Chapter 3: Towards the Simplex Method for Efficient Solution of Linear Programs

The simplex method, invented by George Dantzig in 1947, is the basic workhorse for solving linear programs, even today. While there have been many refinements to the method, especially to take advantage of computer implementations, the essential elements are the same as they were when the method was invented. During the Second World War, calculations were carried out on manual hand calculators by rooms full of clerks. Fortunately today, these calculations are done much more rapidly and accurately by digital computers.

The simplex method has a poor theoretical efficiency, but actually performs extremely well in practice. It is the method of choice for a broad range of small to large problems, but the newer *interior-point methods* are preferred for extremely large problems. The simplex method also has the advantage of making sensitivity analysis easier.

The goal in this chapter is to give you an understanding of the basic mechanics of the method, and to show you why it works. This will equip you to deal with the inevitable unexpected results when you are solving linear programs in practice later in your career.

Some Definitions

Here is some fundamental vocabulary for talking about linear programs when they are expressed in a graphical manner (refer to Figure 3.1 for examples):

- **solution:** any point in the variable space. Could be feasible (satisfies all constraints), or infeasible (violates at least one constraint).
- **cornerpoint solution:** anywhere two or more constraints intersect. Again, such a point might be feasible or infeasible.

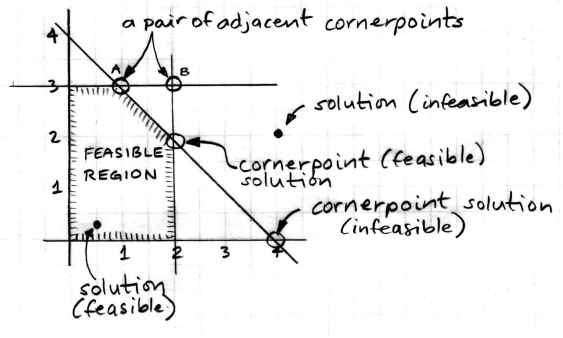

Figure 3.1: Vocabulary for graphical representations of LPs.

- **feasible cornerpoint solution:** a cornerpoint solution that is feasible. As we will see later, the simplex method is *very* interested in feasible cornerpoint solutions.
- **adjacent cornerpoint solutions:** two cornerpoint solutions that are connected by a single constraint line segment are adjacent cornerpoint solutions. Again, there are no assumptions about feasibility.

Key Properties of Linear Programs

Here are the three key properties of linear programs that drive the design of the simplex method:

1. **The optimum point is always at a feasible cornerpoint.** As we saw previously (see Figure 2.3), this is a by-product of the fact that all of the constraints and the objective function are linear. Remember that there can also be multiple optima, but this will happen only when at least two of the optima are adjacent cornerpoint feasible solutions.
2. **If a cornerpoint feasible solution has an objective function value that is better than or equal to all adjacent cornerpoint feasible solutions, then it is optimal.** Again, this is a by-product of modeling with lines.
3. **There are a finite number of cornerpoint feasible solutions.** This means that any method which concentrates on looking only at cornerpoint feasible solutions, as the simplex method does, will eventually terminate. Good news!

Think about point 1 for a moment: is it possible to have 3 or more cornerpoints having the same optimum value? Sure, but you have to go to higher dimensions (i.e. problems with more variables). Imagine a problem in 3 dimensions in which the feasible region resembles a cube. Now imagine that the "slope" of the objective function exactly matches the "slope" of one of the faces of the cube. In this case, there will be 4 cornerpoints all having the exact same optimum value of the objective function!

These properties have tremendous practical impact. Property 1 means that you only need to look at cornerpoints, rather than at the infinite number of points in the feasible region, which is a tremendous boost in efficiency. Property 2 means that you can easily recognize when you have found the optimum point, so you don't have to look at all of the feasible cornerpoints, another boost in efficiency. Finally, property 3 means that the method is guaranteed to terminate, so you *will* get an answer.

