Chapter 5: Solving General Linear Programs

So far we have concentrated on linear programs that are in *standard form*, which have a maximization objective, all constraints of ≤ type, all of the right hand side constants are greater than or equal to zero, and all of the variables are restricted to nonnegative values. As we saw in Chapter 2 (see Figure 2.5), a standard form LP has one big advantage: the origin [the point at (0,0,0,…)] is always a feasible cornerpoint, so the simplex method can always start there. The simplex method then happily proceeds from cornerpoint to better cornerpoint until it recognizes optimality. But the simplex method is in trouble if it can't find that initial cornerpoint to start at.

All forms of LPs arise in practice, not just standard form LPs. You may have minimization objectives, constraints of the ≥ or = type, variables which can take negative values, and right hand side constants which are negative. How can we deal with these forms? That is the subject of this chapter.

Minimization

There are two simple ways to deal with a minimization objective function. The easiest and most popular method is to simply multiply the minimization objective function by -1, and then to *maximize* the resulting function. For example, suppose your objective function is:

$$\text{minimize } Z = 12x_1 + 5x_2 - 7x_3$$

then convert to a maximization:

$$\text{maximize } (-Z) = -12x_1 - 5x_2 + 7x_3$$

As a reminder that the original objective function was a minimization, put a –1 in the Z column of the simplex tableau. After solving the linear program, recover the solution as follows:

- multiply the optimum objective function value by –1 to recover the minimum value of the original minimization objective function.
- the values of the variables as taken from the tableau will be correct.

The second and less popular alternative to dealing with minimization objectives is to keep the minimization objective function in place, unaltered, but to change the rules of tableau manipulation, as follows:

- new test for optimality: are all of the coefficients in the objective function row *nonpositive*? If yes, then stop iterating: you've reached the optimal solution.
- new choice for entering basic variable: choose the *most positive* coefficient in the objective function row.

For most students, it is easiest to use the first option because the rules of tableau manipulation remain unchanged. This is the method that we will assume from now on.

Equality Constraints

Suppose that the Acme Bicycle Company problem included an equality constraint, as below:

$$\begin{aligned} &\text{maximize } Z = 15x_1 + 10x_2 \\ &\text{subject to:} \quad x_1 \le 2 \\ &\qquad\qquad\quad\; x_2 \le 3 \\ &\qquad\qquad\quad\; x_1 + x_2 = 4 \leftarrow \textit{note change to equality constraint!} \end{aligned}$$

As you can see in Figure 5.1, the origin is no longer a feasible cornerpoint at which to start the simplex method. There doesn't seem to be any easy way to find an initial basic feasible solution. How do we get the simplex method started?

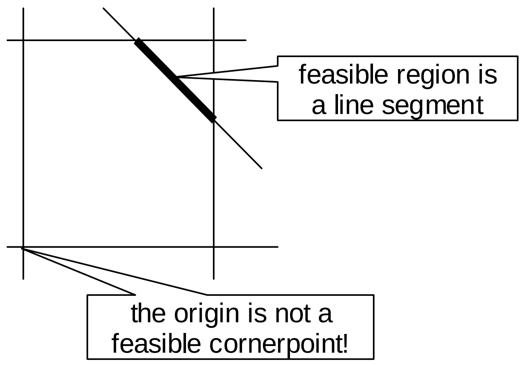
feasible region is a line segment
the origin is not a feasible cornerpoint!

Figure 5.1: An equality constraint eliminates the origin as a feasible cornerpoint.

When converting to equality format, we will add one slack variable for the first constraint, and one slack variable for the second constraint, but will not need a slack variable for the third constraint because it is already in equality format. And therein lies the difficulty! The third constraint will involve only x_1 and x_2, and we may not be able to directly see how to set those two values such that all of the constraints are simultaneously satisfied. You can see how this problem is even more acute in LPs having many constraints.

The answer lies in adding a dimension to the mathematical representation of the original problem by adding a nonnegative *artificial variable* to any equality constraints in the model, in this case to the third constraint. While an artificial variable may *look* like a slack variable, it is treated much differently.

