Chapter 20

Linear Programming

By Sariel Har-Peled, December 7, 2009[①]

20.1 Introduction and Motivation

In the VCR/guns/nuclear-bombs/napkins/star-wars/professors/butter/mice problem, the benevolent dictator, Biga Piguinus, of Penguina (a country in south Antarctica having 24 million penguins under its control) has to decide how to allocate her empire resources to the maximal benefit of her penguins. In particular, she has to decide how to allocate the money for the next year budget. For example, buying a nuclear bomb has a tremendous positive effect on security (the ability to destruct yourself completely together with your enemy induces a peaceful serenity feeling in most people). Guns, on the other hand, have a weaker effect. Penguina (the state) has to supply a certain level of security. Thus, the allocation should be such that:

$$x_{gun} + 1000 * x_{nuclear-bomb} \geq 1000,$$

where x_{guns} is the number of guns constructed, and $x_{nuclear-bomb}$ is the number of nuclear-bombs constructed. On the other hand,

$$100 * x_{gun} + 1000000 * x_{nuclear-bomb} \leq x_{security}$$

where $x_{security}$ is the total Penguina is willing to spend on security, and 100 is the price of producing a single gun, and $1,000,000$ is the price of manufacturing one nuclear bomb. There are a lot of other constrains of this type, and Biga Piguinus would like to solve them, while minimizing the total money allocated for such spending (the less spent on budget, the larger the tax cut).

More formally, we have a (potentially large) number of variables: $x_1, \ldots, x_n$ and a (potentially large) system of linear inequalities. We will refer to such an inequality as a ***constraint***. We would like to decide if there is an assignment of values to $x_1, \ldots, x_n$ where all these inequalities are satisfied. Since there might be infinite number of such solutions, we want the solution that maximizes some linear quantity. See the instance on the right.

$$\begin{aligned}
&a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1 \\
&a_{21}x_1 + \ldots + a_{2n}x_n \leq b_2 \\
&\ldots \\
&a_{m1}x_1 + \ldots + a_{mn}x_n \leq b_m \\
&\max c_1x_1 + \ldots + c_nx_n.
\end{aligned}$$

The linear target function we are trying to maximize is known as the ***objective function*** of the linear program. Such a problem is an instance of ***linear programming***. We refer to linear programming as LP.

20.1.1 History

Linear programming can be traced back to the early 19th century. It started in earnest in 1939 when L. V. Kantorovich noticed the importance 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 his work was too mathematical. Thats how it goes.

20.1.2 Network flow via linear programming

To see the impressive expressive power of linear programming, we next show that network flow can be solved using linear programming. Thus, we are given an instance of max flow; namely, a network flow $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with source s and sink t, and capacities $\mathsf{c}(\cdot)$ on the edges. We would like to compute the maximum flow in G.

To this end, for an edge $(u \to v) \in E$, let $x_{u\to v}$ be a variable which is the amount of flow assign to $(u \to v)$ in the maximum flow. We demand that $0 \leq x_{u\to v}$ and $x_{u\to v} \leq \mathsf{c}(u \to v)$ (flow is non negative on edges, and it comply with the capacity constraints). Next, for any vertex v which is not the source or the sink, we require that $\sum_{(u\to v)\in E} x_{u\to v} = \sum_{(v\to w)\in E} x_{v\to w}$ (this is conservation of flow). Note, that an equality constraint $a = b$ can be rewritten as two inequality constraints $a \leq b$ and $b \leq a$. Finally, under all these constraints, we are interest in the maximum flow. Namely, we would like to maximize the quantity $\sum_{(\mathsf{s}\to u)\in E} x_{\mathsf{s}\to u}$. Clearly, putting all these constraints together, we get the linear program depicted on the right.

