Optimization Methods
Draft of August 26, 2005

I.
Solving Linear Programs by the Simplex Method

Robert Fourer

Department of Industrial Engineering and Management Sciences
Northwestern University
Evanston, Illinois 60208-3119, U.S.A.

(847) 491-3151

4er@iems.northwestern.edu
http://www.iems.northwestern.edu/~4er/

Introduction

As the companion to this volume shows, there are many opportunities for formulating optimization and related problems as the minimization of linear functions of many decision variables, subject to restrictions on other linear functions of the variables. There would be no point to formulating these ***linear programs*** or ***LPs,*** however, if they couldn't be "solved" for an optimal objective value and optimal levels of the variables. Thus, although linear algebra has long been a part of mathematics, the popularity of linear programming dates only back to the early 1950's, when the first computational *algorithms* for solving LPs were developed. Not surprisingly, these algorithms were invented at about the same time that computers were first widely used in government and industry.

In the first part of this volume we introduce one particular method, the *simplex algorithm,* for solving linear programs using a computer. The simplex algorithm was the first practical approach for solving LPs, and it is still the most widely used — although newer approaches may prove to be faster, especially for very large problems. Our development of the simplex algorithm will provide an elementary yet extensive example of the kinds of reasoning involved in deriving methods for solving optimization problems; most importantly, you will see that the algorithm is an *iterative* method for which the number of steps cannot be known in advance. Additionally, many important properties of linear programs will be seen to derive from a consideration of the simplex algorithm.

We begin this part by motivating the simplex algorithm and by deriving formulas for all of its steps. Then we develop just enough theory to prove that the algorithm really works, under some simplifying assumptions. We conclude by reconsidering these assumptions, in order to show that the algorithm is truly of practical value.

1. General Formulations

Different applications of linear programming give rise to many different structures and formulations. When it comes to solving linear programs, however, we do not want to have to devise a different method for every different application. Much of the strength and usefulness of linear programming derives from the fact that a single algorithm can be used to solve just about any LP.

To permit a general solution method, what we need is a general formulation to which all other LP formulations can be transformed. This section presents such a formulation, and describes the most common transformations associated with it. We also introduce a second general formulation that is useful in geometric analyses of linear programming. Later the second form will be seen to also play a very important role as a so-called *dual* of the first.

1.1 A general "algebraic" form

The general form to be employed here consists of the minimization of a linear function of n nonnegative variables, subject to m linear equations:

$$\begin{array}{lrcl}
\text{Minimize} & c_1x_1 + & c_2x_2 + \cdots + & c_nx_n \\
\text{Subject to} & a_{11}x_1 + & a_{12}x_2 + \cdots + & a_{1n}x_n = b_1 \\
& a_{21}x_1 + & a_{22}x_2 + \cdots + & a_{2n}x_n = b_2 \\
& \vdots & & \vdots \\
& a_{m1}x_1 + & a_{m2}x_2 + \cdots + & a_{mn}x_n = b_m \\
& & x_1 \geq 0, x_2 \geq 0, \ldots, & x_n \geq 0
\end{array}$$

The same objective and constraints can be written more concisely using the familiar Σ summation notation from calculus:

$$\begin{array}{ll}
\text{Minimize} & \sum_{j=1}^{n} c_j x_j \\
\text{Subject to} & \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, \ldots, m \\
& x_j \geq 0, \qquad\qquad j = 1, \ldots, n
\end{array}$$

This resembles standard diet, blending and other minimum-cost input models, except that the required amount of output i is required to exactly equal b_i.

In the subsequent discussion of algorithms, it will be desirable to employ an even more concise vector and matrix notation. The data of the general LP can be represented by an $m \times n$ matrix

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

together with an n-vector of "costs"

$$c = (c_1, \ldots, c_n)$$

and an m-vector of "right-hand sides"

$$b = (b_1, \ldots, b_m).$$

The variables can also be grouped into an n-vector:

$$x = (x_1, \ldots, x_n).$$

Then the entire linear program can be written as follows:

$$\begin{array}{ll} \text{Minimize} & cx \\ \text{Subject to} & Ax = b \\ & x \geq 0 \end{array}$$

The notation cx refers to the inner product $\sum_{j=1}^{n} c_j x_j$ of the n-vectors c and x. Often in linear algebra this is written $c^T x$ (c-transpose x), where the "row vector" c^T multiplies the "column vector" x. But for purposes of introducing the simplex method we will not distinguish row and column vectors where the meaning is clear from the context.

1.2 Transformations to the general algebraic form

If you have a linear program whose objective is minimized, whose variables are all nonnegative, and whose constraints are all equalities, then it fits the general form above. As a simple example, a 3×3 assignment problem,

$$\begin{array}{lll} \text{Minimize} & \sum_{i=1}^{3} \sum_{j=1}^{3} c_{ij} x_{ij} & \\ \text{Subject to} & \sum_{j=1}^{3} x_{ij} = 1, & i = 1, \ldots, 3 \\ & \sum_{i=1}^{3} x_{ij} = 1, & j = 1, \ldots, 3 \\ & x_{ij} \geq 0, & i = 1, \ldots, 3;\ j = 1, \ldots, 3 \end{array}$$

can be put into the general form, by taking

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

$$b = (1, 1, 1, 1, 1, 1),$$

$$c = (c_{11}, c_{12}, c_{13}, c_{21}, c_{22}, c_{23}, c_{31}, c_{32}, c_{33}),$$
$$x = (x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}).$$

For other linear programs, some more work is necessary. Fortunately, however, there are simple and convenient ways of transforming different kinds of objectives and constraints so that they conform to the standard algebraic form. The following are the most common cases that require transformation:

Maximization. To find a solution that maximizes some function, all you have to do is minimize its negative. In particular, to find a maximizing solution for

$\sum_{j=1}^{n} c_j x_j$, you can minimize $\sum_{j=1}^{n} (-c_j) x_j$; that is, just reverse the sign of every nonzero coefficient in the objective.

When some computer systems are applied to solve a profit-maximizing LP, the optimal objective value is reported as being negative! This is because the system is really minimizing the negative of the profit.

Inequality constraints. Suppose that a linear program has a constraint that is an inequality, such as

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i,$$

with all $x_j \ge 0$. Clearly this constraint is satisfied if and only if the difference $b_i - \sum_{j=1}^{n} a_{ij} x_j$ is nonnegative. As a result, we can define a nonnegative variable s_i to equal the difference, leading to equivalent constraints

$$\sum_{j=1}^{n} a_{ij} x_j + s_i = b_i, \quad s_i \ge 0,$$

with still all $x_j \ge 0$. We have transformed the inequality constraint into an equality in nonnegative variables, as required by the standard algebraic form.

The variable s_i is called a ***slack*** variable. You can transform any number of $\le$ constraints into $=$ constraints by adding a slack variable to each one. (Remember to add a different slack variable to each constraint, however; it is not equivalent to use the same slack on every one.)

A $\ge$ constraint is handled in much the same way. The equivalent of

$$\sum_{j=1}^{n} a_{ij} x_j \ge b_i$$

is

$$\sum_{j=1}^{n} a_{ij} x_j - s_i = b_i, \quad s_i \ge 0,$$

which again fits the standard algebraic form. The variable s_i subtracted here is also called a slack variable, or sometimes a ***surplus*** variable. (Of course, we could have just exchanged the two sides of the $\ge$ constraint to make it a $\le$ one, and then we could have added a slack. The result would have been the same, though.)

Differently constrained variables. In the unlikely case that the constraints require $x_j \le 0$, we can observe that this is just the same thing as $-x_j \ge 0$. Thus, to get back to the standard form in which all variables are nonnegative, it suffices to replace every occurrence of x_j by $-x_j$. Equivalently, every nonzero coefficient of x_j just needs to be reversed in sign.

What if some variable x_j is not constrained to have either positive or negative sign? We can write such a ***free*** variable as the difference of two nonnegative variables:

$$x_j \equiv x_j^+ - x_j^- \quad \text{where} \quad x_j^+ \geq 0, \quad x_j^- \geq 0.$$

No matter what value x_j takes, there is always a pair of nonnegative values for x_j^+ and x_j^- so that x_j equals $x_j^+ - x_j^-$. Thus we can substitute the expression $x_j^+ - x_j^-$ for every occurrence of x_j in the linear program; the nonstandard free variable is consequently replaced by two standard nonnegative ones.

1.3 A general "geometric" form

An alternative standard form maximizes a linear objective subject only to certain linear inequalities:

$$\begin{array}{ll} \text{Maximize} & \sum_{j=1}^n c_j x_j \\ \text{Subject to} & \sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \end{array}$$

Defining the matrix A and the vectors b, c and x as before, the same thing can be written as

$$\begin{array}{ll} \text{Maximize} & cx \\ \text{Subject to} & Ax \leq b \end{array}$$

We call this the geometric form because it is convenient for discussions of the geometry of linear programming. In contrast, the algebraic form is much more convenient as a standard for defining and implementing the algorithm that will be described. The algebraic and geometric forms are entirely equivalent, however, in the sense that for any linear program expressed in one, there is an equivalent linear program in the other.

Suppose that we start from the algebraic form. The constraints $Ax = b$ are equivalent to $Ax \leq b$ and $Ax \geq b$, or $Ax \leq b$ and $(-A)x \leq (-b)$. The constraints $x \geq 0$ are the same as $-x \leq 0$, or $(-I)x \leq 0$, where I is an $n \times n$ identity matrix. Finally, to minimize cx, we can maximize its negative, $(-c) \cdot x$. Putting this all together, an equivalent linear program is given by

$$\begin{array}{ll} \text{Maximize} & (-c) \cdot x \\ \text{Subject to} & \begin{bmatrix} A \\ -A \\ -I \end{bmatrix} x \leq \begin{bmatrix} b \\ -b \\ 0 \end{bmatrix} \end{array}$$

which is in the geometric form.

Conversely, suppose that we start from the geometric form. Since each variable x_j is unconstrained in sign, we replace it by a difference $x_j^+ - x_j^-$ of nonnegative variables. We also convert each inequality to an equality by adding a slack variable. Now the linear program is

$$\begin{array}{lll} \text{Maximize} & \sum_{j=1}^n c_j (x_j^+ - x_j^-) & \\ \text{Subject to} & \sum_{j=1}^n a_{ij} (x_j^+ - x_j^-) + s_i = b_i, & i = 1, \ldots, m \\ & x_j^+ \geq 0, \; x_j^- \geq 0, & j = 1, \ldots, n \\ & s_i \geq 0 & i = 1, \ldots, m \end{array}$$

To maximize this objective, we can minimize its negative. When we write $x^+ = (x_1^+, \ldots, x_n^+)$, $x^- = (x_1^-, \ldots, x_n^-)$ and $s = (s_1, \ldots, s_m)$, the resulting equivalent linear program is then seen to have the form

$$\begin{array}{ll} \text{Minimize} & (-c, c, 0) \cdot (x^+, x^-, s) \\ \text{Subject to} & [A \ \ -A \ \ I]\,(x^+, x^-, s) = b \\ & (x^+, x^-, s) \geq 0 \end{array}$$

which is in the algebraic form.

Similar results can be proved about other linear programming formulations. All are equivalent under suitable transformations. As a result, it is justifiable to pick one convenient standard LP form, and to develop just one algorithm that applies to it.

2. Geometry

To understand the steps of an algorithm for linear programming, you may find it useful to first understand the underlying geometry. This section develops a two-dimensional example to show how a linear program's optimal solution must occur at a certain "vertex" of a polygon of possible solutions. Then the insight provided by this example is extended to higher dimensions, where several key complications are seen to appear. Finally, we use these examples to suggest the essentials of an algorithm for solving linear programs.

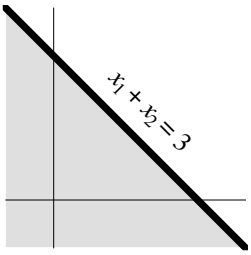
2.1 A two-dimensional example

For a geometric form linear program in two variables, it is easy to graph the objective and constraints and to see what the optimal solution ought to be. For purposes of illustration, we use the following example:

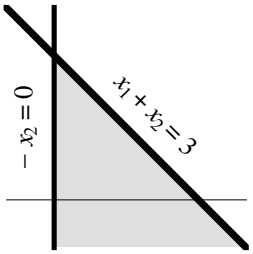
$$\begin{array}{lrcl} \text{Maximize} & x_1 + 4x_2 & & \\ \text{Subject to} & x_1 + x_2 & \leq & 3 \\ & x_1 + 2x_2 & \leq & 4 \\ & -x_1 + x_2 & \leq & 1 \\ & -x_1 & \leq & 0 \\ & -x_2 & \leq & 0 \end{array}$$

We represent any solution vector (x_1, x_2) in the familiar way, as a point in the graph of x_1 versus x_2.

