THE GEOMETRY OF LINEAR PROGRAMMING

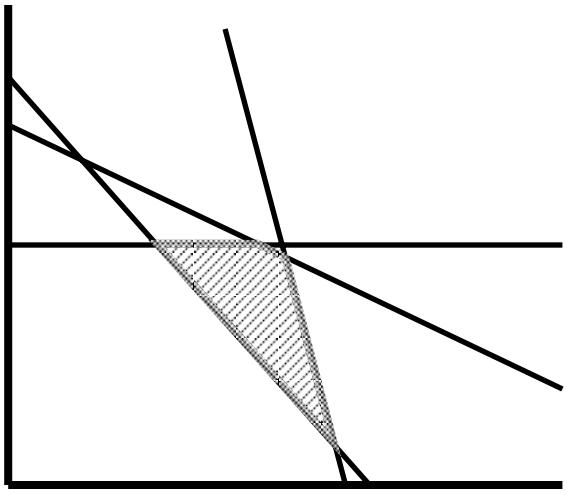
INTRODUCTION

Understanding the relationship between geometry (or algebraic topology) and mathematical programming is critical to being able to apply the methods of mathematical programming effectively. This review note will define ten common terms in detail and illustrate their role in the programming solution. Also, we will consider three frequent problem areas in dealing geometrically with linear programming problems. Some of this note will be part geometry review, some will be part algebra review, and some will be part microeconomics. All of these areas, however, are essential if one is to fully understand the processes underlying linear programming.

10 IMPORTANT TERMS

Frontier

By frontier, what is meant is the boundary of the feasible region. Often, frontier is taken to mean the portion of the boundary which lies in the appropriate direction of the optimal solution. In standard linear programming problems, the solution is guaranteed to lie on the frontier. Determining the optimal point along the frontier involves "pushing" the isocost curve in the direction of optimality (either towards the origin or away from the origin) until it passes the frontier and leaves the feasible region. The last point in the feasible region touched (which is on the frontier) is the optimal point.

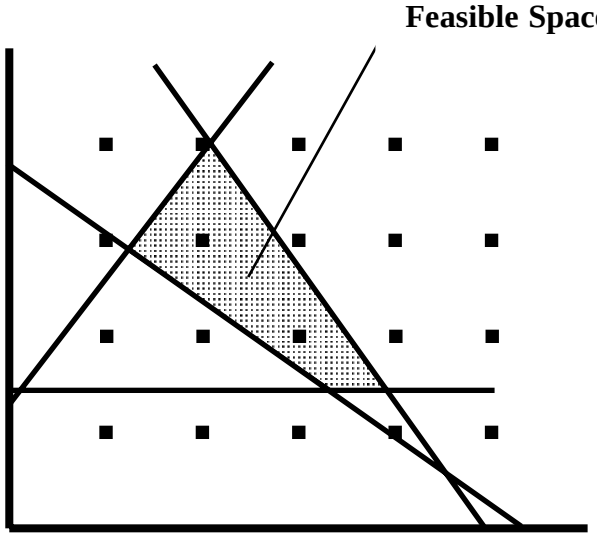


The Frontier

Grid

The term grid becomes important if we are forced only to consider integer-valued solutions or constraints. In standard applications of linear programming, we are able to use the far more (mathematically) convenient solution area of real-valued space, $\mathbb{R}$. However, if integer solutions are required, the solution space becomes a grid or a set of points. The space between the points is not

feasible in these cases. We can only consider points which are both in the feasible region *and* are elements of the grid. For example, in the first quadrant (*q.v.*), a grid with constraints may appear as follows:



Note that there are only two grid points within the feasible space and both are relatively far from the boundary. Thus, the imposition of an integer constraint in this problem is likely to be a costly one. Given such a constraint, the optimal solution to this problem must be either one of the two grid points in the feasible region (as determined by the isocost (*q.v.*) curve).

At this point, it may be valuable to discuss some of the number theoretic and set theoretic symbols and terms you will see in this area. Among other classifications, numbers can be grouped and labeled according to the following distinctions:

Type	Symbol	Description
Natural Numbers	$\mathbb{N}$	Counting Numbers, Positive Integers, {1, 2, 3, ...}
Integers	$\mathbb{I}$, $\mathbb{Z}$	Whole Numbers, {..., -3, -2, -1, 0, 1, 2, 3, ...}
Complex Numbers	$\mathbb{C}$	The even roots of negative numbers, given in the form $a+bi$ with $i^2=-1$
Real Numbers	$\mathbb{R}$, $\boldsymbol{R}$, $\Re$	All noncomplex numbers

With these categories, we use the following symbols to indicate membership in a category or set:

Symbol	Meaning
$\cap$	Intersection: the set of elements contained by both groups
$\cup$	Union: the set of elements contained by all groups

