Introduction to Optimisation, Lecture 4

Augustin Cosse

January 2022

This note was written as part of the series of lectures on Optimisation delivered at ULCO in 2022-2023. The version is temporary. Please direct any comments or questions to augustin.cosse@univ-littoral.fr.

Introduction

When hiring or dispatching workers or sheduling buses, working with real numbers is not possible and the problems are ususally defined on integers. Although the solution to 2D problems can easily be found graphically, higher dimensional problems quickly become difficult. It is however sometimes the case that an almost optimal solution could be obtained by rounding the components of a linear relaxation of the original problem as we will see.

Linear optimisation problems defined on integers are known as integer programs (IPs) and have general form

$$\begin{aligned} \max \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \mathbb{Z}^n \end{aligned}$$

The feasible set is not a convex polyhedron anymore but consists of a lattice of separate (integer) points (see Fig. 1 below).

Solving general integer programs is computationally difficult (i.e. integer programming is NP hard in the general case) and there are integer programs that are intractable for no more than 10 constraints and 10 variables even for the most modern computers. Integer programming can in fact be used to encode logical sentences as the constraints $x_i \in \mathbb{Z}$ and $0 \leq x_i \leq 1$ can be used to encode True/False decisions.

Before discussing how to solve integer programs, we list a couple of classical applications. We start with the Maximum Weight Matching problem.

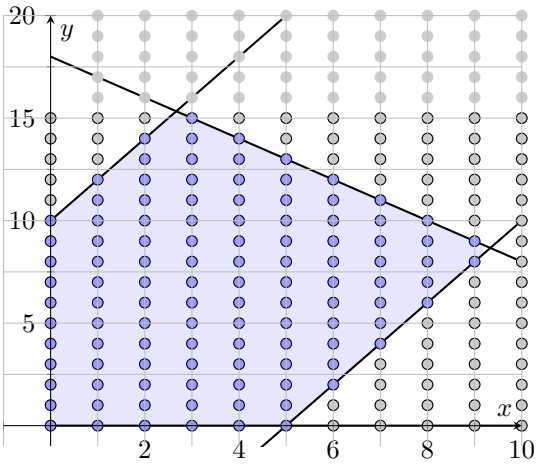

Figure 1: Polytope and corresponding integer lattice. The integer feasible points are highlighted in blue.

Example 1. *A company undergoes a reorganization. 5 positions are available and the company has to assign those positions to 5 employees. In order to assign those positions, the HR manager asks the employees to fill out appropriate questionnaires. Each questionnaire is graded on a scale from 0 to 100 and the scores are encoded as* w_{ij}*. The problem can be represented as a graph shown in Fig. 2.*

The manager would like to select a position for each employee so that the total satisfaction (sum of the scores) is maximized. In the language of graph theory, the graph shown in Fig. 2 is known as a bipartite graph with vertex set $J \cup E$ *and edge set* $\mathcal{E}$*.*

We want to find a set of edges $S \subseteq \mathcal{E}$ *such that each vertex is incident to exactly one edge (such a set is known as a perfect matching). In order to formulate the problem as an integer program, we introduce a variable* x_ℓ *for each edge* $e \in \mathcal{E}$ *such that* $x_e = 1$ *if* $e \in S$ *and* 0 *otherwise. The objective of maximizing the matching can then read as*

$$\max \quad \sum_{e \in \mathcal{E}} w_e x_e$$

and the requirement that every vertex is incident to exactly one edge is given by

$$\sum_{\substack{e \in \mathcal{E} \\ s.t.\ v \in e}} x_e = 1, \quad \textit{for each } v$$

Our final integer program for the maximum weight matching problem read as

$$\begin{aligned} \max \quad & \sum_{e \in \mathcal{E}} w_e x_e \\ s.t. \quad & \sum_{\substack{e \in \mathcal{E} \\ s.t.\ v \in e}} x_e = 1 \quad \textit{for each } v \\ & x_e \in \{0, 1\} \end{aligned} \tag{1}$$

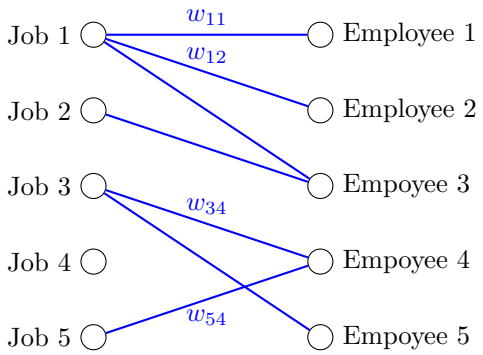

Figure 2: An example of a bipartite graph for the Maximum Weight Matching Problem (1).

If we relax the integrality condition, we get the following LP

$$\begin{aligned} \max \quad & \sum_{e \in \mathcal{E}} w_e x_e \\ s.t. \quad & \sum_{\substack{e \in \mathcal{E} \\ s.t.\ v \in e}} x_e = 1, \quad \textit{for each vertex } v \in V \\ & 0 \leq x_e \leq 1, \quad \textit{for each edge } e \in E \end{aligned} \tag{2}$$

problem (2) is known as the LP relaxation of problem (1). Clearly, any solution to (2) provides an upper bound on any optimal solution to (1). This follows from the fact that any feasible solution to (1) is also a feasible solution to (2) and so we are maximizing over a bigger feasible set in (2). It is sometimes the case that the solution of the LP relaxation provides the exact solution to the original IP.