After converting the problem to equality format and adding the artificial variable (a_1), the initial set of constraint equations looks like this:

$$\begin{array}{llllllll} (1) & x_1 & & +s_1 & & & = & 2 \\ (2) & & x_2 & & +s_2 & & = & 3 \\ (3) & x_1 & +x_2 & & & +a_1 & = & 4 \end{array}$$

If a_1 were a slack variable, we would be able to use the origin as the initial feasible cornerpoint to get started, giving an initial basic feasible solution of $(x_1,x_2,s_1,s_2,a_1) = (0,0,2,3,4)$. But here's the difficulty: a_1 cannot be nonzero in the final solution, or the original 3rd constraint is violated! Somehow we need to force a_1 to zero to make sure that the 3rd constraint is satisfied.

The Phase 1 LP

The solution to this dilemma is to solve *two* LPs. This is where the phase 1 that we have ignored until now finally comes in. The sole purpose of the phase 1 LP is to obtain a basic feasible solution for the original problem, as follows:

- **Phase 1:** solve an LP whose objective is to minimize the value of any artificial variables in the model, in this case a_1. If you can drive all of the artificial variables to zero, then you will be at a feasible cornerpoint of the original problem.
- **Phase 2:** starting at the feasible cornerpoint found during phase 1, switch over to the original objective function and continue iterating until the optimum point is found.

The phase 1 objective function is simply to minimize the sum of the artificial variables. In general this is expressed as:

$$\text{minimize } W = a_1 + a_2 + a_3 + \ldots$$

You can think of W as the sum of the constraint violations because each artificial variable, in a manner similar to slack variables, takes up the slack between the constraint left hand side and the right hand side. But the left hand side should be equal to the right hand side in an equality constraint, so the artificial variable shows the amount by which the constraint is violated. Because we are minimizing W, after we multiply by –1 and move the variables to the left hand side, the phase 1 objective function looks like this:

$$\text{maximize } -W + a_1 + a_2 + a_3 + \ldots = 0$$

In setting up the tableau, the original (or phase 2) objective function is also retained. We use the phase 1 objective during phase 1, but also update the phase 2 objective at the same time so that it is ready to go when phase 1 ends. On the other hand, after phase 1 ends successfully we ignore the phase 1 objective function and the artificial variables thereafter because they are no longer useful.

Here is one important thing to remember: the first phase 1 tableau is never in proper form. This is because the artificial variables will be the initial basic variables for the constraints they appear in, and yet they will appear twice in the column: once in their constraint row, and once in the objective function row. They should appear only once, with a +1 coefficient, in their constraint row. Hence some initial work is required to put the tableau into proper form so that the iterations can start.

Let's go through the iterations for the version of the Acme Bicycle Company problem in which the third constraint is an equality. The initial tableau is as follows:

basic variable	eqn. no.	W	Z	x_1	x_2	s_1	s_2	a_1	RHS	MRT
W	ph1	-1	0	0	0	0	0	1	0	*never*
Z	ph2	0	1	-15	-10	0	0	0	0	*never*
s_1	1	0	0	1	0	1	0	0	2	
s_2	2	0	0	0	1	0	1	0	3	
a_1	3	0	0	1	1	0	0	1	4	

As you can see, this tableau has two objective functions: W for phase 1, which seeks to minimize the sum of the artificial variables (only a_1 in this case), and Z for phase 2, the original objective. Looking at the a_1 column, you can see that this tableau is not in proper form: we need to eliminate the coefficient in the phase 1 (W) objective function row. To do this, subtract each row containing an artificial variable from the phase 1 objective function row. In the tableau above, subtract equation 3 from equation *ph1*. The tableau in proper form is shown below.

basic variable	eqn. no.	W	Z	x_1	x_2	s_1	s_2	a_1	RHS	MRT
W	ph1	-1	0	-1	-1	0	0	0	-4	*never*
Z	ph2	0	1	-15	-10	0	0	0	0	*never*
s_1	1	0	0	1	0	1	0	0	2	
s_2	2	0	0	0	1	0	1	0	3	
a_1	3	0	0	1	1	0	0	1	4	

Notice that the RHS for the phase 1 objective function now reads "-4". But don't forget that we multiplied the original phase 1 minimization objective function by –1 so that we could operate on it as a maximization objective function: the –1 in the W column reminds us of this. So the true value of W at this point is 4, meaning there are 4 units of constraint violation. The goal of phase 1 is to reduce the sum of the constraint violations to zero. During phase 1, we are using the phase 1 objective function, and as the tableau above shows, both x_1 and x_2 are tied for the entering basic variable with objective function coefficients of –1. Let us arbitrarily choose x_2 as the entering basic variable. The calculations are shown below.

