Linear Programming

April 17, 2002

1 Introduction

In the VCR/guns/nuclear bombs/napkins/star wars/professors/butter/mice problem, the benelovent dictator, Bigus Piguinus, of south antartica (having 24 million pinguins under its control) has to decide how to allocate his empire resources to the maximal benefits of his pinguins. In particular, she has to decide how to allocate the money for the next year budget. For example, buying a nuclear bomb has a teremndous positive effect on security (the ability to destruct yourself completely together with your enemy is considered to induce a peaceful security feeling in most people). Guns on the other hand has lesser effect. Piguinga (the state) has to supply a certain level of security. Thus, the allocation should be such that:

\[ x_{\text{gun}} + 1000 \times x_{\text{nuclear-bomb}} \geq 1000 \]

where \( x_{\text{gun}} \) is the number of guns constructed, and \( x_{\text{nuclear-bomb}} \) is the number of NB constructed. On the other hand,

\[ 1000 \times x_{\text{gun}} + 1000000 \times x_{\text{nuclear-bomb}} \leq x_{\text{security}} \]

where \( x_{\text{security}} \) is the total Piguinga is willing to spend on security, and 1000 is the price of producing a single gun, and 1000000 is the price of manufacturing one nuclear bomb. There are a lot of other constrains of this type, and Biguis Piguinus would like to solve them, while minimizing the total money allocated for such spending (the less spend on budget, the larger the tax cut).

More formally, we have a large number of variables: \( x_1, \ldots, x_n \) and a large system of linear inequalities:

\[
\begin{align*}
  a_{11}x_1 + \ldots + a_{1n}x_n &\leq b_1 \\
  a_{21}x_1 + \ldots + a_{2n}x_n &\leq b_2 \\
  \vdots \\
  a_{m1}x_1 + \ldots + a_{mn}x_n &\leq b_m
\end{align*}
\]

we would like to decide if there is an assignment of values to \( x_1, \ldots, x_n \) where those inequalities are satisfied. Since there might be infinite number of such solutions, we decide that we want the solution that maximizes the quantity:

\[ c_1x_1 + \ldots + c_nx_n. \]
This quantity is known as the **objective function** of the linear program.

**Question 1.1** How do we compute such a solution?  
This is known as **linear programming**.

## 2 History

Linear programming can be traced back to the early 19th century. It really started in 1939 when L.V. Kantorovich noticed the importances of certain type of Linear Programming problems. Unfortunately, for several years, Kantorovich work was unknown in the west and unnoticed in the east.

Dantzig, in 1947, invented the simplex method for solving LP problems for the US Air-force planning problems.

T.C. Koopmans, in 1947, showed that LP provide the right model for the analysis of classical economic theories.

In 1975, both Koopmans and Kantorovich got the Nobel prize of economics. Dantzig probably did not get it because it was toooo mathematical.

## 3 Linear programming

### 3.1 Standard form

Maximize $\sum_{j=1}^{n} c_j x_j$

Subject to $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ for $i = 1, 2, \ldots, m$

$x_j \geq 0$ for $j = 1, \ldots, n$.

Setting

\[
    c = \begin{pmatrix}
        c_1 \\ \\
        \vdots \\ \\
        c_n
    \end{pmatrix}, \quad A = \begin{pmatrix}
        a_{11} & a_{12} & \cdots & a_{1(n-1)} & a_{1n} \\
        a_{21} & a_{22} & \cdots & a_{2(n-1)} & a_{2n} \\
        \vdots & \vdots & \cdots & \vdots & \vdots \\
        a_{(m-1)1} & a_{(m-1)2} & \cdots & a_{(m-1)(n-1)} & a_{(m-1)n} \\
        a_{m1} & a_{m2} & \cdots & a_{m(n-1)} & a_{mn}
    \end{pmatrix}, \quad x = \begin{pmatrix}
        x_1 \\
        x_2 \\
        \vdots \\
        x_{n-1} \\
        x_n
    \end{pmatrix}
\]

Then:

Maximize $c^T x$

Subject to $Ax \leq b$

$x \geq 0$
In the following, we are going to do a long long sequence of rewritings. The first one, was already done implicitly, as required all variables to be positive. This is of course an unreasonable requirement, BUT we can take an instance without this requirement and transform it into this “positive” form. This can be easily achieved by introducing two variables for each variable. Thus, we replace a variable $x_i$ by $x_i' - x_i''$. Clearly, we can now require $x_i'$ and $x_i''$ to be positive, as their difference can define any real number.