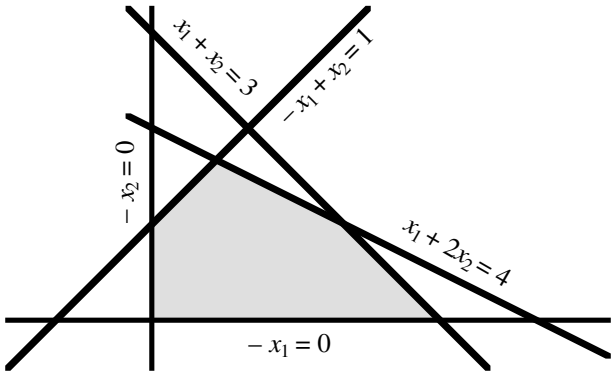
Consider first the constraints. Each linear inequality is satisfied by all points (x_1, x_2) in a certain half-plane. For example, $x_1 + x_2 \leq 3$ is satisfied by all points on or below the line defined by $x_1 + x_2 = 3$, as shown by the shaded region in this graph:



A pair of inequalities is satisfied by all points (x_1, x_2) in a pair of intersecting half-planes. For example, points that satisfy $x_1 + x_2 \leq 3$ as above, and also $-x_2 \leq 0$, must lie both below $x_1 + x_2 = 3$ and to the right of $-x_2 = 0$:

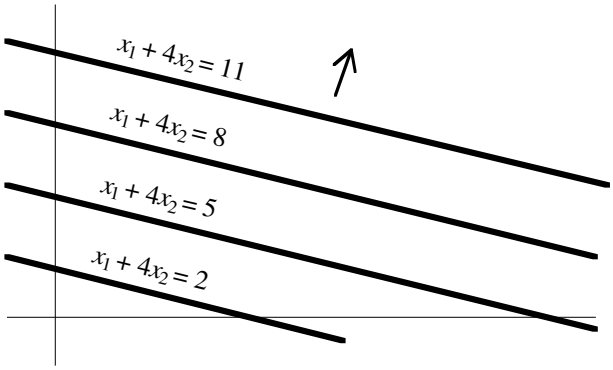


More generally, m different constraints define a ***feasible region*** that is the intersection of m half-spaces. Only the points in this region satisfy all m constraints. For our example, the feasible region is the following shaded area:



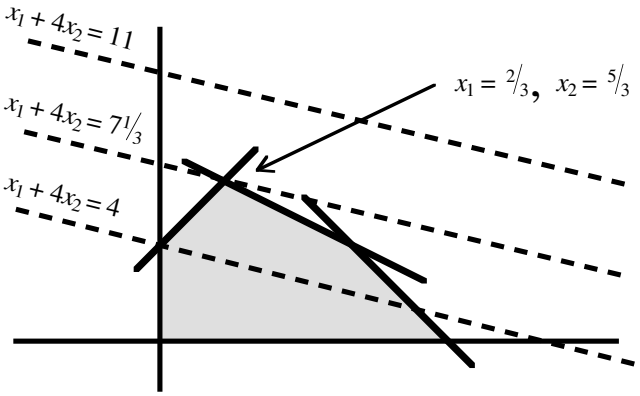
It is the intersection of five half-spaces corresponding to the five constraints.

Now consider the set of all points at which the objective function has some given value. This set is just a line. In our example, the set of points where the objective function $x_1 + 4x_2$ is equal to 8 is just the line $x_1 + 4x_2 = 8$. The set of points where it equals any number K is the line $x_1 + 4x_2 = K$. Here are some examples:



You can see that different values of the objective define different *parallel* lines. As the objective increases, these lines "move" across the plane — up and to the right in this case.

We can now say how to solve the linear program: The optimal solution is given by the "highest" objective line that intersects the feasible region. To see how this is true, look again at the example, with the graphs of some objective lines superimposed on the graph of the constraints:



Three cases can be observed here:

- The line $x_1 + 4x_2 = 4$ intersects the feasible region. At all the points where it intersects, the constraints are satisfied and the objective value equals 4.
- The line $x_1 + 4x_2 = 11$ misses the feasible region entirely. No point satisfying all constraints is able to have an objective value equalling 11.
- The line $x_1 + 4x_2 = 7\frac{1}{3}$ is the highest one (of the family $x_1 + 4x_2 = K$) that intersects the feasible region. It touches just one point at a corner of the region, where all constraints are satisfied and the objective value equals $7\frac{1}{3}$. All higher lines clearly miss the region.

From the third case, we can conclude that $7\frac{1}{3}$ is indeed the ***optimal*** value of the objective. The solution that achieves the optimum is represented by the corner point, or vertex, where the line $x_1 + 4x_2 = 7\frac{1}{3}$. just touches the feasible region. This point is $x_1 = \frac{2}{3}$, $x_2 = \frac{5}{3}$.

We can summarize this example by saying that the feasible region is a polygon, and the optimal solution is found at a vertex. These facts hold for two-dimensional linear programs generally. There do exist some exceptions, but only for special cases; it can happen that there is no finite optimal solution at all, or that the optimal set includes all of the infinitely many points along one "edge" of the feasible polygon (including the vertices at each end).

These observations suggest that, to find an optimal solution, it should be enough to check all vertices and to choose the one with the highest objective value. Indeed, you could save time by starting at any one vertex, and then moving along the feasible region's perimeter to vertices that have better and better objective values. As soon as the objective started to go down again, you could stop and declare an optimum found.

2.2 Aspects of a three-dimensional example

If the two-dimensional case were typical of linear programming generally, then LPs would be truly easy to solve. In fact, however, there are certain complications that become evident only in higher dimensions.

Consider the case of a 3-variable linear program, in the standard geometric form. You can imagine "graphing" each possible solution (x_1, x_2, x_3) as a point in three-dimensional space. Each constraint is now satisfied by all points in a certain half-space, defined by a plane and all points to one side of it. Together, the constraints are satisfied by all points that lie in the intersection of various half-spaces. Such an intersection defines a feasible region that has the form of a ***polyhedron*** in 3-space: a convex body with flat faces, like a pyramid or a cut diamond.

The set of all points where the objective function equals some value K is represented by a plane in 3-space. As K is increased, a family of parallel planes is defined. Thus the optimal objective value is naturally defined by the "highest" plane that still intersects the feasible region. By the same reasoning that applied in two dimensions, the optimal solution should occur at some vertex of the region.

Whereas a vertex in the two-dimensional case is always at the intersection of two edges of the feasible polygon, however, a vertex in the three-dimensional case is at the intersection of three *or more* edges. (Think of the tip of a square pyramid, for example, which is at the intersection of four edges.) Furthermore, whereas a polygon has just one perimeter, there can be many paths along the edges of a polyhedron from one vertex to another. The possibilities multiply rapidly as one moves into higher dimensions (which are hard to visualize, of course).

2.3 Geometry of an algorithm

The above analysis leads to an idea for an algorithm for solving linear programs. Start at any convenient vertex of the feasible region, and repeat the following steps:

- Choose an edge that heads "upward" — in the sense of the objective function — from the current vertex.
- Follow the chosen edge until you reach another vertex, where the objective function is necessarily higher than before.

When you reach a vertex from which all edges head "downward", stop and declare the current vertex to be the optimal solution.

This is, in fact, the idea of the simplex algorithm. To apply it with confidence, we need only show that it really works for all linear programs. In particular, we must show that it can't follow edges upward forever, without ever stopping. And we must show that when all edges from a vertex head downward, that vertex must be really optimal.

We have a further goal, however, which is to make the algorithm practical for running on a computer. Since the computer can't look at graphs, we must recast the algorithm in terms of specific algebraic steps. To this end, we will need to answer questions like the following:

- What are vertices and edges in algebraic terms?

- What algebraic test can tell you whether or not an edge heads upward from the current vertex?
- What algebraic test can tell you which vertex you reach when you are following an edge?

In the process, we will have to deal with various complications that would not be expected from the geometry. Here is a list of just some of them:

- There will have to be a way of finding a feasible vertex to start from.
- If more than one edge head upward from the current vertex, there will have to be a criterion for choosing which one to follow.
- If the current vertex is an intersection of more than three edges, there is a possibility of "degenerate" algebraic steps that fail to either change the solution or increase the objective.
- There is a possibility of following an edge out to infinity, without ever reaching another vertex.
- There is a possibility of having more than one optimal vertex.

The purpose of the next several sections is to develop the algebraic algorithm, to deal with the complications that come up, and to show that the algorithm really works under realistic conditions.

You might wonder why we want to follow edges, rather than moving toward the optimum through the middle of the feasible polyhedron. Indeed, the idea of moving through the middle leads to some very efficient algorithms — but not in the most obvious ways. Because this simplex algorithm is more elementary, it makes for a better introduction to linear programming methods; also, in the course of developing the simplex method we will have a convenient opportunity to derive many important properties of linear programs.

3. Algebraic Interpretation of Vertices: Basic Solutions

Consider now a linear program in the standard algebraic form,

$$\begin{array}{ll} \text{Minimize} & cx \\ \text{Subject to} & Ax = b \\ & x \geq 0 \end{array}$$

As before, A is an $m \times n$ matrix; let a_j denote the jth column of A, and let a^i denote the ith row. To be consistent, b must be an m-vector and c must be an n-vector; x is an n-vector of variables.

After a few preliminaries, we define below the notion of "basic" solutions that correspond to vertices of the geometric form.

3.1 Preliminary definitions and assumptions

We'll find it convenient to say that particular n-vector $\bar{x}$ is a ***solution*** to a linear program if it satisfies the equation constraints: $A\bar{x} = b$. Solutions can be found by standard techniques of linear algebra, like Gaussian elimination; for linear programs of interest, there are infinitely many of them.

To avoid certain uninteresting complications, we will make the following assumption about the equations' coefficient matrix:

The rows of A are linearly independent.

A fundamental property of matrix algebra says that, if this assumption does not hold, then either there are redundant equations in $Ax = b$ that can be dropped, or there are inconsistent equations that prevent $Ax = b$ from having any solutions at all. Thus we are not giving up anything by requiring independence. Moreover, in applications of linear programming the rows of A either turn out to be independent, or can easily be made so.

As an immediate consequence of this assumption, the matrix A cannot have more rows than columns: $m \leq n$. In other words, the linear program cannot have more equations than variables.

What makes linear programming difficult is that not all solutions to the equations also satisfy the nonnegativity constraints. A solution $\bar{x}$ is said to be ***feasible*** if $\bar{x} \geq 0$ as well as $A\bar{x} = b$. If x^* is a feasible solution, and if its objective value $cx^* \leq c\bar{x}$ for any other feasible solution $\bar{x}$, then x^* solves the linear program; it is ***optimal.***

3.2 Basic solutions

We can easily dispose of the case $m = n$. Since we have assumed that the rows of A are linearly independent, the equations $Ax = b$ have exactly one solution in this situation. Either that solution is a vector of nonnegative entries, in which case it is *the* feasible and optimal solution; or else there are no feasible solutions at all.

Now consider the interesting case of $m < n$. The equations $Ax = b$ in this situation have infinitely many solutions. Even so, we can reduce the problem to that of finding a unique solution for just m of the variables. First, we pick m variables that correspond to m linearly independent columns of A; this must be possible, because there are m independent rows. Then, we set the other $n - m$ variables to zero. What's left are m independent equations in m variables, which necessarily have a unique solution. Of course, the solution we get will depend on which m independent columns we pick.

To write the same thing more precisely, we'll use the following notation for a choice of m linearly independent columns of A:

$\mathcal{B}$	a set of column indices (or column numbers) of m linearly independent columns of A
B	the $m \times m$ matrix formed from the columns corresponding to the indices in $\mathcal{B}$
$x_{\mathcal{B}}$	the m-vector of variables corresponding to the indices in $\mathcal{B}$

We use an analogous notation for the $n - m$ other columns:

$\mathcal{N}$	the set of column indices of the columns of A that are not in B
N	the $m \times (n - m)$ matrix formed from the columns corresponding to the indices in $\mathcal{N}$
$x_{\mathcal{N}}$	the $(n - m)$-vector of variables corresponding to the indices in $\mathcal{N}$

Then the kind of solution $\bar{x}$ that we have been talking about is determined uniquely by

$$\begin{aligned} \bar{x}_{\mathcal{N}} &= 0, \\ B\bar{x}_{\mathcal{B}} &= b. \end{aligned}$$

Either the set $\mathcal{B}$ or the matrix B is referred to as a ***basis*** for the linear program. The variables in $x_{\mathcal{B}}$ and $x_{\mathcal{N}}$ are respectively the basic and nonbasic variables, and an $\bar{x}$ determined as above is a ***basic solution.***

There are many different bases for a typical $Ax = b$ with $m < n$, and to each there corresponds a unique basic solution. Conversely, if $\bar{x}$ is a solution whose nonzero entries correspond to linearly independent columns of A, then we can assert the following:

- If there are exactly m nonzero entries in $\bar{x}$, then it is a basic solution for which there is a unique basis.
- If there are less than m nonzero entries in $\bar{x}$, then it is a basic solution for which there are many different bases — a case we'll have more to say about later.

