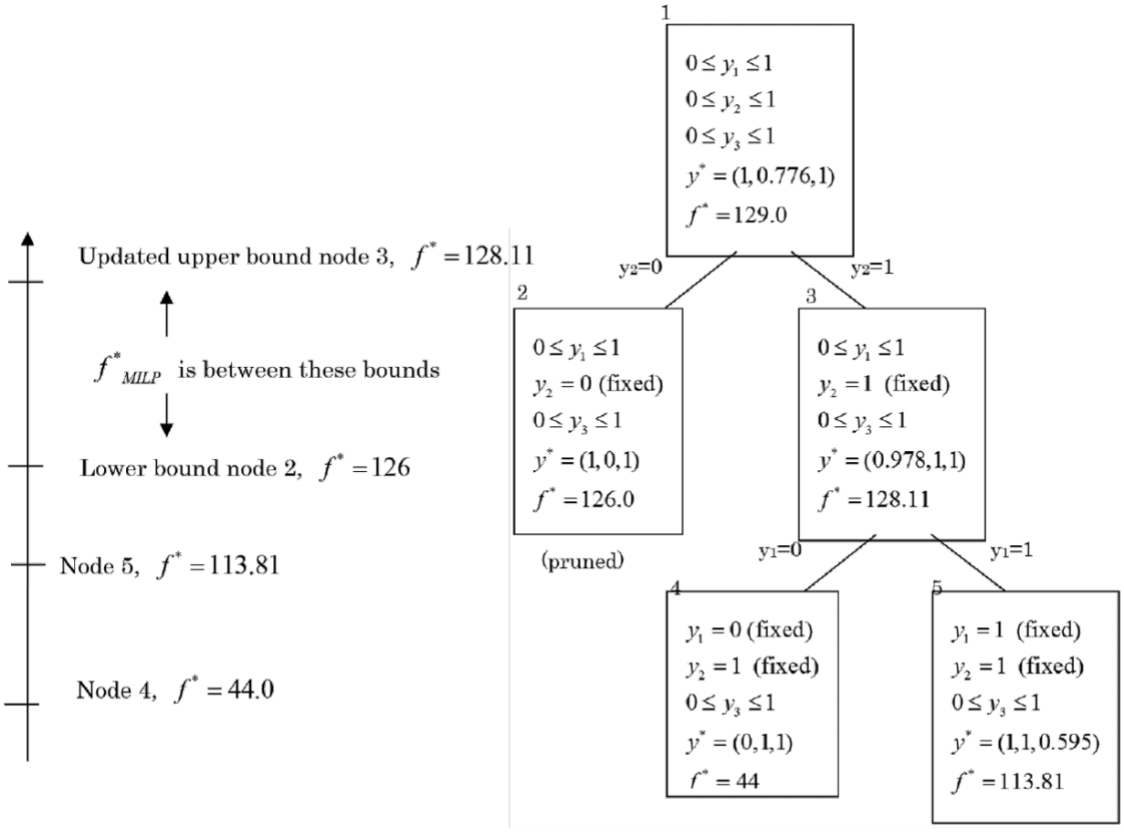
LB

$$(y_1, y_2, y_3) = (1, 0.776, 1), f^*_{relaxed} = 129.1$$

Now we can branch around the second variable, which is non integer, as follows:



Note that we have now both an upper and a lower bound, furthermore we need to explore a bit more node 3, since y_1 is not integer we will branch on this variable.



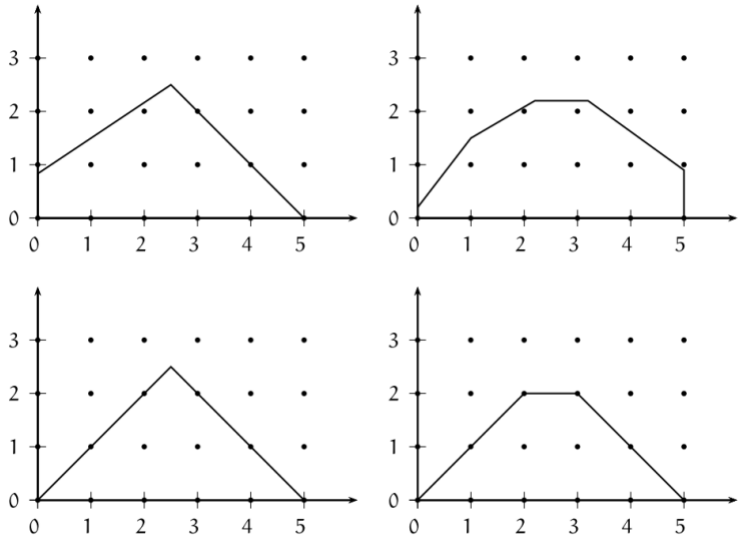
From the results we can see two things, (1) node 4 gives an integer solution but is worst than the one we already have, so that is not a maximum (2) when we solve the relaxed problem for node 5, we get an upper bound which is lower (worst) than our current lower bound, this means we do not need to explore this branch anymore, and we can prune it.