3.2 Slack Form

One can rewrite an LP instance, so that every inequality becomes equality, and all variables must be positive:

\[
\begin{align*}
\text{Maximize} & \quad c^T x \\
\text{Subject to} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

To do that, we introduce new variables (slack variables) replacing each inequality of the form

\[
\sum_{i=1}^{n} a_ix_i \leq b
\]

by

\[
x_{n+1} = b - \sum_{i=1}^{n} a_ix_i \\
x_{n+1} \geq 0
\]

In fact, now we have a special variable for each equality inequality in the LP program. This variables are special, and would be called basic variables. The variable on the right side are nonbasic variables (original isn’t it?). This form, is called SLACK FORM. It is defined by the following parameters:

| $B$ - Set of indices of basic variables |
| $N$ - Set of indices of nonbasic variables |
| $n = |N|$ - number of original variables |
| $b, c$ - two vectors of constants |
| $m = |B|$ - number of basic variables (i.e., number of inequalities) |
| $A$ - The matrix of coefficients |
| $N \cup B = \{1, \ldots, n + m\}$ |

And the slack form itself is:
A tuple $(N,B,A,b,c,v)$
Where the objective function is to maximize:

$$z = v + \sum_{j \in N} c_j x_j$$

And the constraints are:

$$x_i = b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for } i \in B,$$

of course, all variables $x_i$ must be non-negative (i.e., $x_i \geq 0$).

**Exercise 3.1** Show that any linear program can be transformed into equivalent slack form.

**Example 3.2** In the slack form, we have

$$z = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3}$$

$$x_1 = 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3}$$

$$x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3}$$

$$x_4 = 18 - \frac{x_3}{2} + \frac{x_5}{2}$$

Here we have:

$$B = \{1, 2, 4\}, \quad N = \{3, 5, 6\}$$

$$A = \begin{pmatrix} a_{13} & a_{15} & a_{16} \\ a_{23} & a_{25} & a_{26} \\ a_{43} & a_{45} & a_{46} \end{pmatrix} = \begin{pmatrix} \frac{-1}{6} & \frac{-1}{6} & \frac{1}{3} \\ \frac{8}{3} & \frac{2}{3} & \frac{-1}{3} \\ \frac{1}{2} & \frac{-1}{2} & \frac{0} \end{pmatrix}$$

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_4 \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 18 \end{pmatrix} \quad c = \begin{pmatrix} c_3 \\ c_5 \end{pmatrix} = \begin{pmatrix} \frac{-1}{6} \\ \frac{-1}{6} \end{pmatrix} \quad v = 28.$$

Note that indices depend on $N$ and $B$. Note also that $A$ entries are negation of what they appear in the slack form.

### 3.3 Geometric interpretation of linear programming

Each inequality defines a half space. The feasible region is the intersection of those half-spaces. This region is known as a *convex polytope* or just *polytope*.
3.4 A concrete example

\[
\begin{align*}
\text{maximize} & \quad 5x_1 + 4x_2 + 3x_3 \\
\text{subject to} & \quad 2x_1 + 3x_2 + x_3 \leq 5 \\
& \quad 4x_1 + x_2 + 2x_3 \leq 11 \\
& \quad 3x_1 + 4x_2 + 2x_3 \leq 8 \\
& \quad x_1, x_2, x_3 \geq 0
\end{align*}
\]

Next, we introduce slack variables, for example, rewriting \(2x_1 + 3x_2 + x_3 \leq 5\) as the constraints:

\[
\begin{align*}
\text{maximize} & \quad z = 5x_1 + 4x_2 + 3x_3 \\
\text{subject to} & \quad w_1 = 5 - 2x_1 - 3x_2 - x_3 \\
& \quad w_2 = 11 - 4x_1 - x_2 - 2x_3 \\
& \quad w_3 = 8 - 3x_1 - 4x_2 - 2x_3 \\
& \quad x_1, x_2, x_3, w_1, w_2, w_3 \geq 0
\end{align*}
\]

where \(w_1, w_2, w_3\) are the slack variables. Note also that they are currently also the basic variables. We start from the slack representation trivial solution: \(x_1 = x_2 = x_3 = 0\) then \(w_1 = 5, w_2 = 11\) and \(w_3 = 8\). This is a feasible solution, and \(z = 0\).

How to improve this solution? Let increase the value of \(x_1\). The good thing that happened is that the objective function increases, however, the bad thing that might happen is that the solution might stop being feasible. Thus, we are going to increase \(x_1\) as much as possible.

