Linear Programs

The Simplex Algorithm I

Bashar Dudin

August 11, 2017

EPITA

Linear Optimization over Polyhedra

Geometry of the Set of Feasible Solutions

Let L be the following linear program, in standard form :

$$\begin{array}{ll} \text{maximize} & x_1 + 2x_2 \\ \text{subject to} & \\ & x_1 + x_2 \leq 5 \\ & -2x_1 + x_2 \leq 3 \\ \text{with} & x_1, x_2 \geq 0 \end{array}$$

The set of feasible solutions of L is the region of the plane in between the positive parts of both axes and the lines having equations $x_2 = 5 - x_1$ and $x_2 = 3 + 2x_1$.

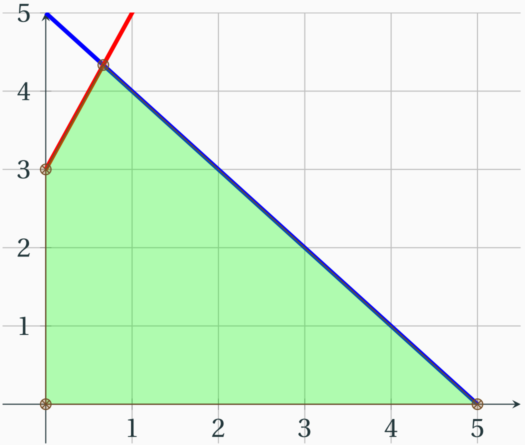
Geometry of the Set of Feasible Solutions

This set of feasible solutions of L has a remarkable geometric property called *convexity*.

Definition

A subset A of $\mathbb{R}^n$ is said to be ***convex*** if the line segment linking any two points of A is contained in A.

In our case, we have a specific type of convex set called a ***polyhedron***. It is the intersection of a finite set of half-spaces.



Geometry of the Set of Feasible Solutions

Proposition

Any linear program has an underlying set of feasible solutions which is convex.

Definition

We call ***interior*** of a convex set A the set of points $x \in A$ for which there is a disk having a positive radius and centered at x contained in A. Any point of A not satisfying this condition is called a ***boundary point***. The ***boundary*** of A is the set of boundary points.

When dealing with convex sets defined by linear constraints, boundary points are points of the hyperplanes defined by replacing inequalities by equalities. This is clearly the case in our 2-dimensional case.

Boundary Optimizes Linear Function

Recall that a function $f : A \to \mathbb{R}$ for a region A in $\mathbb{R}^n$ is said to be ***bounded*** if there is a positive real number M such that : for all $x \in A$, $\left|f(x)\right| \leq M$.

Proposition

Let A be the region defined by the set of feasible solutions of a linear program L. If the objective functional is *bounded* on A then L has an optimal objective value, it is attained on the boundary of A.

Boundary Optimizes Linear Function | Naive Search

Boundary Optimizes Linear Function | A Better Search

Rather than listing all potential optimal points, the ***simplex algorithm*** is a walk along boundary points of the feasible region satisfying the fact :

> *A given step has higher or equal objective value than previous step.*

It has an initialization step, that we're going to keep on the side for now.

Rather than listing all potential optimal points, the ***simplex algorithm*** is a walk along boundary points of the feasible region satisfying the fact :

A given step has higher or equal objective value than previous step.

It has an initialization step, that we're going to keep on the side for now. We shall be working under the following assumption.

Assumption

Given a linear program L in *standard* form, we assume the vector having only zero entries is a feasible solution of L.

In the following example, starting from the zero feasible solution we're going to walk around the boundary of the feasible region to look for an optimal value.