We now have enough information to provide a bird's-eye view of the simplex method. It has two main phases:

1. **Phase 1 (start-up):** find *any* cornerpoint feasible solution. The reason why we start our study of linear programming with standard form LPs is that the origin (the (0,0,…,0) point) is *always* a cornerpoint feasible solution in a standard form LP, so phase 1 is simple. Phase 1 is more complicated for non-standard form LPs, and requires a special method, which we will cover later.
2. **Phase 2 (iterate):** repeatedly move to a better adjacent cornerpoint feasible solution until no further better adjacent cornerpoint feasible solutions can be found. This final cornerpoint feasible solution defines the optimum point. Note that there could be other adjacent cornerpoint feasible solutions with the *same* optimum value.

As shown in Figure 3.2, the sequence of feasible cornerpoints visited in the Acme Bicycle Company problem is as follows:

1. *Phase 1:* choose the origin (0,0). The value of the objective function at that point is zero, expressed as $Z(0,0)=0$.
2. *Iteration 1:* move from (0,0) to (2,0). $Z(2,0)=30$, an improvement.
3. *Iteration 2:* move from (2,0) to (2,2). $Z(2,2)=50$, an improvement.
4. *Stop:* the only feasible cornerpoint not yet visited that is adjacent to (2,2) is at (1,3), and $Z(1,3)=45$. So (2,2) with $Z=50$ is better than both of the adjacent feasible cornerpoints: $Z(2,0)=30$ and $Z(1,3)=45$, so (2,2) is the optimum point, and the best achievable value of the objective function is 50.

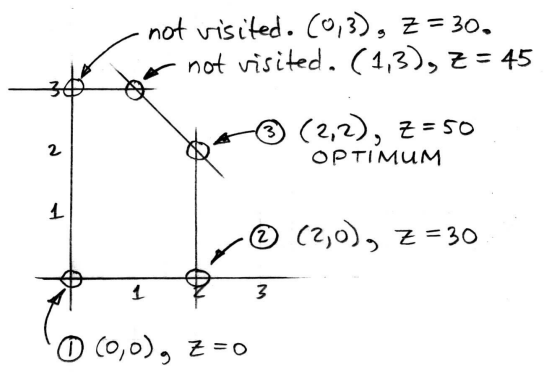

Figure 3.2: Visiting cornerpoints in the Acme Bicycle Company problem.

This means that the Acme Bicycle Company should produce mountain bikes at the rate of 2 per day and racers at the rate of 2 per day to achieve a maximum profit rate of $50 per day.

As we will see later, the simplex method does not actually need to visit all of the feasible cornerpoints adjacent to the optimum point. There is a simple way to recognize that there are no better adjacent feasible cornerpoints left to visit.

Finding Cornerpoints Algebraically

Graphical representations of linear programming problems are only used for teaching the principles; real problems have hundreds, thousands, even millions of variables. For problems at that scale, we need an algebraic way to find the cornerpoints. For standard form LPs, the answer lies in converting the inequalities to equations, and then solving for the intersection of a subset of the equations. There are well-developed methods for finding the intersection of linear equations, such as Gaussian elimination.

Note that we must solve for the intersection of a *subset* of the equations because in the usual case not all of the equations derived from the original inequalities can hold simultaneously. So, in addition to converting the inequalities to equations, we need a way to keep track of which of the equalities are currently selected, or *active*. The resolution of both difficulties lies in the addition of *slack variables* to the inequalities to convert them to equations. For example:

$x_1 \leq 2$ becomes $x_1 + s_1 = 2$ when the nonnegative slack variable s_1 is added.

Slack variables are so named because they "take up the slack" between the left hand side of the equation (in this case just x_1) and the right hand side value. The nonnegativity of the slack variable is essential to this role.

Converting the Acme Bicycle Company to equality format in this way yields the LP shown in Figure 3.3. The problem, originally in two dimensions (x_1 and x_2), is now a problem in five dimensions (x_1, x_2, s_1, s_2, s_3). Note that the slack variables take on positive values only when the constraint that they appear in is *not* active.

	original variables		slack variables			
$Z =$	$15x_1$	$+ 10x_2$				
	x_1		$+ s_1$			$= 2$
		x_2		$+ s_2$		$= 3$
	x_1	$+ x_2$			$+ s_3$	$= 4$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

Figure 3.3: Equality version of the Acme Bicycle Company problem.