Cutting planes

The idea of cutting planes methods is to start from the original formulation of the problem and then include additional constraints that shrink the polyhedron containing the feasible set without modifying the feasible set. These additional constraints are called valid inequalities.

Valid inequalities: let $F \in R^n$ be a set. The inequality $a^T x \geq b$ is a valid inequality for F if it is satisfied by all $x \in F$.

An example that shows that the same feasible set can be characterized by different polyhedral is shown below. Of the presented solutions the most convenient is the one in quadrant 4. Note that if we have this case, then the optimal solution of the relaxed problem is identical to the solution of the original problem, this is because solutions of a linear problem lie at vertexes, and in case 4 all vertexes coincide with integer solutions.



The most popular method to derive valid inequalities in MILPs was proposed by Ralph Gomory in 1959 and it exploits the simplex tableau. Consider a mixed inter linear optimization problem. We solve the relaxed version of this problem, the optimal simplex tableau is given by:

$B^{-1}A$	$B^{-1}b$
$c^T - c_B^T B^{-1} A$	$-c_B^T B^{-1} b$

The upper part of this tableau contains a transformed version of the equalities. If we separate variables in basic and non-basic ($A = [B|N], x = [x_B|x_N]$), then we can write:

$$B^{-1}Ax = B^{-1}b$$

$$B^{-1}Bx_B + B^{-1}Nx_N = B^{-1}b$$

$$x_B + B^{-1}Nx_N = B^{-1}b$$

Based on this last equation, we can write (remember that $B^{-1}b = x_B$):

$$x_i + \sum_{i \in Non-basic} \alpha_{i,j} x_j = (x_B^*)_i$$

Since x is feasible it is positive, therefore if we round down all coefficients $\alpha_{i,j}$ we obtain a valid inequality (round down, in this case, will never lead to an increase in the function).

$$x_i + \sum_{i \in Non-basic} \lfloor \alpha_{i,j} \rfloor x_j \leq (x_B^*)_i \leq x_i + \sum_{i \in Non-basic} \alpha_{i,j} x_j = (x_B^*)_i$$

It is possible to round down both sides of this inequality and obtain:

$$\left\lfloor x_i + \sum_{i \in Non-basic} \lfloor \alpha_{i,j} \rfloor x_j \leq (x_B^*)_i \right\rfloor \leq \lfloor (x_B^*)_i \rfloor$$

Now, if we consider only the x_i that are integers, we can rewrite the former inequality as follows:

$$x_i + \sum_{i \in Non-basic} \lfloor \alpha_{i,j} \rfloor x_j \leq (x_B^*)_i \leq \lfloor (x_B^*)_i \rfloor$$

This inequality is known as a Gomory cuts. They are written for variables that we need to be integer and that in the relaxed solution appear as non-integer.

Example: Let's consider the following problem that we already explored:

$$s.t. \quad \begin{array}{cc} \min x_1 - 2x_2 & \min x_1 - 2x_2 \\ -4x_1 + 6x_2 \leq 5 & -4x_1 + 6x_2 \leq 5 \\ x_1 + x_2 \leq 5 & x_1 + x_2 \leq 5 \\ x_1, x_2 \geq & x_1, x_2 \geq 0 \\ x_1, x_2 \in N & x_1, x_2 \in R \\ Original & Relaxed \end{array}$$

To be able to solve the relaxed version of this we need to reformulate the problem in standard form, so we can use the simplex method, this implies adding couple of slack variables:

$$\begin{array}{c} \min x_1 - 2x_2 \\ -4x_1 + 6x_2 + x_3 = 5 \\ x_1 + x_2 + x_4 = 5 \\ x_1, x_2, x_3, x_4 \geq 0 \\ x_1, x_2, x_3, x_4 \in R \end{array}$$