$\subset$	Proper Subset: R is a proper subset of S iff R is contained within S and $R \neq S$
$\subseteq$	Subset: R is wholly contained within S
$\in$	Element Inclusion: the element is contained by the set
$\notin$	Element Exclusion: the element is not contained by the set
$\not\subset$	Subset exclusion: the set is not a subset
$\varnothing$	The null set; the set which contained no elements

With the above terminology, we may succinctly make the following claims:

$$\mathbb{N} \subseteq \mathbb{I} \subseteq \mathbb{R}$$

$$\mathbb{R} \cap \mathbb{C} = \varnothing$$

$$\text{if } A = B, \text{ then } A \subseteq B \text{ and } B \subseteq A$$

These symbols also allow us to carefully specify the characteristics of solutions or inputs in linear programming problems. For example, if the units of some good, *X*, must be integers or values on a grid, we may represent this by stating that $X \in \mathbb{I}$ or $X \in \mathbb{N}$ meaning that *X* must either be an integer, or, if *X* must be positive, that *X* must be a natural number. You may also see such statements made in set notation using the universal quantifier symbol $\forall$, which means "for all". Thus, to indicate that a solution holds only for positive whole numbers (or the natural numbers or $\mathbb{N}$) we could indicate:

$$\forall X \in [1, \infty) \qquad \textit{("For all X in the interval from 1 to infinity")}$$

The brackets here indicate the inclusion of boundaries. You may think of them as expressing the nature of the inequality. Square brackets, [ ], indicate that the boundary is included in the solution. In the above case, 1 is a valid solution. Rounded brackets, ( ), indicate that the boundary value is *not* contained within the solution. So, for example, $(3, 7]$, would indicate the range of numbers not including 3, but including 7.

The use of these concepts and symbols allows us to succinctly state the format of the variables in problems. It is not as necessary that you be able to use them. However, you will see them often in linear programming (and especially in integer programming) problems and it is important that you know what they mean.

Intercept

The intercept of a line is the point at which the line crosses the axes of the graph. Thus, in our problems, we will have two intercepts: the *x*-intercept and the *y*-intercept. These points are important because they are often the two points used to draw the lines representing constraints or isocurves. In the basic equation of the line, $y = mx + b$, b is the *y*-intercept. It is the value which occurs on the *y* axis (because *x* is set to zero).

However, in linear programming problems, we often face constraints given in the following form:

$$6x + 9y = 36$$

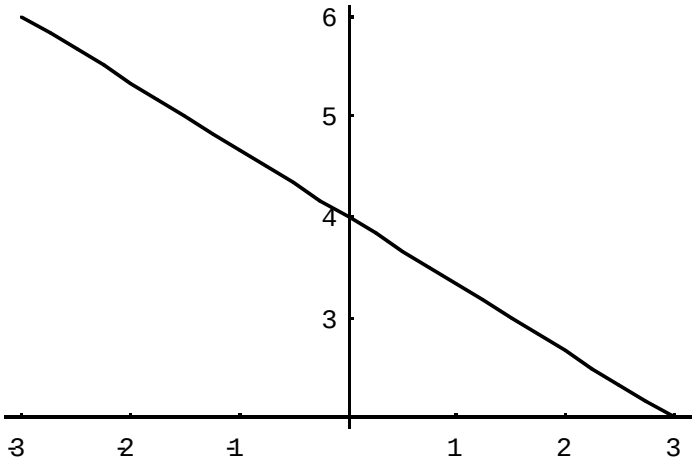
This form could represent some combination of ingredients required to meet the minimum output or the total labor cost if x and y were hours worked and 6 and 9 were wages. In this case, the x-intercept is 6 and the y-intercept is 4. These are the values one gets from solving for x (y) when y (x) is set to zero. When one variable is zero, the remaining variable must lie on the axis.

If we take the above equation, it is easy to see if it is recast in the equation of the line format:

$$6x + 9y = 36$$
$$9y = -6x + 36$$
$$y = -\frac{2}{3}x + 4$$

$$\therefore \quad m = -\frac{2}{3} \text{ and } b = 4$$

Further, we can plot this equation and observe that, in fact, 4 is the point of intersection with the y axis.



Intersection

The intersection of two lines is the point at which they cross or meet. When dealing with linear equations, it is the point which allows a solution to be simultaneously obtained for both x and y. In linear programming, this is important because the optimal solutions almost always occur at the vertex (*q.v.*) or the intersection of two lines.[1]

To solve for the intersection, first obtain the value of one variable in terms of the other. Two lines will intersect so long as they have different slopes. Often, this is given to you if the equation is expressed in the form $y = mx + b$. Then, substitute this value in the second equation. You now have an equation in one variable. Obtain the solution for that variable and substitute that solution into one of the original equations. For example, take the equation we used in the last section, written in the standard form:

[1]The only other case is if the optimal solution *is* the line or line segment rather than a unique point.

