Lecture 4 - Simplex algorithm: Initialization and termination

Optimization and Approximation - ENS M1

Nicolas Bousquet

During the previous lectures, we have seen the global shape of the Simplex algorithm. It can be summarized as follows:

Procedure 1 Scheme of the Simplex Algorithm

Start from a basic feasible solution
while the solution is not optimal **do**
  Apply the ratio test.
  Choose a pivot (variable entering in the basis).
  Apply the ratio test to find a variable leaving the basis
  Apply the pivoting operation
end while
Return the optimal solution and the optimal value.

First notice that, the Simplex algorithm is not "an" algorithm but a family of them. Indeed, each different rule that chooses a pivot might lead to a different algorithm. Moreover we have not how we can find a BFS and when we have to stop.

The goal of today's lecture is the following:

- Detail the stopping rule.
- Explain how we can find a BFS.
- Study the different pivoting operations and their consequences on the complexity of the Simplex algorithm.
- Discuss the complexity (number of pivots) of the Simplex algorithm.

1 Stopping rules

We have already seen a criterion to end the Simplex algorithm during the last lecture. However, there is a second way the Simplex algorithm can stop. In this Section, we will see both stopping rules.

Assume that the Linear Programming is written in the following way:

$$\begin{aligned} \max\ & c_N^T x_N + S \\ \begin{pmatrix} I & A \end{pmatrix} & \begin{pmatrix} x_B \\ x_N \end{pmatrix} = b \\ & x \geq 0 \end{aligned}$$

where $b \geq 0$, x_B is the vector of a basic feasible solution. So in particular, when $x_N = 0$, we obtain a feasible solution of value S. We can stop when:

1. If $c_N \leq 0$, then the optimal value equals S.

2. If there exists a coefficient c_i of c_N such that c_i is positive and all the coefficients of the column of x_i in $\begin{pmatrix} I & A \end{pmatrix}$ are non-positive. Then the solution is infinite

In what follows, we prove that the claim above is true.

1.1 Point 1: finite optimal value

Let $x_B = b$ and let $x_N = 0$. We have

$$\begin{pmatrix} I & A \end{pmatrix} \begin{pmatrix} x_B \\ x_N \end{pmatrix} = b.$$

Moreover the value of this solution is S. Now assume that there exists a strictly better solution $y = \begin{pmatrix} y_B \\ y_N \end{pmatrix}$. Since $y_N \geq 0$ and since $c \leq 0$, we have $c_N^T y_N \leq S$, contradicting the fact that it is a better solution.

1.2 Point 2: infinite optimal value

Assume now that there exists a coefficient of c_N which is positive, say it corresponds to the variable x_i. Assume moreover that all the coefficients of the column of A corresponding to x_i are non-positive.

Let us assume that the current basis is $x_1, \ldots, x_\ell$, the coefficients of column of x_i are a_{ji} and the RHS is b_j. The current solution is $x_j = b_j$ for $j \leq \ell$ gives a feasible solution. As well as:

$$x_j = b_j + a_{ji}M \geq 0 \text{ for } j \leq \ell$$

$$x_i = M \geq 0$$

for any value of $M \geq 0$ since a_{ji} is non-positive for every i. Since this solution has value $S + c_i M$ where S is the value of the current solution, this value tends to infinity when M goes to infinity. So we can find an optimal solution of arbitrarily large value.

So to conclude: **if the ratio test does not give any constraint, the optimal solution is infinite.**

Illustration. Let us see on some example what really happens in this case. Consider the following LP:

$$\max(2x_1 + x_2)$$

Subject to:

$$\begin{array}{rcl} x_1 - x_2 & \leq & 10 \\ 2x_1 - x_2 & \leq & 40 \\ x_1, x_2 & \geq & 0. \end{array}$$

The addition of slack variables permits to write the LP in standard form. The two slack variables are called x_3 and x_4 and the corresponding tableau is

2	1	0	0	0
1	−1	1	0	10
2	−1	0	1	40

Let us first put x_1 into the basis. One can note that x_3 leaves the basis (since $x_1 \leq 10$ is given by the first constraint). We obtain the tableau:

0	3	−2	0	−20
1	−1	1	0	10
0	1	−2	1	20

Since x_2 has a positive coefficient in the objective function, we have no reach the optimal value. This pivoting operation ensures that x_2 enters in the basis and x_4 leaves the basis.

