Linear Programs

The Simplex Algorithm I

Bashar Dudin

November 10, 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 $x_2 = 3 + 2x_1$.

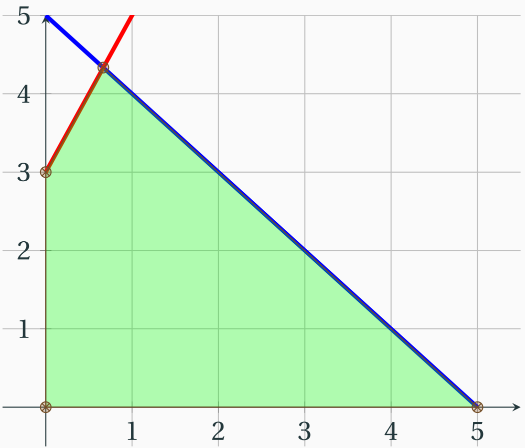
Geometry of the Set of Feasible Solutions

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 (it can be empty).

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 $x \in A$, $|f(x)| \le M$.

Proposition

Let A be the region defined by the set of feasible solutions of a linear program L. If the objective function is *bounded* on A then L has an optimal objective value, it is attained on the boundary of A (if not empty).

The justification for this result says way more!

Extremal Points Optimize Linear Function

Let A be the feasible region of a linear program L in standard form. An ***extremal point*** of A is a point x for which any *disk* centered at x doesn't contain any (open) line segment centered at x and fully contained in A.

Extremal Points Optimize Linear Function

Let A be the feasible region of a linear program L in standard form. An ***extremal point*** of A is a point x for which any *disk* centered at x doesn't contain any (open) line segment centered at x and fully contained in A. Such points are necessarily boundary points, they appear as intersections of as many hyperplanes as dimension of program.

Extremal Points Optimize Linear Function

Let A be the feasible region of a linear program L in standard form. An ***extremal point*** of A is a point x for which any *disk* centered at x doesn't contain any (open) line segment centered at x and fully contained in A. Such points are necessarily boundary points, they appear as intersections of as many hyperplanes as dimension of program.

Proposition

Under previous assumptions, if the objective function of L is *bounded* on A then L has an optimal objective value which is attained at an extremal point of A (if there are any).

Boundary Optimizes Linear Function | Naive Search

Let L be a linear program of dimension n. A naive way to look for optimal points is then to look for intersections of combinations of any n equations composing constraints of the linear program.

Boundary Optimizes Linear Function | Naive Search

Let L be a linear program of dimension n. A naive way to look for optimal points is then to look for intersections of combinations of any n equations composing constraints of the linear program. Looking for one single such intersection means solving a linear system having n equations.

Boundary Optimizes Linear Function | Naive Search

Let L be a linear program of dimension n. A naive way to look for optimal points is then to look for intersections of combinations of any n equations composing constraints of the linear program. Looking for one single such intersection means solving a linear system having n equations.

Question

Let n be a positive integer. Consider the linear constraints

$$\forall i \in \{1, \ldots, n\}, \quad 0 \leq x_i \leq 1.$$

How many extremal points do the following constraints define?

Boundary Optimizes Linear Function | Naive Search

Let L be a linear program of dimension n. A naive way to look for optimal points is then to look for intersections of combinations of any n equations composing constraints of the linear program. Looking for one single such intersection means solving a linear system having n equations.

Question

Let n be a positive integer. Consider the linear constraints

$$\forall i \in \{1, \ldots, n\}, \quad 0 \leq x_i \leq 1.$$

How many extremal points do the following constraints define?

As the example shows, complexity at worst grows exponentially. With number of inequalities defining the constraints.

Boundary Optimizes Linear Function | A Better Search

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

A given step returns a higher or equal objective value than previous step.

It has an initialization step, we're keeping on the side for now by 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.

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

A given step returns a higher or equal objective value than previous step.

It has an initialization step, we're keeping on the side for now by 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 set of extremal points of feasible region to look for an optimal value.