$$y = -\frac{2}{3}x + 4$$

Also, take the following equation, $6x + 4y = 12$, in the standard format and find their intersection.

$$y = -\frac{3}{2}x + 3$$

Since we have already solved for *y*, we can use that relationship to solve directly for *x*.

$$-\frac{2}{3}x + 4 = -\frac{3}{2}x + 3$$

$$6 \cdot \left(-\frac{2}{3}x + 4 = -\frac{3}{2}x + 3\right)$$

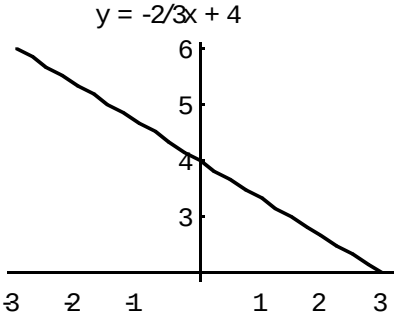
$$-4x + 24 = -9x + 18$$

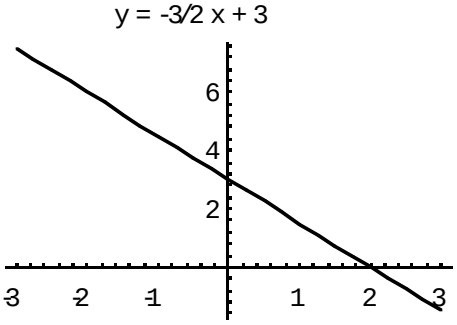
$$5x = -6$$

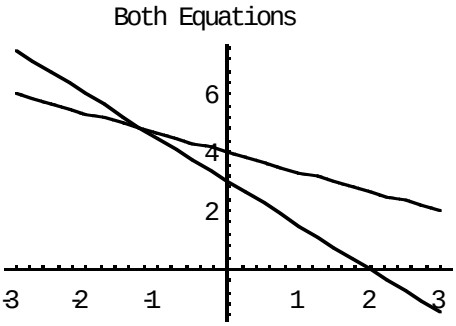
$$x = -\frac{6}{5} \quad \bigstar$$

$$\therefore \quad y = -\frac{2}{3} \cdot -\frac{6}{5} + 4 = \frac{24}{5} \quad \bigstar$$

Therefore, the two lines intersect at the point $\left(-\frac{6}{5}, \frac{24}{5}\right)$ or (-1.2, 4.8). By plotting the lines, we can observe that this is actually where the intersection occurs:







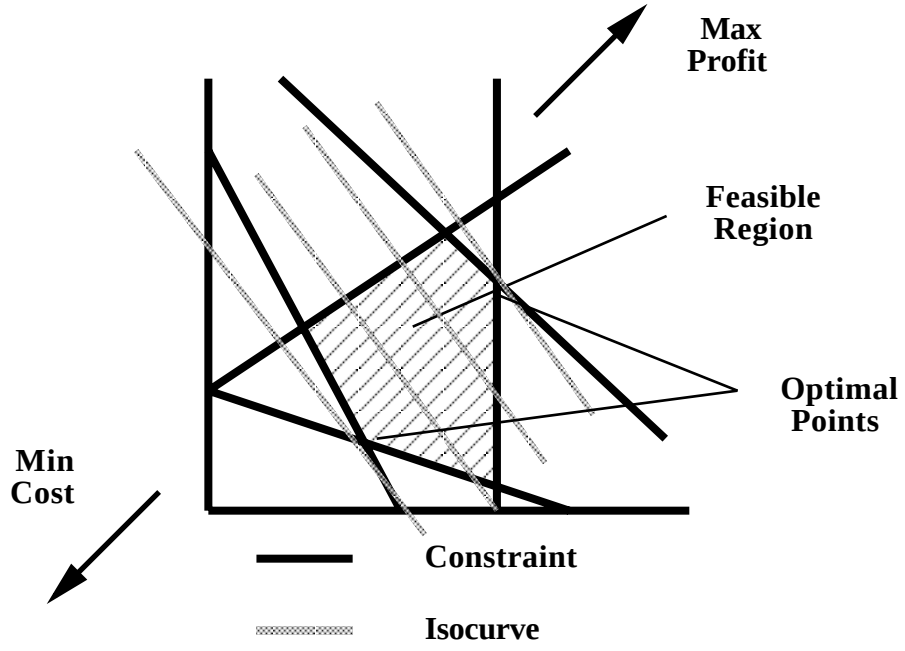
Isocurves

Iso as a prefix means "equal". The isocurves represent the path along which the factor in question possesses a constant value. For example, consider the various combinations of selling prices for two goods which give a firm equal profits. To see this, assume that a firm's profit is represented solely by the prices of the two goods it sells, *X* and *Y*. To make a profit of \$10, the firm could set prices a number of ways:

X = \$10 and Y = \$0
X = \$9 and Y = \$1
X = \$8 and Y = \$2
...
X = \$0 and Y = \$10

Thus, the line along which these values fall is an isoprofit curve (if the figures were costs, it would be an isocost curve) given by $y = x$. What this means is that any points along an isocurve is an optimal solution so long as it appears within the feasible region. The goal is to push the isocurve as far towards the optimal direction as possible while retaining at least one optimal point. Because linear programming problems (and programming problems in general) often have oddly shaped feasible spaces, it is frequently the case that the furthest feasible isocurve intersects (*q.v.*) with only 1 point.