So: \(x_2 = x_3 = 0\) and

\[
\begin{align*}
w_1 &= 5 - 2x_1 - 3x_2 - x_3 = 5 - 2x_1 \\
w_2 &= 11 - 4x_1 - x_2 - 2x_3 = 11 - 4x_1 \\
w_3 &= 8 - 3x_1 - 4x_2 - 2x_3 = 8 - 3x_1
\end{align*}
\]

We want to increase \(x_1\) as much as possible, as long as \(w_1, w_2, w_3\) are non-negative. Formally,

\[
\begin{align*}
w_1 &= 5 - 2x_1 \geq 0, \\
w_2 &= 11 - 4x_1 \geq 0 \quad \text{and} \\
w_3 &= 8 - 3x_1 \geq 0.
\end{align*}
\]

This implies that:

\[
\begin{align*}
x_1 &\leq 2.5 \\
x_1 &\leq 11/4 = 2.75 \text{ and} \\
x_1 &\leq 8/3 = 2.66.
\end{align*}
\]
We must take the strictest condition. Namely, \( x_1 = 2.5 \). Putting it into the system, we now have, that the current solution is: \( x_1 = 2.5, x_2 = 0, x_3 = 0 \) and \( w_1 = 0, w_2 = 1, w_3 = 0.5 \) and the objective function is \( z = 5x_1 + 4x_2 + 3x_3 = 12.5 \).

What happened? One zero nonbasic variable (i.e., \( x_1 \)) became non-zero, and one basic variable became zero (i.e., \( w_1 \)):

\[
\begin{align*}
\text{maximize} & \quad z = 5x_1 + 4x_2 + 3x_3 \\
\text{subject to} & \quad w_1 = 5 - 2x_1 - 3x_2 - x_3 \\
& \quad w_2 = 11 - 4x_1 - x_2 - 2x_3 \\
& \quad w_3 = 8 - 3x_1 - 4x_2 - 2x_3 \\
& \quad x_1, x_2, x_3, w_1, w_2, w_3 \geq 0
\end{align*}
\]

Let us rewrite the first rule for \( w_1 \) so that \( x_1 \) is on the left side:

\[ x_1 = 2.5 - 0.5w_1 - 1.5x_2 - 0.5x_3 \]

The problem is that \( x_1 \) still appears in the right size of the equations for \( w_2 \) and \( w_3 \). We get:

\[
\begin{align*}
z & = 12.5 - 2.5w_1 - 3.5x_2 + 0.5x_3 \\
x_1 & = 2.5 - 0.5w_1 - 1.5x_2 - 0.5x_3 \\
w_2 & = 1 + 2w_1 + 5x_2 \\
w_3 & = 0.5 + 1.5w_1 + 0.5x_2 - 0.5x_3
\end{align*}
\]

Note that we get the solution, by putting \( w_1 = x_2 = x_3 = 0 \). We repeat the above procedure for \( x_3 \) (why? because its coefficient in the objective function is positive.).

Checking carefully, it follows that the maximum we can increase \( x_3 \) is to 1 (why?). We rewrite the last equality for \( x_3 \) and we get:

\[ x_3 = 1 + 3w_1 + x_2 - 2w_3 \]

Substituting this into the LP, we get:

\[
\begin{align*}
z & = 13 - w_1 - 3x_2 - w_3 \\
x_1 & = 2 - 2w_1 - 2x_2 + w_3 \\
w_2 & = 1 + 2w_1 + 5x_2 \\
w_3 & = 1 + 3w_1 + x_2 - 2w_3
\end{align*}
\]

What should we do next?

Nothing. We had reached the optimal solution. Indeed, all the coefficients in the objective function are negative (or zero). As such, the trivial solution (all non-basic variables get zero) is maximal, as they must all be non-negative, and increasing their value decreases the value of the objective function. So we better stop.
3.5 Starting somewhere

OK. We had transformed a linear programming problem into slack form. Intuitively, what the Simplex algorithm is going to do, is to start from a feasible solution and start walking around in the feasible region till it reaches the best possible point as far as the objective function is concerned. But maybe the linear program $L$ is not feasible at all (i.e., no solution exists.). Let $L$ be (in slack form):

$$
\begin{align*}
    z &= v + \sum_{j \in N} c_j x_j \\
    x_i &= b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for } i \in B,
\end{align*}
$$

where all variables $x_i$ must be non-negative (i.e., $x_i \geq 0$).

Clearly, if we set all $x_i = 0$ if $i \in N$ then this determines the values of the basic variables. If they are all positive, we are done, as we found a feasible solution. The problem is that they might be negative.

We generate a new LP problem $L' = \text{Feasible}(L)$:

$$
\begin{align*}
    \text{Minimize } x_0 \\
    x_i &= x_0 + b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for } i \in B,
\end{align*}
$$

where all variables $x_i$ must be non-negative (i.e., $x_i \geq 0$).