If there are more than m nonzero entries in $\bar{x}$, on the other hand, then it cannot be a basic solution.

The variables $\bar{x}_{\mathcal{N}} = 0$ in a basic solution are necessarily nonnegative. There is nothing to guarantee, however, that the variables $\bar{x}_{\mathcal{B}}$ will be nonnegative. If they are, then $\bar{x}$ is a ***basic feasible solution,*** and we refer to the set $\mathcal{B}$ or the matrix B as a ***feasible basis.***

We can now easily state a correspondence between geometry and the algebraic concept of bases:

> Each basic feasible solution
> plays the same role as a vertex of the feasible region.

Thus we can assume that, in particular, if there is an optimal solution at all then there is one that occurs at a basic feasible solution. As a result, we need only search basic feasible solutions to look for an optimum.

Observe that bases, like vertices, are finite in number — because there are only finitely many ways in which m basic columns can be chosen from the n columns of A. Hence the subset of feasible bases is also finite in number. The size of this subset can grow exponentially with the number of variables, however, reaching impractical values for LPs of even modest size. If there are, say, 1000 variables and 100 constraints, then the number of ways to choose 100 different variables out of the 1000 is

$$\begin{pmatrix} 1000 \\ 100 \end{pmatrix} \approx 6.4 \times 10^{139}.$$

Very few of these choices of 100 variables may give feasible bases, but even a tiny fraction would be far too many to search in anyone's lifetime. But as we will see, there are systematic ways to search the basic feasible solutions that in practice visit only a manageably tiny subset of them.

3.3 Examples

To illustrate various kinds of solutions and bases, we introduce the following two equations in five variables ($m = 2$, $n = 5$):

$$\begin{aligned} x_1 - 2x_2 + x_3 - 4x_4 + 2x_5 &= 2 \\ x_1 - x_2 + 2x_3 - 3x_4 - x_5 &= 4 \\ x_1, \ldots, x_5 &\geq 0 \end{aligned}$$

These are the constraints of some linear program, but we don't need to know yet what the objective function is. If the constraints are written in the standard algebraic form $Ax = b$, $x \geq 0$, then

$$A = \begin{bmatrix} 1 & -2 & 1 & -4 & 2 \\ 1 & -1 & 2 & -3 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 4 \end{bmatrix},$$

and x is a 5-vector $(x_1, x_2, x_3, x_4, x_5)$ of variables.

A solution. The vector $\bar{x} = (14, -2, 0, 4, 0)$ satisfies $A\bar{x} = b$, so it is a solution for the linear program. It is not feasible, however, since $\bar{x}_2 = -2$ is negative.

And it is not basic, since it has three nonzero entries, while the number of constraints is only $m = 2$.

A feasible solution. The vector $\bar{x} = (8, 1, 0, 1, 0)$ satisfies $A\bar{x} = b$ and also $\bar{x} \geq 0$, so it is a feasible solution for the linear program. Again it is not basic, however, because it has more nonzero entries than the number of constraints.

A basic feasible solution.. The vector $\bar{x} = (10, 0, 0, 2, 0)$ satisfies $A\bar{x} = b$ and $\bar{x} \geq 0$, so it is another feasible solution. It is also a basic solution, because it has only two nonzero entries.

Since the number of nonzero entries in this solution is equal to the number $m = 2$ of constraints, there is a unique corresponding basis. The basic variables must be the two nonzero ones, x_1 and x_4, and the basis must be

$$B = [a_1\ a_4] = \begin{bmatrix} 1 & -4 \\ 1 & -3 \end{bmatrix}.$$

You can check that this is the right basis by setting $x_2 = x_3 = x_5 = 0$, and solving the resulting equations in x_1 and x_4:

$$\begin{array}{l} x_1 - 4x_4 = 2 \\ x_1 - 3x_4 = 4 \end{array} \Leftrightarrow \begin{bmatrix} 1 & -4 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \Leftrightarrow Bx_{\mathcal{B}} = b.$$

The unique solution to these equations is $\bar{x}_1 = 10$, $\bar{x}_4 = 2$. Together with the other variables set to zero, this gives a solution of $\bar{x} = (10, 0, 0, 2, 0)$, which is just what we started out with.

Other bases. Different choices of two variables (corresponding to independent columns of A) can give different bases and basic feasible solutions. If we take $x_{\mathcal{B}} = (x_2, x_1)$ and fix $x_3 = x_4 = x_5 = 0$, then the equations $Bx_{\mathcal{B}} = b$ become

$$\begin{array}{l} -2x_2 + x_1 = 2 \\ -\ \ x_2 + x_1 = 4 \end{array} \Leftrightarrow \begin{bmatrix} -2 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}.$$

Now the unique solution is $\bar{x}_2 = 2$, $\bar{x}_1 = 6$, giving a different basic feasible solution of $\bar{x} = (6, 2, 0, 0, 0)$. (If we had said $x_{\mathcal{B}} = (x_1, x_2)$ instead, the result would have been exactly the same — it only matters which variables are basic, not what order they're in.)

As another example, let $x_{\mathcal{B}} = (x_2, x_4)$. Fixing $x_1 = x_3 = x_5 = 0$, the equations $Bx_{\mathcal{B}} = b$ are

$$\begin{array}{l} -2x_2 - 4x_4 = 2 \\ -\ \ x_2 - 3x_4 = 4 \end{array} \Leftrightarrow \begin{bmatrix} -2 & -4 \\ -1 & -3 \end{bmatrix} \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}.$$

In this case the unique solution is $\bar{x}_2 = 5$, $\bar{x}_4 = -3$, giving a solution $\bar{x} = (0, 5, 0, -3, 0)$ that is basic but not feasible. In general, there is no easy way to tell whether a basis is feasible, just by looking at which columns are in B; you have to obtain the solution to $Bx_{\mathcal{B}} = b$, and check whether it has any negative entries.

3.4 Degeneracy

It is instructive to consider one more example, provided by the vector $\bar{x} = (0, 0, 2, 0, 0)$. Clearly $\bar{x}$ is a feasible solution to our constraints, since it satisfies both $A\bar{x} = b$ and $\bar{x} \geq 0$. Also it must be basic, because it does not have more nonzero entries than the number of constraints. In fact, there are fewer nonzero entries than constraints. Since $\bar{x}_3$ is positive, we can tell that x_3 is in the basis; but what is the other basic variable?

Suppose we try $x_{\mathcal{B}} = (x_3, x_4)$. We fix $x_1 = x_2 = x_5 = 0$, and solve $Bx_{\mathcal{B}} = b$,

$$\begin{bmatrix} 1 & -4 \\ 2 & -3 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix},$$

for $\bar{x}_3 = 2$, $\bar{x}_4 = 0$. We thus see that this choice of B is indeed a basis for $\bar{x} = (0, 0, 2, 0, 0)$.

Now suppose we try $x_{\mathcal{B}} = (x_3, x_1)$ instead. This time we fix $x_2 = x_4 = x_5 = 0$ and solve $Bx_{\mathcal{B}} = b$,

$$\begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix},$$

for $\bar{x}_3 = 2$, $\bar{x}_1 = 0$. Hence the corresponding basic solution is again $\bar{x} = (0, 0, 2, 0, 0)$; we have found a second basis for the same solution. You can verify that $x_{\mathcal{B}} = (x_3, x_2)$ and $x_{\mathcal{B}} = (x_3, x_5)$ also work, giving four bases for this solution altogether.

A basic solution of this kind — in which the number of nonzero entries in $\bar{x}$ is less than the number of constraints — is called a ***degenerate*** solution. A corresponding basis must include all of the variables that are positive in the solution, but there are only some number $k < m$ of these; the other $m - k$ variables in the basis can be chosen arbitrarily (so long as all the corresponding columns are linearly independent). Thus there can be many bases for the same degenerate basic solution. We will see that degeneracy introduces some unwelcome complications in the analysis and solution of linear programs.

The geometric analogue of degeneracy in two dimensions is trivial. At any degenerate vertex, all but two of the constraints can be discarded without any change to the solutions. (The discarded constraints are said to be *redundant*.)

A nontrivial geometric analogue of degeneracy can be described in three dimensions, however, as follows. A vertex of the feasible polyhedron has a certain number of incident edges, and an equal number of incident faces. If there are exactly three incident edges and faces, then the vertex is nondegenerate, and it is defined uniquely by the intersection of the three incident faces (or, more precisely, by the intersection of the planes that contain these faces). If there are more than three incident edges, however, then a degenerate condition exists. The vertex in this case can be defined in many ways, by the intersection of any three of the incident faces.

4. Algebraic Interpretation of Choosing an Edge: Reduced Costs

In the geometric view, the simplex method prepares to leave the current vertex of the feasible polyhedron by choosing an edge along which the objective function is improving. This section introduces the algebraic equivalent of an edge: a half-line of solutions that is generated when one nonbasic variable is allowed to increase from zero. Using this idea, we derive a formula for a "reduced cost" that indicates whether the objective will improve as the solution moves out along such a line.

4.1 The effect on the basic variables when a nonbasic variable is increased from zero

Suppose that we have found some basis $\mathcal{B}$ that is feasible: there is an associated basic solution $\bar{x}_\mathcal{B}$ that satisfies $B\bar{x}_\mathcal{B} = b$, $\bar{x}_\mathcal{B} \geq 0$. To generate further solutions that might give better objective values, we can let one of the nonbasic variables in $x_\mathcal{N}$ increase from zero. We must then adjust the values of the basic variables, however, so that the constraints $Ax = b$ continue to be satisfied.

Specifically, let x_p, $p \in \mathcal{N}$ denote some nonbasic variable that will be allowed to take positive values, while all the other nonbasic variables continue to be fixed at zero. Then the equations $Ax = b$, with all the other nonbasics dropped out, have the form

$$Bx_\mathcal{B} + a_p x_p = b,$$

where a_p is the column of A corresponding to the nonbasic variable x_p. Since B is a square matrix having linearly independent columns, it is nonsingular; we could move the $a_p x_p$ term to the right and multiply through by B^{-1} to get

$$x_\mathcal{B} = (B^{-1}b) - (B^{-1}a_p)x_p.$$

Thus we see that adjusted basic values $x_\mathcal{B}$ are a linear function of the nonbasic value x_p. As x_p is increased from zero, this function defines a half-line, or ***ray,*** of solutions.

The above formula is easily rearranged to be more computationally appropriate. We have already defined $\bar{x}_\mathcal{B}$ so that

$$B\bar{x}_\mathcal{B} = b \quad \Leftrightarrow \quad \bar{x}_\mathcal{B} = B^{-1}b,$$

and we can similarly define a vector $y_\mathcal{B}$ to satisfy

$$By_\mathcal{B} = a_p \quad \Leftrightarrow \quad y_\mathcal{B} = B^{-1}a_p.$$

Writing θ for the value given to x_p, and substituting in the expression for $x_\mathcal{B}$ above, we have the following expression for the ray of solutions:

$$\left.\begin{array}{l} x_p = \theta \geq 0 \\ x_\mathcal{B} = \bar{x}_\mathcal{B} - \theta y_\mathcal{B} \end{array}\right\} \quad \text{where } By_\mathcal{B} = a_p,$$

with all nonbasic variables other than x_p still fixed at zero. For $\theta = 0$ this just gives the original basic solution with $x_{\mathcal{B}} = \bar{x}_{\mathcal{B}}$ and $x_{\mathcal{N}} = 0$; but for $\theta > 0$ it gives other, generally nonbasic solutions.

For a particular basic variable x_i, $i \in \mathcal{B}$, we have $x_i = \bar{x}_i - \theta y_i$. Thus you can think of y_i as giving the amount by which x_i will change for each increase of 1 in the value of x_p. (In calculus terms, it would be like the derivative of x_i with respect to x_p.)