0	0	4	-3	-80
1	0	-1	1	30
0	1	-2	1	20

Again the tableau is not optimal. Indeed the coefficient of x_3 is positive. So we want to apply a pivot operation on x_3 (the coefficient of x_4 is negative so it is not possible to pivoting on x_4). But the problem is the following: the constraints does not give any constraint on x_3. Indeed:

$$-x_3 \leq 30 \Rightarrow x_3 \geq -30$$

$$-2x_3 \leq 20 \Rightarrow x_3 \geq -40$$

So it means that, if x_3 enters in the basis, then we can arbitrarily increase x_3 without violating any constraint. Since the coefficient in front of x_3 is positive, it means that we can arbitrarily increase the value of the objective function without violating constraints. Thus the optimal value is unbounded !

More formally, what we have is that for any nonnegative δ, the point $(30+\delta, 20+2\delta, \delta, 0)$ is feasible. Since the objective function at this point has value $80+4\delta$, we see that the problem is unbounded.

Clearly, unboundedness of a problem can occur only when the feasible region is unbounded, which, unfortunately, is something we cannot tell in advance. In the above example, we detected unboundedness when we encountered a pivot column that does not contain any positive entry. More generally, we can in fact declare a problem as unbounded if any (nonbasic) column, not necessarily associated with the entering variable, is identified to have the above-stated property at the end of an iteration. Referring back to the initial tableau, we see that, indeed, the x_2-column had this property. Therefore, we could have concluded that the problem is unbounded at the outset.

Note finally that if the function is a minimization function, then the criterion are reversed.

2 Initialization - Big M and Phase I/II

2.1 Artificial variables

We have said that the simplex algorithm needs an initialization. However finding a basic feasible solution for starting the simplex algorithm is not so simple. Indeed, we can try all the possible basis and try to determine if they are basic feasible solutions or not. But this operation may be very long (it can take an exponential time). Let us now see how we can artificially create a basic feasible solution using additional variables. Note nevertheless that this operation has a cost: it creates new solutions which were not solution in the original LP. Thus we will have to be sure that we will finally find an optimal solution which is feasible in the original LP. We will see two methods which ensure that it is possible: the big M method and Phase I/ Phase II procedure.

Let us first introduce the notion of *artificial variables*. Before doing that, let us came back to the dovetail problem where slack variables have been introduced:

$$\begin{pmatrix} 3 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 18 \\ 9 \\ 7 \\ 6 \end{pmatrix}$$

For this problem, we have seen that it is simple to find an basic feasible solution for starting the simplex algorithm: we just have to consider the set $BI = \{3, 4, 5, 6\}$. Indeed the restriction of the matrix to this set of columns induces a squared matrix which is the identity matrix. Since b is nonnegative, we obtain a basic feasible solution.

Nevertheless in the general case, we are not sure that there exists such a square identity matrix (and it probably will not be the case).

When it is not the case, we can have trouble for initializing the simplex algorithm. Consider for instance the following LP:

$$\begin{aligned} &\max x_1 + x_2 \\ &\text{such that} \\ -x_1 - x_2 &\leq -5 \\ x_1 + 2x_2 &\geq 3 \\ 3x_1 + 5x_2 &\leq 20 \\ x_1, x_2 &\geq 0 \end{aligned}$$

When we add the slack and the surplus variables and we multiply by -1 in order to obtain $b \geq 0$ we have:

$$\begin{aligned} x_1 + x_2 - x_3 &= 5 \\ x_1 + 2x_2 - x_4 &= 3 \\ 3x_1 + 5x_2 + x_5 &= 20 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 \end{aligned}$$

Taking as a basis $\{x_3, x_4, x_5\}$ is not a good idea since it would lead to $x_3 = -5$ which is negative. So this basis is not a basic feasible solution. In order to find a basis, we will add *artificial variables* for the each constraint without variables (i) appearing in only that constraint (ii) with a positive coefficient. Usually it means that we *do not* have to add artificial variables for $\leq$ constraints (with positive RHS) and we *add* artificial variables for equality of $\geq$ constraints.