$$\begin{aligned}
&\forall (u \to v) \in E && 0 \leq x_{u\to v} \\
& && x_{u\to v} \leq \mathsf{c}(u \to v) \\
&\forall v \in V \setminus \{\mathsf{s}, \mathsf{t}\} && \sum_{(u\to v)\in E} x_{u\to v} - \sum_{(v\to w)\in E} x_{v\to w} \leq 0 \\
& && \sum_{(u\to v)\in E} x_{u\to v} - \sum_{(v\to w)\in E} x_{v\to w} \geq 0 \\
&\text{maximizing} && \textstyle\sum_{(\mathsf{s}\to u)\in E} x_{\mathsf{s}\to u}
\end{aligned}$$

It is not too hard to write down min-cost network flow using linear programming.

20.2 The Simplex Algorithm

20.2.1 Linear program where all the variables are positive

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \le b_i \\ & \text{for } i = 1, 2, \ldots, m \end{aligned}$$

We are given a LP, depicted on the left, where a variable can have any real value. As a first step to solving it, we would like to rewrite it, such that every variable is non-negative. This is easy to do, by replacing a variable x_i by two new variables x'_i and x''_i, where $x_i = x'_i - x''_i$, $x'_i \ge 0$ and $x''_i \ge 0$. For example, the (trivial) linear program containing the single constraint $2x + y \ge 5$ would be replaced by the following LP: $2x' - 2x'' + y' - y'' \ge 5$, $x' \ge 0$, $y' \ge 0$, $x'' \ge 0$ and $y'' \ge 0$.

Lemma 20.2.1 *Given an instance I of LP, one can rewrite it into an equivalent LP, such that all the variables must be non-negative. This takes linear time in the size of I.*

20.2.2 Standard form

Using Lemma 20.2.1, we can now require a LP to be specified using only positive variables. This is known as ***standard form***.

A linear program in standard form.

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \text{for } i = 1, 2, \ldots, m \\ & x_j \ge 0 \qquad\qquad \text{for } j = 1, \ldots, n. \end{aligned}$$

A linear program in standard form.

(Matrix notation.)

$$\begin{aligned} \max \quad & c^T x \\ \text{subject to} \quad & Ax \le b. \\ & x \ge 0. \end{aligned}$$

Here the matrix notation rises, by setting

$$c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}, b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}, A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2(n-1)} & a_{2n} \\ \vdots & \ldots & \ldots & \ldots & \vdots \\ a_{(m-1)1} & a_{(m-1)2} & \ldots & a_{(m-1)(n-1)} & a_{(m-1)n} \\ a_{m1} & a_{m2} & \ldots & a_{m(n-1)} & a_{mn} \end{pmatrix}, \text{ and } x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}.$$

Note, that c, b and A are prespecified, and x is the vector of unknowns that we have to solve the LP for.

In the following in order to solve the LP, we are going to do a long sequence of rewritings till we reach the optimal solution.

20.2.3 Slack Form

We next rewrite the LP into ***slack form***. It is a more convenient[②] form for describing the **Simplex** algorithm for solving LP.

Specifically, one can rewrite a LP, so that every inequality becomes equality, and all variables must be positive; namely, the new LP will have a form depicted on the right (using matrix notation). To this end, we introduce new variables (***slack variables***) rewriting the inequality

max	$c^T x$
subject to	$Ax = b.$
	$x \geq 0.$

$$\sum_{i=1}^{n} a_i x_i \leq b$$

as

$$\begin{aligned} x_{n+1} &= b - \sum_{i=1}^{n} a_i x_i \\ x_{n+1} &\geq 0. \end{aligned}$$

Intuitively, the value of the slack variable x_{n+1} encodes how far is the original inequality for holding with equality.

In fact, now we have a special variable for each inequality in the LP (this is x_{n+1} in the above example). This variables are special, and would be called ***basic variables***. All the other variables on the right side are ***nonbasic variables*** (original isn't it?). A LP in this form is in ***slack form***.

The slack form is defined by a tuple (N, B, A, b, c, v).

