Introduction to Optimisation, Lecture 1

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

In this course we will be concerned with general problems of the form

$$\min \quad f(x) \tag{1}$$
$$\text{s.t.} \quad x \in S \tag{2}$$

Here $f : S \mapsto \mathbb{R}$ and S denotes the set of feasible solutions. it can be shown (Weiertrass) that if S is compact and f is a continuous function, then the optimum exists. We will cover various instances of problem (2) (see Figure 1 for a (non-exhaustive) diagram).

Linear Programming

The problem of solving a set of linear equations can be traced back to Fourier (1826/1827) who introduced one of the first resolution method known today as the *Fourier-Motzkin* elimination. The creation of linear programming as a discipline, together with its first serious recognition came with the work of Dantzig, Kantorovitch, Koopmans and von Neumann.

Dantzig introduced what is known today as the Simplex algorithm. von Neumann established the theory of duality. The main contribution of Kantorovitch to linear programming, which was rewarded by the Nobel prize in economics, followed from an interest in the distribution of raw material which he developed while working as a consultant around 1938 for the Soviet government's laboratory of the Pywood trust, and with the objective of maximizing equipment productivity under certain

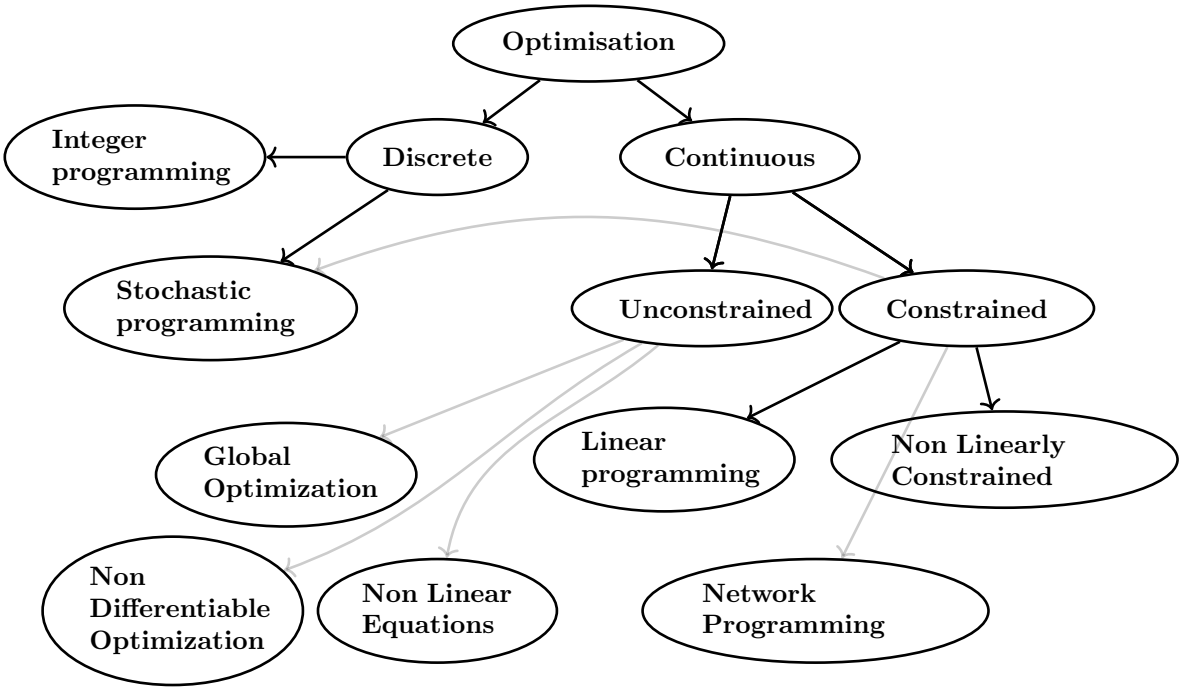

Figure 1: A few of the (many) optimization subfields and the connections among those.

restrictions. Kantorovitch found that many (seemingly) different problems shared a same mathematical form: distribution of equipement, best use of sowing area, distribution of transport flows,... Kantorovitch gathered those findings together with what would constitute the foundations of linear programming in his booklet *The Mathematical Method of Production Planning and Organization*. Similar ideas were developed around the time by Tjalling Koopmans and Georges Dantzig. The career of Kantorovitch would then be devoted to those ideas and their generalizations.

In linear programming, both the function f and the set S are described by linear

combinations of the variables. The general form of a linear program is as follows

$$\begin{aligned}
\min \quad & \sum_{i=1}^{N} c_i x_i \\
\text{s.t.} \quad & \sum_{j=1}^{N} a_{ij} x_j \leq b_i, \quad i \in I_1 \\
& \sum_{j=1}^{N} a_{ij} x_j \geq b_i, \quad i \in I_2 \\
& \sum_{j=1}^{N} a_{ij} x_j = b_i, \quad i \in I_3 \\
& \\
& x_j \geq 0, \quad j \in V_1 \\
& x_j \leq 0, \quad j \in V_2
\end{aligned}$$

Compactly, we can write the problem as

$$\begin{aligned}
\min \quad & \boldsymbol{c}^T \boldsymbol{x} \\
\text{s.t.} \quad & \boldsymbol{a}_i^T \boldsymbol{x} \geq b_i, \quad i \in I_1 \\
& \boldsymbol{a}_i^T \boldsymbol{x} \leq b_i, \quad i \in I_2 \\
& \boldsymbol{a}_i^T \boldsymbol{x} = b_i, \quad i \in I_3 \\
& x_j \geq 0, \quad j \in V_1 \\
& x_j \leq 0, \quad j \in V_2
\end{aligned}$$

Linear optimization spans a wide range of applications including transportation, air traffic control, movements of truck loads, telecommunication, manufacturing, medicine, engineering,... Three particular applications are discussed below.

The transportation problem

In the transportation setting, a given product has to be shipped in distinct amounts $u_1, u_2, \ldots, u_n$ from n service points to m destinations where it is needed in respective amounts $v_1, v_2, \ldots, v_m$. If the cost of sending one unit of the given product from source i to destination j is encoded as c_{ij}, the transportation problem consists in estimating the quantities x_{ij} to be sent from the factories i to the destinations j so that the total transportation cost is minimized. The problem can be represented graphically by means of a complete graph $K_{m,n}$ with edge weights given by c_{ij} (see Fig.2)

From the quantities that are shipped from i to j, the total cost reads as

$$\sum_{i,j} c_{ij} x_{ij}$$

To express the constraints, let us note that for any fixed shipping point i, the total quantity to be shpipped is given by u_i so that

$$\sum_j x_{ij} = u_i, \quad i = 1, \ldots, n$$

Likewise, for any given destination j, the quantity that should be received by j is given by v_j. Hence

$$\sum_i x_{ij} = v_j, \quad j = 1, \ldots, m$$

Note that the two sets of equalities above necessarily imply

$$\sum_{i=1}^{n} u_i = \sum_{j=1}^{m} v_j$$

It also seems natural to require the quantities shipped x_{ij} to be non negative, i.e. $x_{ij} \geq 0$ for every i, j. Altogether, the transportation problem can thus read mathematically as

$$\begin{aligned} \min \quad & \sum_{ij} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_j x_{ij} = u_i, \quad i = 1, 2, \ldots \\ & \sum_i x_{ij} = v_j, \quad j = 1, 2, \ldots \\ & x_{ij} \geq 0, \quad \text{for all } i, j \end{aligned}$$

The diet problem

Another instance of a linear problem is the *diet problem*. In the diet problem, we have at our disposal the nutritional values as well as the prices of a number of products together with a chart indicating the daily requirements for each nutrient. The objective then consists in determining a mix of the different products that could be purchased so as to meet the nutritional requirements at a mimimal cost.