Theorem 1. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an arbitrary bipartite graph with real edge weights w_e. If the LP relaxation (2) of the IP (1) has at least one feasible solution, then it has at least one integral solution. Moreover, this integral solution is an optimal solution for the original integer program (1).*

Proof. TBA □

Example 2 (Minimum Vertex Cover)**.** *As a second example of an integer program, we consider the problem of equipping each connection between two computers of a network with a tracking device. Moreover, we would like to attach the tracking device to some of the computers only in such a way that the total number of tracking devices is minimized. Using a graph terminology, we can view the computers in the network as vertices and the connections between those computers as edges. The problem then reads as*

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ s.t. \quad & x_u + x_v \geq 1, \quad \text{for every edge } \{u, v\} \in \mathcal{E} \\ & x_v \in \{0, 1\} \quad \forall v \in \mathcal{V} \end{aligned} \tag{3}$$

Finding a minimum vertex cover is NP hard in the general case. We can however again study the fractional relaxation

$$\begin{aligned} \min \quad & \sum_{v \in \mathcal{V}} x_v \\ s.t. \quad & x_u + x_v \geq 1, \quad \text{for every edge } \{u, v\} \in E \\ & 0 \leq x_v \leq 1 \end{aligned} \tag{4}$$

The first step of the approximation algorithm to vertex cover consists in computing an optimal solution x^* of the LP relaxation. The components of x^* are real numbers in the $[0, 1]$ interval. In the second step, we define the set

$$S_{LP} = \{v \in \mathcal{V} \mid x_v^* \geq 1/2\}$$

This is a vertex cover since for every edge $\{u, v\}$ we have $x_u^* + x_v^* \geq 1$ hence for every edge $(u, v) \in \mathcal{E}$, $x_u^* \geq 1/2$ or $x_v^* \geq 1/2$. Let S_{opt} denote the minimum possible vertex cover. We will show that

$$|S_{opt}| \leq 2|S_{LP}|$$

To see this, let $\tilde{x}$ denote a solution of the integer program corresponding to the objective S_{opt}. I.e. we have $\tilde{x}_v = 1$ for $v \in S_{opt}$ and 0 otherwise. Since $\tilde{x}$ is a feasible solution of the relaxation, we have

$$\sum_{v \in V} x_v^* \leq \sum_{v \in V} \tilde{x}_v$$

On the other hand,

$$|S_{opt}| \leq S_{round} = |S_{LP}| = \sum_{v \in S_{LP}} 1 \leq 2 \sum_{v \in \mathcal{V}} x_v^*$$

since $x_v^* \geq 1/2$ for every $v \in S_{LP}$. As a result we can write

$$|S_{LP}| = \sum_{v \in S_{LP}} 1 \leq 2 \sum_{v} x_v^* \leq 2 \sum_{v \in V} \tilde{x}_v = 2|S_{opt}|$$

From which we finally obtain the integrality gap

$$\frac{1}{2}|S_{LP}| \leq |S_{opt}| \leq |S_{LP}|$$

Example 3 (Maximum Independent Set)**.** *Maximum Independent Set is a fundamental problem in computer science which has important applications including in network analysis and coding theory. The problem is NP hard in its general form and consequently there have been several attempts at finding approximate solutions. For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a set $I \subseteq \mathcal{V}$ of vertices is called independent (or stable) if no two vertices of I are connected by an edge in $\mathcal{G}$.*

Computing an independent set with the maximum possible number of vertices for a given graph can be expressed by an integer program

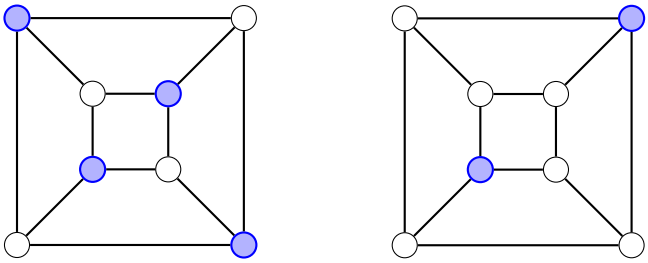

Figure 3: Two examples of maximal independent sets. The left one being maximum

$$\begin{aligned} \max \quad & \sum_{v \in \mathcal{V}} x_v \\ s.t. \quad & x_u + x_v \leq 1, \quad \textit{for each edge } \{u, v\} \in \mathcal{E} \\ & x_v \in \{0, 1\} \quad \textit{for all } v \in \mathcal{V} \end{aligned} \tag{5}$$

Any solution to this problem corresponds to an independent set (i.e. the set $I \subseteq \mathcal{V}$ of all v s.t. $x_v^* = 1$). A maximal independent set is an independent set to which no node can be added without violating the independent set property. A maximal independent set is called a maximum independent set of a graph if it is not possible to find any larger MIS in the graph. As an example, in the complete graph, the largest independent set consists of a single vertex

The problems that we have described in the previous 3 examples are combinatorial in the sense that they consist in finding an optimal object from a finite collection.