There is some special terminology when working with the algebraic, equation-converted version of the original LP model, as follows:

- **augmented solution:** the values for all of the variables are given, including both the original variables and the slacks. For example, at the optimum solution to the Acme Bicycle Company problem, the augmented solution is $(x_1,x_2,s_1,s_2,s_3) = (2,2,0,1,0)$.
- **basic solution:** an augmented *cornerpoint* solution (could be feasible or infeasible). In the ABC problem, (2,3,0,0,-1) is a basic solution that happens to be infeasible.
- **basic feasible solution:** an augmented cornerpoint *feasible* solution. In the ABC model, (0,3,2,0,1) is a basic feasible solution.

As you might imagine, the simplex method is mostly concerned with basic feasible solutions.

Setting the Values of the Variables

Somehow we need to set the values of the variables, and this should be done in such a way that we arrive at a feasible cornerpoint, or basic feasible solution. Consider the ABC problem in which we have 5 variables after conversion to equation format, but only 3 constraints. This must mean that we can set the value of 2 of the variables arbitrarily, and then calculate the values of the other 3 using the equations.

The number of variables whose values can be set arbitrarily is known as the *number of degrees of freedom (df)* of a problem. In general,

df = (number of variables in equation format) – (number of independent equations)

As we will see later, the simplex method will automatically set the values of *df* of the variables, and then solve for the values of the others. In fact, simplex will set those *df*

variables to a value of zero. Why zero? Because this implies that the corresponding constraint is *active*, i.e. right on it's limiting value and actively preventing the solution process from venturing into the infeasible zone. Let's see if this is true for the ABC problem. Considering the equation form of the problem as in Figure 3.3:

- x_1=0 implies that you are on the limiting value of the $x_1 \geq 0$ constraint.
- x_2=0 implies that you are on the limiting value of the $x_2 \geq 0$ constraint.
- s_1=0 implies that you are on the line x1=2.
- s_2=0 implies that you on the line x_2=3.
- s_3=0 implies that you are on the line x_1+x_2=4.

Because *df*=2 in the ABC company, that means that simplex will set the values of two of the 5 variables to zero. In other words, this will make two of the constraints active, and it will thereby define a cornerpoint where these two constraints cross. Figure 3.4 shows that the selection of which two variables are set to zero can define a feasible, or perhaps an infeasible cornerpoint.

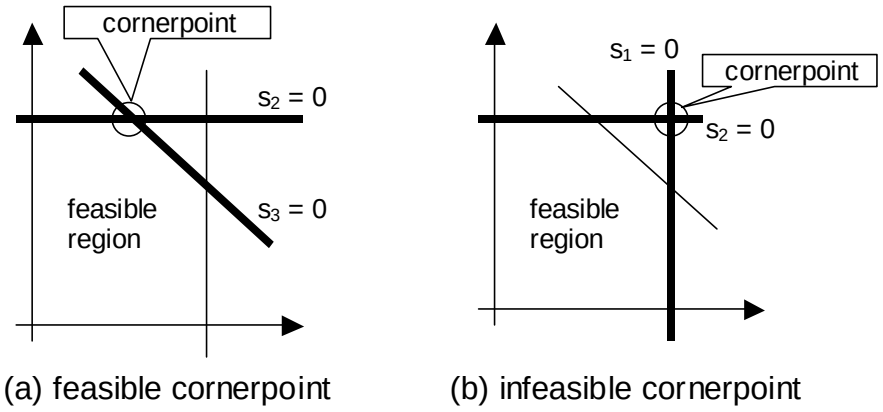

Figure 3.4: Setting variables to zero defines a cornerpoint.

There are two final items of terminology at this point:

- **nonbasic variable:** a variable currently set to zero by the simplex method.
- **basic variable:** a variable that is *not* currently set to zero by the simplex method. Basic variables normally have nonzero values (positive in the case of standard form LPs), but they can also be zero in special circumstances. This means that you can't reliably tell which variables are zero and which are nonzero just by looking at their current values.
- **a basis:** as simplex proceeds, the variables are always assigned to the basic set or the nonbasic set. The current assignment of variables is called the basis.