Illustration. In this set of constraints, we add two artificial variables u_1, u_2:

$$\begin{aligned} x_1 + x_2 - x_3 + u_1 &= 5 \\ x_1 + 2x_2 - x_4 + u_2 &= 3 \\ 3x_1 + 5x_2 + x_5 &= 20 \\ x_1, x_2, x_3, x_4, x_5, u_1, u_2 &\geq 0 \end{aligned}$$

Now, by construction, for each constraint there exists a variable x such that the coefficient of x is positive and x appears in precisely one constraint. Now consider BI a set of variables such that for each constraint BI contains precisely one variable of this type. Note that since $b \geq 0$, this base is a basic feasible solution. So, by adding this set of solution, we have solved our "problem", we can find a basic feasible solution for starting the simplex algorithm. In this case we have $BI = \{u_1, u_2, x_5\}$.

Be careful ! The addition of these artificial variables is not priceless ! Indeed we have seen that adding the slack or the surplus variables does not affect the shape of the feasible region. In some sense, by adding these variables, we just modify the representation of our problem but we do not modify the solution themselves. Adding artificial variables **DO MODIFY** the structure of the feasible region. In other words, the projection of a solution with artificial variables on the original LP is not necessarily a solution of the original LP. Note that it differs from the behavior of the slack / surplus variables for which a projection of a solution was a solution of the original LP.

In particular, if $u_i > 0$ then the set of variables **is not a solution of the initial LP anymore**. Note nevertheless that if $u_i = 0$ for every i then the resulting solution is a feasible solution (actually, it is an equivalence).

Observation 1. *The projection of a solution of the LP with artificial variables is a solution of the LP in standard form iff all the artificial variables equal* 0.

Moral. We have seen a method which permits to compute a basic feasible solution. Nevertheless, this method can introduce some new solutions which are not desired. Let us show how we can treat these artificial solutions in order to be sure that, at the end, we obtain a real feasible solution.

2.2 Big M method

When $u_i = 0$ for every i, then the resulting solution is a feasible solution of the original LP in standard form. So if we can ensure that at the end of the simplex method all the variables u_i equal zero, then we are done. To accomplish this, we put a very large negative cost, say $-M$, where M is a big positive integer (which explains the name of the method), on every variable u_i. How big M should be depends on the problem. It should be much larger than any other numbers we encounter while running the simplex method.

So if we apply this method to our example, the problem becomes:

$$\begin{aligned} \max(x_1 + x_2 - Mu_1 - Mu_2) \\ x_1 + x_2 - x_3 + u_1 = 5 \\ x_1 + 2x_2 - x_4 + u_2 = 3 \\ 3x_1 + 5x_2 + x_5 = 20 \\ x_1, x_2, x_3, x_4, x_5, u_1, u_2 \geq 0 \end{aligned}$$

Nevertheless there are two main drawbacks to the use of the big M method:

- We do not know *a priori* how large M must be in order to be sure that all the variables u_i equal zero (or at least tends to 0) at the end of the algorithm. Recall that M can be much larger than any number in the input.
- It may lead to numerical difficulties. Indeed, because of the bounded precision in the way computers represent floating-point numbers, using values of M that are much larger than other data of the input can become problematic. For example, if M is large, a computer might not be able to distinguish M and $M + 1$.
- Maybe the optimal solution will never put any of the u_i's to 0 and then we will have to slightly approximate the solution.