The analogy of the ray of solutions to the geometry of the linear program is easy to state:

> A ray of solutions corresponds to a half-line
> extending from a vertex and along an edge of the feasible region.

In the case of a degenerate basic solution, this statement does need to be qualified somewhat as we will explain later.

4.2 The effect on the objective value when a nonbasic variable is increased from zero

So far we have seen that it is possible to move along a ray of solutions by increasing a nonbasic variable. But is it desirable? To give an answer, we must analyze the effect on the objective value.

To derive a formula for the answer, we must introduce some notation for the objective evaluated at the current basic solution and along the ray:

$c_{\mathcal{B}}$	the m-vector of coefficients from c that correspond to the basic variables in $x_{\mathcal{B}}$
$\bar{z}$	the value of the objective function at the current basic solution $\bar{x}$
$z(\theta)$	the value of the objective function at the point on the ray of solutions where $x_p = \theta$

At a basic solution the nonbasic variables, being zero, do not contribute to the value of the objective. Hence $\bar{z} = c_{\mathcal{B}}\bar{x}_{\mathcal{B}}$. What we want is a formula for $z(\theta)$, as a function of θ.

On the ray of solutions, only the basic variables together with x_p contribute to the objective value. Thus $z(\theta) = c_{\mathcal{B}}x_{\mathcal{B}} + c_p x_p$. We have already seen, however, that the ray is defined by $x_p = \theta$ and $x_{\mathcal{B}} = \bar{x}_{\mathcal{B}} - \theta y_{\mathcal{B}}$. Substituting and rearranging, we get

$$\begin{aligned} z(\theta) &= c_{\mathcal{B}}x_{\mathcal{B}} + c_p x_p \\ &= c_{\mathcal{B}}(\bar{x}_{\mathcal{B}} - \theta y_{\mathcal{B}}) + c_p(\theta) \\ &= \bar{z} + \theta(c_p - c_{\mathcal{B}}y_{\mathcal{B}}) \end{aligned}$$

where $c_{\mathcal{B}}y_{\mathcal{B}}$ is the inner product of the vectors $c_{\mathcal{B}}$ and $y_{\mathcal{B}}$.

This formula shows that the objective value along a ray of solutions is a linear function of θ, the value of the nonbasic variable. Moreover, the expression $c_p - c_{\mathcal{B}}y_{\mathcal{B}}$ gives the amount that the objective will change for each increase of

1 in the nonbasic variable x_p. (In the terminology of calculus it would be called the derivative of $z(\theta)$ with respect to θ.)

We can now answer the question posed above. If $c_p - c_{\mathcal{B}} y_{\mathcal{B}}$ is negative, then the objective value will continuously decrease as x_p is pushed up from zero; since the problem is to minimize, this will represent an improvement, so it will be desirable to move along the ray. On the other hand, if $c_p - c_{\mathcal{B}} y_{\mathcal{B}} > 0$, then the objective value will increase as x_p is pushed from zero, and moving along the ray will be disadvantageous; while if $c_p - c_{\mathcal{B}} y_{\mathcal{B}} = 0$, the objective value will be the same for any value of x_p and there will be no advantage to changing it.

The crucial quantity $c_p - c_{\mathcal{B}} y_{\mathcal{B}}$ is known as the ***reduced cost*** of the variable x_p. We will call it d_p. Its value can be computed by first solving $B y_{\mathcal{B}} = a_p$ for $y_{\mathcal{B}}$, then accumulating $c_p - \sum_{i \in \mathcal{B}} c_i y_i$. However, if it is desired to compute reduced costs for many nonbasic variables, then this approach will require that a *different* vector $y_{\mathcal{B}}$ be determined for each one.

As an alternative, we can define a new vector $\pi = (\pi_1, \ldots, \pi_m)$ as the solution to

$$\pi B = c_{\mathcal{B}}$$

(or, to say the same thing another way, $B^T \pi = c_{\mathcal{B}}$). Then

$$\begin{aligned} d_p = c_p - c_{\mathcal{B}} y_{\mathcal{B}} &= c_p - (\pi B) y_{\mathcal{B}} \\ &= c_p - \pi (B y_{\mathcal{B}}) = c_p - \pi a_p. \end{aligned}$$

In this formula for the reduced cost, the inner product is between π and a_p. Given a particular B, we need only solve once for π; then the reduced cost for any x_p is easily computed, by just substituting the appropriate c_p and a_p in the formula. It is customary to refer to π as the ***price vector*** for the basis B, and the process of computing reduced costs is consequently sometimes called "pricing out". As the name suggests, we will find that π has important meanings beyond its convenience for computation.

To summarize, for a given basis B and nonbasic variable x_p, the objective value along the associated ray of solutions is

$$z(\theta) = \bar{z} + \theta d_p = \bar{z} + \theta (c_p - \pi a_p).$$

The objective function is improving, as you move out from the current basic solution along the ray, if and only if the reduced cost d_p is negative.

In terms of geometry, the analogy for the reduced cost is not hard to guess:

> If a nonbasic variable has a negative reduced cost,
> its ray of solutions corresponds to an edge
> along which the objective function is improving.

On the other hand, a nonnegative reduced cost gives a correspondence to an edge along which the objective is not improving. Thus you might expect that, when all the reduced costs are nonnegative — corresponding to all incident edges being non-improving — then the current basic feasible solution is optimal. We will eventually establish that this is indeed the case.

4.3 Examples

Suppose that we now add an objective function to our previous example, so that it becomes the following linear program:

$$\begin{array}{ll} \text{Minimize} & -x_1 - x_2 - 2x_3 - 2x_4 - 2x_5 \\ \text{Subject to} & x_1 - 2x_2 + x_3 - 4x_4 + 2x_5 = 2 \\ & x_1 - x_2 + 2x_3 - 3x_4 - x_5 = 4 \\ & x_1, \ldots, x_5 \geq 0 \end{array}$$

Let $\bar{x}_{\mathcal{B}} = (\bar{x}_1, \bar{x}_2) = (6, 2)$ be the current basic solution, and imagine that nonbasic variable x_4 is allowed to move away from zero, while x_3 and x_5 remain fixed. The unique solution to

$$By_{\mathcal{B}} = a_4 \quad \Leftrightarrow \quad \begin{bmatrix} 1 & -2 \\ 1 & -1 \end{bmatrix} y_{\mathcal{B}} = \begin{bmatrix} -4 \\ -3 \end{bmatrix}$$

is $y_{\mathcal{B}} = (-2, 1)$. Thus an associated ray of solutions is given by

$$\begin{aligned} x_4 &= \theta \geq 0 \\ x_{\mathcal{B}} &= \bar{x}_{\mathcal{B}} - \theta y_{\mathcal{B}} = \begin{bmatrix} 6 \\ 2 \end{bmatrix} - \theta \begin{bmatrix} -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 + 2\theta \\ 2 - \theta \end{bmatrix} \end{aligned}$$

and $x_3 = x_5 = 0$; that is, $x = (6 + 2\theta, 2 - \theta, 0, \theta, 0)$ for any $\theta \geq 0$.

To compute the reduced costs at this basis, we observe that $c_{\mathcal{B}} = (-1, -1)$, and solve

$$\pi B = c_{\mathcal{B}} \quad \Leftrightarrow \quad B^T \pi = c_{\mathcal{B}} \quad \Leftrightarrow \quad \begin{bmatrix} 1 & 1 \\ -2 & -1 \end{bmatrix} \pi = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

for $\pi = (2, -3)$. Then we have

$$\begin{aligned} d_3 &= c_3 - \pi a_3 = -2 - (2, -3) \cdot (1, 2) &&= 2 \\ d_4 &= c_4 - \pi a_4 = -2 - (2, -3) \cdot (-4, -3) &&= -3 \\ d_5 &= c_5 - \pi a_5 = -2 - (2, -3) \cdot (2, -1) &&= -9 \end{aligned}$$

Since both d_4 and d_5 are negative, we can improve the objective function by letting either of the nonbasic variables x_4 or x_5 increase from zero. On the other hand, since d_3 is not negative, we cannot improve the objective by letting nonbasic variable x_3 increase (indeed, we can only make the objective worse).

Just to double-check one of the reduced costs, we can substitute the our previously derived expression for the x_4 ray into the objective function:

$$z(\theta) = cx = (-1, -1, -2, -2, -2) \cdot (6 + 2\theta, 2 - \theta, 0, \theta, 0) = -8 - 3\theta.$$

This is precisely $\bar{z} + \theta d_4$, as the derivation has predicted it should be.

5. Algebraic Interpretation of Moving to the Next Vertex: The Minimum Ratio Test

When looking at the graph of a linear program, you have no trouble following an edge of the feasible region from one of its vertices to the other. We now show how an algebraic simplex method can carry out an analogous operation, by following a ray of solutions from one basic feasible solution to another. Since there is no graph to follow, we will have to derive some formulas that indicate which basic feasible solution is reached; specifically, we will show that it is necessary to identify a certain "minimum ratio" corresponding to a basic variable that falls to zero.

This section begins, like previous ones, with a derivation of formulas and an example. We then present some more examples for two special cases—unboundedness and degeneracy—that present additional complications.

5.1 Feasibility along a ray of solutions

A little more geometry will help to motivate the formulas we want to derive here. Imagine that you are at a vertex, and that you identify an edge along which the objective function improves. Because the objective is linear, the further you move along the edge, the more improvement in the objective you get. The only thing that stops you from moving along the edge indefinitely is that you reach the opposite vertex; if you moved any further, you would go past the vertex and outside the feasible region.

Now consider the algebraic analogue. You start at a basic feasible solution, and identify a nonbasic variable x_p whose reduced cost is negative. Next you solve $B y_{\mathcal{B}} = a_p$. Then you can move along the ray of solutions defined by the basis and x_p,

$$\begin{aligned} x_p &= \theta \geq 0, \\ x_i &= \bar{x}_i - \theta y_i, \quad \text{for all } i \in \mathcal{B}, \end{aligned}$$

by steadily increasing θ. The objective value will steadily drop as a result. Obviously, you would like to increase θ as much as possible, so that the objective drops as much as possible.

What can prevent you from increasing θ? If you increase it too much, some of the variables x_i may become negative! In other words, moving too far along the ray of solutions may cause you to leave the feasible region (where all $x_i \geq 0$), just as in the geometric analogy.

The question we need to answer is thus the following: How large can we make the value θ of the entering variable while preserving nonnegativity of the basic variables? In symbols, we want the largest possible θ such that

$$x_i = \bar{x}_i - \theta y_i \geq 0, \quad \text{for all } i \in \mathcal{B}.$$

Clearly if $y_i \leq 0$, however, then x_i will only *increase* as θ increases—it will be nonnegative for any $\theta \geq 0$. Thus we only need to consider the condition that

$$\bar{x}_i - \theta y_i \geq 0, \quad \text{for all } i \in \mathcal{B} \text{ such that } y_i > 0.$$

Rearranging this inequality, we see that θ must satisfy

$$\theta \leq \bar{x}_i / y_i, \quad \text{for all } i \in \mathcal{B} \text{ such that } y_i > 0.$$

Since θ must be $\leq$ all of these ratios $\bar{x}_i / y_i$, we can say equivalently that it must be less than or equal to their minimum:

$$\theta \leq \Theta_p \equiv \min_{i \in \mathcal{B}} \{\bar{x}_i / y_i : y_i > 0\}.$$

We have shown that the solutions along the ray are feasible for $0 \leq \theta \leq \Theta_p$, where Θ_p is the ***minimum ratio*** of $\bar{x}_i$ to y_i for all positive entries in $y_{\mathcal{B}}$.

The geometric analogy can now be made explicit:

> The line segment of solutions given by $0 \leq x_p \leq \Theta_p$
> corresponds to an edge of the feasible region.

Naturally, this segment corresponds to an improving (or "upward") edge if and only if also the reduced cost $d_p < 0$.

5.2 Moving to another basic feasible solution

Suppose now that we let the variable x_p take on the largest value of θ at which feasibility is preserved. Thus

$$\theta = \Theta_p = \bar{x}_q / y_q$$

for some index $q \in \mathcal{B}$ that achieves the minimum ratio above.

Substituting into the formula for the ray of solutions,

$$x_q = \bar{x}_q - \theta y_q = \bar{x}_q - (\bar{x}_q / y_q) y_q = 0.$$

In words, when θ reaches its maximum feasible value, there is a basic variable — the one giving the minimum ratio x_q / y_q — that exactly falls to zero. Indeed, the ratio x_i / y_i just represents the level of θ that results in x_i being exactly zero; the minimum of these ratios represents the level of θ at which a basic variable *first* becomes zero. It is impossible to increase θ any further than the minimum ratio without this variable, which we are calling x_q, becoming negative.