Simplex Algorithm : First Try

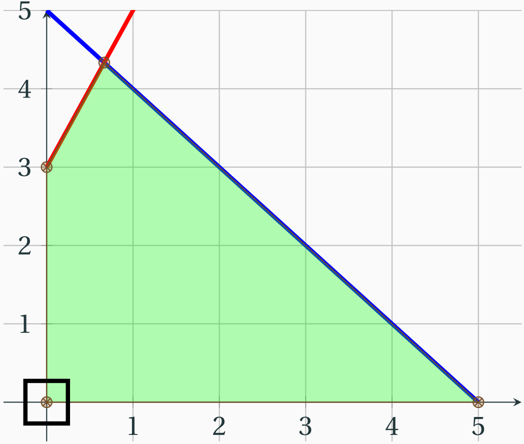
This is the slack form of the linear program L we started with.

$$\begin{aligned} \text{maximize} \quad & x_1 + 2x_2 \\ \text{subject to} \quad & \\ & x_3 = 5 - x_1 - x_2 \\ & x_4 = 3 + 2x_1 - x_2 \\ \text{with} \quad & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Recall that the slack variables x_3 and x_4 tell us how far our initial equalities are from being ***tight***. The zero feasible solution of the standard form of L corresponds to the solution $(0, 0, 5, 3)$ of its slack form. It has objective value 0, the first two entries give a feasible solution of L and the second two tell us how much margin there is in each inequality.

Simplex Algorithm : First Practical Examples

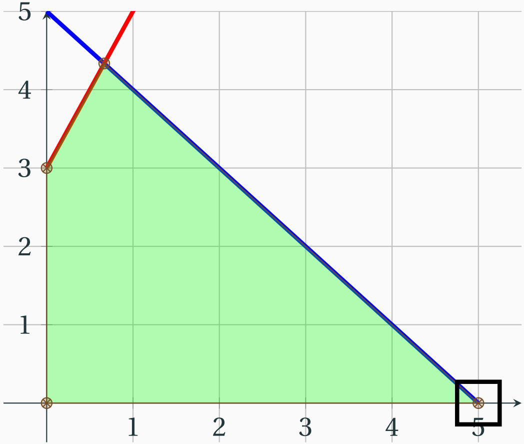
The feasible solution of the slack form $(0, 0, 5, 3)$ has corresponding feasible solution of the standard form at the origin of the plane. To wander around the boundary of the feasible region we have to choose which way to go. Either the vertical or horizontal edges of the boundary respectively keeping x_1 or x_2 to 0. Any choice is fine as long as we are increasing the objective value. We shall give a *rule of thumb* as to what choice one can make at each step later. For now, let us increase x_1 while keeping x_2 at 0.



Simplex Algorithm : First Practical Examples

$$\begin{aligned}
&\text{maximize} && x_1 + 2x_2 \\
&\text{subject to} && \\
& && x_3 = 5 - x_1 - x_2 \\
& && x_4 = 3 + 2x_1 - x_2 \\
&\text{with} && x_1, x_2, x_3, x_4 \geq 0
\end{aligned}$$

x_1 is constrained by the first equation : increasing x_1 indefinitely would violate non-negativity constraints while no such constraints come from the second equation. The highest possible value for x_1 is thus obtained when x_3 is zero. It is $(5, 0, 0, 13)$, of objective value 5.

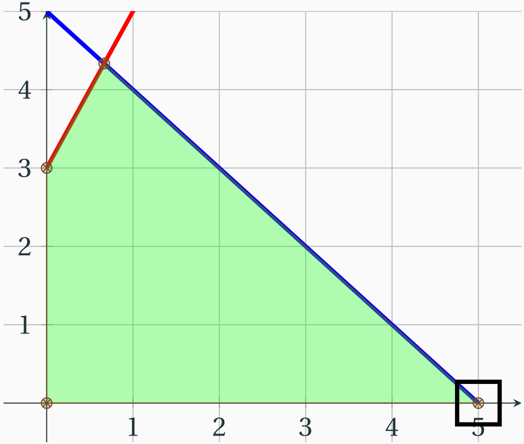


Simplex Algorithm : First Practical Examples

To get the feasible solution $(5,0,0,13)$ one can express x_1 in the first equation in terms of the two other variables. One then replaces x_1 elsewhere with this expression. We get the equivalent program

$$\begin{array}{lll} \text{maximize} & 5 - x_3 + x_2 & \\ \text{subject to} & & \\ & x_1 & = 5 - x_3 - x_2 \\ & x_4 & = 13 - 2x_3 - 3x_2 \\ \text{with} & x_1, x_2, x_3, x_4 \geq 0 & \end{array}$$