2.3 Phase I/II procedure

When have seen that we have to introduce artificial variables and that a solution of the optimization problem will provide a feasible solution of the problem without artificial variables if and only if the artificial variables equal zero. So there is a simple solution: what about changing the objective function and replace it by a minimization of the sum of the artificial variables. This idea seems weird at first glance since we completely loose our initial objective function, but it has one interest: we are sure that if the initial problem has a solution then this solution will have value 0 (*i.e.* the artificial variables will have value 0). And then this solution will provide a basic feasible solution of the initial LP without artificial variables. So we just have to launch the LP with this new basic feasible solution in order to find an optimal solution of the original LP.

Let us illustrate this method. The LP after the introduction of the artificial variables is the following:

$$\begin{aligned} \min u_1 + u_2 \\ x_1 + x_2 - x_3 + u_1 = 5 \\ x_1 + 2x_2 - x_4 + u_2 = 3 \\ 3x_1 + 5x_2 + x_5 = 20 \\ x_1, x_2, x_3, x_4, x_5, u_1, u_2 \geq 0 \end{aligned}$$

So we replace the objective function by a minimization of the sum of the artificial variables, *i.e.*

$$\min u_1 + u_2 = \max -u_1 - u_2$$

The advantage of this method is indeed that it finds an optimal solution. The main drawback is that we have to solve two LP instead of one, which indeed increases the running time of the algorithm.

3 Simplex termination

3.1 Degeneracy and termination

Up to this point, we have only consider "nice" linear programs, *i.e.* LP which ends "correctly". Let us see that the situation is not necessarily so great in general. Consider the linear program:

$$\max 2x_1 + x_2$$

under the constraints:

$$\begin{aligned} 4x_1 + 3x_2 &\leq 12 \\ 4x_1 + x_2 &\leq 8 \\ 4x_1 + 2x_2 &\leq 8 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Let us first transform this problem into a standard LP. The tableau corresponding to the standard form is the following:

2	1	0	0	0	0
4	3	1	0	0	12
4	1	0	1	0	8
4	2	0	0	1	8

Let us first add x_1 as a basic variable. First note that there is a tie for the ratio test between the second and the third row. Assume that we take the second row. So the variable x_4 leaves the basis (and the variable x_1 enters in it). We obtain the following tableau:

0	$\frac{1}{2}$	0	$-\frac{1}{2}$	0	-4
0	2	1	-1	0	4
1	$\frac{1}{4}$	0	$\frac{1}{4}$	0	2
0	1	0	-1	1	0

The current basic feasible solution is the point $(2, 0, 4, 0, 0)$. Note that the variable x_5 equals zero (which is not really surprising, why? hint: think about the tie). We then say that x_5 is *degenerate basic variable*. There is one variable, x_2 which has a positive coefficient in the objective function. So we can try to add x_2 in the solution.

The "problem" is that the ratio test ensures that $x_2 \leq 0$ (which is not really a problem, we can nevertheless make the pivot). After the pivot we obtain:

0	0	0	0	$-\frac{1}{2}$	-4
0	0	1	1	-2	4
1	0	0	$\frac{1}{2}$	$-\frac{1}{4}$	2
0	1	0	-1	1	0

Note that since $x_2 = 0$ in the ratio test, the value of the objective function has not increased even if we have made some pivot operation. More precisely the basic feasible solution after the pivot operation is still the point $(2, 0, 4, 0, 0)$. The fact that two basis give the same point of the polyhedron is called the *degeneracy*.

Note that in this case, it does really have some "effect" on our algorithm since the current bfs at the end is precisely the optimal solution. But in the general case, it is not the case. Even worse, it can happen that the Simplex algorithm make "cycles", *i.e.* that you pass through the same basis several times (if your choice is bad). If the choice of variables leaving and entering the basis is deterministic, it means that the Simplex algorithm might just cycle on this sequence of basis and will never end.... which is fairly bad ! Fortunately, these kinds of cases rarely happen in practice and can be avoided using clever systems of rules, as we'll see later.