When it comes to solving integer programs, there are usually two classical approaches:

1. Cutting planes
2. Branch and Bound

To introduce the cutting plane approach, we first need to introduce the notion of valid inequality

Definition 1. *An inequality $\boldsymbol{a}^T \boldsymbol{x} \leq b$ is a valid inequality for $\boldsymbol{x} \in X \subseteq \mathbb{R}^N$ if $\boldsymbol{a}^T \boldsymbol{x} \leq b$ for all $\boldsymbol{x} \in X$*

Example 4 (Integer rounding)**.** *Consider the integer region $X = P \cap \mathbb{Z}^4$ where*

$$P = \left\{ \boldsymbol{x} \in \mathbb{R}^4_+ \mid 13x_1 + 20x_2 + 11x_3 + 6x_4 \geq 72 \right\}$$

Dividing the inequality by 11, we get

$$\frac{13}{11}x_1 + \frac{20}{11}x_2 + x_3 + \frac{6}{11}x_4 \geq \frac{72}{11}$$

Since the variables are non negative, we can always round up the LHS, which further gives

$$\lceil \frac{13}{11} \rceil x_1 + \lceil \frac{20}{11} \rceil x_2 + x_3 + \lceil \frac{6}{11} \rceil x_4 = 2x_1 + 2x_2 + x_3 + x_4 \geq \frac{72}{11} \tag{6}$$

This inquality clearly remains valid for the set P. If we restrict x_1, x_2, x_3, x_4 to integers, we can go one step further and also round up the RHS in (6). I.e, any integer that has to be larger than $\frac{72}{11}$ also has to be larger than 7.

Why do we want to generate new valid inequalities? Recall that the convex envelope is the smallest convex feasible region that contains all of the integer solutions. If one can generate additional linear inequalities that are valid for all integer solutions until the feasible set reduces to the convex envelope of those integer solutions, one can solve the resulting LP and be guaranteed to find the optimal integer solution because all of the extreme points of the convex envelope are integer solutions. Consider the following example whose graphical representation is given in Fig. 4.

Example 5.

$$\begin{aligned} \max \quad & 3x_1 + 4x_2 \\ s.t. \quad & 5x_1 + 7x_2 \leq 21 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned} \tag{7}$$

Valid inequalities are also known as cuts or cutting planes. In the case of problem (7), both of the inequalities $x_1 \leq 5$ and $x_2 \leq 4$ are valid inequalities. In fact, as we will see, we can do better and derive the constraint

$$x_1 + x_2 \leq 4 \tag{8}$$

as well as the constraint

$$x_2 \leq 3 - \frac{2x_1}{3} \tag{9}$$

Adding those inequalities, we get formulation (10) whose feasible set is represented in Fig 4.

$$\begin{aligned} \max \quad & 3x_1 + 4x_2 \\ s.t. \quad & 5x_1 + 7x_2 \leq 21 \\ & x_1 + x_2 \leq 4 \\ & 2x_1 + 3x_2 \leq 9 \\ & x_1, x_2 \geq 0, \\ & x_1, x_2 \in \mathbb{Z} \end{aligned} \tag{10}$$

From what we saw, generating valid inequalities is relatively simple. What is difficult is to generate them in an efficient manner (especially in a high dimensional setting) in order to quickly find the convex envelope of the integer solutions. This can once again

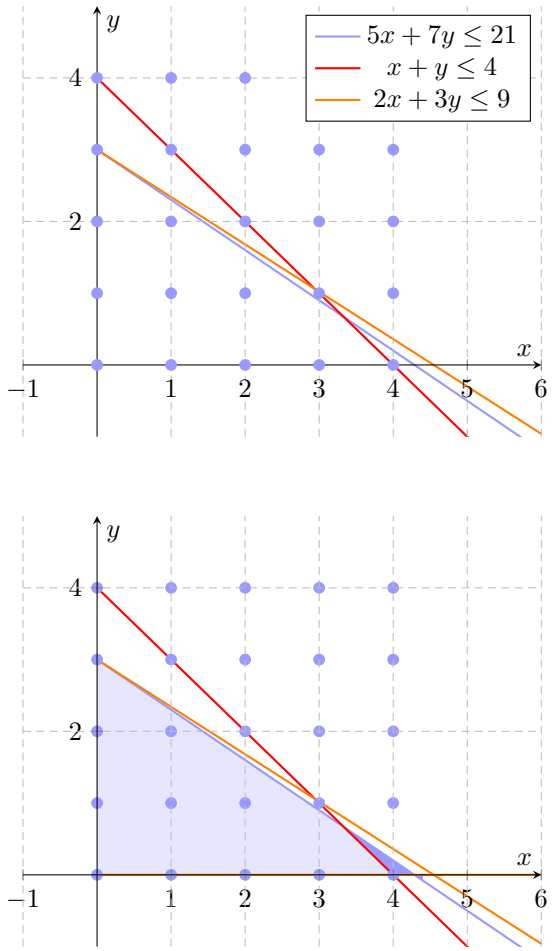

Figure 4: (Top) Representation of the constraints and integer lattice for problem (7). The additional (valid) constraints generated in (8) and (9) are shown in red and orange. (Bottom) Polytope corresponding to the augmented set of constraints (10) (light blue) and difference between this polytope and the polytope associated to the original formulation (7). The reduction is possible because the solutions are *in fine* known to be integer. The restriction does not modify the set of integer solutions and it therefore makes sense to add the supplementary linear constraints (8) and (9).