Simplex Algorithm : First Try

Simplex Algorithm : First Practical Examples

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

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

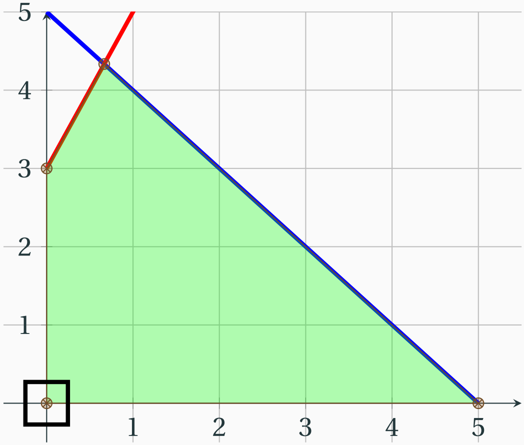
The zero feasible solution of the standard form of L corresponds to the solution $(0,0,5,3)$ of its slack form; it has objective value 0. The slack variables x_3 and x_4 tell us how far solution (x_1, x_2) is from making the constraints

$$\begin{aligned}
x_1 + x_2 &\leq 5 \\
-2x_1 + x_2 &\leq 3
\end{aligned}$$

tight, i.e. equalities.

Simplex Algorithm : First Practical Examples

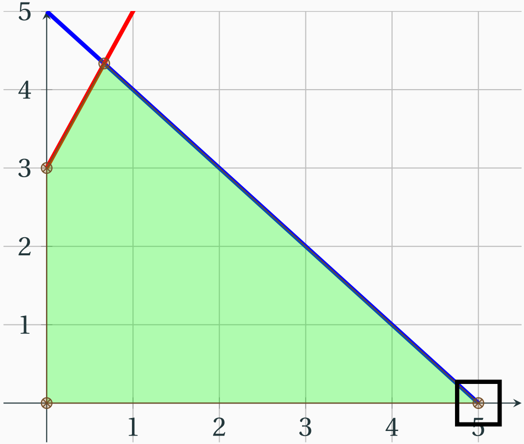
The feasible slack solution $(0,0,5,3)$ has corresponding feasible standard solution the origin of the plane. To wander around the boundary of the feasible region we have to choose which way to go ; we either go vertically keeping x_1 to 0 or horizontally keeping x_2 to 0. Any choice is fine as long 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

$$
\begin{array}{lll}
\text{maximize} & x_1 + 2x_2 & \\
\text{subject to} & & \\
& x_3 = 5 - x_1 - x_2 & (E_1) \\
& x_4 = 3 + 2x_1 - x_2 & (E_2) \\
\text{with} & x_1,\, x_2,\, x_3,\, x_4 \geq 0 &
\end{array}
$$

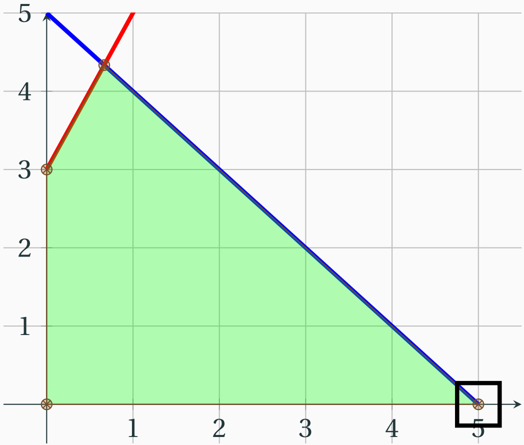
Variable x_1 is constrained by E_1: increasing it indefinitely would violate non-negativity constraints. No such constraints come from E_2. The highest possible value for x_1 is thus obtained when x_3 is 0. Obtained solution is $(5, 0, 0, 13)$ of objective value 5. It forces first equation of initial program tight.