The optimal tableau for the relaxed problem is this:

x_1	x_2	x_3	x_4		
0	1	0.1	0.4	2.5	x_2
1	0	-0.1	0.6	2.5	x_1
0	0	0.3	0.3	2.5	

Based on this tableau we can write two Gomory cuts:

$$x_2 + \lfloor 0.1 \rfloor x_3 + \lfloor 0.4 \rfloor x_4 \leq \lfloor 2.5 \rfloor$$

$$x_2 \leq 2$$

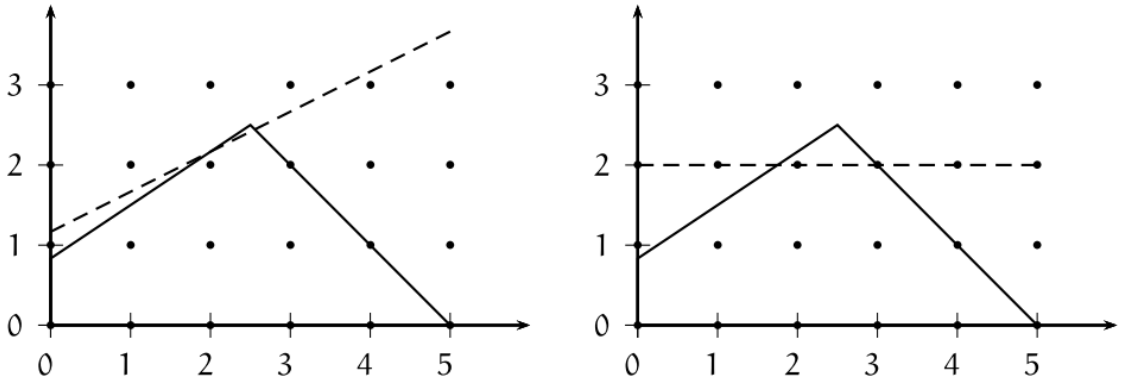
The second constraint can be written as follows:

$$x_1 + \lfloor -0.1 \rfloor x_3 + \lfloor 0.6 \rfloor x_4 \leq \lfloor 2.5 \rfloor$$

$$x_1 - x_3 \leq 2$$

Note that $x_3 = 5 + 4x_1 - 6x_2$, so we can express this inequality in terms of the original problem variables.

$$-3x_1 + 6x_2 \leq 7$$



(a) Gomory cut on x_1 (b) Gomory cut on x_2

Example: Let's consider the following optimization problem

$$\begin{aligned} \min\ & -3x_1 - 13x_2 \\ s.t.\ & 2x_1 + 9x_2 + x_3 = 29 \\ & 11x_1 - 8x_2 + x_4 = 79 \\ & x_1, x_2, x_3, x_4 \geq 0 \\ & x_1, x_2, x_3, x_4 \in N \end{aligned}$$

The optimal solution of the relaxed tableau is as follows:

x_1	x_2	x_3	x_4		
0	1	0.1	-0.02	1.4	x_2
1	0	0.07	0.08	8.2	x_1
0	0	1.45	0.01	42.8	

Then we can generate the following two valid inequalities:

$$x_2 - x_4 \leq 1$$

$$x_1 \leq 8$$

In practical implementations, that is in commercial software packages (e.g., GUROBI, CPLEX, XPRESS) both branch and bound and cutting planes strategies are simultaneously implemented, this type of approach is typically referred as *branch and cut*.

Solving MINLPs

The algorithms used for the solution of MINLPs are more advanced than those discussed in the course, we will not describe them in detail, but we want to make come comments regarding the solution of this type of problem. First, let's remember that in general an MINLP looks as follows:

$$\min Z = f(x, y)$$

$$s.t. \quad \begin{aligned} h(x, y) &= 0 \\ g(x, y) &\leq 0 \\ x \in R^n \; y &\in \{0,1\}^m \end{aligned}$$

However, in many instances we can have a special structure in which the integer part of these problems is linear, if that is the case, the problem can be written as follows:

$$\min Z = c^T x + f(x)$$

$$s.t. \quad \begin{aligned} h(x, y) &= 0 \\ g(x) + By &\leq 0 \\ Ay &\leq a \\ x \in X \; y &\in \{0,1\}^m \end{aligned}$$

There are two major approaches that can be used to solve this problem:

- *The branch and bound method:* this method is similar to solving an MILP, but at each node one solves an NLP instead of an LP. This approach is only practical if the number of discrete variables is small. If the number of discrete variables is too large then the approach is not practical as solving NLPs is much more computationally intensive than solving a LP. Solvers that rely on this approach include SBB and BONMIN
- *Decomposition approaches:* this is a collection of methods, in general they iterate between the solution of an NLP subproblem with fixed binaries, and an MILP master problem that predicts new values for the binary variables. The NLP subproblem solution provide an upper bound to the problems, while the MILPs provide a lower bound. Iterations continue until the bounds lie within an acceptable range.