Let us now try to understand geometrically the notion of genereracy. The geometrical representation of the first LP is given in Figure 1. Notice that three, and not two, constraint equalities pass through

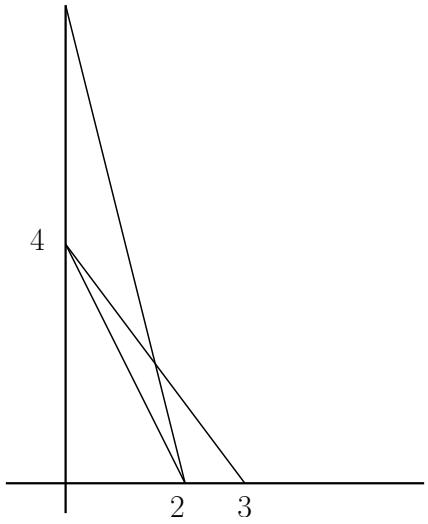

Figure 1: Geometrical representation of the degenerated example

the point $(x_1, x_2) = (2, 0)$. These equations are: $x_2 = 0$, $4x_1 + x_2 = 8$, and $4x_1 + 2x_2 = 8$. Since only two lines are needed to define such an intersection (indeed a point is just the intersection of two affinely independent equations of the plane), we see that degeneracy is a manifestation of redundancy in the information of the data. More formally, any pair of these three equations are enough to find the point $(2, 0)$.

For example, if we choose $x_2 = 0$ and $4x_1 + x_2 = 8$ as the defining equations, then, since the solution to this pair of equations will automatically satisfy equation $4x_1 + 2x_2 = 8$, and then the value of the slack variable associated with the inequality $4x_1 + 2x_2 \leq 8$, namely x_5, must turn out to be 0. The defining equations corresponds to the set of equations which are binding. The sets of equations which are forced to be binding are the set of variables which are constraint to be equal to zero: the variable x_2 and the variable x_4 in this case. So this ensures that when x_2 and x_4 are equal to 0 then the variable x_5 is forced to be equal to zero too. It is precisely what we obtained after the first pivot operation.

Similarly, if we choose $4x_1 + x_2 = 8$ and $4x_1 + 2x_2 = 8$ as the defining equations, then the inequality constraint $x_2 \geq 0$ will turn out to be binding. So when the variables x_4 and x_5 are non-basic variables, the variable x_2 will be equal to 0, which corresponds to the tableau after the second pivot. Finally, even if we have not made the corresponding calculations, if we chooce $x_2 = 0$ and $4x_1 + 2x_2 = 8$ as the defining equations, then the constraint $4x_1 + x_2 \leq 8$ will be automatically binding and the corresponding tableau will give the same point.

If the input of the Simplex algorithm is degenerated, then the Simplex algorithm can modify its basis without strictly improving the objective function since the vertex of the polyhedron considered by the Simplex algorithm is still the same.

Remark: An example and a pivoting rule on which the Simplex algorithm is cycling will be seen during the exercise Session.

The simplex method without degeneracy	**The simplex method with degeneracy**
The solution changes after each pivot.	The solution may stay the same after a pivot.
The obj. value strictly improves after a pivot.	The objective value may stay the same.
The simplex method is guaranteed to be finite.	The simplex method may not ends.
Distinct tableaux give distinct solutions.	Several tableaux can give the same solution.

The rest of this section is devoted to explain methods to ensure that the Simplex algorithm stops even with degeneracy.

3.2 Small random perturbations.

A basic feasible solution is called *non-degenerate* if none of the RHS' are 0. An LP is *non-degenerate* if **all** its basic feasible solutions are non-degenerate. Pivoting on a non-degenerate basis strictly increases

the objective function (since we have assumed that all the basic feasible solutions are non-degenerate, it means that in **every** case we will increase our objective function).