What is the new feasible solution like at $\theta = \Theta_p$? Potentially the variables that are positive include x_p (which we have increased from zero) and all of the basic variables except x_q (which has just dropped to zero). Hence it appears that we have moved to a new *basic* feasible solution, by adding x_p to the basis and dropping x_q.

If x_p was chosen to have a negative reduced cost d_p, then the objective value drops by d_p for every 1 that θ increases. Thus the new basic feasible solution has an objective value of $\Theta_p d_p$ less than the old one.

The analogy between the algebra and the geometry can now be completed:

> The basic solutions given by $x_p = 0$ and $x_p = \Theta_p$
> correspond to two adjacent vertices
> connected by an edge of the feasible region.

To prove this claim fully, we would have to show that, when x_p is added to the basis and x_q is dropped, the resulting matrix B still has linearly independent columns. A straightforward algebraic argument can establish this fact, however.

5.3 A simple example

Continuing the example from previous sections,

$$\begin{array}{ll} \text{Minimize} & -x_1 - x_2 - 2x_3 - 2x_4 - 2x_5 \\ \text{Subject to} & x_1 - 2x_2 + x_3 - 4x_4 + 2x_5 = 2 \\ & x_1 - x_2 + 2x_3 - 3x_4 - x_5 = 4 \\ & x_1, \ldots, x_5 \geq 0 \end{array}$$

suppose x_1 and x_2 are basic. Then we have

$$\bar{x}_{\mathcal{B}} = (\bar{x}_1, \bar{x}_2) = (6, 2),$$

with objective value $\bar{z} = -8$. We have determined that $d_4 = -3$, so x_4 can enter the new basis. Next we solve $By_{\mathcal{B}} = a_4$ to get

$$y_{\mathcal{B}} = (y_1, y_2) = (-2, 1).$$

Since y_1 is negative and only y_2 is positive, the minimum ratio is trivially established to be

$$\Theta_4 = \min_{i \in \mathcal{B}} \{\bar{x}_i / y_i : y_i > 0\} = \bar{x}_2 / y_2 = 2/1 = 2.$$

Hence a better basic feasible solution ought to be attained by adding x_4 to the basis, and dropping x_2. Indeed, it is easy to check that the resulting basic solution has $\bar{x}_{\mathcal{B}} = (\bar{x}_1, \bar{x}_4) = (10, 2)$, with an improved objective of $\bar{z} = -14$.

To further see what is going on here, look at the explicit formula for the ray of solutions that we calculated previously:

$$x = (x_1, x_2, x_3, x_4, x_5) = (6 + 2\theta, 2 - \theta, 0, \theta, 0).$$

Because $y_1 < 0$, $x_1 = 6 + 2\theta$ is increasing along the ray; it will be nonnegative for any θ. On the other hand, since $y_2 > 0$, $x_2 = 2 - \theta$ is decreasing along the ray; at the value of $\theta = \Theta_4 = 2$, this variable decreases to zero. Substituting $\theta = 2$ in the formula for the ray, we get exactly $x = (10, 0, 0, 2, 0)$, the new basic feasible solution.

By our previous reasoning, the objective value should have been changed by the minimum ratio times the reduced cost, or $\Theta_4 d_4 = 2 \cdot (-3) = -6$. That is just what happened; the objective decreased from -8 to -14.

5.4 The degenerate case

We have been claiming that a basis change as determined by the minimum ratio test must reduce the objective value. The reasoning is that the value changes by $\Theta_p d_p$, where d_p is chosen to be negative, and Θ_p is positive.

If some of the elements of $\bar{x}_{\mathcal{B}}$ are zero, however, then it is possible that

$$\bar{x}_q = 0 \quad \text{and} \quad y_q > 0$$

for some index q. In this case the minimum ratio is necessarily $\Theta_p = \bar{x}_q / y_q = 0$—how could it be any smaller? Consequently $\Theta_p d_p = 0$, and it would seem that the objective does not change at all.

The problem here can be appreciated by looking at the formula for the basic variable x_q along the ray of solutions:

$$x_q = \bar{x}_q - \theta y_q = 0 - \theta y_q, \quad y_q > 0.$$

The value of x_q starts out at zero, and for *any* positive value of θ it becomes negative. Thus the maximum value of θ that preserves a feasible solution is $\Theta_p = 0$; there is no way to increase the nonbasic variable x_p to any positive value without forcing x_q to an unacceptable negative value.

As a specific illustration, suppose that in our previous example we have x_1 and x_3 basic. Thus

$$\bar{x}_{\mathcal{B}} = (\bar{x}_1, \bar{x}_3) = (0, 2),$$

giving a basic feasible solution of $\bar{x} = (0, 0, 2, 0, 0)$, with objective value $\bar{z} = -4$. The price vector and reduced costs are computed by the usual formulas:

$$B^T \pi = c_{\mathcal{B}} \quad \Leftrightarrow \quad \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \pi = \begin{bmatrix} -1 \\ -2 \end{bmatrix} \quad \Leftrightarrow \quad \pi = (0, -1)$$

$$\begin{aligned} d_2 &= c_2 - \pi a_2 = -1 - (0, -1) \cdot (-2, -1) = -2 \\ d_4 &= c_4 - \pi a_4 = -2 - (0, -1) \cdot (-4, -3) = -5 \\ d_5 &= c_5 - \pi a_5 = -2 - (0, -1) \cdot (2, -1) = -3 \end{aligned}$$

All of the reduced costs are negative, so any of the three nonbasic variables can be chosen. If you choose x_5, then

$$B y_{\mathcal{B}} = a_5 \quad \Leftrightarrow \quad \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} y_{\mathcal{B}} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \quad \Leftrightarrow \quad y_{\mathcal{B}} = (y_1, y_3) = (5, -3).$$

The only positive component of $y_{\mathcal{B}}$ is y_1, so the minimum ratio is $x_1 / y_1 = 0/5 = 0$. Hence x_5 cannot be increased at all without x_1 becoming negative.

Suppose that you decide to bring x_5 into the basis anyway, while dropping x_1. Then the new basis gives $\bar{x}_{\mathcal{B}} = (\bar{x}_5, \bar{x}_3) = (0, 2)$; the basic feasible solution is still $\bar{x} = (0, 0, 2, 0, 0)$, and the objective value is still -4. All you have done is to move from one basis for the degenerate solution $\bar{x}$ to another basis for this solution. This is a ***degenerate iteration.***

If you instead choose x_2, then you get a different $y_{\mathcal{B}}$:

$$B y_{\mathcal{B}} = a_2 \quad \Leftrightarrow \quad \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} y_{\mathcal{B}} = \begin{bmatrix} -2 \\ -1 \end{bmatrix} \quad \Leftrightarrow \quad y_{\mathcal{B}} = (y_1, y_3) = (-3, 1).$$

Here the only positive component of $y_{\mathcal{B}}$ is y_3, and the minimum ratio is $x_3 / y_3 = 2/1 = 2$. Thus x_2 can be increased up to a value of 2, before x_3 falls to zero. When x_2 replaces x_3 in the basis, we get a new basic solution at the next iteration of $\bar{x}_{\mathcal{B}} = (\bar{x}_1, \bar{x}_2) = (6, 2)$, and the objective value declines to $\bar{z} = -8$.

This example is typical of what happens when one or more basic variables are at a value of zero. Even though you pick a nonbasic variable that has a negative reduced cost, you may not be able to increase it more than zero before some basic variable becomes negative. In that case you can make a basis change, but you will just arrive at another basis for the same degenerate solution, and the objective value will not be improved.

It may happen that if you pick some other nonbasic variable that has a negative reduced cost, then you will be able to increase it a positive amount, and hence you will be able to improve the objective value. But can you be sure that such a nonbasic variable exists? This is an issue that we will come back to, once we have shown that the simplex method works in the nondegenerate case.

5.5 The unbounded case

Another assumption we have been making is that the minimum ratio,

$$\Theta_p = \min_{i \in \mathcal{B}} \{\bar{x}_i / y_i : y_i > 0\},$$

is a well-defined, finite number. But what if $y_i \leq 0$ for every one of the components of $y_{\mathcal{B}}$? Then we would be taking the minimum over an empty set!

It's not hard to see what is happening here. Recall that the ray of solutions extending from a basic feasible solution is given by

$$\begin{aligned} x_p &= \theta \geq 0, \\ x_i &= \bar{x}_i - \theta y_i, \quad \text{for each } i \in \mathcal{B}. \end{aligned}$$

If all $y_i \leq 0$, then not only is $\bar{x}_i \geq 0$, but $x_i \geq 0$ for any positive value of θ—no matter how large. As a result, x_p can be increased as much as you like, without any basic variable going negative and making the solution infeasible.

For each 1 that x_p increases from zero, however, the objective value declines by an amount equal to the reduced cost, d_p. Hence if x_p can be increased any amount, the objective can be made to decline any amount. The linear program in this case is said to be ***unbounded.*** You can think of it as having an objective that can be minimized all the way down to $-\infty$.

As an example, consider for a final time the preceding two-constraint linear program with basic variables x_1 and x_2. The basic feasible solution is $\bar{x}_{\mathcal{B}} = (\bar{x}_1, \bar{x}_2) = (6, 2)$, or $\bar{x} = (6, 2, 0, 0, 0)$. We have seen before that $d_5 = -9$, so that x_5 can be picked to enter the basis. Then

$$B y_{\mathcal{B}} = a_5 \quad \Leftrightarrow \quad \begin{bmatrix} 1 & -2 \\ 1 & -1 \end{bmatrix} y_{\mathcal{B}} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \quad \Leftrightarrow \quad y_{\mathcal{B}} = (y_1, y_2) = (-4, -3).$$

No component of $y_{\mathcal{B}}$ is positive, and hence this linear program is unbounded.

To verify this conclusion, you can check that the ray of solutions given by $x_5 = \theta$ and $x_i = \bar{x}_i - \theta y_i$ is

$$\bar{x} = (6 + 4\theta, 2 + 3\theta, 0, 0, \theta).$$

Obviously $\bar{x}$ remains feasible as θ increases. Moreover, plugging into the objective function, we get $c\bar{x} = -8 - 9\theta$, which obviously goes off to $-\infty$ as θ increases.

6. The Simplex Algorithm

The preceding sections have developed a procedure for finding an optimal solution to any linear program in the standard algebraic form. This procedure is the ***simplex*** algorithm or method.

We begin by providing a full outline and then a full algebraic description of the algorithm. Then we prove that, under certain assumptions, it really does always find an optimal solution (if any exists). We conclude with an extended example.

6.1 Outline of the algorithm

The simplex algorithm consists of a series of ***iterations.*** The first iteration of the algorithm starts from some given basic feasible solution, and moves to a new basic feasible solution that achieves a better value of the objective function. The second iteration starts from the basic feasible solution that was found at the end of the first iteration, and moves to a newer basic feasible solution that achieves an even better value of the objective. The third iteration starts from the basic feasible solution that was found at the end of the second iteration, ... and so forth. Each iteration works in the same way; it starts from the basic feasible solution provided by the previous iteration, and follows a certain procedure to find a better such solution.

The simplex algorithm does not iterate indefinitely. After some finite number of iterations, it either arrives at a basic feasible solution that cannot be improved upon, or it discovers that the linear program is unbounded.

The steps of a simplex iteration have been developed in the previous sections. We summarize them first in outline form, along with the geometric analogues that motivated them:

Algebraic	*Geometric*
Given: A basic feasible solution satisfying $B\bar{x}_{\mathcal{B}} = b$, $\bar{x}_{\mathcal{B}} \geq 0$.	Given: A vertex of the feasible region.
(1) Compute reduced costs $d_j = c_j - \pi a_j$, where $\pi B = c_{\mathcal{B}}$.	(1) Test whether edges from this vertex head up or down.
(2) If all $d_j \geq 0$, stop with an optimal solution.	(2) Are there any upward edges? If *no*, stop with an optimal solution.
(3) Otherwise, choose p such that $d_p < 0$.	(3) If *yes*, pick an upward edge.
(4) Solve $By_{\mathcal{B}} = a_p$.	(4) Travel up the chosen edge.