basic variable	eqn. no.	W	Z	x_1	x_2	s_1	s_2	a_1	RHS	MRT
W	ph1	-1	0	-1	-1	0	0	0	-4	*never*
Z	ph2	0	1	-15	-10	0	0	0	0	*never*
s_1	1	0	0	1	0	1	0	0	2	no limit
s_2	2	0	0	0	1	0	1	0	3	3/1=3
a_1	3	0	0	1	1	0	0	1	4	4/1=4

basic variable	eqn. no.	W	Z	x_1	x_2	s_1	s_2	a_1	RHS	MRT
W	ph1	-1	0	-1	0	0	1	0	-1	*never*
Z	ph2	0	1	-15	0	0	10	0	30	*never*
s_1	1	0	0	1	0	1	0	0	2	2/1=2
x_2	2	0	0	0	1	0	1	0	3	no limit
a_1	3	0	0	1	0	0	-1	1	1	1/1=1

The phase 1 objective function value, the sum of the constraint violations, has been reduced to 1 in the tableau above. Phase 1 is not yet complete though: there is still a negative coefficient in the phase 1 objective function row. Note that the phase 2 objective has also been updated to reflect its value at this new (infeasible) cornerpoint.

basic variable	eqn. no.	W	Z	x_1	x_2	s_1	s_2	a_1	RHS	MRT
W	ph1	-1	0	0	0	0	0	1	0	*never*
Z	ph2	0	1	0	0	0	-5	15	45	*never*
s_1	1	0	0	0	0	1	1	-1	1	
x_2	2	0	0	0	1	0	1	0	3	
x_1	3	0	0	1	0	0	-1	1	1	

The tableau above has at last managed to force the phase 1 objective function to zero. Phase 1 is now complete: no constraints are violated, and we are at a feasible cornerpoint for the original problem. Now it is time to discard the phase 1 objective function, the *W* column, and all of the artificial variables (just a_1 in this case), which results in the tableau below. Note that the phase 2 objective function, which has been carried through all of the calculations so far, is updated, in proper form, and ready to go. And because there is a negative coefficient in the phase 2 objective function row, the iterations must continue.

basic variable	eqn. no.	Z	x_1	x_2	s_1	s_2	RHS	MRT
Z	ph2	1	0	0	0	-5	45	*never*
s_1	1	0	0	0	1	1	1	1/1=1
x_2	2	0	0	1	0	1	3	3/1=3
x_1	3	0	1	0	0	-1	1	no limit

basic variable	eqn. no.	Z	x_1	x_2	s_1	s_2	RHS	MRT
Z	ph2	1	0	0	5	0	50	*never*
s_2	1	0	0	0	1	1	1	
x_2	2	0	0	1	-1	0	2	
x_1	3	0	1	0	1	0	2	

After this final phase 2 iteration, the solution is complete: there are no negative coefficients remaining in the phase 2 objective function. The final solution is (x_1,x_2,s_1,s_2)

= (2,2,0,1) with $Z = 50$. As before, Acme should produce mountain bikes at the rate of 2 per day and racers at the rate of 2 per day to achieve a profit rate of $50 per day.

Because of the added dimensions due to the slack and the artificial variables, it is not strictly possible to draw a picture illustrating what happens during a two-phase solution. But we can project what is happening onto a 2-dimensional, x_1-x_2 plane, as in Figure 5.2. The figure shows that the phase 1 solution still starts at the origin, which is *feasible* as far as the phase 1 formulation is concerned, because of the added artificial variable, but which is *infeasible* for the phase 2 formulation. The phase 1 procedure moves through other cornerpoints that are phase-1-feasible but phase-2-infeasible until it reaches a cornerpoint that is feasible for *both* phase 1 and phase 2. It is at this point that the phase 1 formulation is discarded in favor of the phase 2 formulation. The phase 2 solution then iterates from there to problem optimality.