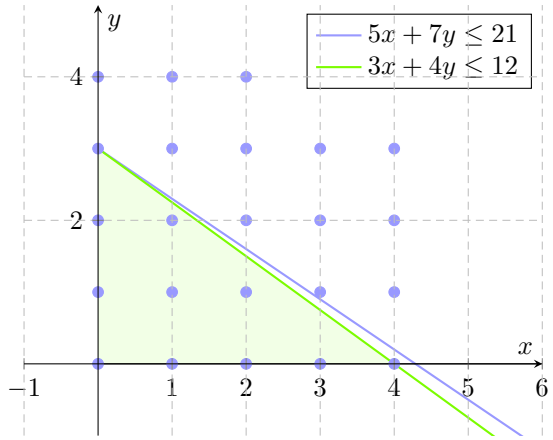

Figure 5: Final convex envelope obtained for problem (7) after adding the valid inequality $4y + 3x \leq 12$. One can check that every extreme point of the convex envelope is an integer solution.

be illustrated by problem (7). Although undeniably more constrained, formulation (10) still does not correspond to the convex envelope of the original IP (7). To finally obtain this envelope (shown in Fig. 5), we need to add the final valid inequality $4x_2+3x_1 \leq 12$ *(which can be derived by combining* $5x_1 + 7x_2 \leq 21$ *with* $x_1 + x_2 \leq 4$ *then dividing by 2 and rounding both sides of the resulting inequality following the Chvátal-Gomory procedure which we introduce next)*

The remaining of this lecture will precisely discuss how to generate valid inequalities efficiently.

Proposition 1. *The inequality* $\boldsymbol{\pi}^T\boldsymbol{x} \leq \pi_0$ *is valid for* $P = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq 0\}$ *if and only if there exists* $\boldsymbol{u} \geq 0$ *such that* $\boldsymbol{\pi} \leq \boldsymbol{u}^T\boldsymbol{A}$ *and* $\boldsymbol{u}^T\boldsymbol{b} \leq \pi_0$

Now that we clarified the general notion of the valid inequality, we discuss the particular setting of integer programming. From our previous example, we can already highlight the following

Proposition 2. *Let* $X = \{y \geq 0 \mid y \leq b\} \cap \mathbb{Z}$*, then the inequality* $y \leq \lfloor b \rfloor$ *is valid for* X

Now that we have covered a few examples of how to generate valid inequalities, let us try to come up with a general procedure. Before formulating the general procedure, let us briefly recap what we just learned. let us consider the feasible set

$$\begin{aligned} 7x_1 - 2x_2 &\leq 14 \\ x_2 &\leq 3 \\ 2x_1 - 2x_2 &\leq 3 \\ \boldsymbol{x} &\geq 0 \\ \boldsymbol{x} &\in \mathbb{Z}^2 \end{aligned} \tag{11}$$

To generate valid inequalities, we saw that we could follow three main steps:

1. We can first combine the constraints according to any weight vector $\boldsymbol{w}$ with non negative weights. For problem (11) we can for example take $\boldsymbol{w} = \left(\frac{2}{7}, \frac{37}{63}, 0\right)$. Combining the constraints in (11) according to this vector, we get the new valid inequality
$$2x_1 + \frac{1}{63}x_2 \leq \frac{121}{21}$$
2. Since the RHS gives an upper bound on the sum $2x_1 + \frac{1}{63}x_2$ and since the variables are non negative, we can always derive a valid constraint by rounding the LHS. I.e.
$$\lfloor 2 \rfloor x_1 + \lfloor \frac{1}{63} \rfloor x_2 \leq 2x_1 + \frac{1}{63}x_2 \leq \frac{121}{21}$$
3. Now because we focus on integer solutions, and because after rounding the multiplicative coefficients on the LHS, this side is guaranteed to take an integer value, we can also round the right-hand side (see Proposition (2) and note that any integer smaller than 121/21 necessarily has to be smaller than 5)
$$2x_1 \leq \lfloor \frac{121}{21} \rfloor = 5$$

Our previous observations can be summarized by the following general procedure known as *Chvátal-Gomory cut*. We let $X = P \cap \mathbb{Z}^n$ where $P = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq b\}$, $u \in \mathbb{R}_+$.

(i) The inequality
$$\sum_{j=1}^{n} u a_j x_j \leq ub$$
is valid for P as long as $u \geq 0$ and $\sum_j a_j x_j \leq b$

(ii) The inequality
$$\sum_{j=1}^{n} \lfloor u a_j \rfloor x_j \leq ub$$
is valid for P as $x_j \geq 0$.

(iii) Finally, the inequality
$$\sum_{j=1}^{n} \lfloor u a_j \rfloor x_j \leq \lfloor ub \rfloor$$
is valid for any integer solution $\boldsymbol{x}$

As indicated by the following theorem, that simple procedure is sufficient to generate all valid inequalities for integer programs. This idea is summarized by the following theorem

Theorem 2. *Every valid inequality for X can be obtained by applying the* Chvátal-Gomory *procedure a finite number of times.*

In many cases, the number of valid inequalitites we would have to add is enormous. Moreover, given a specific objective function, we are in general not interested in finding the complete convex envelope but we are instead usually happy with a good approximation of this envelope in the neighborhood of the solution.