As we develop the simplex method, we will be working intensively with this idea of variables being set to zero and thereby activating constraints. You should memorize the

following mantra, and chant it to yourself or work it into casual conversations at every opportunity:

Nonbasic, variable set to zero, corresponding constraint is active.

The simplex method is all about deciding what the current basis is, i.e. which variables are currently basic and which are currently nonbasic. In fact, if you were exceedingly lucky, you could solve LPs directly just by guessing which variables should be basic and which nonbasic. This then defines the active set of constraints, so you can solve for the intersection point of the active constraints to find the optimum point and the value of the objective function at that point. But there are some pitfalls if you guess incorrectly! Figure 3.4 (b) shows how you could select a basis that defines an infeasible cornerpoint. Figure 3.5 shows how you might even select a basis that does not define a cornerpoint at all.

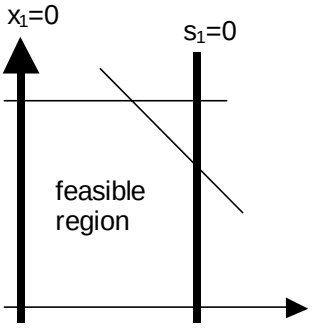

Figure 3.5: A basis that does <u>not</u> define a cornerpoint.

Moving to a Better Basic Feasible Solution

Given that phase 1 provides us with a basic feasible solution to start from, the simplex method then just needs to move to a better basic feasible solution, and to continue doing this until the stopping conditions (no better adjacent cornerpoint feasible solution) are met. As we've seen above, choosing the next basis cannot be done at random: you might choose an infeasible basis, a worse basis, or a basis that does not define a cornerpoint at all.

It turns out that if you are at basic feasible solution (feasible cornerpoint), then the easiest basic feasible solution to find next will be adjacent. This is partly due to the following property of adjacent cornerpoints:

- in two adjacent cornerpoints, the nonbasic sets will be identical, except for one member
- in two adjacent cornerpoints, the basic sets will be identical, except for one member.

For example, consider the two adjacent cornerpoints labeled in Figure 3.1 (these are the same cornerpoints shown in Figure 3.4):

- point A: nonbasic set = $\{s_2, s_3\}$, basic set = $\{x_1, x_2, s_1\}$
- point B: nonbasic set = $\{s_1, s_2\}$, basic set = $\{x_1, x_2, s_3\}$

This seems to imply that we can move from one cornerpoint to the next simply by swapping a pair of variables between the basic and the nonbasic sets. Unfortunately, this condition is necessary, but *not* sufficient for adjacency. For example, the pair of cornerpoints at (0,4) and at (4,0) also has this property, but those cornerpoints are not

adjacent. So there are three conditions that must be taken care of while moving from one cornerpoint to the next:

- the cornerpoints must be adjacent,
- the cornerpoints must be feasible,
- the new cornerpoint must have a better value of the objective function than the current cornerpoint.

The simplex method has an impressively clever yet simple way of making sure that all three conditions are satisfied. There are two steps:

1. Determine which nonbasic variable (remember! nonbasic variables are set to zero) will increase the objective function value most swiftly if allowed to take on a positive value. Move this variable from the nonbasic set to the basic set. This is called the *entering basic variable* because it is entering the basic set.
2. Allow the entering basic variable to increase only until one of the basic variables is forced to a value of zero. Move the variable that is forced to zero from the basic set to the nonbasic set, where it will retain its value of zero. This is called the *leaving basic variable* because it is leaving the basic set.

How do we determine the variable that most swiftly increases the value of the objective function in step 1? At the origin, this is simply done by looking at the objective function. For the ABC problem, $Z = 15x_1 + 10x_2$, so it is immediately obvious that x_1 provides the swiftest rate of increase in Z, because it increases Z by 15 per unit increase in x_1, where x_2 increases Z by only 10 per unit increase in x_2. So if we are currently at the basic feasible solution at the origin point (0,0), then x_1 is chosen as the entering basic variable. At other basic feasible solutions we will work with rewritten versions of the objective function, but the basic principle is the same.