(5) If all $y_i \le 0$, stop with an unbounded solution.	(5) Do you come to another vertex? If *no*, stop with an unbounded solution.
(6) Otherwise, choose q that achieves a minimum ratio: $\bar{x}_q/y_q = \Theta_p = \min\{-\bar{x}_i/y_i : y_i > 0\}$.	(6) If *yes*, proceed up the edge as far as you can.
(7) Compute the new feasible solution: $\bar{x}_p \leftarrow \Theta_p$, $\bar{x}_{\mathcal{B}} \leftarrow \bar{x}_{\mathcal{B}} - \Theta_p y_{\mathcal{B}}$.	(7) Stop at the new vertex that you reach.
(8) Change the basis: Make the qth variable nonbasic, and the pth variable basic. Repeat from (1).	(8) Now you've found a new vertex higher the one you started with. So, begin the steps anew.

6.2 Statement of the algorithm

In more formal terms, the simplex algorithm — or at least the version of it developed here — is a method for solving linear programs in the following form:

$$\begin{array}{ll} \text{Minimize} & cx \\ \text{Subject to} & Ax = b \\ & x \ge 0 \end{array}$$

The algorithm starts with a square nonsingular submatrix B of A — and a corresponding subset $\mathcal{B}$ of basic variables — that give a basic solution $\bar{x}$,

$$\begin{aligned} \bar{x}_{\mathcal{N}} &= 0, \\ B\bar{x}_{\mathcal{B}} &= b, \end{aligned}$$

that is feasible: $\bar{x}_{\mathcal{B}} \ge 0$. We write $c_{\mathcal{B}}$ for the vector of elements from c that correspond to the basic variables, and a_p for the column of A that corresponds to the nonbasic variable x_p.

The algorithm executes the following steps repeatedly, as many times as necessary until it stops:

(1) *Solve* $\pi B = c_{\mathcal{B}}$ (equivalently, $B^T\pi = c_{\mathcal{B}}$) for the m-vector π.

(2) *Test for optimality:* If every nonbasic variable x_j, $j \in \mathcal{N}$, satisfies $d_j = c_j - \pi a_j \ge 0$, then $\bar{x}$ is an optimal solution. *Stop.*

(3) *Choose entering variable:* Pick a nonbasic variable x_p, $p \in \mathcal{N}$, such that $d_p = c_p - \pi a_p < 0$.

(4) *Solve* $By_{\mathcal{B}} = a_p$ for the m-vector $y_{\mathcal{B}}$.

(5) *Test for unboundedness:* If every basic variable x_i has $y_i \le 0$, then the objective can be decreased without bound. *Stop.*

(6) *Choose leaving variable:* Pick a basic variable x_q , $q \in \mathcal{B}$, such that $\Theta_p = x_q/y_q$ minimizes x_i/y_i over all $i \in \mathcal{B}$ such that $y_i > 0$.

(7) *Update the solution:* Reset $\bar{x}_p \leftarrow \Theta_p$, and $\bar{x}_\mathcal{B} \leftarrow \bar{x}_\mathcal{B} - \Theta_p y_\mathcal{B}$.

(8) *Change the basis:* Replace q by p in $\mathcal{B}$, and replace the column corresponding to the qth variable with the column corresponding to the pth variable in B.

Notice that the steps of an iteration fall into three main parts. Steps (1)–(3) are concerned with choosing an entering variable; (4)–(6) are concerned with choosing a leaving variable; and (7)–(8) make the resulting change of basis.

6.3 Proof of the algorithm

We have argued that the simplex algorithm ought to work, because its steps make a lot of sense. This is not enough, however, to guarantee that it really will find an optimal solution to any linear program that has one. To ensure that the algorithm can be relied upon in all cases, we need to offer some kind of *proof* that it works. The same must be done for any optimization algorithm, before it can by used confidently. Thus the proofs given below are examples of certain kinds of arguments that must be made frequently in optimization.

To avoid inessential complications, we begin by making two assumptions about the linear programs that are to be solved:

> At least one basic feasible solution can be found.
>
> Every basic feasible solution is nondegenerate, in the sense that every basic variable takes a strictly positive value.

In the following section, we will see that the first assumption is easily accommodated. The second assumption is much more problematical; without it, some degenerate iterations are unavoidable, but fortunately there are both theoretical and practical ways of insuring that not all iterations are degenerate and that an optimal basic solution is eventually reached.

The first assumption guarantees that, given any linear program (that has been put into the form we require), the simplex algorithm can be started up. There are then two aspects to a proof: to show that the algorithm eventually stops, and to show that it can only stop with an optimal or unbounded solution.

Proof of termination. We have already observed that each full iteration of the simplex algorithm steps from a basic feasible solution at which the objective value is $\bar{z}$, to a basic feasible solution at which the objective is $\bar{z} + \Theta_p d_p$, where

$$\Theta_p = \bar{x}_q/y_q = \min_{i \in \mathcal{B}} \{-\bar{x}_i/y_i : y_i > 0\}$$

is the minimum ratio defined by the entering variable x_p. Step (3) of the algorithm chooses $d_p < 0$. The nondegeneracy assumption ensures that $x_i > 0$ for all $i \in \mathcal{B}$, and hence that $x_i/y_i > 0$ whenever $y_i > 0$. Thus Θ_p is the minimum of

positive ratios, implying that $\Theta_p > 0$. We conclude that $\Theta_p d_p < 0$, from which it follows that the objective value *decreases* at every full iteration.

Since each basis is defined by a choice of m basic variables from the linear program's n variables, there can be only a finite number of different bases. If the algorithm never stopped, therefore, it would eventually run out of different bases to visit, and it would have to visit one of them twice.

Since the objective is decreasing at every iteration, however, the simplex algorithm cannot possibly visit any basis more than once. Hence it is contradictory to assume that the algorithm never stops. The only consistent conclusion left to us is that it always eventually terminates, in step (2) or step (5), after finitely many iterations.

Proof of unboundedness. If the algorithm stops in step (5), then it demonstrates the existence of a ray of feasible solutions,

$$\begin{aligned} x_p &= \theta \geq 0, \\ x_i &= \bar{x}_i - \theta y_i, \quad y_i \leq 0, \end{aligned}$$

along which the objective value $z(\theta) = \bar{z} + \theta d_p$ is continually decreasing. It follows that the linear program is unbounded.

Proof of optimality. Imagine now that the algorithm has stopped in step (2), and let x^* denote the solution it has found. Since x^* is a basic feasible solution, we have $Bx^*_{\mathcal{B}} = b$, $x^*_{\mathcal{B}} \geq 0$, and $x^*_{\mathcal{N}} = 0$. The current objective value is cx^*, or $c_{\mathcal{B}} x^*_{\mathcal{B}}$. Let π^* be the price vector that was determined in step (1), and let the reduced costs be $d^*_j = c_j - \pi^* a_j$ for all $j \in \mathcal{N}$.

To show that x^* is optimal, we need only show it achieves at least as good an objective value as *any* other feasible solution — basic or otherwise. So, let x' denote some other feasible solution: $Ax' = b$, $x' \geq 0$.

Define $x'_{\mathcal{B}}$ to be the part of x' corresponding to the basic variables in x^*, and $x'_{\mathcal{N}}$ to be the part corresponding to the nonbasic variables in x^*. Then the equations $Ax' = b$ can be solved for x' as follows:

$$\begin{aligned} Ax' = b \quad &\Leftrightarrow \quad Bx'_{\mathcal{B}} + \textstyle\sum_{j\in\mathcal{N}} a_j x'_j = b \\ &\Leftrightarrow \quad x'_{\mathcal{B}} = B^{-1}(b - \textstyle\sum_{j\in\mathcal{N}} a_j x'_j) \\ &\Leftrightarrow \quad x'_{\mathcal{B}} = x^*_{\mathcal{B}} - \textstyle\sum_{j\in\mathcal{N}} (B^{-1}a_j) x'_j \end{aligned}$$

Substituting this expression for $x'_{\mathcal{B}}$ in the formula cx' for the objective function, we get:

$$\begin{aligned} cx' &= c_{\mathcal{B}} x'_{\mathcal{B}} + \textstyle\sum_{j\in\mathcal{N}} c_j x'_j \\ &= c_{\mathcal{B}}(x^*_{\mathcal{B}} - \textstyle\sum_{j\in\mathcal{N}} (B^{-1}a_j) x'_j) + \textstyle\sum_{j\in\mathcal{N}} c_j x'_j \\ &= c_{\mathcal{B}} x^*_{\mathcal{B}} + \textstyle\sum_{j\in\mathcal{N}} (c_j - c_{\mathcal{B}} B^{-1} a_j) x'_j \\ &= c_{\mathcal{B}} x^*_{\mathcal{B}} + \textstyle\sum_{j\in\mathcal{N}} (c_j - \pi^* a_j) x'_j \\ &= c_{\mathcal{B}} x^*_{\mathcal{B}} + \textstyle\sum_{j\in\mathcal{N}} d^*_j x'_j \end{aligned}$$

Since the algorithm stopped in step (2), however, it must be that $d_j^* \geq 0$ for all nonbasic x_j. And, since x' is feasible, $x'_j \geq 0$. Thus the sum in the last line above is nonnegative, from which it follows that $cx' \geq c_B x_B^*$.

We have shown that the objective value at x' cannot be less than the objective value at x^*. Since this is true for any feasible x', the solution x^* must be minimal, which is what we set out to prove.

6.4 An example

As an illustration of several consecutive iterations of the simplex algorithm, we introduce the following small linear program:

$$\begin{array}{lrcrcrcrcl}
\text{Minimize} & -3x_1 & - & x_2 & - & 3x_3 & & & & \\
\text{Subject to} & 2x_1 & + & x_2 & + & x_3 & + & x_4 & & = 2 \\
& x_1 & + & 2x_2 & + & 3x_3 & & & + x_5 & = 5 \\
& 2x_1 & + & 2x_2 & + & x_3 & & & + x_6 & = 6 \\
& & & & & & & & x_1, \ldots, x_6 & \geq 0
\end{array}$$

The relevant vectors and matrices are

$$c = \begin{bmatrix} -3 & -1 & -3 & 0 & 0 & 0 \end{bmatrix},$$

$$A = \begin{bmatrix} 2 & 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 3 & 0 & 1 & 0 \\ 2 & 2 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 5 \\ 6 \end{bmatrix}.$$

An obvious starting point is the basis that consists of the variables x_4, x_5 and x_6. Then the basic feasible solution, with its associated basis matrix and objective coefficients, is

$$c_B = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\bar{x}_B = \begin{bmatrix} 2 & 5 & 6 \end{bmatrix} = \begin{bmatrix} \bar{x}_4 & \bar{x}_5 & \bar{x}_6 \end{bmatrix},$$

with nonbasic variables $\bar{x}_1 = \bar{x}_2 = \bar{x}_3 = 0$. The objective value is $\bar{z} = 0$.

1st Iteration.

(1) $B^T\pi = c_{\mathcal{B}} \;\Rightarrow\; \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \pi = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \;\Rightarrow\; \pi = (0,0,0).$

(2) $d_1 = c_1 - \pi a_1 = -3 - (0,0,0)\cdot(2,1,2) = -3,$
$d_2 = c_2 - \pi a_2 = -1 - (0,0,0)\cdot(1,2,2) = -1,$
$d_3 = c_3 - \pi a_3 = -3 - (0,0,0)\cdot(1,3,1) = -3.$

Not optimal, since there are negative reduced costs.

(3) Pick x_2, with $d_2 = -1 < 0$, to enter the basis.
(We could also have picked x_1 or x_3.)

(4) $B y_{\mathcal{B}} = a_p \;\Rightarrow\; \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} y_{\mathcal{B}} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$

$$\Rightarrow \quad y_{\mathcal{B}} = (y_4, y_5, y_6) = (1,2,2).$$

(5) Not unbounded, since there are positive entries in $y_{\mathcal{B}}$.

(6) $y_4 > 0$: $\bar{x}_4/y_4 = 2/1 = 2$
$y_5 > 0$: $\bar{x}_5/y_5 = 5/2 = 2.5$
$y_6 > 0$: $\bar{x}_6/y_6 = 6/2 = 3$

Pick x_4, with minimum ratio $\Theta_2 = \bar{x}_4/y_4 = 2$, to leave the basis.

(7) $\bar{x}_2 \leftarrow \Theta_2 = 2;$

$$\begin{bmatrix} \bar{x}_4 \\ \bar{x}_5 \\ \bar{x}_6 \end{bmatrix} \leftarrow \bar{x}_{\mathcal{B}} - \Theta_2 y_{\mathcal{B}} = \begin{bmatrix} 2 \\ 5 \\ 6 \end{bmatrix} - 2\begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}.$$

(8) The new basic feasible solution is

$$c_{\mathcal{B}} = \begin{bmatrix} -1 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix},$$

$$\bar{x}_{\mathcal{B}} = \begin{bmatrix} 2 & 1 & 2 \end{bmatrix} = \begin{bmatrix} \bar{x}_2 & \bar{x}_5 & \bar{x}_6 \end{bmatrix},$$

with nonbasic variables $\bar{x}_1 = \bar{x}_3 = \bar{x}_4 = 0$. The objective value is $\bar{z} = -2$.