If we let x_i to denote the amount of product i that should be purchased, and if c_i is used to denote the unit price for product i, the total cost of the combination is then given by

$$\sum_{i=1}^{n} c_i x_i.$$

For each nutrient j, we must make sure that the requirement is met. For this, we introduce the variables a_{ij}; which encode the nutritional value (in the j^{th} nutrient)

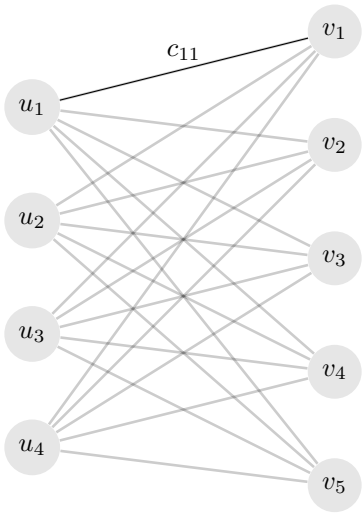

Figure 2: Graphical representation of the transportation problem. The cost associated to the path from u_1 to v_1 is highlighted and indicated as c_{11}

of product i; and b_j the minimal requirement in the same nutrient. The nutritional constraint then reads as

$$\sum_{i=1}^{n} a_{ji} x_i \geq b_j, \quad \text{for every } j$$

As in the transportation problem, the amount in each nutrient is required to be non negative $x_i \geq 0$. The final problem then reads as

$$\begin{aligned} \min \quad & \sum_i c_i x_i \\ \text{s.t.} \quad & \sum_i a_{ji} x_i \geq b_j, \quad \text{for every } j \\ & x_i \geq 0, \quad \text{for every } i \end{aligned}$$

Portfolio optimization

As a last application, we consider the problem of determining, for a given portfolio, which stocks and how many of those stocks one should sell to free a given amount C of cash while maintaining maximum value of the portfolio. Let us use p_i to denote the current price of shares from company i and let us assume that the portfolio is currently made of x_i shares from company i that were purchased at an initial price of q_i. Let us first use f_i to denote the expected future price of the shares from company i. We assume that the owner of the portfolio pays a tax of 30% on every capital gained by selling a share and that he/she pays an additional 1% as transaction cost (i.e we assume that the owner uses an external broker). The question we want to answer is then "*which amount of what shares do we have to sell in order to free an amount* C

of cash while keeping the highest possible value for the portfolio?". Once again, one can check that the problem just described can be expressed as a linear program

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} f_i(s_i - x_i) \\ \text{s.t.} \quad & \sum_{i=1}^{n} p_i x_i - \frac{30}{100}\sum_{i=1}^{n}(p_i - q_i)x_i - \frac{1}{100}\sum_{i=1}^{n} p_i x_i \geq C \\ & 0 \leq x_i \leq s_i \end{aligned}$$

The geometry of linear programming

Each inequality in linear programming represents a *half-space*. The feasible region being given by the intersection of all those halfspaces $\cap_i \left\{ \boldsymbol{x} \mid \boldsymbol{a}_i^T \boldsymbol{x} \leq b_i \right\}$.

Such an intersection is known as a *polytope* (polygon in 2D and polyhedron in 3D).

Given a feasible point $\boldsymbol{x} \in \mathbb{R}^n$, a constraint $\boldsymbol{a}_i^T \boldsymbol{x} \leq b_i$ is called *active* (or binding) if $\boldsymbol{a}_i^T \boldsymbol{x} = b_i$ otherwise it is called *inactive*.

Let us consider the mathematical formulation of the Lipton problem (see recitation 1). The cost in this problem being linear, it has level curves that are straight lines of equation $x_1 + 1.5x_2 = c$. When varying the value of c, we thus obtain parallel lines. The question is then *how big can we take c for a feasible pair (x_1, x_2)*?. Since all the pairs that achieve a profit of c are located along the line $x_1 + 1.5x_2 = c$, we can move the line to maximize the value of c and study when the intersection of this line with the feasible set remains non empty for increasing values of c (see Fig. 4 for an illustration of this).

From the discussion above, it is not difficult to see that under the assumption that the linear program has a solution, at least one of the optimal solutions will be located at a vertex of the feasible set.

To clarify what we mean by a vertex, we start by recalling a few geometric notions.

Definition 1. *Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k \in \mathbb{R}^n$ and let $\lambda_1, \ldots, \lambda_k \in \mathbb{R}_{\geq 0}$ such that $\sum_{i=1}^k \lambda_i = 1$. The vector $\sum_{i=1}^k \lambda_i \boldsymbol{x}_i$ is called a* convex combination *of the $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$. The set of all convex combinations of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ is called* convex hull *of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$*

Clearly every convex hull is a polytope. We can now proceed with a formal characterization of vertices and extreme points.

Definition 2 (extreme point)**.** *$\boldsymbol{x}$ is an* extreme point *of the polytope P if it cannot be written as a convex combination of two other points in P. I.e. there do not exist $\boldsymbol{y}, \boldsymbol{z} \in P$ and $\lambda \in (0, 1)$ s.t. $\boldsymbol{x} = \lambda \boldsymbol{y} + (1 - \lambda)\boldsymbol{z}$.*

Unlike the more general extreme points, vertices are characterized by the existence of at least one objective vector with respect to which no improvement can be achieved without leaving the feasible set.

Figure 3: Illustration of the distinction between vertices that are also extreme points (blue) and extreme points that are not vertices or exposed (red). The convex hull is shown in green.

Definition 3 (vertex or exposed point)**.** $\boldsymbol{x}$ *is a* vertex *of the polytope* P *if there exists a vector* $\boldsymbol{c}$ *such that* $\boldsymbol{c}^T\boldsymbol{x} < \boldsymbol{c}^T\boldsymbol{y}$ *for all* $\boldsymbol{y} \in P, \boldsymbol{y} \neq \boldsymbol{x}$

In a convex polytope defined by a finite intersection of half-spaces, all the extreme points are also vertices. The two notions are illustrated in Fig. 3.

The connection between the optimal solution of a linear program and the extreme points of its associated polytope can be summarized by the following theorem

Theorem 1. *Given a bounded linear program whose feasible set contains at least one extreme point, there is always an extreme point that is an optimal solution*

Proof. Let $V = \left\{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} = b, \boldsymbol{x} \geq 0, \boldsymbol{c}^T\boldsymbol{x} = v\right\}$. The set of feasible points with minimal objective value v. The feasible set S contains no line since it contains one extreme point. Similarly, the set V cannot contain any line either as $V \subseteq S$. As a result, V must contain an extreme point. Let us denote this point as $\boldsymbol{x}^*$.

From the above, we thus have $\boldsymbol{x}^* \in V \subseteq S$. Let us assume that $\boldsymbol{x}^*$ is not an extreme point of S. In this case, there must exist $\lambda \in (0, 1)$ such that $\boldsymbol{x}^* = \lambda\boldsymbol{y} + (1-\lambda)\boldsymbol{z}$ and $\boldsymbol{y}, \boldsymbol{z} \in S$. Now take $\boldsymbol{c}^T\boldsymbol{x}^* = \lambda\boldsymbol{c}^T\boldsymbol{y} + (1-\lambda)\boldsymbol{c}^T\boldsymbol{z}$. Since $\boldsymbol{x}^*$ is an extreme point of V, either $\boldsymbol{y} \notin V$ or $\boldsymbol{z} \notin V$. As a result, at least one of $\boldsymbol{y}$ and $\boldsymbol{z}$ must have a value $\boldsymbol{c}^T\boldsymbol{y}$ (resp. $\boldsymbol{c}^T\boldsymbol{z}$) smaller than v which implies the contradiction

$$\lambda\boldsymbol{c}^T\boldsymbol{y} + (1-\lambda)\boldsymbol{c}^T\boldsymbol{z} < v = \boldsymbol{c}^T\boldsymbol{x}^*$$

□

Standard form

Linear programs admit several equivalent forms.

Definition 4. *The standard form of a linear program is given by*

$$\min \quad \boldsymbol{c}^T\boldsymbol{x}, \quad s.t. \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq 0$$

The main ingredients of every LP are thus

- an $m \times n$ matrix $\boldsymbol{A}$ with $n > m$
- a vector $\boldsymbol{b} \in \mathbb{R}^m$
- a vector $\boldsymbol{c} \in \mathbb{R}^n$

Any linear program can be transformed into the standard form through 3 three main steps

(i) *Variable without sign restriction.* In this case, we introduce a decomposition into a positive and a negative part x^+ and x^- so that $x = x^+ - x^-$, $|x| = x^+ + x^-$ and $x^+, x^- \geq 0$. I.e. any variable x_i not restricted in sign can be written as the difference of two new variables that are both non negative.

(ii) *Transforming inequalities into equalities.* Very often, linear programs will be formulated in terms of inequalities. I.e.

$$\min \quad \boldsymbol{c}^T\boldsymbol{x}, \quad s.t. \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{A}'\boldsymbol{x} = \boldsymbol{b}',\ \boldsymbol{x} \geq 0$$

First note that by multiplying an inequality by -1 we can always change the direction of the inequality. To turn an inequality into an equality, we can rely on the notion *slack variable*. I.e. for any constraint $\boldsymbol{a}^T\boldsymbol{x} \leq b$, we can introduce a variable s as $s = b - \boldsymbol{a}^T\boldsymbol{x}$ and rewrite the constraint as $\tilde{\boldsymbol{a}}^T\tilde{\boldsymbol{x}} = \boldsymbol{b}$ where $\tilde{\boldsymbol{a}} = [\boldsymbol{a}, 1]$, $\tilde{\boldsymbol{x}} = [\boldsymbol{x}, s]$. The number of slack variables in the standard form is equal to the number of inequality constraints in the original formulation.

