

26. Linear Programming Part I (Background & Theory)

26.1 Introduction to Linear Programming

Linear programming is a central part of a field called Operations Research. Developed after the Second World War, its applications range from supply chain management, to air traffic control. In particular, the Sloan School of Business at MIT has the best Operations Research department in the United States. If you decide that this topic interests you, I would encourage you to sign up on the Sloan Lottery and take 15.764, “The Theory of Operations Management.”

Linear programming is a bit of a misnomer, since it is not what we commonly consider to be “programming.” Linear programming, in the context of operations research, involves solving certain optimization problems, which are called linear programs. It is also commonly utilized in the fields of applied sciences, finance, and economics.

A linear program represents a problem in which you are asked to find the maximum value of a linear function of variables, subject to linear **constraints**.

The following is a simple example of a linear program.

Suppose we have two variables x_1 and x_2 , and we wish to maximize the value of the linear function $.5x_1 + .1x_2$. Both variables obey the following constraints:

1. Both variables are non-negative
2. The function $3x_1 + 2x_2$ can have a value of at most 4
3. The function $x_1 + x_2$ can have a value of at most 2

As explained above we are maximizing a linear function with linear constraints.

The linear function we wish to maximize will be called the **objective function**.

Constraints on x_1 and x_2 could be that $x_1 < 8$ and $x_2 = 50$, and also x_i is greater or equal to 0.

26.2 Standard Mathematical Form of a Linear Program Problem.

We have n variables $\{x_k\}$ (for $k=1 \dots n$)

The value of each variable is non-negative.

We have m constraints, each of the form

$$A_{j1}x_1 + A_{j2}x_2 + \dots + A_{jn}x_n \leq b_j.$$

We seek to find the maximum value of an objective function of the following form

$$V_0 + V_1 X_1 + V_2 X_2 + \dots + V_n X_n.$$

Besides the form described above, there are a multitude of other forms of linear programs. For instance some of our variables may not have to be non-negative, so they can take on any real value. Some of our inequalities might actually be equalities.

Such variations do not make the problem more difficult, in fact it becomes a bit easier for each such occurrence. However we have to introduce notation to describe such variables and constraints. For the moment, we will ignore this and introduce the notation later.

When the variables have the additional constraint that each (or some) of the x_j must be an integer, the problem is called an **integer linear program** or a **mixed linear program**.

In some cases the fractional parts of solution values for the variables are uninteresting and can be ignored. Typically when the value of the solution variables are large, fractions are unimportant. In other cases they are important, and integer or mixed linear programs can be much harder to solve than ordinary linear programs.

26.3 The General Resource Allocation Problem.

We begin by considering some examples of problems that give rise to linear programs. In our discussion we will see some approaches to solving linear programs, and some of their curious properties.

Suppose you are engaged in some sort of a manufacturing process. That is, you put together various raw materials into objects that you make. The raw materials are called **inputs**. These raw materials can be bulk items, parts, wrappings, usages of machinery, time needed by those who perform the manufacture, and so on.

Suppose further that you have the capability of manufacturing a number of different products using the resources available to you. The classic example of several resources producing different products would be using metal and oil to produce guns and butter. The metal and oil can be used to manufacture either guns or butter or both. The amount of metal and oil needed to create guns is different than the proportion of metal and oil needed to create butter. The amount of guns and butter produced depends on the allocation of metal and oil to either manufactured good.

You are then confronted with the problem “how much of each possible product should you produce in a given time period?”

This problem, in its simplest form is a linear program.

Let us index the raw materials or resources from 1 to m, and the potential products from 1 to n.

Let A_{jk} represent the amount, in appropriate units, of resource j that will be used in producing one unit of product k, and let x_k represent the number of units of product k that you might produce.

With the assumption that you cannot unmake your products and that you can not have negative amounts of inputs, you have the constraint.

$$x_k \geq 0 \text{ for all } k.$$

Furthermore, we can assume that you have a limited capacity in any given time period for each of your resources; if the amount of resource j available to you in the period under consideration is b_j , you have, for each resource j, the constraint...

$$\sum_{k=1}^n A_{jk} x_k \leq b_j.$$

Each resource has a cost associated with it, and your finished products each have "sale" values that you can estimate for them. Let v_k represent the value of a unit of product k beyond the total cost of the resources that go into its manufacture. In other words, v_k is the value added by creating it out of its constituent parts.

It makes sense that in order to maximize the utility of your efforts, you want to maximize the following.

$$\sum_{k=1, n} v_k x_k,$$

subject to the constraints previously mentioned.

In practice there are various complications that we will mention here before continuing but not discuss in detail until later.

First, in real life there are uncertainties both in the valuation of your products and the replacement values of your raw materials, as well as in the functioning of your processes. This means that there is a wide berth of uncertainty in all the parameters here that must be considered in practice. We ignore this for the purpose of simplicity in the course.

Second, as stated above, x_k represents a potential number of units of product k. When your product k is a discrete item, and a large one in particular, you have to consider the additional constraint that x_k must be an integer. In many circumstances the optimal x_k will be large enough or the integer condition does not exist, in which case you can safely ignore this additional constraint. We assume so here.