It is important to keep in mind that there exist an infinite number of isocurves - only one is optimal. The isocurve which contains the optimal solution is the one which intersects the point (or set of points) which are the last to exit the feasible set. Observe.



As illustrated by the two points identified as optimal, if this were a minimization problem, the optimal point would be the intersection of the constraint and the isocurve closest to the origin. If it were a maximization problem, the optimal point would be the intersection of the constraint and the isocurve furthest from the origin. Any combinations of the inputs which lie on these lines will produce identical profits or costs.

Lines and Regions, Planes and Spaces

The linear programming problems we are concerned with in this class typically involve two variables which must be adjusted to produce the maximum profit or minimum cost. In actuality, mathematical programming is capable of solving problems with virtually unlimited numbers of variables. However, because it is often difficult for us to conceive of objects with more than two or three dimensions, our use of the graphical method is restricted to two variables (and thus lines).

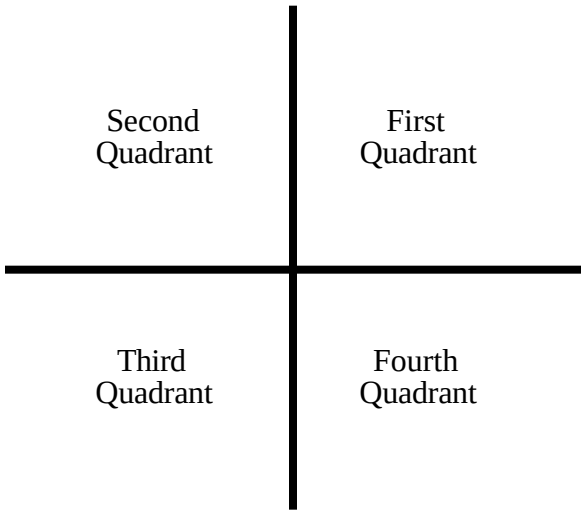
A **line** is formally the set of all points (x, y) which satisfy a given linear equation, $ax + by + c = 0 \quad \forall\, a, b \neq 0$. Thus, a line is continuous to infinity in either direction but does not possess width. On the other hand, a plane is a line which possesses width. A **plane** has area. A plane is a surface such that a straight line joining any two of its points lies entirely in the surface. The equation of a plane incorporates a third dimension, *z*, and is given by $ax + by + cz + d = 0$ $\forall\, a,\ b,\ c \neq 0$.

A **region** is a set that is the union of an open connected set along with none, some, or all of its boundary points. If a set includes its boundary points it is a **closed region**. The inclusion of boundary points is an important issue (see Topic #3 below) in solving programming problems. A **space** is simply a three-dimensional region.

Formally, linear programming problems with two variables are concerned only with lines and regions (constraints are lines bounding the feasible region). When more than two variables are involved, more complicated topological spaces are required and we must speak of constraints as planes (known as *hyperplanes*) bounding optimal spaces. In practice, however, the terms are often used interchangeably - especially region and space. You will often see people speak of feasible sets, feasible regions, and feasible spaces regardless of the number of variables actually being used. Still, it is important to keep in mind the true definitions of those terms and recognize the differences between them.

Quadrant

Programming problems involve solutions and variables which are always contained within $\mathbb{R}$. Within this space, however, we can further divide the areas based on the sign taken by numbers within certain regions of the graph. This is done by defining four quadrants on the set of real numbers in two variables, *x* and *y*.



Pairs of numbers which are contained within each quadrant take on different signs depending on which side of the origin they find themselves on. They take the following distinctions:

First Quadrant	(+, +)
Second Quadrant	(–, +)
Third Quadrant	(–, –)
Fourth Quadrant	(+, –)

This is important in linear programming because we can often restrict ourselves to only considering the first quadrant through the use of feasibility constraints. It is often the case that negative numbers have no meaning. For example, one cannot produce negative units of some tangible good or work a negative number of hours. It is advantageous to reduce the size of the problem space whenever possible. This is particularly true in complex problems involving integer, nonlinear, or dynamic programming since these problems often involve semi-probabilistic search algorithms. Limiting the size of the problem space can greatly reduce the computational time required to find a solution.