Simplex Algorithm : First Practical Examples

$$\begin{array}{lrl} \text{maximize} & & x_1 + 2x_2 \\ \text{subject to} & & \\ & x_3 & = 5 - x_1 - x_2 \qquad (E_1) \\ & x_4 & = 3 + 2x_1 - x_2 \qquad (E_2) \\ \text{with} & & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

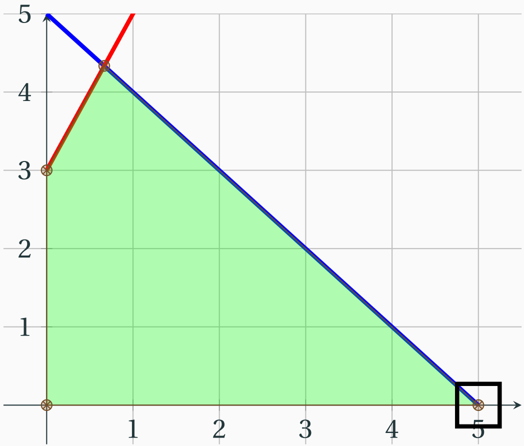
Solution $(0,0,5,3)$ can be understood as setting x_1 and x_2 to 0. Solution $(5,0,0,13)$ is about doing so to x_3 and x_2. The thing to notice is that extremal points are exactly those for which 2 variables are set to 0. To keep pattern *set variables on the left to* 0 we're going to exchange sides of x_1 and x_3 (variables that entered into choice of walk).



To get the feasible solution $(5,0,0,13)$ one can express x_1 in E_1 in terms of the two other variables. One then replaces x_1 elsewhere with this expression. We get the equivalent program

$$
\begin{array}{ll}
\text{maximize} & 5 - x_3 + x_2 \\
\text{subject to} & \\
& x_1 \quad = 5 - x_3 - x_2 \qquad (E_1) \\
& x_4 = 13 - 2x_3 - 3x_2 \qquad (E_2) \\
\text{with} & x_1, x_2, x_3, x_4 \geq 0
\end{array}
$$

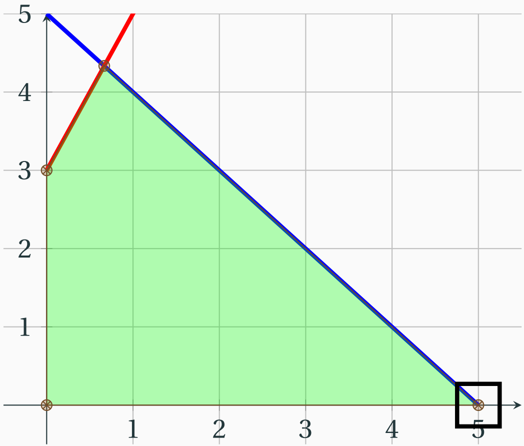
Putting x_3 and x_2 to zero in both equation gives back the expected feasible solution.



Simplex Algorithm : First Practical Examples

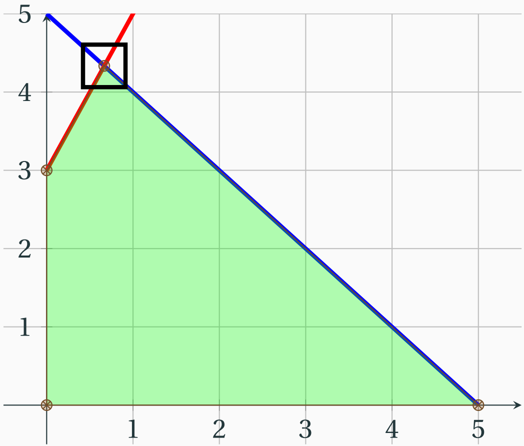
$$\begin{array}{lll} \text{maximize} & \frac{28}{3} - \frac{5}{3}x_3 - \frac{1}{3}x_4 & \\ \text{subject to} & & \\ & x_1 = 2/3 - (1/3)x_3 + (1/3)x_4 & (E_1) \\ & x_2 = 13/3 - (2/3)x_3 - (1/3)x_4 & (E_2) \\ \text{with} & x_1, x_2, x_3, x_4 \geq 0 & \end{array}$$

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 $28/3$.



$$\begin{aligned}
&\text{maximize} && 5 - x_3 + x_2 \\
&\text{subject to} \\
& && x_1 = 5 - x_3 - x_2 && (E_1) \\
& && x_4 = 13 - 2x_3 - 3x_2 && (E_2) \\
&\text{with} && x_1, x_2, x_3, x_4 \geq 0
\end{aligned}$$