Third, there may be other costs associated with your manufacturing process, such as transition costs on your machines in shifting from one product to another, which are not included in the model described above, but which can affect the optimal policy. In any given situation it may well be possible to model such costs but we will not talk further about them here. Fixed costs, including overhead, rent, and initial capital costs are easy to model here because they just add a constant to the cost of the enterprise.

Finally, in many cases production is geared to orders on hand, or are dictated by priorities other than value added. These may or may not be modeled in similar ways.

Linear Programs arise in many other contexts, and there are variations in them from the standard form exhibited in this model.

In particular, some of the variables may take on negative values in a sensible way, and some of the constraints may be equalities rather than inequalities, and the resulting problem is still called an LP or linear program. We shall encounter at least one such problem later.

There are many other practical problems which can, under favorable circumstances, be modeled as linear programs. Flows of commodities in networks, assignments of jobs to machines, are examples of problems that are linear programs.

26.4 Basic Properties and Forms for describing a Linear Program

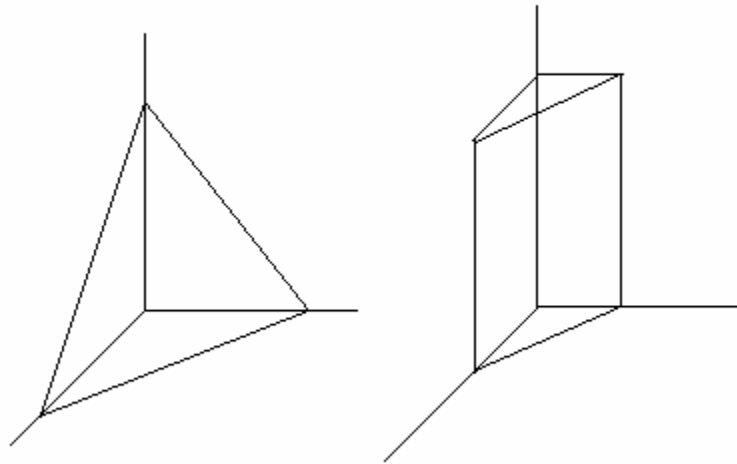
In the description of an LP that we start with, we have $n + m$ **hyperplane** constraints in an n -dimensional space, each of which confines solutions to one side of it, and we have a linear objective function that we want to maximize subject to these conditions. Hyperplanes are $n - 1$ dimensional surfaces in n dimensional space. Hyperplanes are defined by a single linear equation. These hyperplanes outline the edges of our constraints. Think of a plane in these dimensions and you have a reasonable idea of a hyperplane in n dimensions.

Let us call a 0-dimension intersection of any of these hyperplanes a **vertex**, and a 1-dimensional intersection a **line**. Since the requirement that a point lies on a single hyperplane cuts the dimensions down by at most 1, ordinarily n hyperplanes define a vertex and $n-1$ determine a line. It can also be possible that more than n hyperplanes pass through a vertex. This vertex is then said to be **degenerate**.

The **feasible** or allowed region, is the set of points of our n dimensional space that obey all of our $n + m$ constraints.

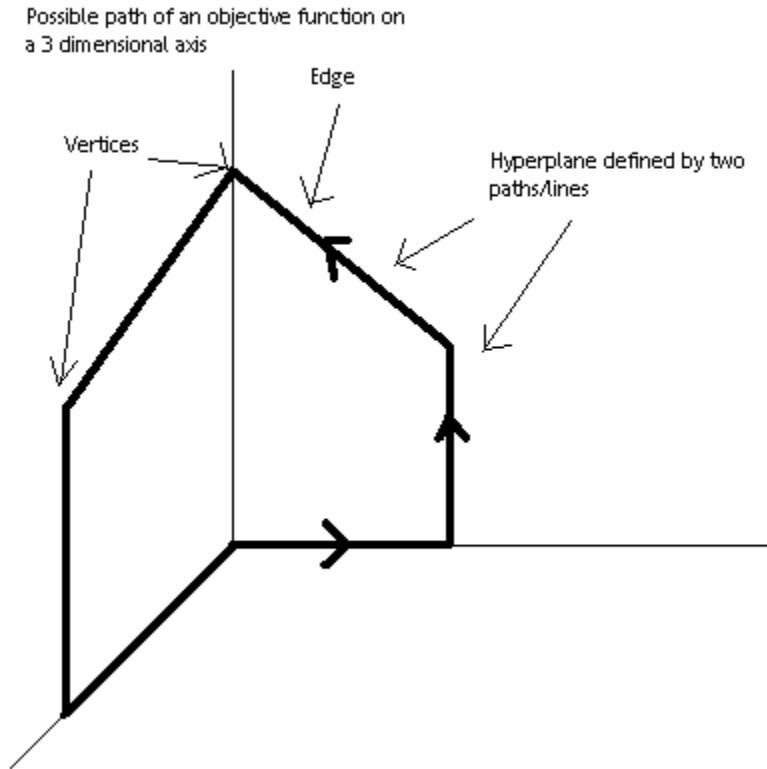
This region does not have to exist at all, since our constraints could contradict one another. However, it often does exist in practice, particularly when we are modeling a real system, and when it does, it normally consists of some n dimensional convex polygon or **polytope**, which means a region bounded by hyperplanes.