Shadow Price

The shadow price is a price which is implicitly (not market) determined or determined as a consequence. They represent the marginal values of the constraints or the opportunity costs, per unit, of the constraints. Mathematically, they arise from the relationship between the primal and the dual in the linear programming and can be solved for analytically by way of the simplex algorithm.

Shadow prices are especially useful in postoptimality analysis because they can be used to measure slack utilization. Shadow prices can be used to analyze the change in the objective function given some increase or decrease in the various constraints. For example, say a current constraint is that no more than 5 employees may be used to create a certain product. What would the change in the overall cost be if that constraint was relaxed to 6 employees? The shadow price of labor would be able to tell us how much costs would change by and thus could be used in evaluating the efficiency of hiring an additional employee.

Clearly, this information is important to have. However, there are usually limits to how large the changes in the constraints can be. We may be interested in establishing the **permissible range** within which the shadow prices accurately represent price changes. Also, it is often the case that a constraint may have a zero shadow price. This is because slack is only valuable if the current optimal solution makes full use of the resources. If the current optimal solution above, for example, didn't use all 5 employees, the value of adding a sixth would be zero.

Now we have two limits on the usefulness and range of shadow prices. First, there is often a maximum size of the change they are accurate for. This is called the permissible range. Second, there may be no shadow price at all. If the constraint in some resource is non-binding, that resource has no marginal value. Each of these pieces of information is valuable to managers. The permissible range establishes boundaries on sensitivity analysis and can help shape future strategic plans by directing resources where the results are clear rather than on areas outside of the permissible range where the forecasts are unknown. If the shadow price of a constraint is zero, that means that managers should not be devoting additional resources to that area. In fact, it may cause managers to rethink the importance of that resource in the production of the good or service.

Slope

The slope of a line is the degree to which the line is "not horizontal". That's a crude definition, but descriptively accurate. A perfectly horizontal line has a slope of 0. More correctly, however, the slope of a line is the tangent of the angle that the line makes with the positive *x*-axis or the rate of change of the ordinate with respect to the abscissa:

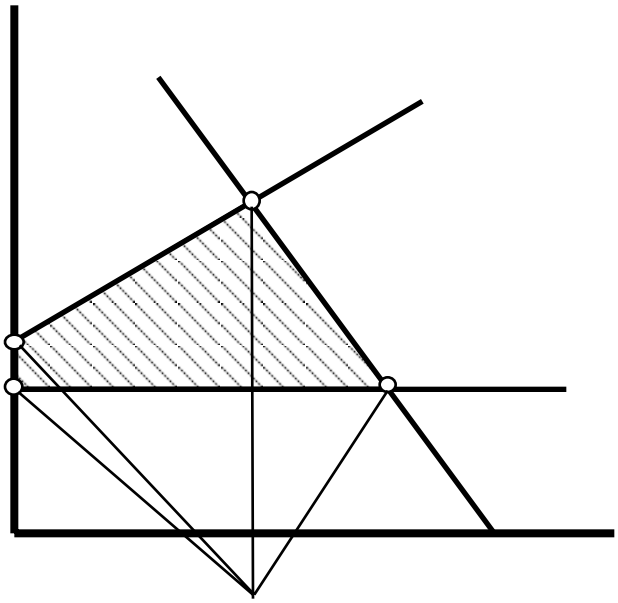
$$\frac{y_2 - y_1}{x_2 - x_1}$$

Note, however, that the slope of a perfectly vertical line ($x_2 = x_1$) is not defined (but division by zero is sometimes taken to be infinite). In the standard form, the slope if represented by m.

In linear programming, the slope is important for the same reasons the isocurves are important: the optimal solutions have identical slopes (given certain parameter changes in the problems). The most common mistake made with regard to slopes is that people often reverse the positions of the first and second points. This causes them to invert the sign of the slope, labeling the slope positive when it should be negative and vice versa. Make sure you choose the correct points in calculating your slope. Also, check the number you get. It is trivial to see (in two dimensions) that your slope is accurate. Plot it out and make sure it heads in the correct direction!

Vertex

A vertex represents the corner point or intersection of two lines. Corner points are important in linear programming because unless an entire line or line segment is optimal, the optimal solution (if it exists) is guaranteed to lie at a vertex. Thus, a possible solution algorithm would simply be to test each of the corner points or vertices and select this one with the highest (lowest) value. In practice, the simplex algorithm, follows a procedure similar to this. In solving these problems by hand, however, such methods are often impractical. As the number of constraints increases, so does the number of vertices. For complicated problems, there could be hundreds or thousands of constraints. In fact, if we allow for more than two variables, complicated problems could have not only thousands of constraints but thousands of variables. The increase in the dimensionality of the problem makes testing each vertex impractical.