This first step does two things. First, moving a variable out of the nonbasic set inactivates one of the equations (remember! nonbasic – set to zero – constraint active). Second, because that variable is going to the basic set, we know that it is now allowed to increase in value, so this gives us a direction in which to move. Figure 3.6 shows the effect of starting at the origin and choosing x_1 as the entering basic variable. The dashed line indicates the position of the x_1=0 constraint, which has just been released.

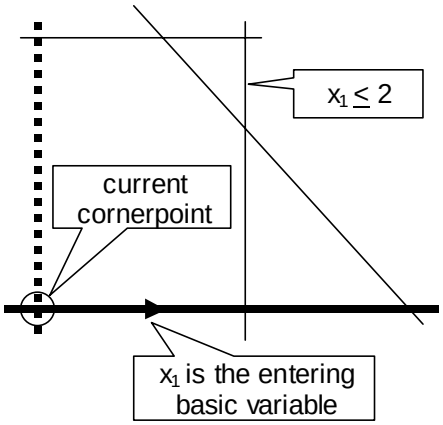

Figure 3.6: Choosing x_1 as the entering basic variable.

Step 2 then tells us when to *stop* increasing the value of the entering basic variable. We can see in Figure 3.6 that the correct place to stop increasing x_1 is when we bump into the limiting value of the $x_1 \leq 2$ constraint, otherwise we will move out of the feasible region. But how do we algebraically detect that we have reached the point at which x_1=2, i.e. the point at

which the $x_1 \leq 2$ constraint becomes active? The answer is that the variable associated with the equation form of $x_1 \leq 2$ is driven to zero before any other variable.

Look at Figure 3.6 again. At the origin, the situation is this:

- basic variables: s_1, s_2, s_3
- nonbasic variable: x_2
- entering basic variable: x_1

Let us consider the effect of increasing the entering basic variable value on each of the constraint equations, as in the Table 3.1. Basic variables are underlined in the table. Note that there is exactly one basic variable in each of the constraint equations. Its value at the origin is shown.

basic variable	**constraint equation**	**bound on increase in x_1**
$\underline{s_1} = 2$	$x_1 + \underline{s_1} = 2$	$x_1 \leq 2$
$\underline{s_2} = 3$	$x_2 + \underline{s_2} = 3$	no limit
$\underline{s_3} = 4$	$x_1 + x_2 + \underline{s_3} = 4$	$x_1 \leq 4$

Table 3.1: Finding the leaving basic variable.

Considering the first equation in Table 3.1, you can see that as x_1 increases, s_1 decreases to keep the equation satisfied. Because s_1 must be nonnegative, x_1 can only increase until s_1 becomes zero. s_1 reaches a value of zero when x_1 reaches a value of 2. Because s_1 is the only basic variable in that equation, it is the only one whose value you have to worry about. Similarly, s_3 is the only basic variable in the third constraint equation, so it is the only variable that can compensate for an increase in the value of x_1 by decreasing its own value (x_2 is nonbasic, and remember! nonbasic variables are set to zero). s_3 reaches a value of zero when x_1 reaches a value of 4. Because x_1 does not even appear in the second constraint equation, that constraint places no limit on the increase in x_1.

Table 3.1 shows that it is the first constraint equation (based on $x_1 \leq 2$) that most limits the increase in x_1. When the basic variable in that equation (s_1) is driven to zero, the constraint associated with s_1 ($x_1 \leq 2$) becomes active. As you can see in Figure 3.6, this is just the constraint to activate to prevent straying out of the feasible region. Hence s_1 is chosen as the leaving basic variable.

At the new cornerpoint defined by the intersection of the limiting values of the two constraints $x_2 \geq 0$ and $x_1 \leq 2$, i.e. the point (2,0), the basis is this:

- basic variables: x_1, s_2, s_3
- nonbasic variable: x_2, s_1

Compared to the basis at the origin point, this represents a swap of x_1 and s_1 between the basic and nonbasic sets.