Examples of Possible Polytopes



The region is **convex**: which means that if two points lie in it, any point in between does as well. This is trivially true for the region on one side of each constraint, which is called a **halfspace**, so it is true for our feasible region as well: (because a statement like this that is true in both of two regions is also true in their intersection, and our feasible region is the intersection of halfspaces)

The feasible region has **vertices**, and **edges**. The edges are segments of lines between two of its vertices. The hyperplanes defining the edge are the hyperplanes that pass through both of its end vertices.

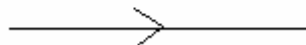


A feasible region can as well have infinite edges that are half lines if the region is infinite in extent.

Not all vertices are vertices of our feasible region. To be a vertex of the feasible region requires being the intersection of enough constraint hyperplanes to determine a point, and *that point must lie on the allowed side of each of the other constraints as well.*

The first fundamental property of this system is that *we need only look for maxima for our objective function on vertices of the feasible region.* This is not to say that maximum cannot occur elsewhere. It means only that if a maximum occurs anywhere it occurs on a vertex as well. This statement is a consequence of the linear nature of the objective function. If the objective function decreases when you move in one direction it will increase in the opposite direction.

Objective function increases in this direction
and decreases in opposite direction



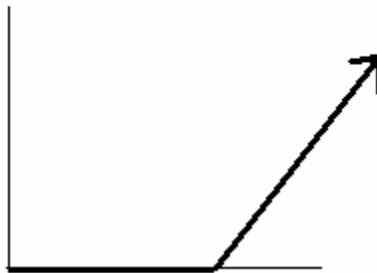
Therefore if you have a maximum on a face of the feasible region with dimension at least 1 that is not a vertex, there is a direction that you can move either forward or back and still stay on that face.

If you are at a maximum, the objective function must be constant in that direction, so you can move as far as you can (until you reach a new constraint), which will take you to a lower dimensional face, without changing the objective function.

If you keep doing this you will eventually arrive at a vertex, and the objective function will still be the maximum it was at first, which proves the statement.

If your feasible region is unbounded it is possible that the objective function grows without limit along some edge, in which case it has no maximum value.

Example of an unbounded objective function.



Our problem, as we have seen, is described by an m by n matrix, A_{jk} , an m -vector $\{b_j\}$ of bounds, and an n -vector $\{v_k\}$ of values, and consists of the problem of maximizing

$$\sum_{k=1, n} v_k x_k$$

subject to the constraints that for each k and j we have

$$x_k \geq 0, \text{ and } \sum_{k=1, n} A_{jk} x_k \leq b_j.$$

Each constraint here can be described by a hyperplane in the n -dimensional space of our n -vectors, namely the hyperplane on which it holds as an equality. The constraint then requires that any solution must lie on one side of the hyperplane.

Thus the constraint $x_1 \geq 0$ can be described by the statement that x_1 must lie to the right of the hyperplane $x_1 = 0$, which means that x_1 must be positive.

We can write all the constraints here in exactly this form by introducing slack variables for each of the m constraints above. We define **the slack variable** s_j by the equation:

$$(\sum_{k=1,n} A_{jk}x_k) + s_j = b_j,$$

whereupon our constraint takes the form $s_j \geq 0$. So slack variables are what we need to add to our constraint inequalities to make them into equalities. This we can visualize our tableau as being described by $m \times n$ inequality constraints in n dimensions or by $n + m$ dimensions.

In this written form we have $n + m$ variables, each either an x_k or an s_j , all of which must be positive and these are subject to m equations as just described.

This formulation of the problem is symmetric between our original variables x and slack variables s , but as you can see, the way our equations are written, it is not symmetric in this way.

In particular, each s occurs in exactly one equation, with coefficient 1, and does not occur in the objective function ($\sum (v_k x_k)$) at all.

We now discuss the second fundamental property of systems of equations like this.

*Given a set of m equations among variables as we have here, any linear combination of these equations is also an equation. We can describe the same equation set by choosing any m linearly independent linear combinations of these equations. In other words these linearly independent equations form a **basis** for all other possible linear combinations in a **vector space**.*

In particular, we can single out other choices of m of our $n + m$ variables x and s and arrange it such that *each one of the chosen variables appears in exactly one equation, and we can normalize that equation so that the given chosen variable has coefficient 1 in it.*

If we do so we will have changed nothing except the way we choose to describe our equations: which is the basis we choose to use for our vector space of equations.

The form of the equations initially presented to us allows us to deduce the values of the s variables and therefore of all the variables immediately at the **origin** of the x variables. At that vertex all the x variables are 0 and we have $s_j = b_j$ for each j .

When we choose a different set of m variables to solve our equations for, we associate the vertex at which all the other non-chosen variables are 0 to that form of the equation.

We can then relate the geometric problem of moving from vertex to vertex, along an edge, on a feasible polytope to the algebraic problem of changing of chosen variables. Both can be used to describe how to achieve the maximization of a function.

- Jonathan Lii