General useful inequalities can be generated by alternating between (i) solving the linear program through the simplex algorithm and (ii) refinining the feasible set through the addition of a valid inequality that "cuts off" the solution returned by the simplex if this solution is not an integral solution. By doing so, we are guaranteed to gradually refine the polytope until we reach a set whose vertices are all integral solutions.

We summarize this idea below in the general case and we then derive the explicit procedure in the case of integer programs.

The problem of "cutting off" a solution that is not feasible for the original (i.e. integral) formulation by adding a constraint to the relaxation is known as the *separation problem.*

Definition 2. *The separation problem associated with a combinatorial optimization problem (COP)* $\max\left\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{x} \in X \subseteq \mathbb{R}^n\right\}$ *can be formulated as follows*

- *Given* $\boldsymbol{x}^* \in \mathbb{R}^n$*, is* $\boldsymbol{x}^* \in conv(X)$*?*
- *If not, find an inequality* $\boldsymbol{a}^T\boldsymbol{x} \leq b$ *satisfied by all points in* X *but violated by the point* $\boldsymbol{x}^*$*.*

We can now introduce our general procedure to derive valid inequalities efficiently. In the pseudo-code below, we assume that we don't have access to the set X except through an oracle to which we submit our queries.

Algorithm 1: Naive Algorithm

1. We consider the set $X \subseteq P$ (any finite subset of P). Set $t = 0$, $P^0 = P$

   a) For $t = 0, \ldots,$

      (i) Solve the linear program

      $$\max \left\{ \boldsymbol{c}^T \boldsymbol{x} \mid \boldsymbol{x} \in P^t \right\}$$

      Let $\boldsymbol{x}^t$ be an optimal solution. if $\boldsymbol{x}^t \in X$ stop. $\boldsymbol{x}^t$ is an optimal solution for the COP. Else, solve the separation problem for $\boldsymbol{x}^t$ and the current family of constraints.

      (ii) if an inequality is found in V (family of valid inequalities) with $\boldsymbol{a}_i^T \boldsymbol{x}^t \leq b_i$ so that it cuts off $\boldsymbol{x}^t$, set $P^{t+1} = P^t \cap \left\{ \boldsymbol{x} \mid \boldsymbol{a}_i^T \boldsymbol{x} \leq b_i \right\}$ and augment i. Otherwise stop.

   b) If the algorithm iterates without finding a solution in X, then

   $$P^T = P \cap \cup_{i=1}^{T} \left\{ \boldsymbol{a}_i^T \boldsymbol{x} \leq b_i \right\}$$

   can be used to initialize a Branch and Bound algorithm.

Gomory's Fractional Cutting Plane Algorithm

We now discuss the particular case of integer programs. The restriction of the above procedure to that setting is known as *Gomory's fractional cutting plane algorithm*. In this particular setting, we focus our attention on the integer program

$$\max \left\{ \boldsymbol{c}^T \boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \in \mathbb{Z}_+^n \right\} \tag{12}$$

The idea is to first solve the LP relaxation with the SIMPLEX and find an optimal basis. Then choose one of the basic variable that is not integer and solve the separation problem for the corresponding constraint so as to "cut off" the fractional solution returned by the SIMPLEX.

Let B and N respectively denote the sets of basic and non basic variables. Suppose that after solving the original LP, one can write the problem as

$$\max \quad \overline{a}_{00} + \sum_{j \in N} \overline{a}_{0j} x_j \tag{13}$$

$$\text{s.t.} \quad x_{B_k} + \sum_{j \in N} \overline{a}_{kj} x_j = \overline{b}_k, \quad k = 1, \ldots, m \tag{14}$$

$$x \geq 0, \quad x \in \mathbb{Z} \tag{15}$$

with $\overline{a}_{0j} \leq 0$ for $j \in N$ (recall that the SIMPLEX terminates when all the residual costs are negative) and $\overline{b}_k \geq 0$ for $u = 1, \ldots, m$ (note that if this is not the case, we can always multiply the equation by -1).

If the basic optimal solution is not an integer, there must exist a row k with $b_k \notin \mathbb{Z}$.

Choosing this row and applying the Chvátal-Gomory procedure, we can generate an inequality of the form

$$x_{B_k} + \sum_{j \in N} \lfloor a_{kj} \rfloor x_j \leq \lfloor b_k \rfloor \tag{16}$$

Subtracting this to the original constraint,

$$x_{B_k} + \sum_{j \in N} a_{kj} x_j = b_k \tag{17}$$

we get (keep in mind that given the inequality (16), when subtracting, we subtract to the left-hand side of (17) something that is smaller than what we subtract to the right-hand side. Consequently, we have to flip the sign of the inequality)

$$x_{B_k} + \sum_{j \in N} a_{kj} x_j - \left(x_{B_k} + \sum_{j \in N} \lfloor a_{kj} \rfloor x_j \right) \geq b_k - \lfloor b_k \rfloor$$

The final constraint is then given by

$$\sum_{j \in N} \underbrace{(a_{kj} - \lfloor a_{kj} \rfloor)}_{f_{kj}} x_j \geq \underbrace{b_k - \lfloor b_k \rfloor}_{f_{k0}} \tag{18}$$

Now recall that in the solution of our original LP, all the non basic variables took the value 0 by construction. As a result, our additional inequality (18) (whose LHS is a combination of non basic variables and whose RHS is non zero) "cuts off" the solution returned by the first SIMPLEX run. As a result, we can then add this inequality to the definition of the polytope and rerun the SIMPLEX a second time on the new formulation.