(iii) Finally, note that any maximization can always be turned into a minimization by flipping the sign of the objective function

$$\max \quad f(x) = -\min\ -f(x).$$

Example 1. *Consider the following program*

$$\begin{aligned} \max \quad & 4x_1 - 2x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 = 2 \\ & x_1 - 2x_2 - x_3 \leq 1 \\ & x_1 + x_3 \geq -2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

1. *For the variables that are restricted in sign, x_3 in this case, we introduce a decomposition into positive and negative part,*

$$x_3 = x_3^+ - x_3^-, \quad x_3^+, x_3^- \geq 0$$

From this, the original problem turns into

$$\begin{aligned} \max \quad & 4x_1 - 2x_3 \\ & x_1 + x_2 + x_3^+ - x_3^- = 2 \\ & x_1 - 2x_2 - x_3^+ + x_3^- \leq 1 \\ & x_1 + x_3^+ - x_3^- \geq -2 \\ & x_1, x_2, x_3^+, x_3^- \geq 0 \end{aligned}$$

2. *We now introduce non negative slack variables to turn the inequalities into equalities. Starting with the first inequality, and adding a non negative correction to the LHS, we get*

$$x_1 - 2x_2 - x_3^+ + x_3^- + x_4 = 1, \quad x_4 \geq 0$$

For the second one, a similar reasoning yields

$$x_1 + x_3^+ - x_3^- - x_5 = -2, \quad x_5 \geq 0$$

After this second step, our problem thus reads as

$$\begin{aligned} \max \quad & 4x_1 - 2x_3 \\ & x_1 + x_2 + x_3^+ - x_3^- = 2 \\ & x_1 - 2x_2 - x_3^+ - x_3^- + x_4 = 1 \\ & x_1 + x_3^+ - x_3^- - x_5 = -2 \\ & x_1, x_2, x_3^+, x_3^-, x_4, x_5 \geq 0 \end{aligned}$$

3. *As our final step, we turn the maximization into a minimization which finally gives*

$$\begin{aligned} \min \quad & 2x_3 - 4x_1 \\ & x_1 + x_2 + x_3^+ - x_3^- = 2 \\ & x_1 - 2x_2 - x_3^+ + x_3^- + x_4 = 1 \\ & x_1 + x_3^+ - x_3^- - x_5 = -2 \\ & x_1, x_2, x_3^+, x_3^-, _4, x_5 \geq 0 \end{aligned}$$

Given a linear program, the following situations may arise:

1) There is no admissible solution

2) There can be no solution because the value of the objective decreases indefinitely towards $-\infty$ for feasible vectors

3) There can be a unique optimal solution (desirable situation)

4) The problem can have multiple (i.e. infinitely many) optimal solutions. In fact for 2 optimal solutions x_1, x_2, any convex combination $tx_1 + (1-t)x_2$, $t \in [0,1]$ is again an optimal solution.

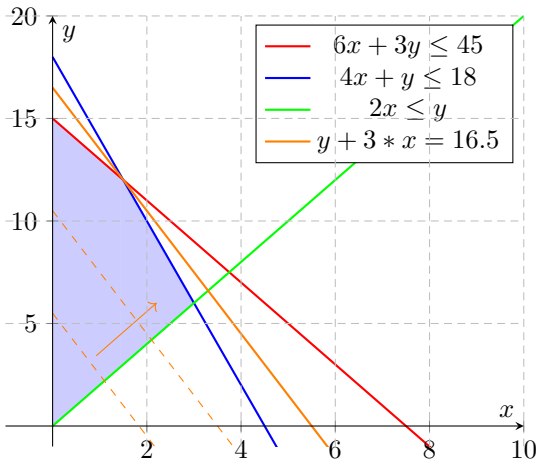

Figure 4: Graphical solution of the linear program (3).

Graphical solution

As already mentioned above, each inequality in a linear program defines a half-space. The feasible region being defined by the intersection of all those halfspaces $\cap_i \left\{\boldsymbol{x} | \boldsymbol{a}_i^T \boldsymbol{x} \leq b_i\right\}$. Such an intersection defines the feasible set of all admissible solutions (a polytope in this case).

The cost function $\boldsymbol{c}^T \boldsymbol{x}$ being also linear, it has level curves that are straight lines of equation $\boldsymbol{c}^T \boldsymbol{x} = \gamma$ for some constant γ. When varying γ, we obtain parallel lines (shown in orange in Fig. 4).

Solving a linear program is therefore equivalent to studying how large (resp. small) one can take the constant γ while maintaining a non empty intersection between the line $\boldsymbol{c}^T \boldsymbol{x} = \gamma$ and the feasible set. In the example of problem (3) and Fig. 4, this is equivalent to increasing the value of γ all the way to 16.5 and hence moving the line $y + 3x = \gamma$ along the positive y axis until there is no intersection between this line and the polytope anymore (this last point corresponding in this case to the intersection between the blue and red constraints)

$$\begin{aligned} \max \quad & x_2 + 3x_1 \\ \text{s.t.} \quad & 6x_1 + 3x_2 \leq 45 \\ & 4x_1 + x_2 \leq 18 \\ & 2x_1 \leq x_2 \end{aligned} \tag{3}$$

A first naive algorithm

With Theorem 1, we saw that provided that the linear program had at least one extreme point, one of the extreme points had to be optimal. From this, it thus seems natural to design a first naive algorithm that would simply generate all the vertices and then compare the objective values of those vertices. However, although it was easily done in 2D, by looking at the feasible set, we still haven't discussed how to generate the vertices in arbitrary LPs

To extend our naive vertex search from 2D to an arbitrary number of dimensions, we need the notion of Basic solution which we define next

Definition 5. *Given the polytope* $P = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$*, we call a vector* $\boldsymbol{x}$ *a basic solution if the subspace spanned by the constraints that are tight at* $\boldsymbol{x}$ *is of dimension* n*. I.e. span* $\left\{\boldsymbol{a}_i \mid \boldsymbol{a}_i^T \boldsymbol{x} = \boldsymbol{b}_i\right\} = \mathbb{R}^n$

Consider the standard form of an LP, $\{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq 0\}$. Note that we can always write the equality constraints as $\{\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\} \cap \{-\boldsymbol{A}\boldsymbol{x} \geq -\boldsymbol{b}\} \cap \{\boldsymbol{x} \geq 0\}$. Moreover, in this case both the $\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$ and $-\boldsymbol{A}\boldsymbol{x} \geq -\boldsymbol{b}$ are tight at any feasible $\boldsymbol{x}$. As a result, the only freedom we have (if we look for an optimal solution) is regarding our choice of the non negativity constraints (that is to say, which of those do we want to be tight/binding). In the sense of Definition 5, if we assume that the rows of $\boldsymbol{A}$ are linearly independent, basic solutions can thus be generated by combining the rows of $\boldsymbol{A}$ with $n-m$ canonical vectors $\boldsymbol{e}_i$ such that the resulting subspace, $\text{span}\,\{\boldsymbol{a}_i\}_{i=1}^N \cup \text{span}\ \{\boldsymbol{e}_j\}_{j \in I} = \mathbb{R}^n$, and then solve the system resulting from the combination $\tilde{\boldsymbol{A}}\boldsymbol{x} = \boldsymbol{b}$ where $\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{E} \end{bmatrix}$ and $\boldsymbol{E}$ is given by the set of canonical vectors $\boldsymbol{e}_i$ such that $\tilde{\boldsymbol{A}}$ has linearly independent rows. Now the row rank (number of linearly independent rows) and the column rank (number of linearly independent columns) are equivalent. Basic solutions can then be obtained equivalently by combining $\boldsymbol{A}$ with canonical vectors $\boldsymbol{e}_i$ such that the columns of the resulting matrix are independent (see below)

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \\ 1 & 0 & \dots & 0 \\ \vdots & & & \\ 0 & 1 & \dots & 0 \end{bmatrix}$$

Now since the columns corresponding to the non zero entries in the canonical vectors are independent by construction, we are left with requiring that the remaining m columns of $\boldsymbol{A}$ are linearly independent.

Basic solutions can then be obtained by finding m linearly independent columns of $\boldsymbol{A}$ (let's say corresponding to the variables $\boldsymbol{x}_i, i \in B$), solving the system $\boldsymbol{A}_B \boldsymbol{x}_B = \boldsymbol{b}$ and setting the remaining $n - m$ variables (in $[n] \setminus B$) to zero (corresponding to the canonical constraints $\boldsymbol{e}_i^T \boldsymbol{x} = 0$ associated to the last n-m rows of $\tilde{\boldsymbol{A}}$)

Keep in mind that basic solutions are <u>not</u> required to be feasible. Those are sim-

ply required to "activate" some of the constraints. This subtlety is highlighted by Example 2 below.

Example 2. *Consider the following feasible set shown in Fig. 5.*

$$\begin{aligned} x_2 &\leq 10 - x_1 \\ x_2 &\geq 1 - x_1 \\ x_2 &\geq x_1 - 1 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned} \tag{4}$$

The standard form of the problem is given by

$$\begin{aligned} x_2 + x_1 + x_3 &= 10 \\ x_2 + x_1 - x_4 &= 1 \\ x_2 - x_1 - x_5 &= 1 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 \end{aligned}$$

The rows of $\boldsymbol{A}$ *are clearly independent in this case. If we set* $\boldsymbol{x}_2$ *and* $\boldsymbol{x}_3$ *to zero, we get the system*

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ -1 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 10 \\ 1 \\ 1 \end{pmatrix}$$

The columns of the matrix corresponding to this choice are again clearly linearly independent. The associated basic solution is given by $\boldsymbol{x}_1 = 10$, $\boldsymbol{x}_2 = 0$ *which is* <u>not</u> *a feasible solution. Now if instead we had retained the columns of* $\boldsymbol{A}$ *corresponding to* x_2, x_4 *and* x_5*, and had set* x_1 *and* x_3 *to zero, the system would have turned into*

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_2 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 10 \\ 1 \\ 1 \end{pmatrix}$$

which now gives $x_2 = 10$, $x_1 = 0$*. This time we get a basic solution that belongs to the polytope.*

We call basic solutions that belong to the polytope Basic Feasible Solutions (BFS).