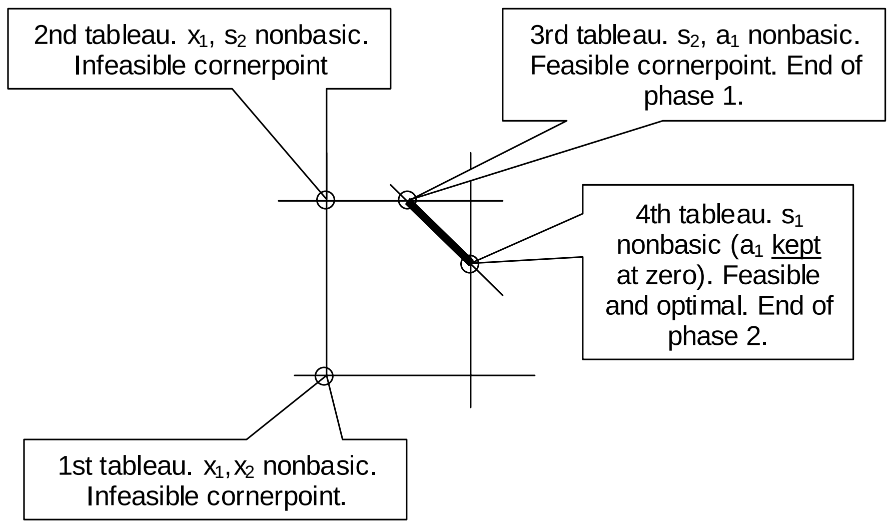
2nd tableau. x_1, s_2 nonbasic. Infeasible cornerpoint
3rd tableau. s_2, a_1 nonbasic. Feasible cornerpoint. End of phase 1.
4th tableau. s_1 nonbasic (a_1 kept at zero). Feasible and optimal. End of phase 2.
1st tableau. x_1,x_2 nonbasic. Infeasible cornerpoint.

Figure 5.1:Projecting the cornerpoints onto the x1-x2 plane.

It's no surprise to see that we arrive at the same solution obtained when we first solved the Acme Bicycle Company problem in Chapter 4. The problems are identical, except that the feasible region for this version of the problem is much smaller. The important point is that the two feasible regions overlap on (2,2), which is the optimal point.

Recognizing Infeasible Linear Programs

It is not difficult to construct linear programs for which there is no feasible solution. The infeasibility can be as simple as two parallel and oppositely-oriented constraints (e.g. $x_1 \leq 8$ and $x_1 \geq 10$), or it could be caused by a number of constraints interacting in such a way that the feasible region is completely eliminated. Fortunately, infeasibility is simple to recognize: if the phase 1 LP terminates (all of the phase 1 objective function coefficients are nonnegative), and yet W is still positive, then not all of the constraint violations have been eliminated. This means that the LP is infeasible. One or more of the artificial variables could not be forced to zero, and for feasibility, *all* of the artificial variables must be forced to zero.

The solution procedure terminates at this point without going on to the phase 2 solution. Now you need to analyze your linear program to determine *why* it is infeasible. It may be that you have correctly expressed the constraints, so there really is no solution to the problem. But it's much more likely that you have made an error in writing the constraints, such as expressing a $\leq$ constraint as a $\geq$ constraint, or added a superfluous, contradicting constraint.

Fortunately, there are tools that can assist in your analysis. Most modern commercial solvers incorporate special routines to find an *Irreducible Infeasible Set (IIS)* of constraints. An IIS is usually a very small subset of the constraints in the model and has a special property: the IIS is infeasible, but taking away one or more of the constraints in the IIS makes it feasible. This means that every member of the IIS contributes directly to the infeasibility. Instead of laboriously working through the thousands of constraints in a model, an IIS brings the problematic constraints directly to the modeler's attention. Human insight is then needed to determine which of the constraints in the IIS to correct.