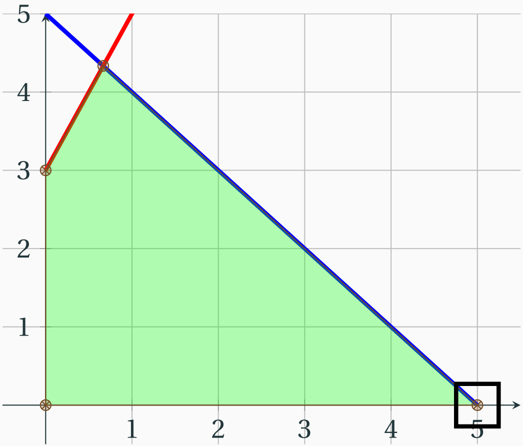
Putting x_3 and x_2 to zero in both equation gives back the expected feasible solution.



Simplex Algorithm : First Practical Examples

$$
\begin{aligned}
\text{maximize} \quad & 5 - x_3 + x_2 \\
\text{subject to} \quad & \\
& x_1 = 5 - x_3 - x_2 \\
& x_4 = 13 - 2x_3 - 3x_2 \\
\text{with} \quad & x_1, x_2, x_3, x_4 \geq 0
\end{aligned}
$$

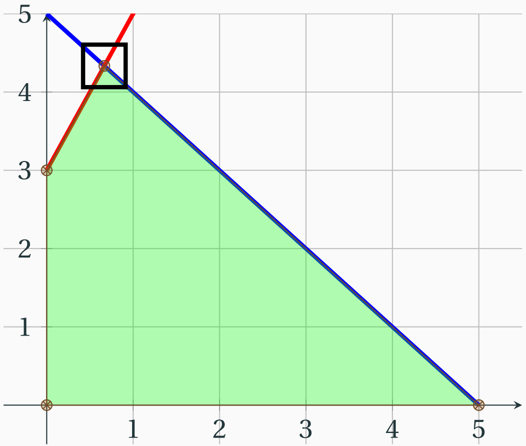
One can hope to maximize this linear program by increasing x_2. Any increase in x_3 would decrease the objective value. The most restrictive equation for x_2 is the second, indeed one can only increase x_2 up to 13/3 while in the second up to 5.



Simplex Algorithm : First Practical Examples

$$\begin{aligned}
\text{maximize} \quad & \frac{28}{3} - \frac{5}{3}x_3 - \frac{1}{3}x_4 \\
\text{subject to} \quad & \\
& x_1 = 2/3 - (1/3)x_3 + (1/3)x_4 \\
& x_2 = 13/3 - (2/3)x_3 - (1/3)x_4 \\
\text{with} \quad & x_1, x_2, x_3, x_4 \geq 0
\end{aligned}$$

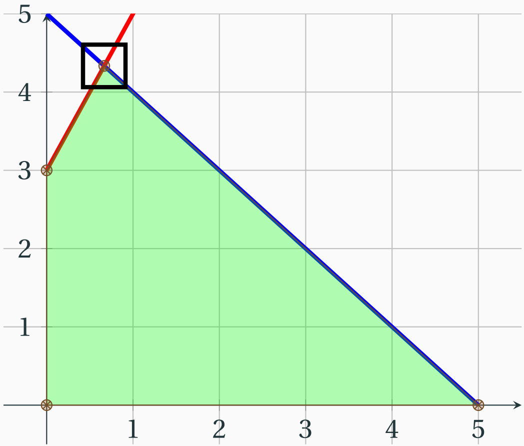
Playing the same previous trick using x_2 and the second equation we get the above equivalent linear program. Putting both x_3 and x_4 to zero, we get the feasible solution $(2/3, 13/3, 0, 0)$ which has objective value $28/3$.