We can now summarize our naive algorithm (see Algorithm 1 below).

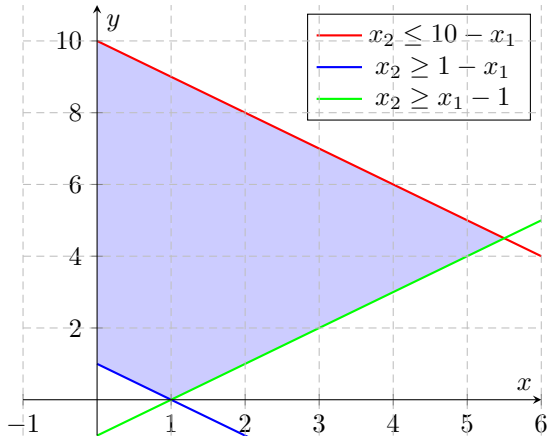

Figure 5: Feasible set for the program (4).

Algorithm 1: Naive Algorithm

1. Until there is no new vertex:
   a. Pick any subset of m linearly independent columns of $\boldsymbol{A}$
   b. Label the indices of those columns as $\{i_1, i_2, \ldots, i_m\} = B$ and solve $\boldsymbol{A}_B \boldsymbol{x}_B = \boldsymbol{b}$ where $\boldsymbol{x}_B$ encodes the variables associated to the indices in B.
   c. Set the variables corresponding to the remaining indices in $N = [n] \backslash B$ to zero
2. Select the vertex with the largest (or smallest) value $\boldsymbol{c}^T \boldsymbol{x}$

The naive algorithm introduced above will work but it will clearly be slow as it needs to visit every vertex of the feasible set. The situation is in fact worst than that. Since the number of vertices is in general combinatorial in the number of variables and constraints, the runtime of our naive algorithm will be exponential in the worst case.

How could we improve this runtime then?

Although it might not always be possible to improve the exponential runtime in the worst case if we restrict ourselves to generating and comparing basic feasible solutions (we will see later that this is in fact possible provided that we turn to a more advanced method known as the Ellipsoid method), for some instances it might be possible to visit the vertices in a more clever manner. In the next section we introduce a more efficient procedure due to Dantzig (1963) which explores the candidate solutions by jumping from vertex to vertex while making sure that the objective decreases at each jump.

The Simplex Method

The Simplex algorithm starts at a particular extremal feasible vector $\boldsymbol{x}$ which we can write as $(\boldsymbol{x}_B, \mathbf{0})$, $\boldsymbol{x}_B \in \mathbb{R}^m$, $\mathbf{0} \in R^{n-m}$, $\boldsymbol{x}_B \geq 0$. From here on, I will use the notation $\boldsymbol{x}_B$ to denote the basic variables (i.e. the ones that are non zero) and correspondingly $\boldsymbol{x}_N$ to denote the non basic variables.

After defining an initial candidate solution, the algorithm iteratively sets one of the components of $\boldsymbol{x}_B$ to zero while moving one of the variables from the set of zero variables to become positive. This step corresponds to moving from one extremal point to another adjacent extremal point of the feasible set. The main idea of the Simplex is to choose the next vertex so as to lower the cost as much as possible. we also need to decide when to stop the exploration of the vertices.

Let $\boldsymbol{A} = [\boldsymbol{A}_B, \boldsymbol{A}_N]$ where N gathers the columns associated to the indices of the variables in $\boldsymbol{x}_N$. Under this decomposition, we can write our system as

$$\boldsymbol{A}_B \boldsymbol{x}_B = [\boldsymbol{A}_B, \boldsymbol{A}_N] \begin{bmatrix} \boldsymbol{x}_B \\ \mathbf{0} \end{bmatrix} = \boldsymbol{b}$$

Similarly the cost associated to the decomposition can be written as

$$\boldsymbol{c}^T \boldsymbol{x} = [\boldsymbol{c}_B, \boldsymbol{c}_N]^T \boldsymbol{x} = \boldsymbol{c}_B^T \boldsymbol{x}_B = \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1} \boldsymbol{b}$$

The basic idea of the simplex algorithm consists in selecting the entering variables from $\boldsymbol{x}_N$ so that the cost associated to the new feasible solution $\overline{\boldsymbol{x}} = (\overline{\boldsymbol{x}}_B, \overline{\boldsymbol{x}}_N)$ is lower than the cost associated to the previous candidate solution. Note that we have

$$\boldsymbol{A}\overline{\boldsymbol{x}} = [\boldsymbol{A}_B,\ \boldsymbol{A}_N] \begin{bmatrix} \overline{\boldsymbol{x}}_B \\ \overline{\boldsymbol{x}}_N \end{bmatrix} = \boldsymbol{A}_B \overline{\boldsymbol{x}}_B + \boldsymbol{A}_N \overline{\boldsymbol{x}}_N = \boldsymbol{b} = \boldsymbol{A}_B \boldsymbol{x}_B$$

hence

$$\begin{aligned} \overline{\boldsymbol{x}}_B &= \boldsymbol{A}_B^{-1} \left(\boldsymbol{A}_B \boldsymbol{x}_B - \boldsymbol{A}_N \overline{\boldsymbol{x}}_N \right) \\ &= \boldsymbol{x}_B - \boldsymbol{A}_B^{-1} \boldsymbol{A}_N \overline{\boldsymbol{x}}_N \end{aligned}$$

and the cost of this new solution is then given by

$$\begin{aligned} & \left[\boldsymbol{c}_B^T, \boldsymbol{c}_N^T \right] \begin{bmatrix} \boldsymbol{x}_B - \boldsymbol{A}_B^{-1} \boldsymbol{A}_N \overline{\boldsymbol{x}}_N \\ \overline{\boldsymbol{x}}_N \end{bmatrix} \\ &= \boldsymbol{c}_B^T \boldsymbol{x}_B + \boldsymbol{c}_N^T \overline{\boldsymbol{x}}_N - \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1} \boldsymbol{A}_N \overline{\boldsymbol{x}}_N \\ &= \underbrace{\boldsymbol{c}_B^T \boldsymbol{x}_B}_{\text{previous cost}} + \left(\boldsymbol{c}_N^T - \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1} \boldsymbol{A}_N \right) \overline{\boldsymbol{x}}_N \end{aligned}$$

As a result, if $\left(\boldsymbol{c}_N^T - \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1} \boldsymbol{A}_N \right) \overline{\boldsymbol{x}}_N$ is negative, the update will have lowered the cost.

We call $\boldsymbol{r} = (\boldsymbol{c}_N^T - \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1} \boldsymbol{A}_N)$ the reduced cost vector. If none of the entries in the reduced cost vector are negative (resp. positive), then there is no possibility to further lower (resp. increase) the value of the objective.

As long as the reduced cost vector has some negative components, there is however a chance that we can lower the cost provided that there is a corresponding feasible $\overline{\boldsymbol{x}}$. Instead of looking for complex updates, the Simplex method considers updates of the form $\overline{\boldsymbol{x}}_N = t\boldsymbol{v}$ where $\boldsymbol{v}$ is a canonical vector having all but one non zero components (in other words, we only change one component at a time, the so-called "entering variable"). Since our objective is to lower the cost as much as possible, it makes sense to start by taking the basis associated to the most negative component in $\boldsymbol{r}$ (This is known as Dantzig's pivot rule)

Following this idea, we will choose t so that $\boldsymbol{r}^T\overline{\boldsymbol{x}}_N = \boldsymbol{r}^T t\boldsymbol{v}$ is as negative as possible while maintaining $\overline{\boldsymbol{x}}$ in the feasible set. To ensure feasibility of $\overline{\boldsymbol{x}}$, recall that we must have

$$\begin{aligned}\boldsymbol{A}\overline{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\\ \boldsymbol{A}_B\overline{\boldsymbol{x}}_B + \boldsymbol{A}_N\overline{\boldsymbol{x}}_N &= \boldsymbol{A}_B\boldsymbol{x}_B = \boldsymbol{b}\end{aligned}$$

as well as

$$\begin{aligned}\overline{\boldsymbol{x}}_B &= \boldsymbol{x}_B - \boldsymbol{A}_B^{-1}\boldsymbol{A}_N\overline{\boldsymbol{x}}_N\\ &= \boldsymbol{A}_B^{-1}\boldsymbol{b} - t\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{v} \geq 0\end{aligned}$$

Since all the other variables in $\overline{\boldsymbol{x}}_N$ will be zero, we are left with checking that the variables in $\overline{\boldsymbol{x}}_B$ also remain non negative. The case $t = 0$ corresponds to the original extreme feasible point.