The Vertices of the Feasible Region

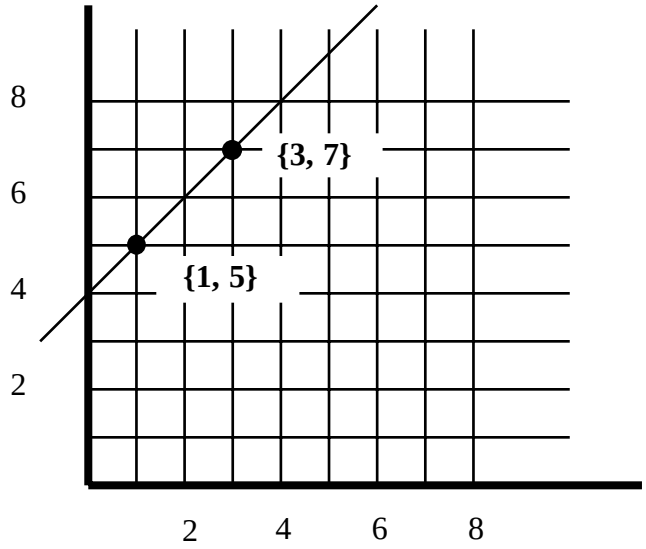
Still, it may be advantageous to use the positions of the vertices as initial guesses when calculating a solution - particularly when under time pressure (such as on an exam). Unfortunately, too often in the past students have thought that simply being able to test the vertices on the exams was sufficient. In actuality, the exams will frequently ask you to solve a programming problem explicitly by using the isocurves (or some other method such as branch and bound, etc.). Therefore keep in mind that testing the vertices is a quick method, but not the "correct" method in many cases. Know how to use the isocurves to identify the optimal solutions, but also know when you can easily substitute the vertex test method instead.

COMMON PROBLEMS ENCOUNTERED WITH LINEAR PROGRAMMING

Problem #1: Plotting Points, Drawing Lines

Surprisingly, many people have problems in setting up the problem visually. Regardless of whether this stems from forgetting their geometry classes or from carelessness, it is a costly error. Often the hardest part of solving programming problems is setting them up correctly. The actual mathematical calculations involved in solving the problems you will encounter in this class are trivial. That said, this first problem considers the task of drawing lines given a series of points, drawing lines given an equation of the line, drawing lines of a given slope, and deriving the equation of the line from a set of points.

Given any two points, you can draw a line. Further, there is only one line which can be drawn through any two points. Therefore, once you have two points on a graph, you can draw a line connecting those two points and also solve for the equation of that line. First, let's simply consider drawing a line. Let's assume the two points we are given are $\{3,7\}$ and $\{1,5\}$. To draw the line, simply plot the two points and draw a straight line through both points connecting them.



Next, we are interested in identifying the equation of that line. Recall that the equation of the line is given by $y = mx + b$. Merely from observing the above graph, what can we observe? Well, we know that the line intersects the *y*-axis at 4. This is *b*. Also, we now know the values of all the (x,y) combinations on that line. We can use those values to solve for the slope, *m*. Recall from the definition we introduced earlier in this note that the slope is given by:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Therefore, we have

$$m = \frac{5-7}{1-3} = \frac{-2}{-2} = 1$$

The slope in this problem is equal to 1. Now we know everything about this line. We have its defining equation:

$$y = x + 4$$

The more commonly encountered scenario in linear programming is that you will be given the equation of a line and need to plot it. This, too, is a simple procedure. On the other hand, the information is often *not* given in terms of the equation of the line. Frequently you are given an equation such as:

$$6x - 2y = 4$$

Now, you can use this equation to plot a line. But, by converting all equations into the standard equation of the line format, it is far easier to observe the slope and the intercept. Identifying these parameters is the most critical part of characterizing and drawing a line.

$6x - 2y = 4$	*The original equation*
$2y = 6x - 4$	*Isolate the LHS of the equation*

$y = 3x - 2$ *Divide through by 2 to solve for* y

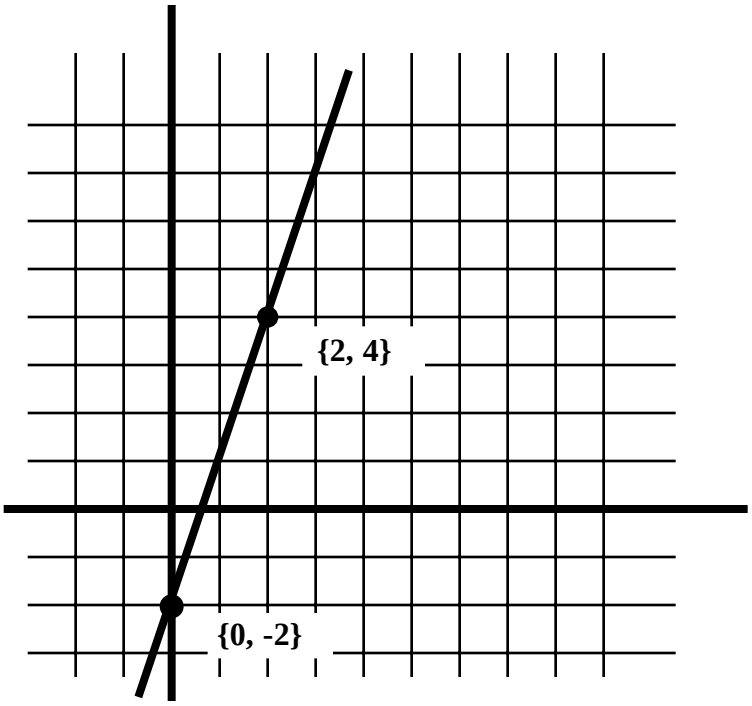
From here, it's easy to see that the slope of this line is 3 and the *y*-intercept is -2. Armed with this data, we can quickly draw the line. To draw the line, we know from above that all we need are two points. We already have one because we know the *y*-intercept. Therefore, we know the point (0, -2) is on the line. Now we only need one additional point. This can be obtained by simply picking *any* value for *x* and plugging it in to the equation. Let's take *x* = 2.