Most of the routines for finding IISs and analyzing infeasible linear programs in general were developed by, ahem, your humble author. I would very much enjoy going into great detail on this *fascinating* topic, but alas, time and space are limited in this introductory exposition…

The Big-M Method

The Big-M method is an alternative to the two-phase method that we have described above. It is presented in many textbooks, but is not used in commercial solvers because it causes numerical problems in the computer calculations. The main idea in the Big-M method is to combine the phase 1 and phase 2 objective functions into a single objective function. This is done by writing the original objective function, and then adding the artificial variables to it. Each artificial variable appears in the objective function with a large penalty coefficient: this is the "Big-M", which stands for "big multiplier". For example, the Big-M version of the objective function for the problem solved above is:

$$\text{maximize } Z = 15x_1 + 10x_2 - \mathrm{M}a_1$$

As you can see, if M is a large positive number then a straightforward solution of the problem will drive a_1 to zero because it has such a reducing effect on Z, which simplex is trying to maximize. When a_1 is zero, the solution will be feasible for the original problem.

Greater-than-or-Equal Constraints

Constraints of the $\geq$ type with a positive right hand side constant are not allowed in a standard form LP, again because such constraints eliminate the origin as a feasible cornerpoint, similar to the way in which equality constraints eliminate the origin. You can't simply multiply through by –1 because this then gives a negative right hand side. The problem with negative right hand sides is that they imply a negative value for a basic variable in the tableau, which is not possible if the variables are restricted to nonnegativity.

The solution to this problem is to convert the $\geq$ constraint to an equality constraint by including a *surplus variable*. A surplus variable is exactly the same as a slack variable, except that it expresses the surplus of the left hand side over the right hand side of the constraint. For example:

$$3x_1 + 5x_2 \geq 20 \;\Rightarrow\; 3x_1 + 5x_2 - s_1 = 20.$$

A surplus variable cannot be used as the basic variable for the constraint it appears in because its coefficient is –1, and we need +1 for the coefficients of basic variables. On the other hand, the constraint is now an equality, so we can treat it exactly as we treat any other equality constraint: add an artificial variable and use a phase 1 procedure! In the end, our example constraint would look like this:

$$3x_1 + 5x_2 - s_1 + a_1 = 20$$

So there will be *two* variables added to every $\geq$ constraint: one surplus variable (with a coefficient of –1) and one artificial variable (with a coefficient of +1). Any artificial variables added to $\geq$ constraints will be driven to zero during phase 1.

Negative Right Hand Side

As described above, a negative right hand side constant presents a problem because it implies that the basic variable for the constraint must take on a negative value, which is not possible for nonnegative variables. The solution to this problem is simple: just multiply by –1 to make the right hand side positive, and then treat the resulting $\leq$, $\geq$, or = as appropriate.

How Many Variables?

The reformulation of the problem for solution via the simplex method does add a number of variables. Let's suppose we have a problem which has n original variables, l less-than-or-equal constraints, g greater-than-or-equal constraints, and e equality constraints. The reformulated model will then have:

- n original variables,
- l slack variables,
- g surplus variables, and

- $g + e$ artificial variables.

All of the g + e artificial variables will be eliminated at once during the same phase 1 procedure using a phase 1 objective of minimizing $W = \sum$(artificial variables).

The speed of the simplex solution depends mostly on the number of constraints in the model, not the number of variables, so the blow-up in the number of variables is not important. Solution speed depends on the number of constraints because these define the cornerpoints, and it is the number of cornerpoints that must be traversed that determines how long the solution takes.

An Example Conversion

Here is another non-standard-form linear program that is vaguely related to the Acme Bicycle Company problem:

minimize	$Z = 15x_1 + 10x_2$	← minimization instead of maximization
	$x_1 \leq 2$	
	$x_2 = 3$	← equality instead of $\leq$
	$x_1 + x_2 \geq 4$	← $\geq$ instead of $\leq$

When converted for solution in the simplex tableau, this becomes:

maximize	$(-W) + a_1 + a_2 = 0$	phase 1 objective function
maximize	$(-Z) + 15x_1 + 10x_2 = 0$	phase 2 objective function
	$x_1 + s_1 = 2$	
	$x_2 + a_1 = 3$	
	$x_1 + x_2 - s_2 + a_2 = 4$	

The initial basic variables will be s_1 for the first constraint, a_1 for the second constraint, and a_2 for the third constraint. These equations can be transported directly to the tableau, which must then be put into proper form (it will not be in proper form because the two artificial variables, both basic, will appear in the phase 1 objective function row as well as in their respective constraints).