We can therefore gradually increase the value of t until the first entry in $\overline{\boldsymbol{x}}_B$ becomes negative. If increasing the value of t immediately leads to the introduction of a negative variable in $\boldsymbol{x}_N$, the update can be done via one of the following alternatives:

- We either keep our original entering variable and we set the first variable that becomes negative as our leaving variable (i.e. we set it to 0). In this case the cost will not change.
- Or we switch to a new entering variable corresponding to the second largest component in the reduced cost vector.

If on the contrary, there is a threshold $\bar{t}$ before which none of the variables vanish, we set t to $\bar{t}$, select as our entering variable the variable corresponding to the non zero entry in $\boldsymbol{v}$ and as our leaving variable the first variable that vanishes for $t = \bar{t}$. If no matter how large t becomes, none of the variables are becoming negative, the problem does not admit a solution (i.e the problem is unbounded). Similarly if $r_j \geq 0$ for all j, the current vertex $\boldsymbol{x}$ is optimal. Instead of screening all the candidate entering variables with the risk of not being able to grow the value of t at all, since $\overline{\boldsymbol{x}}_B$ can read as

$$\overline{\boldsymbol{x}}_B = \boldsymbol{A}_B^{-1}\boldsymbol{b} - t\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{v} \tag{5}$$

We study the ratios

$$\frac{\boldsymbol{A}_B^{-1}\boldsymbol{b}}{\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{e}_j} \tag{6}$$

(including 0 numerators with positive denominators) and select as the leaving variables the variables corresponding to the smallest ratio. Once the leaving variable has been selected, we can thus set the value of this variable to

$$0 = \left(\boldsymbol{A}_B^{-1}\boldsymbol{b} - \bar{t}\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{v}\right)_j$$

provided that j corresponds to the smallest ratio. The value of the entering variable is set to $\bar{t}$.

Before going further, we illustrate the SIMPLEX method on a few simple examples

Example 3. *We consider the problem*

$$\begin{aligned} \min \quad & 2x_1 + x_2 + 7x_3 + x_4 \\ s.t. \quad & x_1 + x_3 + 2x_4 = 4 \\ & x_2 + x_3 - x_4 = 3 \\ & x_i \geq 0 \end{aligned}$$

Let us start by writing the problem in standard form. we have

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & -1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \quad \boldsymbol{c} = (2,\ 1,\ 7,\ 1)$$

1. **Initialization**. *We choose for example* $\boldsymbol{A}_B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $\boldsymbol{A}_N = \begin{pmatrix} 1 & 2 \\ 1 & -1 \end{pmatrix}$ *Hence* $\boldsymbol{c}_B = (2,1)$, $\boldsymbol{c}_N = (7,1)$. *From this, inverting the system* $\boldsymbol{A}_B\boldsymbol{x} = \boldsymbol{b}$ *we obtain* $\boldsymbol{x}_B = (4,3)$. *Checking the cost vector, we have* $\boldsymbol{c}^T\boldsymbol{x} = 11$. *We now compute the vector of reduced cost*

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{c}_N^T - \boldsymbol{c}_B^T\boldsymbol{A}_B^{-1}bsA_N \\ &= (7,1) - (2,1)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}^{-1}\begin{pmatrix} 1 & 2 \\ 1 & -1 \end{pmatrix} = (4,2) \end{aligned}$$

Since we have at least one negative entry in $\boldsymbol{r}$ *(corresponding to* x_4 *in this case), we can proceed with the* SIMPLEX *iterations.*

2. ***Entering variables***. *In this case, the most (and actually the only) negative entry in* $\boldsymbol{r}$ *corresponds to the variable* x_4. *So we choose this variable as our entering variable. Choosing* x_4 *as our entering variable, we have*

$$\begin{aligned} \boldsymbol{A}_B^{-1}\boldsymbol{b} - t\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{v} &= \begin{pmatrix} 4 \\ 3 \end{pmatrix} - t\begin{pmatrix} 1 & 2 \\ 1 & -1 \end{pmatrix}\boldsymbol{v}, \quad \text{taking } \boldsymbol{v} = \boldsymbol{e}_4 \\ &= \begin{pmatrix} 4 \\ 3 \end{pmatrix} - t\begin{pmatrix} 2 \\ -1 \end{pmatrix} \\ &= \begin{pmatrix} 4-2t \\ 3+t \end{pmatrix} \end{aligned}$$

From this, the first variable to exit the basis for $t > 0$ *will thus be* x_1 *and the new system following from the swap is then given by*

$$\boldsymbol{A}'_B = \underbrace{\begin{pmatrix} 0 & 2 \\ 1 & -1 \end{pmatrix}}_{\text{Basic variables } x_2, x_4}, \quad \boldsymbol{b}' = \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \quad \boldsymbol{c}'_B = (1,1), \quad \boldsymbol{c}'_N = (2,7)$$

$$\boldsymbol{A}'_N = \underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_{\text{Non-Basic variables } x_1, x_3}$$

Calculating the residual cost vector, we get

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{c}'_N - \boldsymbol{c}'_B \left(\boldsymbol{A}'_B\right)^{-1} \boldsymbol{A}'_N \\ &= \begin{pmatrix} 1 \\ 7 \end{pmatrix} - (1,1) \begin{pmatrix} 1/2 & 1 \\ 1/2 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 2 \\ 7 \end{pmatrix} - (1,1) \begin{pmatrix} 1/2 & 3/2 \\ 1/2 & 1/2 \end{pmatrix} \\ &= \begin{pmatrix} 2 \\ 7 \end{pmatrix} - \begin{pmatrix} 1 \\ 2 \end{pmatrix} > 0 \end{aligned}$$

Since all the components of $\boldsymbol{r}$ have now become non negative, we can take the vector $(0, 5, 0, 2)$ (in this case we can push x_4 up to 2). Recall that the SIMPLEX *iterates are given by $(\boldsymbol{x}_B - \boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{x}_N, \boldsymbol{x}_N)$, taking $\boldsymbol{x}_N = t\boldsymbol{v}$, we get $(\boldsymbol{x}_B - t\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{v}, t\boldsymbol{v})$. The corresponding cost is then given by*

$$\boldsymbol{c}^T\boldsymbol{x} = (2,1,7,1)^T \begin{pmatrix} 0 \\ 5 \\ 0 \\ 2 \end{pmatrix} = 7$$

So far we have covered the ideal situation in which a solution exists and the SIMPLEX algorithm is able to return it. When running the SIMPLEX though, several situations can occur:

- Unboundedness
- Multiple solutions
- Degeneracy
- Infeasibility

We discuss some of those issues in more details below.

Unboundedness

A first situation that can occur is when whatever the value of t, we always keep a feasible solution. As an example, consider the problem

$$\begin{aligned} \max \quad & x_1 \\ \text{s.t.} \quad & x_1 - x_2 \leq 1 \\ & -x_1 - x_2 \leq 2 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{7}$$

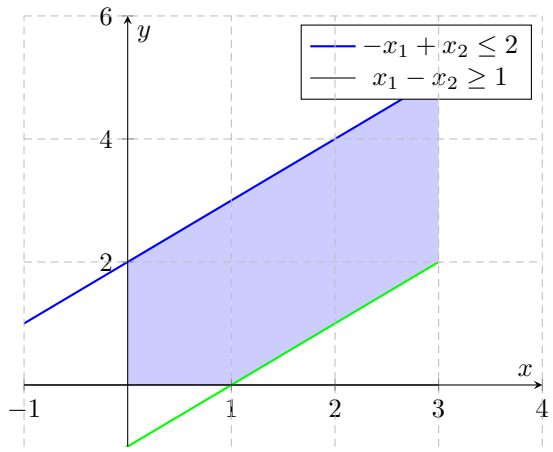

Figure 6: Feasible set for the program (7).

whose feasible set is illustrated in Fig. 6 below.