Note that there are at most $\binom{n}{m}$ possible basis (which is the number of ways to choose m variables from n). Hence there is a finite number of such bases. It follows that if our LP is non-degenerate, then as we pivot from basis to basis, we are strictly increasing the objective function and hence we cannot obtain twice the same basis. Thus the LP stops in finite time.

If the LP is degenerate (which is often the case), then one may perturb the data slightly to produce a non-degenerate LP whose optimal solution is very close to the original (if we put a random perturbation on the coefficient, then the LP has no solution with probability zero). One could then run Simplex Algorithm in finite time on the perturbed LP, and this solution is almost equal to the optimal solution. But there is an even better fact: the optimal basis for the perturbated LP (this basis is unique since the LP is non-degenerated) also is a optimal basis for the initial LP.

There are many distinct forms of the Simplex which choose the entering or leaving variables differently. The version we used was simply to pick the entering variable that gives the biggest bang for the buck: *i.e.* choose x_s such that $c_s = \max_j\{c_j : c_j > 0\}$.

3.3 Bland's rule

Between 1946 and 1977, no general pivoting rules ensured that the Simplex Algorithm ends in finite time except small perturbations. In 1977, Bland proposed (and proved) the first rule that guarantees that the Simplex algorithm stops in finite time.

The Brand's rule is defined as follows:

- Among all candidates for the entering variable in the set BI of basic variables (*i.e.*, those with a positive coefficient in the objective function), choose the one with the smallest index, say s.
- Among all the constraints for which the minimum ratio test results in a tie, choose the constraint r for which the corresponding basic variable has the smallest index, j_r.

Bland showed that using these conditions, the Simplex Algorithm runs in finite time **even without any perturbation**.

Before proving it, we need the following lemma (left as an exercise):

Lemma 2. *Assume that the Simplex algorithm do not stricly improve the objective value at step t. Then the two solutions correspond to the same point.*

The proof of this lemma is left as an exercise. Let us now prove that Bland's rule stops in finite time. The proof of this statement is surprisingly not that simple.

Theorem 3. *Using Bland's rule for the pivot operation, the Simplex algorithm does not cycle.*

Proof. Assume by contradiction that the Simplex algorithm is cycling. Let us denote by $B_1, \dots, B_k$ the basis of a cycle. Between two consecutive basis, there is only one variable leaving the basis and one variable entering in the basis. Amongst all variables leaving the basis during the procedure, let x_t be the one with the largest index. Witout loss of generality we can assume that it left the basis between B_1 and B_2. And let x_s be the variable entering in the basis in B_2. A step B_1 we have:

$$\begin{aligned} x_j &= b_j - \sum a_{ji}x_i \text{ for all } j \in B \\ z &= C + \sum_{i \notin B} c_i x_i \text{ where } c \text{ is the current value of the solution.} \end{aligned} \tag{1}$$

Since we are cycling, x_t must enter in some basis. Let B' be a basis on which x_t is entering. We have

$$z = C + \sum_i c'_i x_i \tag{2}$$

where $c'_i = 0$ is $i \in B'$. Indeed the coefficient for variable of the basis is equal to 0 at this step of the simplex algorithm.

Note that we did not improve the solution so C is the same. Moreover the two objective functions are the same (the system of equation are equivalent), the value of z of the objective function is the same in (1) and (2) (see TDs if you are not convinced). Since

$$\begin{aligned} x_s =& y \\ x_i =& 0 \qquad\qquad \text{for all } i \notin B, i \neq s \\ x_i =& b_i - a_{i,s} y \qquad\qquad \text{for all } i \in B \end{aligned}$$

So we have:

$$C + c_s y = C + c'_s y + \sum_{i \in B} c'_i (b_i - a_{is} y)$$

$$\Leftrightarrow y(c_s - c'_s + \sum_{i \in B} c'_i a_{is}) = \sum_{i \in B, i \neq s} c'_i b_i.$$

Since the former equation is true for any value of y, we have:

$$c_s - c'_s + \sum_{i \in B} c'_i a_{is} = 0$$

Let us now show that this leads to a contradiction.