As an illustration of Gomory's fractional cutting plane algorithm, consider the following example which is taken from [1]

Example 6.

$$\begin{array}{ll} \max & 4x_1 - x_2 \\ s.t. & 7x_1 - 2x_2 \leq 14 \\ & x_2 \leq 3 \\ & 2x_1 - 2x_2 \leq 3 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{array} \tag{19}$$

Adding slack variables s_3, s_4, s_5 and solving the resulting LP relaxation we get the tableau 1

Recall that as we maximize, the SIMPLEX *stops when none of the entries in the reduced cost vector (last row in the simplex tableau) are positive anymore. Tableau 1*

x_1	x_2	s_3	s_4	s_5	
1	0	1/7	2/7	0	20/7
0	1	0	1	0	3
0	0	-2/7	10/7	1	23/7
0	0	-4/7	-1/7	0	

Table 1: Final tableau obtained after a complete run of the SIMPLEX method on problem (19)

x_1	x_2	s_3	s_4	s_5	s_6	
1	0	0	0	0	1	2
0	1	0	0	-1/2	1	1/2
0	0	1	0	-1	-5	1
0	0	0	1	1/2	6	5/2
0	0	0	0	-1/2	-3	

Table 2: Final tableau obtained after a second run of the Simplex on problem (19) augmented with the valid inequality (22)

corresponds to a termination step as the entries in the reduced cost vector are all negative.

The tableau corresponds to the optimal solution $x = (\frac{20}{7}, 3, 0, 0, \frac{23}{7})$. *one can thus generate a first cut by focusing on* x_1 *or* x_5 *(the non integral basic variables). Starting with* x_1 *and applying the Chvátal-Gomory procedure, we get the new cut constraint*

$$\frac{1}{7}s_3 + \frac{2}{7}s_4 \geq \frac{6}{7} \tag{20}$$

which corresponds to taking the constraint

$$x_1 + \frac{1}{7}s_3 + \frac{2}{7}s_4 = \frac{20}{7} \tag{21}$$

generating the constraint

$$x_1 + \lfloor\frac{1}{7}\rfloor s_3 + \lfloor\frac{2}{7}\rfloor s_4 \leq \frac{20}{7}$$

rounding the RHS,

$$x_1 + \lfloor\frac{1}{7}\rfloor s_3 + \lfloor\frac{2}{7}\rfloor s_4 \leq \lfloor\frac{20}{7}\rfloor$$

and subtracting this constraint from (21)*, we get the cut*

$$\frac{1}{7}s_3 + \frac{2}{7}s_4 \geq \frac{6}{7} \tag{22}$$

Adding (22) *to formulation* (19) *and running the* SIMPLEX *a second time, we get the final tableau 2. The optimal solution for this tableau is now given by* $\boldsymbol{x} =$

$(2, 1/2, 1, 5/2, 0)$. *From this, we can again choose either* x_2 *or* x_4 *to generate the new cut constraint. Taking* x_2*, we get the Chvátal-Gomory cut*

$$\begin{aligned} & x_2 - \frac{1}{2}s_5 + s_6 = 1/2 \\ \Rightarrow \quad & x_2 - \lceil 1/2 \rceil s_5 + s_6 \leq 1/2 \\ \Rightarrow \quad & x_2 - s_5 + s_6 \leq 1/2 \\ \Rightarrow \quad & x_2 - s_5 + s_6 \leq 0, \quad \textit{for } x, s \in \mathbb{Z}_+ \end{aligned}$$

subtracting from the original constraint, we derive the inequality

$$\frac{1}{2}s_5 \geq 1/2 \tag{23}$$

Adding this new constraint to the LP (19) *and running the* SIMPLEX *a third time, we end up with the final tableau 3*

x_1	x_2	s_3	s_4	s_5	s_6	s_7	
1	0	0	0	0	1	0	2
0	1	0	0	0	1	-1	1
0	0	1	0	0	-5	-2	2
0	0	0	1	0	6	1	2
0	0	0	0	1	0	-1	1
0	0	0	0	0	-3	-1	

Table 3: Final tableau obtained after a third run of the SIMPLEX on problem (19) augmented with the Chvàtal-Gomory cuts (22) and (23).

Branch and Bound

The most natural way to solve a combinatorial problem is by enumeration. We start by giving an example of a pure enumerative approach. We then introduce the "Branch and Bound" procedure in more details.

Consider the following instance of the knapsack problem with 5 items

$$\begin{aligned} \max \quad & 5x_1 + 3x_2 + 6x_3 + 4x_4 + x_5 \\ \text{s.t.} \quad & 2x_1 + 7x_2 + 3x_3 + x_4 + 2x_5 \leq 14 \\ & x_1, x_2, x_3, x_4, x_5 \in \{0, 1\} \end{aligned} \tag{24}$$

For a set of n variables, complete enumeration in the case of the knapsack problem would consists in investigating each of the 2^n candidate solutions. Such an enumeration can be carried out by means of a tree. I.e. start with the root node and "Branch" the possibilities for the first variable. Then "branch" possibilities for the second variable x_2, and so on. The first two levels of the tree are shown in Fig. 6. Continuing like this for problem (24) will yield a tree of depth 5 in which each node will correspond to a partial solution. Each leaf node will correspond to a particular candidate solution.