As usual, we first turn the problem into standard form by introducing the auxilliary slack variables x_3, x_4 as

$$\begin{aligned} x_1 - x_2 + x_3 &= 1 \\ -x_1 + x_2 + x_4 &= 2 \end{aligned}$$

and turning the max into a min. From this, we can define

$$\boldsymbol{A} = \begin{pmatrix} 1 & -1 & 1 & 0 \\ -1 & 1 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{A}_B = \begin{matrix} x_3 & x_4 \\ \end{matrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad , \quad \boldsymbol{A}_N = \begin{matrix} x_1 & x_2 \\ \end{matrix} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

We also have $\boldsymbol{c} = (1, 0, 0, 0)$, $\boldsymbol{c}_B = (0, 0)$, $\boldsymbol{c}_N = (-1, 0)$. We compute the vector of reduced costs as

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{c}_N - \boldsymbol{c}_B \boldsymbol{A}_B^{-1} \boldsymbol{A}_N \\ &= (-1, 0) - (0, 0) \end{aligned}$$

We can thus take $\boldsymbol{x}_1$ as our entering variable. From this, setting $\boldsymbol{v} = \boldsymbol{e}_1$, we get

$$\begin{aligned} \boldsymbol{x}'_B &= \boldsymbol{A}_B^{-1}\boldsymbol{b} - t\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{e}_1 \\ &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} - t \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \boldsymbol{e}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix} - \begin{pmatrix} t \\ -t \end{pmatrix} \end{aligned}$$

The first variable to leave will therefore be x_3 for $t = 1$. Our next iteration will be

$$\boldsymbol{A}_B = \begin{matrix} x_1 & x_4 \\ \end{matrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \quad , \quad \boldsymbol{A}_N = \begin{matrix} x_2 & x_3 \\ \end{matrix} \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \quad , \quad \boldsymbol{b} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\boldsymbol{c}_B = (1,0)\,, \quad \boldsymbol{c}_N = (0,0)$$

The corresponding reduced cost vector is then given by

$$\boldsymbol{r} = \boldsymbol{c}_N - \boldsymbol{c}_B \boldsymbol{A}_B^{-1} \boldsymbol{A}_N$$

Noting that $\boldsymbol{A}_B^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ we can express this vector as

$$\begin{aligned} \boldsymbol{r} &= (0,0) - (-1,0) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \\ &= (0,0) - (-1,0) \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix} \\ &= (0,0) - \begin{pmatrix} 1 \\ -1 \end{pmatrix} = (-1,1) \end{aligned}$$

It thus seems natural to choose x_2 as our entering variable. Now let us look at the result of such a choice. We have

$$\begin{aligned} \boldsymbol{x}'_B &= \boldsymbol{A}_B^{-1}\boldsymbol{b} - t\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{e}_3 \\ &= \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} - t \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 3 \end{pmatrix} - t \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 3 \end{pmatrix} - t \begin{pmatrix} -1 \\ 0 \end{pmatrix} \end{aligned}$$

We see that we can take t arbitrarily large without violating any of the positivity constraints.

Degeneracy

Another situation that might occur with the SIMPLEX is degeneracy. As an illustration of this situation, consider the following example:

Example 4.

$$\begin{array}{lll} \min & x_1 + 2x_2 + x_3 & \\ s.t. & x_1 + x_2 + 2x_3 & = 3 \\ & -x_1 + x_2 + 3x_3 & = 2 \\ & 3x_2 + 6x_3 & = 4 \\ & 2x_3 + x_4 & = 1 \\ & x_1, x_2, x_3, x_4 & \geq 0 \end{array}$$

In this case, we have

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 1 \\ 1 & -1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \quad \boldsymbol{c} = (3,1,8,1)$$

Let us initialize the SIMPLEX *with* $\boldsymbol{A}_B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ *(which again simplifies the calculation of the inverse* $\boldsymbol{A}_B^{-1}$*). From this, we thus have*

$$\boldsymbol{A}_B = \overset{x_1 \quad x_2}{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}} \quad , \quad \overset{x_3 \quad x_4}{\begin{pmatrix} 2 & 1 \\ 1 & -1 \end{pmatrix}}$$

together with $\boldsymbol{c}_B = (3,1)$ *and* $\boldsymbol{c}_N = (8,1)$*. We start by computing the reduced cost vector*

$$\begin{aligned} \boldsymbol{r} = \boldsymbol{c}_N - \boldsymbol{c}_B \boldsymbol{A}_B^{-1} \boldsymbol{A}_N &= (8,1) - (3,1) \begin{pmatrix} 2 & 1 \\ 1 & -1 \end{pmatrix} \\ &= (8,1) - (7,2) \\ &= (1,-1) \end{aligned}$$

From this reduced cost vector, we see that since at least one of the residual costs is negative, there is a chance we might be able to further reduce the cost. Choosing x_4 *as our next entering variable, we get*

$$\begin{aligned} \boldsymbol{A}_B^{-1}\boldsymbol{b} - t\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{v} &= \boldsymbol{b} - t\boldsymbol{A}_N\boldsymbol{v} && (8) \\ &= \begin{pmatrix} 0 \\ 2 \end{pmatrix} - t \begin{pmatrix} 2 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} && (9) \\ &= \begin{pmatrix} 0 \\ 2 \end{pmatrix} - t \begin{pmatrix} 1 \\ -1 \end{pmatrix} && (10) \end{aligned}$$

The leaving variable in this case will be x_1*. For such a choice, a rapid analysis of* (10) *shows that as soon as* t *becomes positive,* x_1 *takes a negative value. This in particular means that we won't be able to make any progress with respect to the value of the objective. Nonetheless, if the stopping of our algorithm is based on the absence of negative residual costs, it will proceed and remove* x_1 *from the basis (although it does not improve the objective). From this, we get the decomposition*

$$\boldsymbol{A}_B = \overset{x_2 \quad x_4}{\begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix}} \quad , \quad \overset{x_1 \quad x_3}{\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}}$$

together with $\boldsymbol{c} = (3,1,8,1)$ *and* $\boldsymbol{c}_B = (1,1)$*,* $\boldsymbol{c}_N = (3,8)$*. we get the new reduced cost vector as*

$$\begin{aligned} \boldsymbol{A}_B^{-1} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \quad \boldsymbol{r} &= \boldsymbol{c}_N - \boldsymbol{c}_B \boldsymbol{A}_B^{-1} \boldsymbol{A}_N \\ &= (3,8) - (1,1)\,(1,1) \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \\ &= (3,8) - (1,1) \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} = (3,8) - (2,4) \geq 0 \end{aligned}$$

The SIMPLEX *algorithm is guaranteed to terminate in a finite number of iterations provided that there are no degenerate solutions. In the presence of degenerate solutions, the algorithm has to be modified to guarantee its termination in a finite number of iterations.*

Simplex tableaux

A classical and efficient way to implement the SIMPLEX method is by means of simplex tableaux. Consider the problem

$$\begin{aligned} \min \quad & 4x_1 + 6x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \leq 11 \\ & x_1 + x_2 \leq 27 \\ & 2x_1 + 5x_2 \leq 90 \end{aligned}$$

introducing slack variables s_1, s_2, s_3, we can write the problem in standard form as

$$\begin{array}{lrrrrrl} \min & -4x_1 & -6x_2 & & & & \\ \text{s.t.} & -x_1 & +x_2 & +s_1 & & & = 11 \\ & x_1 & +x_2 & & +s_2 & & = 27 \\ & x_1 & +x_2 & & & +s_3 & = 90 \end{array}$$

The corresponding SIMPLEX tableau then reads as

x_1	x_2	s_1	s_2	s_3	b
-1	1	1	0	0	11
1	1	0	1	0	27
1	1	0	0	1	90
-4	-6	0	0	0	0

In this case x_1 and x_2 are the non basic variables. As we start by choosing x_1 and x_2 as our non basic variables, the values of those variables are set to 0 and the values of the slack variables are thus given as $s_1 = 11$, $s_2 = 27$ and $s_3 = 90$. As a result we have $x_1 = 0, x_2 = 0, s_1 = 11, s_2 = 27, s_3 = 90$. The entry in the lower right corner of the tableau corresponds to the value of the obejctive.

To perform optimality check for a solution represented by a simplex tableau, it suffices to look at the entries in the bottom row. If any of those entries are negative, the solution is not optimal.

To improve the solution, we choose x_2 as our entering variable as this is the variable that has the most negative entry in the last row

x_1	x_2	s_1	s_2	s_3	b
-1	1	1	0	0	11
1	1	0	1	0	27
1	1	0	0	1	90
-4	(-6)	0	0	0	0

To determine the index of the leaving variable, we locate the entries that are positive in the column of the entering variable and we form the ratios given by the entries of

the last column and the entries of the column corresponding to the entering variable. I.e

x_1	x_2	s_1	s_2	s_3	b
-1	1	1	0	0	11
1	1	0	1	0	27
1	1	0	0	1	90
-4	-6	0	0	0	0

In this case the ratios are given by $11/1, 27/1$ and $90/5 = 18$. The basic variables are given by the slack variables s_1, s_2 and s_3 and the matrix $\boldsymbol{A}_B = \boldsymbol{I}$ so that $\boldsymbol{A}_B^{-1} = \boldsymbol{I}$. As pointed out in (5), (6), the first variable $\boldsymbol{x}_{B_i}$ to leave the basis corresponds to the first entry that becomes negative in the vector

$$\boldsymbol{A}_B^{-1}\boldsymbol{b} - t\boldsymbol{A}_B^{-1}\boldsymbol{A}_N\boldsymbol{e}_i$$

when increasing the value of t. Concretely this variable is thus given by the smallest of the ratios (6) (In the case of a simplex tableau, since the operations on the tableau always maintain $\boldsymbol{A}_B = \boldsymbol{I}$ this corresponds to taking the variable corresponding to the smallest entry in the vector given by the entrywise ratio of the last column ($\boldsymbol{b}$) to the column of the entering variable $\boldsymbol{A}_N\boldsymbol{e}_i$). Applying this idea to the above tableau, we choose s_1 as our departing variable. To apply the change, we simply implement one step of Gauss-Jordan elimination to turn the column of the entering variable (x_2 in our example) into the canonical vector. Note that in order to keep the canonical columns for s_2 and s_3, we are restricted to operations involving the row (the first one in this case) that has zero entries in the columns associated to those variables. In the illustration that we make of this step, we highlight the column of the leaving (resp. entering) variable in red (resp. green). The Gauss-Jordan operations on the rows are indicated on the side of the tableau.

x_1	x_2	s_1	s_2	s_3	b		x_1	x_2	s_1	s_2	s_3	b	
-1	1	1	0	0	11		-1	1	1	1	0	11	
1	1	0	1	0	27		2	0	-1	1	0	16	R2 - R1
1	1	0	0	1	90	$\Rightarrow$	7	0	-5	0	1	35	R3 - 5R1
-4	-6	0	0	0	0		-10	0	6	0	0	66	R4 + 6R1

Since the last row (i.e. corresponding to the vector of reduced costs) still contains a negative entry, the iterations will continue with x_1 as the next leaving variable.