- First notice that since s is entering in the basis at step B_1, we have $c_s > 0$ (by definition of Bland's rule).
- Moreover x_t is entering at step B' and by maximality of t we have $c'_s \leq 0$.
  Indeed if $x_s \in B'$ the $c'_s = 0$ (by the simplex algorithm) or $x_s \notin B'$ and $c'_s \leq 0$ by maximality of t (and x_t is entering).

Thus we have

$$\sum_{i \in B} c'_i a_{is} < 0 \Rightarrow \text{ there exists } r \in B \text{ such that } c'_r a_{rs} < 0$$

The variable x_r is basic in B_1 and non-basic in B'. So it must leave the basis at step B'' where B'' is between B_1 and B' in the order. Note moreover that:

- $r \leq t$ by maximality of t;
- $r \neq t$. Indeed $c'_t > 0$ (x_t enters in the basis at step B') and $a_{ts} > 0$ (x_t leaving the basis at B_1). Thus $a_{ts} c'_t > 0$

We now have all the ingredients to conclude. Since x_r and x_t are non basic in B', Bland's Rule ensures that $c'_r \leq 0$. Thus a_{rs} must be positive. Finally

$$x_r = 0 \cdots - a_{rs} x_s = 0$$

(the last equality comes from Lemma 2). So x_r is a candidate for leaving B_1 (it reaches the minimal value for the ratio test since it gives $x_r \leq 0$), contradicting the definition of t ! □

Since there is a finite number of basis, the Simplex algorithm stops as long as there is no cycle.

4 Other leads: Complexity and multiple solutions

These two problems will be treated during the exercise session. Let us nevertheless briefly discuss them here.

Running time Empirically, the number of iterations needed to solve the LP usually lies between 1.5 and 2 times the number of constraints (i.e., between 1.5m and 2m) when we are using classical pivoting rules. Nevertheless it could be worse. Actually, the number of steps can be exponential in the number of variables (recall that the number of vertices of a polytope can be exponential in the number of constraints, for instance for the cube).

Examples on which the Simplex algorithm might run in exponential time will be seen treated the exercise session. Actually even the Bland's rule might need an exponential running time. It is still open to determine if the Simplex algorithm runs in polynomial time in the worst case for some "good" pivoting rule choice.

Multiple solutions Determining if a LP has a unique or multiple optimal solutions might be read on the final tableau of the LP. For more details, go to the exercises session !

Final remarks: Advantages of the Simplex Algorithm. The simplex algorithm will take as data a LP in standard form (called canonical form in Bradley, Hax, Magnanti) and solve it. As we already mentioned, the Simplex Algorithm will consist in looking for a local maximizer which is also a global maximizer. The simplex algorithm has many advantages:

- It is *robust* since it can solve any linear program (unbounded linear programs or linear programs with empty feasible regions). Moreover it can detect redundant constraints.
- It is *self-initiating* since it uses itself to generate an initial feasible solution or show that the problem does not have any solution.
- It provides *more information than the optimal solution*. It indicates how the optimal solution is modified in function of the data (sensitivity analysis).

Before introducing the simplex algorithm, we need the following notion of standard form.

4.1 Historical insight on the Simplex Algorithm

1947 Simplex Algorithm (Dantzig)
1953 First examples of cycling (Hoffman)
1960's Interest in achieving a polynomially bounded number of pivots. **1972** Klee Minty give their squashed cube example which shows that Dantzig's rule may take exponentially many pivots.
1977 Bland's Rule: first finite termination rule.
1992 Steepest Edge (Forrest Goldfarb) pivot rule proves to be very successful in practice.
2011 Friednman shows that Zadeh's (least recently used) pivot rule may take $2^{\theta(n)}$ pivots.
2011 Friedman, Hansen, Zwick show a similar lower bound for Simplex with Random Edge pivoting.