$$x := 2$$
$$y = 3 \cdot 2 - 2$$
$$y = 6 - 2$$
$$y = 4$$

We have our second point: (2, 4). From here we can check that our calculation of the slope above is accurate:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{4 - (-2)}{2 - 0} = \frac{6}{2} = 3$$

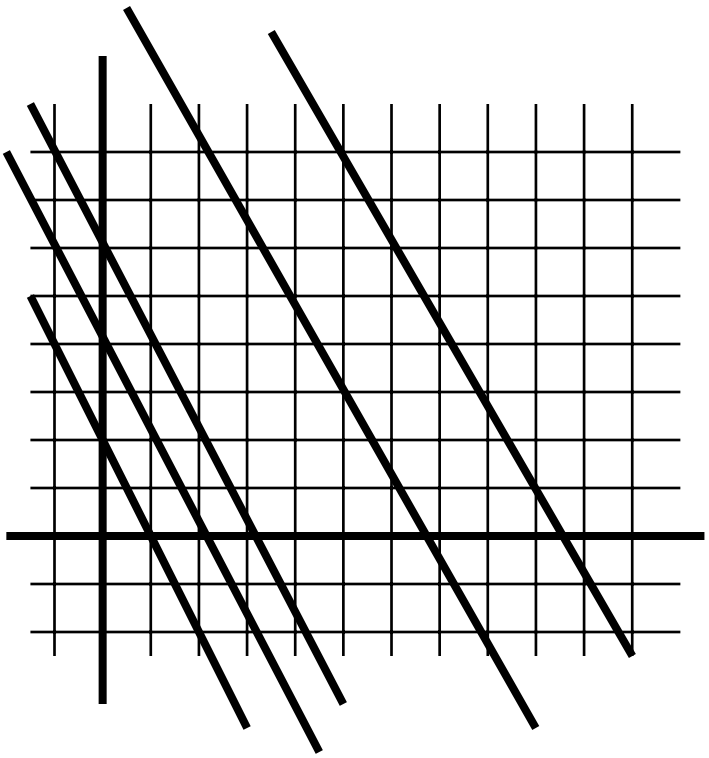
So we've correctly calculated the slope. We have all the information we need to draw this line. Simply plot the two points and connect the points with a straight line.



The third type of line we may be interested in drawing is one for which we are only given the slope of a line. This situation may arise in linear programming when we are solving for the optimal point by using isocurves. We are looking for a line going through a certain point with the same slope as the isocurve. Also, for sensitivity analysis we may be interested in the effect of relaxing a constraint on the optimal solution. In this case, the slope of the isocurve remains

constant. Only the location of that isocurve will change. Thus, given a change in the constraints (within the permissible region) we are looking to identify the new point lying along the same slope.

We will consider the case of drawing a line with a slope of -2. The problem is that there exist an infinite number of lines with a slope of -2. Which should be draw? The answer depends entirely on the specifics of each problem. Each of the lines will differ only in where that line intersects the *y*-axis (represented by the *b* value). Thus we attempt to fit $y = -2x + b$ to the problem at hand. The thing to remember is that the important element is fitting the line to the problem, not solving for *b* and then applying that solution to the problem. In a very real sense, the actual value of *b* is irrelevant. The slope is the critical factor in evaluating changes in the optimal solution.



Each of these lines has a slope of -2. The only difference between them is the location of their *y*-intercept. The way to draw these lines is to simply plug various combinations of numbers for *x* and *b* into the given equation and solve for *y*.