Initialization and the last row

The last row of the simplex should always be initialized with an empty space for the cost value (using a zero can be misleading as we explain next). The absence of a value simply indicates that we haven't computed the value of the cost yet.

Example 5. *Consider the following problem*

$$\begin{array}{lccccc} \min & x_1 & & +2x_3 & & \\ s.t. & x_1 & +x_2 & +x_3 & & = 2 \\ & 2x_1 & +3x_2 & & +x_4 & = 1 \end{array} \tag{11}$$

The original tableau for this problem is given by

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & b \\ 1 & 1 & 1 & 0 & 2 \\ 2 & 3 & 0 & 1 & 1 \\ \hline 1 & 0 & 2 & 0 & \end{array} \tag{12}$$

In this original tableau, the last row indicates the original cost function and the empty space that appears on the right (this space is sometimes filled with a zero) simply means that we haven't added anything yet to this cost function to write it as a constant.

At each iteration of the simplex, the tableau highligths a subset of the variables that take non zero values (the basis) and another subset of the variables that are set to zero. At each of those iterations, the operations we apply to the last row are equivalent to removing the basic variables from the expression of the objective (or substituting the expression of those variables as functions of the non basic variables obtained from the equality constraints) in order to reduce the objective to a simple constant + terms that are multiplying vanishing variables. Consider problem (11). In the setting of this problem, in order to "get rid" of the "2" that appears under the variable x_3 in the objective, we use the row associated to this variable (i.e the pivot row) and combine the last row with -2 times this first row which gives

$$\begin{array}{ll} & x_1 + 2x_3 \\ -2 & (x_1 + x_2 + x_3) \\ = & -x_1 - 2x_1 - 2x_2 \\ = & -x_1 - 2x_2 \end{array} \tag{13}$$

The corresponding tableau is given by

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & b \\ 1 & 1 & 1 & 0 & 2 \\ 2 & 3 & 0 & 1 & 1 \\ \hline -1 & -2 & 0 & 0 & -4 \end{array} \tag{14}$$

When applying this operation to the tableau, we in fact combine a corresponding linear combination of the equality constraints to the objective function. I.e. the mathematical operation corresponding to the execution of the simplex update is actually given by

$$\begin{array}{lll} & x_1 + 2x_3 & \text{the original objective} \\ -2 & (x_1 + x_2 + x_3 - 2) & \text{the constraint } x_1 + x_2 + x_3 - 2 = 0 \\ = & -x_1 - 2x_2 + 4 & \end{array} \tag{15}$$

Moreover, since we have eliminated all the basic variables from the objective and since all the non basic variables are zero, the constant term that appears in this objective (the "4" in this case) is the value of this objective at the current vertex/simplex iterate.

Adding a linear combination of the (equality) constraints to the original objective obviously does not change the value of this objective (provided that we stay inside the feasible set) and we can thus always perform such an operation. However, when combining the rows in the tableau, we leave aside the constants in the equations (i.e. the right-hand sides) (compare (13) to (15)) and we must therefore keep track of those right-hand sides. By performing the same operations on the right-hand sides as those that we apply on the rows, i.e.

$$\begin{array}{rll} & x_1 + 2x_3 & \\ & -2(x_1 + x_2 + x_3) & = -2(2) \\ = & -x_1 - 2x_2 & = -4 \end{array} \tag{16}$$

one can see that the value we get is not the value of the objective but the opposite of this value (we simply move the constant on the other side of the equality). From now on, this value should therefore be kept in the tableau as we have modified the objective by adding to it a non zero expression.

Artificial variables

In order to start the simplex, we need to find an initial basic feasible solution. Sometimes this is straightforward, for example when we have constraints of the form $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ with $\boldsymbol{b} \geq 0$. In this case we can introduce the usual slack variables and rewrite the constraints as $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{b}$ and because $\boldsymbol{b} \geq 0$, the vector defined by $\boldsymbol{x} = 0$, $\boldsymbol{s} = \boldsymbol{b}$ is a basic feasible solution by definition. In general though, finding a basic feasible solution might not always be that easy. What we can always do is multiply the RHS by ± 1 in order to obtain non negative b_i. We can then introduce a vector $\boldsymbol{y} \in \mathbb{R}^m$ of "artificial" variables and consider the problem

$$\begin{aligned} \min \quad & y_1 + y_2 + \ldots + y_m && (17) \\ s.t. & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{b} && (18) \\ & \boldsymbol{x} \geq 0 && (19) \\ & \boldsymbol{y} \geq 0 && (20) \end{aligned}$$

For this auxiliary problem, initialization is easy since we can just take $\boldsymbol{x} = 0$ and $\boldsymbol{y} = \boldsymbol{b}$. Note that for any feasible $\boldsymbol{x}$, the choice $\boldsymbol{y} = 0$ gives a feasible solution to the auxiliary problem. Moreover, since all the variables in $\boldsymbol{y}$ are non negative, and since the objective is given by $y_1 + y_2 + \ldots + y_m$, if the objective of the auxilliary problem is different from 0, this means that this auxilliary problem does not have a solution of the form $[\boldsymbol{x}, \boldsymbol{0}]$ and hence that the original problem does not have a feasible solution.

We can thus apply the simplex to the auxiliary problem and this will tell us if the original problem is feasible or not. The auxiliary problem however only provides a partial answer to our original question as the final tableau will not necessarily terminate with a basic feasible solution of the original problem.

If the Simplex on the auxilliary problem terminates with a basis corresponding to the original variables, we can just drop the columns corresponding to the auxiliary variables. If it is not the case (meaning we have a degenerate basic feasible solution), and some of the auxiliary variables are in the basis (although with zero value), since the auxilliary variables must be zero at the optimum, let $A_{B(1)}, \ldots, A_{B(k)}$ denote the columns of $\boldsymbol{A}$ that belongs to the basis and such that $x_{B(1)}, \ldots x_{B(k)}$ are non zero. Since at least one of the artificial variables with value 0 is in the basis, there must be a total of $k < m$ non zero basic variables. If we assume that the matrix $\boldsymbol{A}$ has rank m, one can choose an additional $m - k$ columns from $\boldsymbol{A}$ to obtain a set of linearly independent columns. In other words, we replace the columns associated to the artificial variables by columns associated to the original variables $\boldsymbol{x}_i$. The procedure is known as "driving the artificial variables out of the basis".

To find a non artificial variable to add to the basis, if the artificial variable is the ℓ^{th} variable in the basis (i.e. whose column corresponds to the ℓ^{th} canonical basis vector), then we choose a column $\boldsymbol{B}^{-1}\boldsymbol{A}_j$ in the tableau corresponding to a non basic variable whose ℓ^{th} entry is non zero. Since the ℓ^{th} basic variable is zero (i.e. the RHS entry corresponding to the ℓ^{th} row is zero), using this row to bring the new variable into the basis (and remove the zero artificial variable) will not affect the values of the other basic variables. Neither will it affect the value of the objective. Since the objective does not change and since an objective of value 0 is known to be optimal, the change of basis will only replace a non negative reduced cost vector by another non negative reduced cost vector. In other words, swapping columns (whenever the entry corresponding to the original artificial variable is non zero) will not change the nature of the solution and we can thus always do this.

If instead we cannot find a column with a non zero ℓ^{th} entry among the original variables (meaning all the entries corresponding to the original variables in the row are zero) then it means that by applying operations on the rows of $\boldsymbol{A}$ we were able to reduce one of the rows of this matrix to zero. This in turns means that this row can be written as a linear combination of the other rows (it is thus linearly dependent on the other rows) and we can thus remove this row from $\boldsymbol{A}$ without modifying the problem.