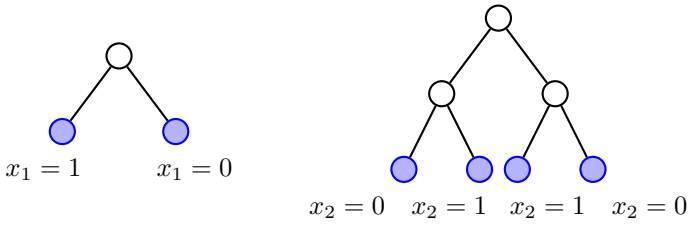

Figure 6: Enumeration tree for the Knapsack instance (24).

To reduce the complexity, the idea is to stop branching as early as possible. If we look at a node close to the root and conclude that none of its descendants can be optimal, this might be used to reduce the computation drastically. This desire to reduce the complexity as much as possible lies at the core of the Branch and Bound procedure which we discuss next.

The general idea of Branch and Bound is to use the LP relaxations as way to "sort" the nodes. A subdomain on which the corresponding LP relaxation yields poor upper (for a maximization problem) or lower (for a minimization problem) bounds should not be investigated further.

We illustrate this procedure through the following example

Example 7.

$$\begin{array}{ll} \max & 4x_1 - x_2 \\ s.t. & 7x_1 - 2x_2 \leq 14 \\ & x_2 \leq 3 \\ & 2x_1 - 2x_2 \leq 3 \\ & \boldsymbol{x} \in \mathbb{Z}^2_+ \end{array} \tag{25}$$

The final SIMPLEX tableau for problem (25) is given by

$$\begin{array}{llllll} \max & & -4/7x_3 & -1/7x_4 & & \\ & x_1 & +1/7x_3 & +2/7x_4 & & = 20/7 \\ & x_2 & & +x_4 & & = 3 \\ & & -2/7x_3 & +10/7x_4 & +x_5 & = 23/7 \\ & & & x_1,\ x_2,\ x_3,\ x_4,\ x_5 & \geq 0 & \end{array} \tag{26}$$

The corresponding objective is given by 59/7. Since no feasible solution is available yet and since problem (26) is a maximization problem, we can set our intial lower bound as $\underline{z} = -\infty$.

The idea of Branch and Bound is to continue to explore as long as the lower bound is smaller than the upper bound.

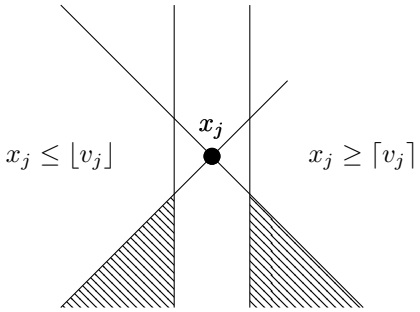

Figure 7: Graphical representation of the "branching" step in Branch and Bound.

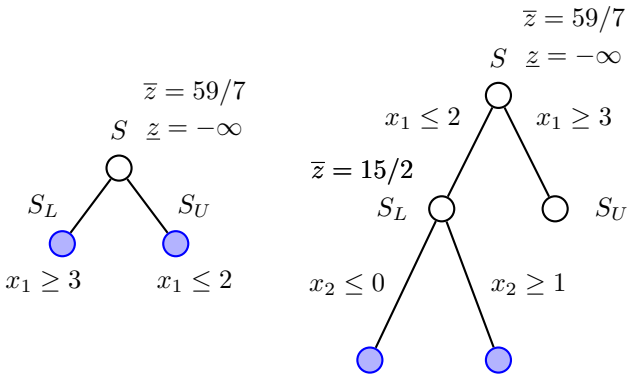

Figure 8: Evolution of the Branch and Bound tree for the integer program (25) (Part I).

A good approach for the split is to take one of the basic variables that takes a fractional value. If we let x_j and v_j to denote such a fractional variable and its associated value, and if we split the feasible set S into the two subsets

$$S_B = S \cap \{x_j \leq \lfloor v_j \rfloor\}, \quad S_U = S \cap \{x_j \geq \lceil v_j \rceil\} \tag{27}$$

we do not lose any information as all the integer solutions are maintained in the feasible set (i.e. the white strip in Fig. 7 cannot contain any integer solution by construction). Moreover, we are guaranteed to get rid of the fractional solution and hence to reduced the size of the polytope. I.e.

$$\{\boldsymbol{x} \in S_B\} \vee \{\boldsymbol{x} \in S_U\} \leq \max\{\boldsymbol{x} \in S\}$$

Applying this idea to the tableau (26), we can start building our solution tree by adding the two nodes $x_1 \leq 2$ and $x_1 \geq 3$ (see Fig. 8).