Linear program in slack form.

$$\begin{aligned} \max \quad & z = v + \sum_{j \in N} c_j x_j, \\ \text{s.t.} \quad & x_i = b_i - \sum_{j \in N} a_{ij} x_j \quad for \quad i \in B, \\ & x_i \geq 0, \quad \forall i = 1, \ldots, n+m. \end{aligned}$$

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)
$A = \left\{a_{ij}\right\}$ - The matrix of coefficients
$N \cup B = \{1, \ldots, n+m\}$
v - objective function constant.

Exercise 20.2.2 Show that any linear program can be transformed into equivalent slack form.

Example 20.2.3 Consider the following LP which is in slack form, and its translation into the tuple (N, B, A, b, c, v).

[②] The word *convenience* is used here in the most liberal interpretation possible.

$$\begin{aligned}
\max \quad z &= 29 - \frac{1}{9}x_3 - \frac{1}{9}x_5 - \frac{2}{9}x_6 \\
x_1 &= 8 + \frac{1}{6}x_3 + \frac{1}{6}x_5 - \frac{1}{3}x_6 \\
x_2 &= 4 - \frac{8}{3}x_3 - \frac{2}{3}x_5 + \frac{1}{3}x_6 \\
x_4 &= 18 - \frac{1}{2}x_3 + \frac{1}{2}x_5
\end{aligned}$$

$$\begin{aligned}
B &= \{1,2,4\}, N = \{3,5,6\} \\
A &= \begin{pmatrix} a_{13} & a_{15} & a_{16} \\ a_{23} & a_{25} & a_{26} \\ a_{43} & a_{45} & a_{46} \end{pmatrix} = \begin{pmatrix} -1/6 & -1/6 & 1/3 \\ 8/3 & 2/3 & -1/3 \\ 1/2 & -1/2 & 0 \end{pmatrix} \\
b &= \begin{pmatrix} b_1 \\ b_2 \\ b_4 \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 18 \end{pmatrix} \quad c = \begin{pmatrix} c_3 \\ c_5 \\ c_6 \end{pmatrix} = \begin{pmatrix} -1/9 \\ -1/9 \\ -2/9 \end{pmatrix} \quad v = 29.
\end{aligned}$$

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

20.2.4 The Simplex algorithm by example

Before describing the **Simplex** algorithm in detail, it would be beneficial to derive it on an example. So, consider the following LP.

$$\begin{aligned}
\max \quad & 5x_1 + 4x_2 + 3x_3 \\
s.t. \quad & 2x_1 + 3x_2 + x_3 \leq 5 \\
& 4x_1 + x_2 + 2x_3 \leq 11 \\
& 3x_1 + 4x_2 + 2x_3 \leq 8 \\
& x_1, x_{2,} x_3 \geq 0
\end{aligned}$$

Next, we introduce slack variables, for example, rewriting $2x_1 + 3x_2 + x_3 \leq 5$ as the constraints: $w_1 \geq 0$ and $w_1 = 5 - 2x_1 - 3x_2 - x_3$. The resulting LP in slack form is

$$\begin{aligned}
\max \quad z &= 5x_1 + 4x_2 + 3x_3 \\
s.t. \quad w_1 &= 5 - 2x_1 - 3x_2 - x_3 \\
w_2 &= 11 - 4x_1 - x_2 - 2x_3 \\
w_3 &= 8 - 3x_1 - 4x_2 - 2x_3 \\
& x_1, x_{2,} x_3, w_1, w_2, w_3 \geq 0
\end{aligned}$$

Here w_1, w_2, w_3 are the slack variables. Note also that they are currently also the basic variables. Consider the slack representation trivial solution, where all the non-basic variables are assigned zero; namely, $x_1 = x_2 = x_3 = 0$. We then have that $w_1 = 5$, $w_2 = 11$ and $w_3 = 8$. Fortunately for us, this is a feasible solution, and the associated objective value is $z = 0$.