Clearly, if we pick $x_j = 0$ for all $j \in N$ (all the nonbasic variables), and a value large enough for $x_0$ then all the basic variables would be non-negatives, and as such, we have found a feasible solution for $L'$. Let $\text{StartingSolution}(L')$ denote this easily computable feasible solution.

We can now use the SimplexInner algorithm to find this optimal solution to $L'$ (because we have a feasible solution to start from!).

**Lemma 3.3** $L$ is feasible if and only if the optimal objective value of $L'$ is zero.

**Proof:** A feasible solution to $L$ is immediately an optimal solution to $L'$ with $x_0 = 0$, and vice versa. Namely, given a solution to $L'$ with $x_0 = 0$ we can transform it to a feasible solution to $L$ by removing $x_0$. □

We can now present a first version of the Simplex algorithm:
Simplex( LP: linear system of inequalities )
Transform LP into slack form. Let L be the resulting slack form.
Compute \( L' \leftarrow \text{Feasible}(L) \) (as described above)
\( x \leftarrow \text{StartingSolution}(L') \)
\( x' \leftarrow \text{SimplexInner}(L', x) \)
If objective function value of \( x' \) is > 0 then
return “No solution”
\( x'' \leftarrow \text{SimplexInner}(L, x') \)
return \( x'' \)

Thus, in the following, we have to describe \( \text{SimplexInner} \) - a procedure to solve an LP in slack form, when we start from a feasible solution.

4 The Simplex (Inner) Algorithm

Slack form encodes \((N, B, A, b, c, v)\) inside it the feasible solution. We assign all the Nonbasic variables the value zero. This implies immediately a value for the nonbasic variables, and for the objective function.

A tuple \((N,B,A,b,c,v)\)
Where the objective function is to maximize:
\[
z = v + \sum_{j \in N} c_j x_j
\]
And the constraints are:
\[
x_i = b_i - \sum_{j \in N} a_{ij} x_j \quad \text{for} \quad i \in B,
\]
of course, all variables \( x_i \) must be non-negative (i.e., \( x_i \geq 0 \)).

Since every basic variable is defined as a linear combination of the nonbasic variables, it follows that we can always assume that the objective function is defined by nonbasic variables. Thus, once we set all the nonbasic variables to zero, we immediately get the optimal objective value which is \( v \).

Let assume that all the basic variables for the feasible solutions are positive, and we have a nonbasic variable \( x_j \) that appears in the objective function, and furthermore \( c_j \) is positive.

Clearly, if we increase the value of \( x_e \) then one of the basic variables is going to vanish (i.e., become zero). Let \( x_l \) be this basic variable. What we are going to do is the following: increase the value of \( x_e \) (the entering variable) till \( x_l \) (the leaving variable) becomes zero.

We pick \( e \) to be one of the indices of
\[
\left\{ j \mid c_j > 0, \ j \in N \right\}
\]
\( x_e \) is the entering variable.
Setting all nonbasic variables to zero, and letting $x_e$ grow, implies that:

$$x_i = b_i - a_{ie}x_e.$$  

All those variables must non negative, thus we require:

$$\forall i \in B \quad x_i = b_i - a_{ie}x_e \geq 0$$

namely, $x_e \leq (b_i/a_{ie})$ or alternatively, $\frac{1}{x_e} \geq \frac{a_{ie}}{b_i}$. Namely, $\frac{1}{x_e} \geq \max_{i \in B} \frac{a_{ie}}{b_i}$ and, the new value of $x_e$ is

$$U = \left(\max_{i \in B} \frac{a_{ie}}{b_i}\right)^{-1}.$$  

We pick $l$ (the index of the leaving variable) from the set all basic variables that vanish to zero when $x_e = U$. Namely, $l$ is from:

$$\left\{ j \mid \frac{a_{je}}{b_j} = U \text{ where } j \in B \right\}.$$

Now, we know $x_e$ and $x_l$. We rewrite the equation that $x_l$ so that it has $x_e$ on the left size. Formally,

$$x_l = b_l - \sum_{j \in N} a_{lj}x_j \quad \Rightarrow \quad x_e = \frac{b_l}{a_{le}} - \sum_{j \in N \cup \{l\}} \frac{a_{lj}}{a_{le}}x_j \quad \text{where } a_{ll} = 1.$$

We do Guassian elimination, to remove any appearance of $x_e$ on the right side of the equalities in the LP (and also from the objective function). In the end of this process, we have a new equivalent LP where the basic variables are $B' = (B - \{l\}) \cup \{e\}$ and the non-basic variables are $N' = (N - e) \cup \{l\}$.