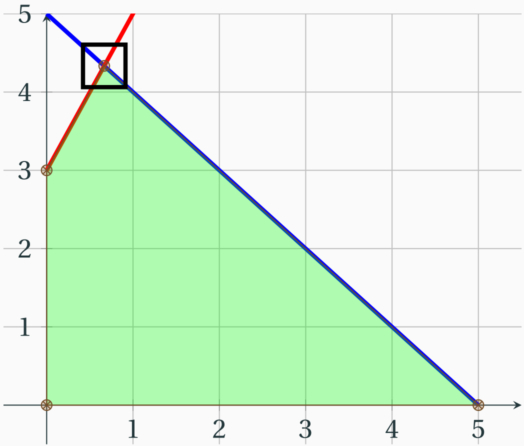
One can hope to maximize this linear program by increasing x_2. Any increase in x_3 would decrease the objective value. The most restrictive equation for x_2 is the second, indeed one can only increase x_2 up to 13/3 while in the second up to 5.



Simplex Algorithm : First Practical Examples

$$
\begin{aligned}
\text{maximize} \quad & \frac{28}{3} - \frac{5}{3}x_3 - \frac{1}{3}x_4 \\
\text{subject to} \quad & \\
& x_1 = 2/3 - (1/3)x_3 + (1/3)x_4 \qquad (E_1) \\
& x_2 = 13/3 - (2/3)x_3 - (1/3)x_4 \qquad (E_2) \\
\text{with} \quad & x_1, x_2, x_3, x_4 \geq 0
\end{aligned}
$$

We can't hope to increase the objective value anymore. 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 $\{1, \dots, n\}$ of indexes of variables, the one of B is the set $\{n+1, \dots, n+m\}$ indexing linear constraints[1]. The scalar v was zero in our previous examples. Lastly $(c_1, \dots, c_n)$ is a vector of size n while $(b_{n+1}, \dots, b_{n+m})$ is one of size m.

[1] The need for these two sets will be clear later on.

Notation and Terminology

The slack form of the previous linear program is written

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

The set of variables (indexed by B) are called ***basic*** variables. Variables on the right (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 ; it might have negative entries.

Practical Steps of the Simplex Algorithm

Linear programs having feasible basic solution are easier to deal with. The ones having non-feasible basic solution need a work around. The assumption under which we've been working can be rephrased as:

Assumption (BF)

The *basic solution* of initial linear program in standard form is a feasible solution.

We first treat programs satisfying (***BF***) before going onto general case.

Practical Steps of the Simplex Algorithm

The simplex algorithm is a procedure exchanging a basic variable with a non-basic one at each step.

Practical Steps of the Simplex Algorithm

The simplex algorithm is a procedure exchanging a basic variable with a non-basic one at each step. Here is the main steps of the simplex algorithm in case ($\boldsymbol{BF}$) is satisfied.

- Choose a non-basic variable x_e increasing the objective value.

Practical Steps of the Simplex Algorithm

The simplex algorithm is a procedure exchanging a basic variable with a non-basic one at each step. Here is the main steps of the simplex algorithm in case ($\boldsymbol{BF}$) is satisfied.

- 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.

Practical Steps of the Simplex Algorithm

The simplex algorithm is a procedure exchanging a basic variable with a non-basic one at each step. Here is the main steps of the simplex algorithm in case ($\boldsymbol{BF}$) is satisfied.

- 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.

Practical Steps of the Simplex Algorithm

The simplex algorithm is a procedure exchanging a basic variable with a non-basic one at each step. Here is the main steps of the simplex algorithm in case ($\boldsymbol{BF}$) is satisfied.

- 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.

Practical Steps of the Simplex Algorithm

The simplex algorithm is a procedure exchanging a basic variable with a non-basic one at each step. Here is the main steps of the simplex algorithm in case ($\boldsymbol{BF}$) is satisfied.

- 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 one cannot increase objective value anymore.

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 objective function is bounded on the feasible region, then there is a maximum attained at the boundary of this region. But there are cases when the objective function is not bounded on the feasible region.

Unboundedness

Can you find a linear program which is not bounded?

Practical Steps of the Simplex Algorithm

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

A 3-dimensional case

Following previous steps try solving the linear program

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

That’s it for today