Simplex Algorithm : First Practical Examples

$$\begin{aligned} \text{maximize} \quad & \frac{28}{3} - \frac{5}{3}x_3 - \frac{1}{3}x_4 \\ \text{subject to} \quad & \\ & x_1 = 2/3 - (1/3)x_3 + (1/3)x_4 \\ & x_2 = 13/3 - (2/3)x_3 - (1/3)x_4 \\ \text{with} \quad & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

We can't hope to increase the objective value. Any increase in the values of x_3 or x_4 would decrease the object value. The objectve function is maximal when both are zero. The corresponding maximal solution is $(2/3, 13/3, 0, 0)$ of objective value $28/3$.



Simplex Algorithm : A Step Towards Rigorousness

Notation and Terminology

Consider a linear program given in standard form as

$$
\begin{aligned}
&\text{maximize} && z = v + \sum_{j=1}^{n} c_j x_j \\
&\text{subject to} && \\
& && \forall i \in B, \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \\
&\text{with} && \forall j \in N, \quad x_j \geq 0
\end{aligned}
$$

where $(a_{ij})_{i,j}$ is an (m, n) matrix. The set N is the set $\{1, \ldots, n\}$ of indexes of variables, the one of B is the set $\{n+1, \ldots, n+m\}$ indexing linear constraints. The need for these two sets will be clear later on. Lastly, $(c_1, \ldots, c_n)$ is a vector of size n while $(b_{n+1}, \ldots, b_{n+m})$ is one of size m.

Notation and Terminology

The slack form of the previous linear program is written

$$
\begin{array}{ll}
\text{maximize} & z = v + \sum_{j=1}^{n} c_j x_j \\
\text{subject to} & \\
& \forall i \in B, \quad x_i = b_i - \sum_{j=1}^{n} a_{ij} x_j \\
\text{with} & \forall j \in N \cup B, \quad x_j \geq 0
\end{array}
$$

The set of variables on the left (and indexed by B) are called ***basic*** variables. Variables on the left and indexed by N are called ***non-basic*** ones. The ***basic solution*** of a linear program in slack form is the one obtained by putting all non-basic variables to zero. This is not, in general, a feasible solution of the linear program (but we'll learn to deal with it).

Practical Steps of the Simplex Algorithm

The simplex algorithm is a procedure exchanging a basic variable with a non-basic one at each step. Recall that up so far we have been working under the following assumption:

Assumption

The zero tuple is a feasible solution of the linear program in standard form. Equivalently, the *basic solution* of the slack form of our linear program is a feasible solution.

We shall take care of the general case later. It does build up on this case.

Practical Steps of the Simplex Algorithm

Here is the main steps of the simplex algorithm

- Choose a non-basic variable x_e increasing the objective value.
- Use the most restrictive constraint ℓ on x_e to express x_e in terms of the remaining variables.
- Replace x_e in remaining linear equalities and linear functional by the expression previously obtained. The variable x_e becomes a basic variable while x_ℓ becomes non-basic.
- Putting all non-basic variables to zero we get a tuple whose first n entries are a feasible solution of the linear program we started with.
- Repeat previous steps until there is no room for increasing the objective value.

Practical Steps of the Simplex Algorithm

The previous steps are precisely what we've been doing while wandering around the boundary of our case. This approach does however exceed the 2-dimensional case.

A 3-dimensional case

Following previous steps try solving the linear program

$$
\begin{array}{ll}
\text{maximize} & 3x_1 + x_2 + 2x_3 \\
\text{subject to} & \\
& x_1 + x_2 + 3x_3 \leq 30 \\
& 2x_1 + x_2 + 5x_3 \leq 24 \\
& 4x_1 + x_2 + x_3 \leq 36 \\
\text{with} & x_1, x_2, x_3 \geq 0
\end{array}
$$

A Finishing Blow about Boundedness

Notice that there is no garantee that the previous algorithm would get you a maximal solution in general. We do know that, if the linear functional is bounded on the feasible region, then there is a maximum attained at the boundary of this region. But there are cases when the linear functional is not bounded on the feasible region.

Unboundedness

Can you find a linear program which is not bounded?

That’s it for today