Variables that can be Negative

In some problems, the most natural formulation of the problem allows some variables to possibly take on negative values. For example, the variable might express the difference between the production level now and the production level in the last fiscal quarter, so it will take on a negative value if the production level falls. There are two cases to consider, as described below.

There is a Lower Bound on the Negative Value

You may know, for example, that the production level cannot drop more than 25 units from its present value, that is that $x_1 \geq -25$, or in general that $x_j \geq L_j$, where L_j is a negative number. In this case we can use a change of variables: $x_j = x_j' + L_j$. Now we replace x_j by $x_j' + L_j$ everywhere in the problem and solve the resulting LP for the value of x_j'. Afterwards, the value of x_j is recovered from the variable relationship: $x_j = x_j' + L_j$.

For example, consider the following linear program in which one of the variables can be negative:

$$\begin{array}{ccccc}
Z = 15x_1 - 5x_2 & & Z = 15(x_1'-25) - 5x_2 & & Z = 15\,x_1' - 5x_2 - 375 \\
2x_1 \leq 10 & & 2(x_1'-25) \leq 10 & & 2x_1' \leq 60 \\
4x_2 \leq 25 & \Rightarrow & 4x_2 \leq 25 & \Rightarrow & 4x_2 \leq 25 \\
3x_1 - 2x_2 \leq 8 & & 3(x_1'-25) - 2x_2 \leq 8 & & 3x_1' - 2x_2 \leq 83 \\
\boldsymbol{x_1 \geq -25} & & (x_1'-25) \geq -25 & & x_1' \geq 0 \\
x_2 \geq 0 & & x_2 \geq 0 & & x_2 \geq 0
\end{array}$$

As you can see, the final converted form of the problem is an ordinary linear program with nonnegative variables that we already know how to solve! After solving, the value of x_1 is found from the original change of variables relationship: $x_1 = x_1' - 25$. By the way, the constant in the converted objective function is easy to handle: it just appears on the right hand side in the initial tableau and is carried along. The final value of Z is given correctly in the final tableau and is not affected by the change of variables.

Unrestricted Variables

Variables can also take on negative values when they are totally unrestricted; when their value may be anything between minus infinity and plus infinity. Unrestricted variables must be handled in a different way: each unrestricted variable x_j is replaced by the difference of two new nonnegative variables: $x_j = x_j^+ - x_j^-$. Even though x_j^+ and x_j^- are nonnegative, x_j can take on any value between minus and plus infinity by choosing appropriate values for x_j^+ and x_j^-.

As it happens, the simplex method will automatically set one or the other of x_j^+ or x_j^- to zero, or it will set both to zero, but it will never happen that both are positive simultaneously. Why is this? Notice that when you make the change of variables, replacing x_j by $x_j^+ - x_j^-$, you get two tableau columns that are identical, except that one is the negative of the other. This property is maintained throughout the solution. Recall that when a tableau is in proper form, the basic variable has a +1 coefficient, and the rest of the column is filled with zeros. If the x_j^+ and x_j^- columns are negative copies of each other, then it is not possible for both to be basic at the same time: the +1 entry in one column should be mirrored by a –1 entry in the other column, and this is not possible when the tableau is in proper form.

This means that reading the value of x_j after the revised problem is solved is straightforward:

- if $x_j^+ > 0$ then $x_j = x_j^+$,
- if $x_j^- > 0$ then $x_j = - x_j^-$,
- if $x_j^+ = x_j^- = 0$ then $x_j = 0$.

You might think that if all of the variables in the problem are unrestricted, then there must be twice as many variables in the revised problem, but this is not so. In this case we just use the *same* x_j^- in *all* of the change of variable expressions. Upon solution, x_j^- is the absolute value of the most negative variable. The various x_j^+ then express the amount by which each x_j exceeds that most negative value in a manner similar to the case of lower-bounded negative variables.

In Practice

In practice, you don't have to worry about any of the details of converting a linear program into a form that can be solved by the simplex method. The solver software carries out any necessary conversions automatically. Your responsibility is making sure that the mathematical model correctly represents reality, or more accurately, makes a relevant approximation of reality!

Some of the conversions mentioned in this chapter are not in fact needed at all in some LP solvers. For example, many solvers implicitly assume that all variables (and row constraints) are upper and lower bounded. Whether the bounds are positive or negative does not matter: all bounds are handled in a similar manner.