This last situation is illustrated in the example below

Example 6. *We consider the following problem which is taken from [3]*

$$\begin{array}{lccccc} \min & x_1 & +x_2 & +x_3 & & \\ s.t. & x_1 & +2x_2 & +3x_3 & & = 3 \\ & -x_1 & +2x_2 & +6x_3 & & = 2 \\ & & 4x_2 & +9x_3 & & = 5 \\ & & & 3x_3 & +x_4 & = 1 \\ & x_1, & x_2, & x_3, & x_4 & \geq 0 \end{array} \tag{21}$$

To find a feasible solution, we form the following auxilliary problem

$$
\begin{array}{lcccccccccc}
\min & & & & & & x_5 & +x_6 & +x_7 & +x_8 & \\
s.t. & x_1 & +2x_2 & +3x_3 & & & +x_5 & & & & = 3 \\
& -x_1 & +2x_2 & +6x_3 & & & & +x_6 & & & = 2 \\
& & +4x_2 & +9x_3 & & & & & +x_7 & & = 5 \\
& & & +3x_3 & +x_4 & & & & & +x_8 & = 1 \\
& x_1, & x_2, & x_3, & x_4, & & x_5, & x_6, & x_7, & x_8 & \geq 0
\end{array}
\tag{22}
$$

A first basic feasible solution can now be obtained as $(x_1, x_2, \ldots, x_4) = 0$ *and taking* $(x_5, x_6, x_7, x_8) = \boldsymbol{b} = (3, 2, 5, 1)$*. After updating the reduced cost vector, we get the tableau*

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8		
1	2	3	0	1	0	0	0	3	x_5
-1	2	6	0	0	1	0	0	2	x_6
0	4	9	0	0	0	1	0	5	x_7
0	0	3	1	0	0	0	1	1	x_8
0	-8	-21	-1	0	0	0	0	-11	

(23)

Having x_8 *exit and* x_4 *enter the basis (although not optimal in terms of the reduced cost, this can rapidly be done to reduce the cost since* x_4 *already has a canonical column) gives*

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8		
1	2	3	0	1	0	0	0	3	x_5
-1	2	6	0	0	1	0	0	2	x_6
0	4	9	0	0	0	1	0	5	x_7
0	0	3	1	0	0	0	1	1	x_4
0	-8	-18	0	0	0	0	1	-10	

(24)

We now bring x_3 *into the basis and have* x_4 *exit the basis*

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8		
1	2	0	-1	1	0	0	-1	2	x_5
-1	2	0	-2	0	1	0	-2	0	x_6
0	4	0	-3	0	0	1	-3	2	x_7
0	0	1	1/3	0	0	0	1/3	1/3	x_3
0	-8	0	6	0	0	0	7	-4	

(25)

Next we bring x_2 *into the basis and let* x_6 *exit the next tableau is then given by*

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8		
2	0	0	1	1	-1	0	1	2	x_5
$-1/2$	1	0	-1	0	1/2	0	-1	0	x_2
2	0	0	1	0	-2	1	1	2	x_7
0	0	1	1/3	0	0	0	1/3	1/3	x_3
-4	0	0	-2	0	4	0	-1	-4	

(26)

Finally we let x_1 enter the basis and x_5 leave, which gives the tableau

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8		
1	0	0	1/2	1/2	−1/2	0	1/2	1	x_1
0	1	0	−3/4	1/4	1/4	0	−3/4	1/2	x_2
0	0	0	0	−1	−1	1	0	0	x_7
0	0	1	1/3	0	0	0	1/3	1/3	x_3
0	0	0	0	2	2	0	1	0	

(27)

From this tableau, one can see that the objective drops to 0 which means we have a feasible solution to the auxiliary problem. However, the auxiliary variable x_7 is still in the basis (in this case x_7 is the third variable in the basis) and all the entries corresponding to the original variables are now zero in the third row which means this row can be written as a linear combination of the other rows (i.e. by applying operations on the rows of $\boldsymbol{A}$ we were able to reduce this row to zero). We can thus delete this row from the matrix $\boldsymbol{A}$ in this case. This leaves the following reduced tableau.

x_1	x_2	x_3	x_4	
1	0	0	1/2	1
0	1	0	−3/4	1/2
0	0	1	1/3	1/3

(28)

From this, we can now compute te reduced cost and then start the Simplex.

Following [3], we summarize our conclusion from example 6 below.

Two-phase Simplex:

Phase I

1. By multiplying some of the constraints by -1, change the problem so that $\boldsymbol{b} \geq 0$
2. Introduce artificial variables $y_1, \ldots, y_m$ (if needed) and apply the SIMPLEX method to the auxiliary problem with cost $\sum_{i=1}^{m} y_i$
3. If the optimal cost in the auxiliary problem is positive, the original problem is infeasible and the algorithm terminates
4. If the optimal cost in the auxiliary problem is zero, a feasible solution to the auxiliary problem has been found

If no artificial variable is in the final basis, the artificial variables and the corresponding columns can be eliminated and a feasible basis for the original problem can be derived

5. If the ℓ^{th} basic variable is an artificial one, examine the ℓ^{th} entry of the columns of the tableau. if all the entries are zero then the ℓ^{th} constraint represent a redundant constraint and can be eliminated. Otherwise, if the ℓ^{th} entry of the j^{th} column is non zero, apply a change of basis and repeat until all artificial variables are out of the basis.

Phase II

1. Let the final basis and tableau obtained from Phase I be the initial basis and tableau obtained for Phase II
2. Compute the reduced cost of all variables for this initial basis
3. Apply the simplex method to the original problem.

The "big M" method

Instead of using the auxiliary problem as an initialization step for the simplex, one can also incorporate that step directly into the simplex iterations. The result is known as the "big M" method.

The idea of the "big M" method is to introduce an objective of the form

$$\sum_{j=1}^{n} c_j x_j + M \sum_{i=1}^{m} y_i$$

where the y_i are the artificial variables which were introduced above. For a sufficiently large value of M, in the minimization setting, we can expect the variables y_i to be driven towards zero. In particular, there is no reason to fix the value of M and we can just leave it as an arbitrary constant and then implement the simplex on that constant.

Consider the following example

Example 7. *We consider the problem*

$$\begin{array}{lccccccc} \min & x_1 & +2x_2 & +x_3 & & & & \\ s.t. & x_1 & +x_2 & +x_3 & & & = 2 & \\ & -x_1 & +x_2 & +2x_3 & & & = 3 & \\ & & 4x_2 & +6x_3 & & & = 5 & \\ & & & 2x_3 & & +x_4 & = 1 & \\ & x_1, & x_2, & x_3, & & x_4 & \geq 0 & \end{array} \tag{29}$$

Adding artificial variables and using the "big-M" method, the problem turns into

$$\begin{array}{lccccccccc} \min & x_1 & +2x_2 & +x_3 & & +Mx_5 & +Mx_6 & +Mx_7 & & \\ s.t. & x_1 & +x_2 & +x_3 & & +x_5 & & & & = 2 \\ & -x_1 & +x_2 & +2x_3 & & & +x_6 & & & = 3 \\ & & 4x_2 & +6x_3 & & & & +x_7 & & = 5 \\ & & & 2x_3 & +x_4 & & & & & = 1 \\ & x_1, & x_2, & x_3, & x_4 & x_5 & x_6 & x_7 & & \geq 0 \end{array} \tag{30}$$

In this instance, we omit the artificial variable x_8 as x_4 only appears in the last row and has a multiplicative coefficient of 1*. The initial tableau is given by*

x_1	x_2	x_3	x_4	x_5	x_6	x_7	
1	1	2	0	1	0	0	2
-1	1	2	0	0	1	0	3
4	6	0	0	0	0	1	5
0	0	2	1	0	0	0	1
1	2	3	0	M	M	M	

(31)

Using the basic operations on the rows in order to set the reduced costs associated to the basic variables to zero we get the tableau

x_1	x_2	x_3	x_4	x_5	x_6	x_7	
1	1	2	0	1	0	0	2
-1	1	2	0	0	1	0	3
4	6	0	0	0	0	1	5
0	0	2	1	0	0	0	1
$1-4M$	$2-8M$	$3-4M$	0	0	0	0	$-10M$

(32)

For a sufficiently large value of M, we see that the first 3 non zero entries of the reduced cost vector will be negative. Starting with the most negative entry (x_2 in this case) as the entering variable, we can look at the ratios $(2, 3, 5/6)$ given by the RHS over the second column. The smallest ratio in this case being given by $5/6$, we therefore take x_7 as the leaving variable.

We continue like this until we are left with a reduced cost vector that does not contain any negative entry for a sufficiently large M. In the last tableau all the artificial variable must have been driven outside the basis as any optimal solution to problem (30) for sufficiently large M must necessarily be of the form $(x_1^*, x_2^*, x_3^*, x_4^*, 0, \ldots, 0)$.

References

[1] Pablo Pedregal, *Introduction to optimization*, volume 46, Springer 2004.

[2] Jiří Matoušek, Bernd Gärtner, *Understanding and using linear programming*, volume 33, Springer 2007.

[3] Dimitris Bertsimas, John Tsitsiklis, *Introduction to linear optimization*, (Vol. 6, pp. 479-530), Belmont, MA: Athena Scientific, (1997)