2nd Iteration.

(1) $B^T\pi = c_{\mathcal{B}} \;\Rightarrow\; \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\pi = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \;\Rightarrow\; \pi = (-1,0,0).$

(2) $d_1 = c_1 - \pi a_1 = -3 - (-1,0,0)\cdot(2,1,2) = -1,$
$d_3 = c_3 - \pi a_3 = -3 - (-1,0,0)\cdot(1,3,1) = -2,$
$d_4 = c_4 - \pi a_4 = 0 - (-1,0,0)\cdot(1,0,0) = 1.$

Not optimal, since there are negative reduced costs.

(3) Pick x_3, with $d_3 = -2 < 0$, to enter the basis.
(We could also have picked x_1.)

(4) $By_{\mathcal{B}} = a_p \;\Rightarrow\; \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} y_{\mathcal{B}} = \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}$

$$\Rightarrow \quad y_{\mathcal{B}} = (y_2, y_5, y_6) = (1,1,-1).$$

(5) Not unbounded, since there are positive entries in $y_{\mathcal{B}}$.

(6) $y_2 > 0$: $\bar{x}_2/y_2 = 2/1 = 2$
$y_5 > 0$: $\bar{x}_5/y_5 = 1/1 = 1$
$y_6 \le 0$: *no ratio*

Pick x_5, with minimum ratio $\Theta_3 = \bar{x}_5/y_5 = 1$, to leave the basis.

(7) $\bar{x}_3 \leftarrow \Theta_3 = 1;$

$$\begin{bmatrix} \bar{x}_2 \\ \bar{x}_5 \\ \bar{x}_6 \end{bmatrix} \leftarrow \bar{x}_{\mathcal{B}} - \Theta_3 y_{\mathcal{B}} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - 1\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}.$$

(8) The new basic feasible solution is

$$c_{\mathcal{B}} = \begin{bmatrix} -1 & -3 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 1 & 1 & 0 \\ 2 & 3 & 0 \\ 2 & 1 & 1 \end{bmatrix},$$

$$\bar{x}_{\mathcal{B}} = \begin{bmatrix} 1 & 1 & 3 \end{bmatrix} = \begin{bmatrix} \bar{x}_2 & \bar{x}_3 & \bar{x}_6 \end{bmatrix},$$

with nonbasic variables $\bar{x}_1 = \bar{x}_4 = \bar{x}_5 = 0$. The objective value is $\bar{z} = -4$.

3rd Iteration.

(1) $B^T\pi = c_{\mathcal{B}} \;\Rightarrow\; \begin{bmatrix} 1 & 2 & 2 \\ 1 & 3 & 1 \\ 0 & 0 & 1 \end{bmatrix}\pi = \begin{bmatrix} -1 \\ -3 \\ 0 \end{bmatrix} \;\Rightarrow\; \pi = (3,-2,0).$

(2) $d_1 = c_1 - \pi a_1 = -3 - (3,-2,0)\cdot(2,1,2) = -7,$
$d_4 = c_4 - \pi a_4 = 0 - (3,-2,0)\cdot(1,0,0) = -3,$
$d_5 = c_5 - \pi a_5 = 0 - (3,-2,0)\cdot(0,1,0) = 2.$

Not optimal, since there are negative reduced costs.

(3) Pick x_1, with $d_1 = -7 < 0$, to enter the basis.
(We could also have picked x_4 to re-enter the basis.)

(4) $By_{\mathcal{B}} = a_p \;\Rightarrow\; \begin{bmatrix} 1 & 1 & 0 \\ 2 & 3 & 0 \\ 2 & 1 & 1 \end{bmatrix} y_{\mathcal{B}} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}$

$$\Rightarrow \quad y_{\mathcal{B}} = (y_2, y_3, y_6) = (5,-3,-5).$$

(5) Not unbounded, since there is a positive entry in $y_{\mathcal{B}}$.

(6) $y_2 > 0$: $\bar{x}_2/y_2 = 1/5 = 0.2$
$y_3 \le 0$: *no ratio*
$y_6 \le 0$: *no ratio*

Pick x_2, with minimum ratio $\Theta_1 = \bar{x}_2/y_2 = 0.2$, to leave the basis.

(7) $\bar{x}_1 \leftarrow \Theta_1 = 0.2;$

$$\begin{bmatrix} \bar{x}_2 \\ \bar{x}_3 \\ \bar{x}_6 \end{bmatrix} \leftarrow \bar{x}_{\mathcal{B}} - \Theta_1 y_{\mathcal{B}} = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix} - 0.2\begin{bmatrix} 5 \\ -3 \\ -5 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 1.6 \\ 4.0 \end{bmatrix}.$$

(8) The new basic feasible solution is

$$c_{\mathcal{B}} = \begin{bmatrix} -3 & -3 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 0 \\ 2 & 1 & 1 \end{bmatrix},$$

$$\bar{x}_{\mathcal{B}} = \begin{bmatrix} 0.2 & 1.6 & 4.0 \end{bmatrix} = \begin{bmatrix} \bar{x}_1 & \bar{x}_3 & \bar{x}_6 \end{bmatrix},$$

with nonbasic variables $\bar{x}_2 = \bar{x}_4 = \bar{x}_5 = 0$.
The objective value is $\bar{z} = -5.4$.

4th Iteration.

(1) $B^T\pi = c_{\mathcal{B}} \;\Rightarrow\; \begin{bmatrix} 2 & 1 & 2 \\ 1 & 3 & 1 \\ 0 & 0 & 1 \end{bmatrix}\pi = \begin{bmatrix} -3 \\ -3 \\ 0 \end{bmatrix} \;\Rightarrow\; \pi = (-1.2, -0.6, 0).$

(2) $d_2 = c_2 - \pi a_2 = -1 - (-1.2, -0.6, 0) \cdot (1, 2, 2) = 1.4,$
$d_4 = c_4 - \pi a_4 = 0 - (-1.2, -0.6, 0) \cdot (1, 0, 0) = 1.2,$
$d_5 = c_5 - \pi a_5 = 0 - (-1.2, -0.6, 0) \cdot (0, 1, 0) = 0.6.$

Stop with optimal solution, since all reduced costs are nonnegative.

Here is a summary of the iterations:

		Basic solution						
	Basis	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_3$	$\bar{x}_4$	$\bar{x}_5$	$\bar{x}_6$	$\bar{z}$
1	$x_4\ x_5\ x_6$	0.0	0.0	0.0	2.0	5.0	6.0	0.0
2	$x_2\ x_5\ x_6$	0.0	2.0	0.0	0.0	1.0	2.0	−2.0
3	$x_2\ x_3\ x_6$	0.0	1.0	1.0	0.0	0.0	3.0	−4.0
4	$x_1\ x_3\ x_6$	0.2	0.0	1.6	0.0	0.0	4.0	−5.4

Nonbasic variable x_2 entered the basis at the first iteration, but was forced to leave at the third iteration. In general, individual variables will enter and leave the basis numerous times during the simplex algorithm's progress. It is only the full set $\mathcal{B}$ of basic variable indices that always changes.

7. Finding a Feasible Solution

Our development up to this point has assumed that we can always find a basic feasible solution from which to start the simplex algorithm. This section shows how the simplex algorithm itself can be used to find such a solution, or to demonstrate that no feasible solution exists. Combined with the algorithm of the previous section, the result is a "two-phase" approach to finding a feasible, optimal basis.

Later we will show that finding a feasible solution to $Ax = b$, $x \geq 0$ is "just as hard" as finding an optimal solution, in a sense that can be made precise. Thus it makes sense to use the same algorithm for finding feasible solutions as for finding optimal ones; it is unlikely that there will be an easier way.

7.1 The two-phase approach

For some linear programs of simple structure, there is an obvious basic feasible solution. As an example, there is the prototypical maximum revenue production model,

$$\begin{array}{lll} \text{Maximize} & \sum_{j=1}^{n} c_j x_j & \\ \text{Subject to} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, & i = 1, \ldots, m \\ & x_j \geq 0, & j = 1, \ldots, n \end{array}$$

where b_i is the amount of scarce resource i available. This LP can be converted to the standard algebraic form, as you have seen, by the addition of slack variables:

$$\begin{array}{lll} \text{Maximize} & \sum_{j=1}^{n} c_j x_j & \\ \text{Subject to} & \sum_{j=1}^{n} a_{ij} x_j + s_i = b_i, & i = 1, \ldots, m \\ & x_j \geq 0, & j = 1, \ldots, n \\ & s_i \geq 0, & i = 1, \ldots, m \end{array}$$

There are m constraints, and we can take the m slack variables s_i to be basic, leaving the n variables x_j nonbasic. The basis matrix is then just

$$B = I = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix},$$

the matrix of coefficients of the s_i variables; it is an "identity" matrix whose columns are trivially independent for any m. The associated basic solution is $\bar{x}_j = 0$, $j = 1, \ldots, n$ and $\bar{s}_i = b_i$, $i = 1, \ldots, m$. Such a solution is feasible for any sensible LP of this kind, because all the b_i values are nonnegative — it wouldn't make sense to have a negative amount of a scarce resource available. You can think of this solution as being the obvious starting point at which no products are manufactured, and no resources are used.

In many cases, unfortunately, the identity of basic feasible solutions is far from obvious. A minimum cost blending model, for instance, has the following standard form after the addition of surplus variables:

$$\begin{array}{lll} \text{Minimize} & \sum_{j=1}^{n} c_j x_j & \\ \text{Subject to} & \sum_{j=1}^{n} a_{ij} x_j - s_i = b_i, & i = 1, \ldots, m \\ & x_j \geq 0, & j = 1, \ldots, n \\ & s_i \geq 0, & i = 1, \ldots, m \end{array}$$

The values $b_i \geq 0$ represent the amounts of different qualities required in the blend. The basis of all s_i variables is not feasible in this case, since it gives $\bar{s}_i = -b_i$, $i = 1, \ldots, m$. To provide a feasible blend at all, some positive x_j variables will be needed in the basis, yet there is no simple way to know which ones will be needed. The models used in many applications are a good deal more complicated in structure than this one, and the chance of happening upon a basic feasible solution is even more remote.

To deal with this difficulty, you can imagine that the simplex algorithm is applied in two "phases" as follows:

> ***Phase I:*** Starting from a basis that isn't feasible, iterate until you find a feasible one.
>
> ***Phase II:*** Starting from the feasible basis that you found in phase I, iterate until you find an optimal one.

Clearly phase II can be done by the simplex algorithm as already described. The remainder of this section explains several ways in which the simplex algorithm can also be used for phase I.

7.2 A simple phase I: Minimizing a sum of artificial variables

The problem in phase I is essentially just

$$\begin{array}{lll} \text{Solve} & \sum_{j=1}^{n} a_{ij} x_j = b_i, & i = 1, \ldots, m \\ & x_j \geq 0, & j = 1, \ldots, n \end{array}$$

This is not a linear program, since there is no objective function. We can transform it, however, to an equivalent problem that is a linear program. Furthermore, we can arrange that the equivalent LP has an obvious basic feasible solution; this is essential, since otherwise we would still have the problem of finding a starting point for the simplex algorithm.

First, we can arrange that all $b_i \geq 0$. If any $b_i < 0$, we just transform the problem by multiplying both sides of the ith constraint by -1.

Next, we make a transformation to the following linear program, in which one extra variable has been added to each constraint:

$$\begin{array}{lll} \text{Minimize} & \sum_{i=1}^{m} u_i & \\ \text{Subject to} & \sum_{j=1}^{n} a_{ij} x_j + u_i = b_i, & i = 1, \ldots, m \\ & x_j \geq 0, & j = 1, \ldots, n \\ & u_i \geq 0, & i = 1, \ldots, m \end{array}$$

Much as in the production model example, the obvious basis is $B = I$, with feasible solution $\bar{x}_j = 0$ and $\bar{u}_i = b_i$. However, the objective is now the sum of ***artificial*** variables u_i, in place of the true objective.

The minimum of this linear program cannot be negative, since the objective is the sum of nonnegative variables. Writing x_j^* and u_i^* for the optimal values of the variables, there are two possibilities to consider.