We are interested in further improving the value of the objective function (i.e., z), while still having a feasible solution. Inspecting carefully the above LP, we realize that all the basic variables $w_1 = 5$, $w_2 = 11$ and $w_3 = 8$ have values which are strictly larger than zero. Clearly, if we change the value of one non-basic variable a bit, all the basic variables would remain positive (we are thinking about the above system as being function of the nonbasic variables x_1, x_2 and x_3). So, consider the objective function $z = 5x_1 + 4x_2 + 3x_3$. Clearly, if we increase the value of x_1, from its current zero value, then the value of the objective function would go up, since the coefficient of x_1 for z is a positive number (5 in our example).

Deciding how much to increase the value of x_1 is non-trivial. Indeed, as we increase the value of x_1, the the solution might stop being feasible (although the objective function values goes up,

which is a good thing). So, let us increase x_1 as much as possible without violating any constraint. In particular, for $x_2 = x_3 = 0$ we have that

$$\begin{aligned} w_1 &= 5 - 2x_1 - 3x_2 - x_3 &&= 5 - 2x_1 \\ w_2 &= 11 - 4x_1 - x_2 - 2x_3 &&= 11 - 4x_1 \\ w_3 &= 8 - 3x_1 - 4x_2 - 2x_3 &&= 8 - 3x_1. \end{aligned}$$

We want to increase x_1 as much as possible, as long as w_1, w_2, w_3 are non-negative. Formally, the constraints are that $w_1 = 5 - 2x_1 \geq 0$, $w_2 = 11 - 4x_1 \geq 0$, and $w_3 = 8 - 3x_1 \geq 0$.

This implies that whatever value we pick for x_1 it must comply with the inequalities $x_1 \leq 2.5$, $x_1 \leq 11/4 = 2.75$ and $x_1 \leq 8/3 = 2.66$. We select as the value of x_1 the largest value that still comply with all these conditions. Namely, $x_1 = 2.5$. Putting it into the system, we now have a solution which is

$$x_1 = 2.5,\ x_2 = 0,\ x_3 = 0,\ w_1 = 0,\ w_2 = 1,\ w_3 = 0.5 \quad \Rightarrow \quad z = 5x_1 + 4x_2 + 3x_3 = 12.5.$$

As such, all the variables are non-negative and this solution is feasible. Furthermore, this is a better solution than the previous one, since the old solution had (the objective function) value $z = 0$.

What really happened? One zero nonbasic variable (i.e., x_1) became non-zero, and one basic variable became zero (i.e., w_1). It is natural now to want to exchange between the nonbasic variable x_1 (since it is no longer zero) and the basic variable w_1. This way, we will preserve the invariant, that the current solution we maintain is the one where all the nonbasic variables are assigned zero.

So, consider the equality in the LP that involves w_1, that is $w_1 = 5 - 2x_1 - 3x_2 - x_3$. We can rewrite this equation, so that x_1 is on the left side:

$$x_1 = 2.5 - 0.5w_1 - 1.5x_2 - 0.5\ x_3. \tag{20.1}$$

The problem is that x_1 still appears in the right size of the equations for w_2 and w_3 in the LP. We observe, however, that any appearance of x_1 can be replaced by substituting it by the expression on the right side of Eq. (20.1). Collecting similar terms, we get the equivalent LP.

$$\begin{aligned} \max \quad z &= 12.5 - 2.5w_1 - 3.5x_2 + 0.5x_3 \\ x_1 &= 2.5 - 0.5w_1 - 1.5x_2 - 0.5x_3 \\ w_2 &= 1 + 2w_1 + 5x_2 \\ w_3 &= 0.5 + 1.5w_1 + 0.5x_2 - 0.5x_3. \end{aligned}$$

Note, that the nonbasic variables are now $\{w_1, x_2, x_3\}$ and the basic variables are $\{x_1, w_2, w_3\}$. In particular, the trivial solution, of assigning zero to all the nonbasic variables is still feasible; namely we set $w_1 = x_2 = x_3 = 0$. Furthermore, the value of this solution is 12.5.