This process is called pivoting. Everytime we pivot, we make progress. Note, that the linear system is completely defined by which variables are basic, and which are non-basic. Furthermore, pivoting never returns to a combination (of basic/non-basic variable) that was already visited. Thus, we can do at most

$$\binom{n + m}{n} \leq \left(\frac{n + m}{n} \cdot e\right)^n$$

pivoting steps. And this is close to tight in the worst case (there are examples where $2^n$ pivoting steps are needed).

Each pivoting step takes polynomial time in $n$ and $m$. Thus, the overall running time of simplex is exponential in the worst case. However, in practice, Simplex is extremely fast.

### 4.1 Degeneracies

If you inspect carefully the Simplex algorithm, you would notice that it might get stuck if one of the $b_i$s is zero. This corresponds to a case where $> m$ hyperplanes passes through the same point. This might cause the effect, that you might not be able to make any progress at all in pivoting. In this case, we might get stuck.

There are several solutions, the simplest one, is to add tiny random noise to each coefficient. You can even do this symbolically. Intuitively, the degeneracy, being a local accident, disappears with high probability.
The larger danger, is that you would get into cycling - namely a sequence of pivoting operations that do not improve the objective function, and the bases you get are cyclic (i.e., infinite loop).

There is a simple scheme based on using the symbolic perturbation, that avoids cycling, by carefully choosing what is the leaving variable. We omit all details here.

There is an alternative approach, called Bland’s rule, which always choose the lowest index variable for entering and leaving out of the possible candidates. We will not prove the correctness of this approach here.

**Theorem 4.1 (Fundamental theorem of Linear Programming)** *For an arbitrary linear program, the following statements are true:*

1. If there is no optimal solution, the problem is either infeasible or unbounded.
2. If a feasible solution exists, then a basic feasible solution exists.
3. If an optimal solution exists, then a basic feasible solution exists.

*Proof:* Proof is constructive by running the simplex algorithm.

**4.2 Even more about the geometry**

The LP $L$ takes place in $\mathbb{R}^n$, where $n$ is the number of non-basic variables in $L$. The feasible solution encoded inside the slack form (where all nonbasic variables are zero) is just the origin. Each variable corresponds to one constraint. The basic variables corresponds to half-spaces (which we identify by the half-space of $\mathbb{R}^n$ where they receive a non-negative value), while the non-basic variables corresponds to the $n$ half-spaces that define the positive octant - by the requirement that they would be non-negative (i.e., in two dimensions those are the inequalities $x \geq 0$ and $y \geq 0$).

The objective function, corresponds to a direction in space defined by a vector $c$. At each stage, we walk from the origin, to a different vertex of the feasible polytope around us, such that this direction improves the objective function (i.e., $c$ has a positive projection on this axis. Namely, the coefficient that corresponds to this variable in the objective function is positive). During this walk, the vertex “leaves” one of the constraints (i.e., one of the non-basic variables is becoming a basic variable) by walking along an axis, till it hits a new constraint. The point when we stop walking is a new vertex, where we stop. The new constraint, that corresponds to a basic variables now becomes a non-basic variable.

Here is what happened in the pivoting process: We performed a transformation. The new vertex becomes the origin. Since we change the $n$ dimensions we are looking at in any point in time, the current constraints that corresponds to the nonbasic variables, always look orthogonal.

We continue till the objective direction is completely negative. At this point, we can stop, as we reached the vertex of the polytope which is extreme in the objective direction.
4.3 On the ellipsoid method and interior point methods

Simplex algorithm is exponential in the worst case.

The ellipsoid method is weakly polynomial (namely, it is polynomial in the number of bits of the input). Khachian 1979 came up with it. It turned out to be completely useless in practice.

In 1984, Karmakar came up with a different method, called the interior-point method which is also weakly polynomial. However, it turned out to be quite useful in practice, resulting in a arm race between the interior-point method and the simplex method.

The question of whether there is a strongly polynomial time algorithm for linear programming, is one of the major open questions in computer science.

4.4 Further reading

An excellent introduction text on linear programming is the book by Vanderbei [Van97] (see http://www.princeton.edu/~rvdb/). A more advanced (and somewhat cryptic) is the book by Grötschel et al. [GLS88]. They have a nice presentation of LP in the setting of various oracles, and the polynomiality of LP under those models.

There is also a lot of work in linear programming in the numerical analysis community. Programming a program that works to solve linear programs is extremely hard, because of floating point errors.

References