If the minimum is zero, then all of the artificial variables must be at zero; there is no way to have $\sum_{i=1}^m u_i^* = 0$ and $u_i^* \geq 0$, $i = 1, \ldots, m$ without having $u_i^* = 0$ for all $i = 1, \ldots, m$. Thus in fact the x_j^* values satisfy $\sum_{j=1}^n a_{ij} x_j^* + 0 = b_i$, $i = 1, \ldots, m$ and $x_j^* \geq 0$, $j = 1, \ldots, n$, which is a feasible solution to the original linear program's constraints.

If the minimum is greater than zero, on the other hand, then the original constraints have no feasible solution. Indeed, suppose that a feasible solution $\bar{x}_j$, $j = 1, \ldots, n$, does exist. Then by taking also $\bar{u}_i = 0$, $i = 1, \ldots, m$, we get a feasible solution to the phase I linear program above, with objective value $\sum_{i=1}^m u_i = 0$. It follows that, if the minimum objective value for phase I is greater than zero, then feasible solutions to the original constraints cannot possibly exist.

Suppose, then, that we solve the phase I linear program by applying the simplex algorithm, starting from the obvious feasible basis. If it finds a positive minimum, we can stop and declare that no feasible solution is possible. Otherwise, it finds a basic optimal solution that is feasible for the original LP, and we can use that solution as the starting point for phase II.

This form of phase I is attractive for its simplicity. However, it suffers from a number of drawbacks:

- ◇ Whenever the linear program is modified, then — unless the optimal basis remains feasible — the simplex algorithm must be restarted from the basis of all artificial variables.
- ◇ At least one iteration is required for the removal of each artificial variable from the basis.
- ◇ The feasible basis found by phase I may still contain some artificial variables, at degenerate values of 0, which must be dealt with.

If there are slack variables in the formulation, then it is sometimes possible to use a combination of slacks and artificials as a starting basis, cutting down on the number of artificial variables needed. Rather than pursue this and other refinements, however, we move on to a slightly more complicated phase I that is more appropriate for computational purposes.

7.3 A practical phase I: Minimizing the sum of infeasibilities

Suppose that $\bar{x}$ is any solution at all to $Ax = b$. If $\bar{x}_j$ is negative, we can imagine that it represents an "infeasibility" in the amount of $-\bar{x}_j$; for example,

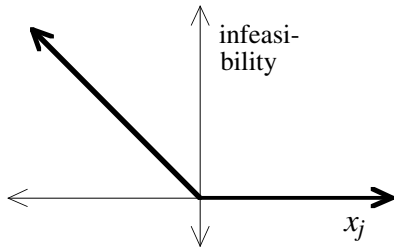
if $\bar{x}_j$ is -10 then there is an infeasibility of 10. On the other hand, if $\bar{x}_j \geq 0$ then it contributes no infeasibility.

To find a feasible solution, we can just minimize the sum of all infeasibilities:

$$\begin{array}{ll} \text{Minimize} & \sum_{j:x_j<0} -x_j \\ \text{Subject to} & Ax = b \end{array}$$

This objective is readily seen to have all the same properties as the artificial objective analyzed above. It is always nonnegative. If a minimum of zero is achieved by some solution, then it is a feasible solution. If the minimum is greater than zero, then no feasible solution exists.

The infeasibility contributed by a variable can be plotted against the value of the variable as follows:



This is not quite a linear function. It is "piecewise" linear: linear with a slope of -1 at negative values of x_j, and linear with a slope of 0 at positive values. Nevertheless, with a few simple modifications, the simplex method can be used to find a minimum of zero or to show that the minimum must be positive.

Imagine that a basic but *infeasible* solution is known: $B\bar{x}_{\mathcal{B}} = b$ and $x_{\mathcal{N}} = 0$, but some components $\bar{x}_i < 0$, $i \in \mathcal{B}$. Finding such a solution is much easier than finding a basic feasible solution; all that is necessary is to identify m linearly independent columns of A. Moreover, at such a solution, the sum of infeasibilities can be written in the following way:

$$\sum_{j:x_j<0} -x_j = f_{\mathcal{B}}x_{\mathcal{B}} + f_{\mathcal{N}}x_{\mathcal{N}},$$

$$\text{where } f_i = \left\{ \begin{array}{rl} -1 & \text{if } x_i < 0 \\ 0 & \text{if } x_i \geq 0 \end{array} \right\} \text{ for all } i \in \mathcal{B}$$

$$\text{and } f_j = 0 \text{ for all } j \in \mathcal{N}.$$

The nonbasic variables all have coefficients of 0 because they are all fixed at feasible values of zero, and will subsequently only be allowed to increase to feasible values. Only the currently basic variables may take infeasible values.

The expression $f_{\mathcal{B}}x_{\mathcal{B}}$ for the sum of infeasibilities looks much like the expression $c_{\mathcal{B}}x_{\mathcal{B}}$ for the true objective function. The only difference is that f_i is not quite constant; it may be either -1 or 0, depending on the sign of x_i. In spite of this difference, however, we can carry out the first part of the simplex algorithm as before, with the values f_i and f_j taking the place of c_i and c_j. Thus we first solve

$$\pi B = f_{\mathcal{B}}$$

for the price vector π, and then compute the reduced costs by

$$d_j = f_j - \pi a_j = -\pi a_j,$$

since f_j is 0 for all $j \in \mathcal{N}$. If we can find a $d_p < 0$, then the sum of infeasibilities can be reduced by letting x_p increase from zero, and adjusting the basic variables accordingly. If all $d_j \geq 0$, then we can conclude that there is no feasible solution to be found.

Once an x_p has been chosen to enter the basis, we can still meaningfully solve $By_{\mathcal{B}} = a_p$, and compute a minimum ratio

$$\Theta_p = \min_{i \in \mathcal{B}} \{-\bar{x}_i / y_i : \bar{x}_i \geq 0 \text{ and } y_i > 0\}.$$

The value of Θ_p indicates how large the entering variable x_p can get before some *nonnegative* basic variable falls to zero. As in the phase II simplex method, we want to consider choosing some x_q to leave the basis such that $\bar{x}_q / y_q$ achieves the above minimum.

There is an extra complication is this case, however, because some of the basic variables are at negative values. Along the ray of solutions defined by x_p,

$$\begin{aligned} x_p &= \theta \geq 0, \\ x_i &= \bar{x}_i - \theta y_i, \end{aligned}$$

those basic variables with $\bar{x}_i < 0$ and $y_i < 0$ are rising toward zero. How big can θ get before one of these infeasible variables reaches zero? None of them will have passed zero yet, provided that

$$x_i = \bar{x}_i - \theta y_i \leq 0 \quad \text{for all } \bar{x}_i < 0 \text{ with } y_i < 0,$$

or equivalently, rearranging the inequality, provided that

$$\theta \leq \min_{i \in \mathcal{B}} \{-\bar{x}_i / y_i : \bar{x}_i < 0 \text{ and } y_i < 0\}.$$

Thus, if we want to increase the entering variable only until one of the basic variables hits zero, either from above or below, then we should pick the minimum ratio to be

$$\Theta_p = \min_{i \in \mathcal{B}} \{-\bar{x}_i / y_i : \bar{x}_i \geq 0 \text{ and } y_i > 0 \text{ or } \bar{x}_i < 0 \text{ and } y_i < 0\}.$$

The leaving variable is chosen as one that achieves this minimum, after which the rest of the iteration — updating the solution and the basis — proceeds as before.

Putting this all together, we have the following steps for phase I of the simplex algorithm, starting from any basis with $B\bar{x}_{\mathcal{B}} = b$, $\bar{x}_{\mathcal{N}} = 0$ and at least one $\bar{x}_i < 0$. They are repeated until the algorithm stops in step (2) or step (8).

(1) *Solve* $\pi B = f_{\mathcal{B}}$ (equivalently, $B^T \pi = f_{\mathcal{B}}$) for the m-vector π, where

$$f_i = \left\{ \begin{array}{rl} -1 & \text{if } \bar{x}_i < 0 \\ 0 & \text{if } \bar{x}_i \geq 0 \end{array} \right\}.$$

(2) *Test for infeasibility:* If every nonbasic variable x_j satisfies $d_j = -\pi a_j \geq 0$, then there is no feasible solution. *Stop.*

(3) *Choose entering variable:* Pick a nonbasic variable x_p such that $d_p = -\pi a_p < 0$.

(4) *Solve* $B y_{\mathcal{B}} = a_p$ for the m-vector $y_{\mathcal{B}}$.

(5) *Test for unboundedness:* There cannot be an unbounded solution in phase I—skip this step.

(6) *Choose leaving variable:* Pick a basic variable x_q such that $\Theta_p = \bar{x}_q / y_q$ minimizes $\bar{x}_i / y_i$ over all i such that $\bar{x}_i \geq 0$ and $y_i > 0$, or $\bar{x}_i < 0$ and $y_i < 0$.

(7) *Update the solution:* Reset $\bar{x}_p \leftarrow \Theta_p$, and $\bar{x}_{\mathcal{B}} \leftarrow \bar{x}_{\mathcal{B}} - \Theta_p y_{\mathcal{B}}$.

(8) *Change the basis:* Replace q by p in $\mathcal{B}$, and replace the column corresponding to the qth variable with the column corresponding to the pth variable in B.

If all $\bar{x}_i \geq 0$, a basic feasible solution has been found; *stop* and use it as an initial basis for phase II.

The proof of this algorithm is much like that for phase II. It must eventually stop, under a suitable nondegeneracy assumption, because the sum of infeasibilities decreases at every iteration. If it stops in step (8) then obviously it has found a basic feasible solution. An argument must be made that, if it stops in step (2), then no feasible solution is possible; the proof can be adapted from the proof of optimality for phase II.

7.4 A refined phase I: Reducing the number of infeasibilities

So far we have only viewed the phase I simplex algorithm as reducing the sum of infeasibilities, much as phase II reduces the linear objective cx. However, the algorithm can be refined somewhat by also putting some emphasis on reduction in the *number* of infeasible variables.

In the algorithm above, if a negative variable is the first to hit zero then it is chosen to leave the basis. Suppose instead that we let it rise past zero, in the hope that some other infeasible basic variables will reach zero—further reducing the number of basic variables that remain negative. Given again the ray of solutions,

$$\begin{aligned} x_p &= \theta \geq 0, \\ x_i &= \bar{x}_i - \theta y_i, \end{aligned}$$

how big can θ get before all the rising variables have become nonnegative? Recall that the negative variables are the ones with $\bar{x}_i < 0$, and the variables that rise in value are those for which $y_i < 0$. Thus we are looking for the smallest θ at which

$$x_i = \bar{x}_i - \theta y_i \geq 0 \quad \text{for all } \bar{x}_i < 0 \text{ with } y_i < 0,$$

or, rearranging, the smallest θ such that

$$\theta \geq \bar{x}_i / y_i \text{ for all } \bar{x}_i < 0 \text{ with } y_i < 0.$$

Clearly the smallest θ that is $\geq$ all these ratios is just their maximum.

This analysis suggests that we define two critical ratios:

$$\Theta'_p = \min_{i \in \mathcal{B}} -\bar{x}_i / y_i : \bar{x}_i \geq 0 \text{ and } y_i > 0\text{''},$$
$$\Theta''_p = \max_{i \in \mathcal{B}} -\bar{x}_i / y_i : \bar{x}_i < 0 \text{ and } y_i < 0\text{''}.$$

The first ratio is the amount by which the entering variable can be pushed up from zero before some feasible basic variable falls to zero. The second ratio is the amount by which the entering variable can be pushed up from zero before the last infeasible basic variable rises to zero. We want to stop according to whichever of these events happens *sooner,* so we take

$$\Theta_p = \min(\Theta'_p, \Theta''_p).$$

As before, the leaving variable is some x_q such that $\bar{x}_q / y_q = \Theta_p$. This variable reaches zero either by being the first to decrease to zero ($\bar{x}_q / y_q = \Theta'_p$) or the last to increase to zero ($\bar{x}_q / y_q = \Theta''_p$).

So long as no infeasible variable becomes positive, this version of phase I is no different from the previous one, and the sum of infeasibilities declines as before. At iterations where one or more infeasible variables do become positive, however, the sum of infeasibilities can actually increase overall; all we know is that the number of infeasibilities must be lower, since some formerly negative variables became positive, while no positive variables were allowed to become negative. Hence this version of phase I succeeds at each iteration in either reducing the sum of infeasibilities while not letting the number of infeasibilities go up, or in reducing the number of infeasibilities. This observation is enough to guarantee termination after finitely many iterations, by the usual argument that no basis can possibly be repeated.