This rewriting step, we just did, is called ***pivoting***. And the variable we pivoted on is x_1, as x_1 was transfered from being a nonbasic variable into a basic variable.

We would like to continue pivoting till we reach an optimal solution. We observe, that we can not pivot on w_1, since if we increase the value of w_1 then the objective function value goes down, since the coefficient of w_1 is -2.5. Similarly, we can not pivot on x_2 since its coefficient in the objective function is -3.5. Thus, we can only pivot on x_3 since its coefficient in the objective function is 0.5, which is a positive number.

Checking carefully, it follows that the maximum we can increase x_3 is to 1, since then w_3 becomes zero. Thus, rewriting the equality for w_3 in the LP; that is,

$$w_3 = 0.5 + 1.5w_1 + 0.5x_2 - 0.5x_3,$$

for x_3, we have

$$x_3 = 1 + 3w_1 + x_2 - 2w_3,$$

Substituting this into the LP, we get the following LP.

$$\begin{aligned} \max \quad z &= 13 - w_1 - 3x_2 - w_3 \\ s.t. \quad x_1 &= 2 - 2w_1 - 2x_2 + w_3 \\ w_2 &= 1 + 2w_1 + 5x_2 \\ x_3 &= 1 + 3w_1 + x_2 - 2w_3 \end{aligned}$$

Can we further improve the current (trivial) solution that assigns zero to all the nonbasic variables? (Here the nonbasic variables are $\{w_1, x_2, w_3\}$.)

The resounding answer is no. We had reached the optimal solution. Indeed, all the coefficients in the objective function are negative (or zero). As such, the trivial solution (all nonbasic 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.

The crucial observation underlining our reasoning is that at each stage we had replace the LP by a completely equivalent LP. In particular, any feasible solution to the original LP would be feasible for the final LP (and vice versa). Furthermore, they would have exactly the same objective function value. However, in the final LP, we get an objective function that can not be improved for any feasible point, an we stopped. Thus, we found the optimal solution to the linear program.

This gives a somewhat informal description of the simplex algorithm. At each step we pivot on a nonbasic variable that improves our objective function till we reach the optimal solution. There is a problem with our description, as we assumed that the starting (trivial) solution of assigning zero to the nonbasic variables is feasible. This is of course might be false. Before providing a formal (and somewhat tedious) description of the above algorithm, we show how to resolve this problem.

20.2.4.1 Starting somewhere

$$\begin{aligned} \text{Max} \quad & z = v + \sum_{j \in N} c_j x_j, \\ \text{s.t.} \quad & x_i = b_i - \sum_{j \in N} a_{ij} x_j \quad for \quad i \in B, \\ & x_i \geq 0, \quad \forall i = 1, \ldots, n+m. \end{aligned}$$

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 a linear program (in slack form depicted on the left. 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' from L. This LP $L' =$ **Feasible**(L) is depicted on the right. 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 **LPStartSolution**(L') denote this easily computable feasible solution.

$$\begin{array}{ll} \min & x_0 \\ \text{s.t.} & x_i = x_0 + b_i - \sum_{j \in N} a_{ij} x_j \quad for \quad i \in B, \\ & x_i \geq 0, \quad \forall i = 1, \ldots, n+m. \end{array}$$

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

Lemma 20.2.4 *The LP L is feasible if and only if the optimal objective value of LP 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. ■

One technicality that is ignored above, is that the starting solution we have for L', generated by **LPStartSolution**(L) is not legal as far as the slack form is concerned, because the non-basic variable x_0 is assigned a non-zero value. However, this can be easily resolve by immediately pivoting on x_0 when we run the **Simplex** algorithm. Namely, we first try to decrease x_0 as much as possible.