Problem #2: Solving for Isocurves

The process involved in drawing the isocurves for a particular problem is easy. In nearly every case, the formula is given to you. Once you know the objective function of the problem (what you are maximizing or minimizing) you can easily calculate the isoprofit or isocost curve. First, realize that there exist an infinite number of isocurves for any given problem. The goal is to find the one "most optimal" which contains points satisfying the constraints. Let's work with an example. Assume that *x* and *y* represent labor hours for two workers. A manager is interested in minimizing the total cost of labor, *C*, which is a function of the hourly wages paid to each worker and the number of hours worked. Assume that *x* makes $5 per hour and *y* makes $10 per hour. What is the objective function of this problem?

Clearly, this is a minimization problem. We know that the manager wants to minimize the

total amount being paid for labor. This amount is equal to the following:

$$\bar{C} = 5x + 10y$$

So, the manager's problem is

$$\min_{x,\,y} \ \bar{C} = 5x + 10y$$

In some sense, the isocost (in this case) curves represent iterative attempts to arrive at the optimal solution. We calculate the isocost curves by simply guessing (carefully) at what the optimal total cost will be and working backwards. Thus, we are working with the objective function by arbitrarily choosing *C*. In this problem, we are looking for the isocost curve closest to the origin which still possesses a point within the feasible region: smaller costs are better. To begin with, let's set the total cost to $50.

$$5x + 10y = 50$$

This equation should look familiar. It's the equation of the line in one of the traditional formats. Let's rearrange the terms to get the standard equation of the line so we have the slope and intercept:

$$\begin{aligned} 5x + 10y &= 50 \\ 10y &= -5x + 50 \\ y &= -\frac{1}{2}x + 5 \end{aligned}$$

The slope is $-\frac{1}{2}$ and the intercept is 5. That information, particularly the slope, is very important. Even though we don't *exactly* know the optimal solution, we now know that the optimal solution must lie on a line with a slope of $-\frac{1}{2}$.

At this point, we need to plot this isocost curve and examine its location. Does it intersect a region with feasible points? If not, it's not even in consideration. If it does, are there still feasible points closer to the origin? If so, we need to adjust the total cost and solve again. If not, then we have identified the optimal solution as the intersection of the isocost curve and the constraint along which the furthest point lies. If we do need to adjust the isocost curve, we only need to draw a parallel line closer to the origin. We have no need to re-solving for the equation of the line with a different value for *C*. The point of solving for the isocurves is to gradually and continually partition the feasible region until only 1 point remains - the optimal point. This partitioning is done by "pushing" the isocurves in the optimal direction (towards or away from the origin) by drawing new curves parallel to some curve initially drawn with a guess.

Problem #3: Boundary Distinctions

One of the most frequent mistakes made on exams and homework assignments in linear programming stems from carelessness about the treatment of boundaries. The problem simply comes down to the differences in the relations $\{<, >\}$ and $\{\leq, \geq\}$. The distinction may seem subtle in practice, but is actually quite important. The importance stems from the notion of indifference which is implicit in the second set of relations, $\{\leq, \geq\}$. In problems which ask for the most a person would pay for a certain good or the fewest number of hours a person would consider working, it frequently becomes necessary to precisely define the limit point.

If we say $x < 10$, we mean that any amount up to, but not including 10 is admissible. In many models, if 10 were included, it would dramatically change the value of the solution. In basic linear programming problems, this may seem a picky distinction. After all, is 10.00000000001 all that different from 10? Well, perhaps not. However, in integer programming, the distinction is quite important. In these cases, 10 is often vastly different than 10.00000000001. One solution is feasible. The other is not. Likewise, in applications of nonlinear programming or programming with discontinuous constraint or objective functions, the distinction is very important. Consider finding a number measured as $\sqrt{x}$. The difference between 0 and -0.0000000001 is insignificant under comparative conditions. Here, however, one value is real, the other complex.

It should not be inferred, however, that this problem is only important in esoteric or degenerate cases! Similar situations arise constantly when considering sensitivity or postoptimality analysis of standard linear programming problems. Often questions are phrased as: What would the value of _____ need to become before it was included in the optimal solution? How low would _____ need to be before the slack went to zero?

The confusion arises from the fact that although the constraints are given in terms of weak or strict inequalities, once they are graphed, they are often treated as equalities - as lines. Then, when it comes time to consider variation in their values, the original *in*equalities are ignored. The sensitivity problems are solved for the points at which there is indifference (=) because they are mathematically easier to deal with. Then, they are arbitrarily adjusted afterwards to become ≥ or ≤. The problem is that using ≥ or ≤ still indicates that indifference (=) is acceptable in the solution. *The questions, however, ask for the point at which behavior changes*. This point is given by $x + \varepsilon$ or $x - \varepsilon$, values strictly greater (>) than or less (<) than *x*. Make sure when you are solving for these types of problems that after you find a numerical solution, you check that the solution and its boundary conditions make sense in the scheme of the problem. Is there indifference in your answer? Should it be there?

David Rode
Decision Analysis and Decision Support Systems
Department of Social and Decision Sciences
Carnegie Mellon University
June 4, 1997