To proceed further, we arbitrarily pick S_L as our next subproblem. Adding the

constraint $x_1 \leq 2$ to the original IP, we get the new problem

$$\begin{array}{ll} \max & 4x_1 - x_2 \\ \text{s.t.} & 7x_1 - 2x_2 + x_3 = 14 \\ & x_2 + x_4 = 3 \\ & 2x_1 - 2x_2 + x_5 = 3 \\ & x_1 + x_6 = 2 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \in \mathbb{Z}_+ \end{array} \tag{28}$$

Solving the LP relaxation for this new formulation with the SIMPLEX, we get the final tableau

$$\begin{array}{lllllll} \max & & & -1/2x_5 & -3x_6 & & \\ & x_1 & & & & +x_6 & = 2 \\ & & x_2 & & -1/2x_5 & +x_6 & = 1/2 \\ & & & x_3 & -x_5 & -5x_6 & = 1 \\ & & & x_4 & +1/2x_5 & +6x_6 & = 5/2 \\ & & & & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 & & \end{array} \tag{29}$$

corresponding to the objective $15/2$ (which is again an upper bound on the objective of the original IP) and solution $x_1 = 2,\ x_2 = 1/2$.

Since x_1 takes integer value, we cannot use it to define the next split. Instead, we now select x_2 and define the two subsets

$$S_{LL} = \left\{ \boldsymbol{x} \in S_L \mid x_2 \leq \lfloor \frac{1}{2} \rfloor \right\}, \quad S_{LU} = \left\{ \boldsymbol{x} \in S_L \mid x_2 \geq \lceil \frac{1}{2} \rceil \right\} \tag{30}$$

The corresponding tree is shown in Fig. 8 (right).

Continuing, arbitrarily choosing S_U as our next node and adding the constraint $x_1 \geq 3$ to formulation (28), we get the following problem

$$\begin{array}{ll} \max & 4x_1 - x_2 \\ \text{s.t.} & 7x_1 - 2x_2 + x_3 = 14 \\ & x_2 + x_4 = 3 \\ & 2x_1 - 2x_2 + x_5 = 3 \\ & x_1 - x_6 = 3 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{array} \tag{31}$$

By inspection (use $x_1 \leq 3$ together with $x_2 \geq 3$, $x_3 \geq 0$ and combine with the first constraint in (31)), one can conclude that the subdomain is empty. We can thus stop exploring the S_U branch of the tree. Continuing with S_{LU}, and adding the constraint

$x_1 \geq 1$ to problem (28), we get the program

$$\begin{aligned} \max \quad & 4x_1 - x_2 \\ \text{s.t.} \quad & 7x_1 - 2x_2 + x_3 = 14 \\ & x_2 + x_4 = 3 \\ & 2x_1 - 2x_2 + x_5 = 3 \\ & x_1 + x_6 = 2 \\ & x_2 - x_7 = 1 \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \end{aligned} \tag{32}$$

The solution to problem (32) is given by $x_1 = 2, x_2 = 1$ with objective value 7. Since the solution in this case is integral, the objective not only constitutes an upper bound on the solution of the problem on the subdomain S_{LU}, but it also constitutes a lower bound on the objective of the original problem (25). The updated tree is shown in Fig. 9 (left). Since the subdomain S_U is empty, and since our solution for S_{LU} is both an upper and a lower bound, we are left with investigating the subdomain S_{LL} (which could still yield a better solution). For this last subdomain, we get the formulation

$$\begin{aligned} \max \quad & 4x_1 - x_2 && (33) \\ \text{s.t.} \quad & 7x_1 - 2x_2 + x_3 = 14 && (34) \\ & 2x_1 - 2x_2 + 5 = 3 && (35) \\ & x_1 + x_6 = 2 && (36) \\ & x_2 = 0 && (37) \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 && (38) \end{aligned}$$

Solving the LP will yield the solution $x = (3/2, 0)$ with optimal value $z = 6$. Since the solution is fractional, this last bound only constitute an upper bound for the subdomain. Since this upper bound is smaller than our current lower bound, we can disregard S_{LL}. The resulting complete tree with the final solution is shown in Fig. 9 (right).

References

[1] Laurence A Wolsey, *Integer programming*, 2020, John Wiley & Sons

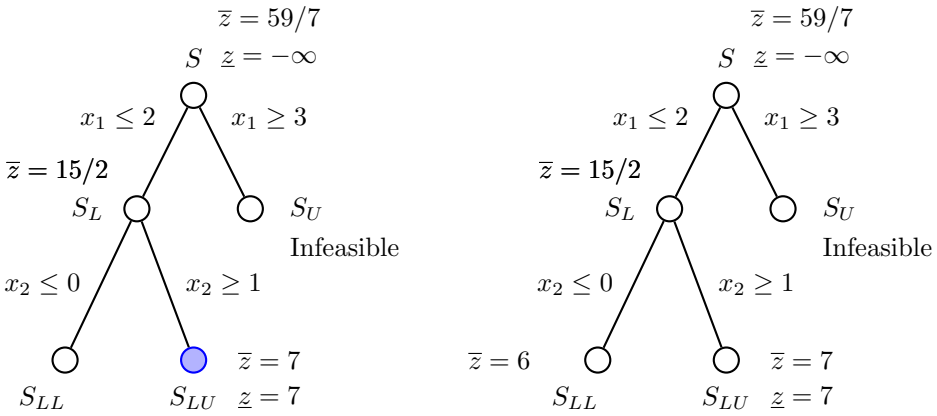

Figure 9: Evolution of the Branch and Bound tree for the integer program (25) (Part II).