Finding the leaving basic variable via the method demonstrated in Table 3.1, as required in step 2, depends on having exactly one basic variable per constraint equation, and the coefficient of the basic variable being exactly 1. As we will see later, the simplex method makes sure that these conditions are met. Once this condition is met, we need only to look at the value of the following ratio to find the limiting value placed on the entering basic variable by a particular constraint equation:

$$\frac{\text{right hand side value of constraint equation}}{\text{coefficient of entering basic variable in the equation}}$$

Because we are always looking for the most limiting bound on the entering basic variable, we need the *smallest* value of this ratio, so the process of finding the leaving basic variable in this manner is called the *minimum ratio test.* The coefficients of the entering basic variable will not always have a value of 1 as they do in Table 3.1: that's an artifact of this simple example.

There are two special cases of the minimum ratio test, as illustrated in Figure 3.7:

- **if the coefficient of the entering basic variable is zero:** this means that the constraint in question does not intersect with the still-active constraints represented by the remaining nonbasic variables, so it places no limit on the increase in the entering basic variable.
- **if the coefficient of the entering basic variable is negative:** this means that the constraint in question does intersect with the still-active constraints, but the direction of increase of the entering basic variable is *away* from the intersection point. Hence the constraint in question places no limit on the increase in the entering basic variable.

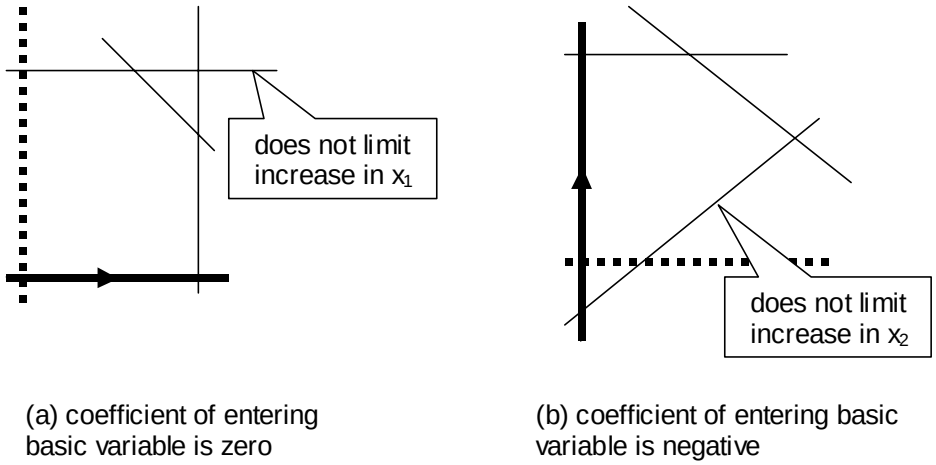

Figure 3.7: Why zero or negative coefficients of the entering basic variable do not limit its increase.

Finding the New Basic Feasible Solution

Now that the new basis is known, how do you actually find the point associated with the next basic feasible solution? A possible, but inefficient, method is to set the nonbasic variables to zero and then solve the remaining $m \times m$ system of linear equations, perhaps by Gaussian elimination.

A much more efficient method is to *update* the current set of equations using just a little bit of Gaussian elimination. This step can put the equations back into the format needed for applying both the test for selecting the entering basic variable (objective function must be rewritten), and the minimum ratio test for selecting the leaving basic variable (constraint equations must have exactly one basic variable each, and the coefficient of that basic variable must be 1).

We will hear more about this in the next chapter, which introduces the simplex tableau, which is a way of formalizing the steps in the simplex method.

A related question is this: how do we know when to stop iterating? This happens when we are unable to find an entering basic variable. In other words, in the updated version of the objective function, there is *no* nonbasic variable that, if allowed to become positive, would increase the value of *Z*. This means that we don't actually need to visit the next cornerpoint and see that the value of the objective function has worsened: we will already know that there is no improving direction to go in! Details follow in the next chapter.