Simplex algorithm

From Wikipedia, the free encyclopedia

This article is about the linear programming algorithm. For the non-linear optimization heuristic, see Nelder–Mead method.

In mathematical optimization, Dantzig's **simplex algorithm** (or **simplex method**) is a popular algorithm for linear programming.[1]

The name of the algorithm is derived from the concept of a simplex and was suggested by T. S. Motzkin.[2] Simplices are not actually used in the method, but one interpretation of it is that it operates on simplicial *cones*, and these become proper simplices with an additional constraint.[3][4][5][6] The simplicial cones in question are the corners (i.e., the neighborhoods of the vertices) of a geometric object called a polytope. The shape of this polytope is defined by the constraints applied to the objective function.

Contents [hide]

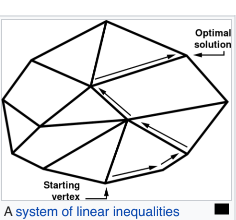
Overview [edit]

Further information: Linear programming

The simplex algorithm operates on linear programs in the canonical form

maximize $\mathbf{c^T x}$
subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$

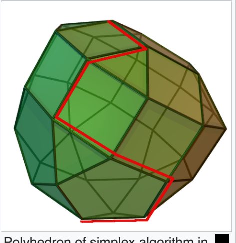
with $\mathbf{x} = (x_1, \ldots, x_n)$ the variables of the problem, $\mathbf{c} = (c_1, \ldots, c_n)$ the coefficients of the objective function, A a *p×n* matrix, and $\mathbf{b} = (b_1, \ldots, b_p)$ nonnegative constants ($\forall j, b_j \geq 0$). There is a straightforward process to convert any linear program into one in standard form, so using this form of linear programs results in no loss of generality.



A system of linear inequalities defines a polytope as a feasible region. The simplex algorithm begins at a starting vertex and moves along the edges of the polytope until it reaches the vertex of the optimal solution.

In geometric terms, the feasible region defined by all values of $\mathbf{x}$ such that $A\mathbf{x} \leq \mathbf{b}$ and $\forall i, x_i \geq 0$ is a (possibly unbounded) convex polytope. An extreme point or vertex of this polytope is known as *basic feasible solution* (BFS).

It can be shown that for a linear program in standard form, if the objective function has a maximum value on the feasible region, then it has this value on (at least) one of the extreme points.[7] This in itself reduces the problem to a finite computation since there is a finite number of extreme points, but the number of extreme points is unmanageably large for all but the smallest linear programs.[8]



Polyhedron of simplex algorithm in 3D

It can also be shown that, if an extreme point is not a maximum point of the objective function, then there is an edge containing the point so that the objective function is strictly increasing on the edge moving away from the point.[9] If the edge is finite, then the edge connects to another extreme point where the objective function has a greater value, otherwise the objective function is unbounded above on the edge and the linear program has no solution. The simplex algorithm applies this insight by walking along edges of the polytope to extreme points with greater and greater objective values. This continues until the maximum value is reached, or an unbounded edge is visited (concluding that the problem has no solution). The algorithm always terminates because the number of vertices in the polytope is finite; moreover since we jump between vertices always in the same direction (that of the objective function), we hope that the number of vertices visited will be small.[9]

The solution of a linear program is accomplished in two steps. In the first step, known as Phase I, a starting extreme point is found. Depending on the nature of the program this may be trivial, but in general it can be solved by applying the simplex algorithm to a modified version of the original program. The possible results of Phase I are either that a basic feasible solution is found or that the feasible region is empty. In the latter case the linear program is called *infeasible*. In the second step, Phase II, the simplex algorithm is applied using the basic feasible solution found in Phase I as a starting point. The possible results from Phase II are either an optimum basic feasible solution or an infinite edge on which the objective function is unbounded above.[10][11][12]

History [edit]

George Dantzig worked on planning methods for the US Army Air Force during World War II using a desk calculator. During 1946 his colleague challenged him to mechanize the planning process to distract him from taking another job. Dantzig formulated the problem as linear inequalities inspired by the work of Wassily Leontief, however, at that time he didn't include an objective as part of his formulation. Without an objective, a vast number of solutions can be feasible, and therefore to find the "best" feasible solution, military-specified "ground rules" must be used that describe how goals can be achieved as opposed to specifying a goal itself. Dantzig's core insight was to realize that most such ground rules can be translated into a linear objective function that needs to be maximized.[13] Development of the simplex method was evolutionary and happened over a period of about a year.[14]

After Dantzig included an objective function as part of his formulation during mid-1947, the problem was mathematically more tractable. Dantzig realized that one of the unsolved problems that he had mistaken as homework in his professor Jerzy Neyman's class (and actually later solved), was applicable to finding an algorithm for linear programs. This problem involved finding the existence of Lagrange multipliers for general linear programs over a continuum of variables, each bounded between zero and one, and satisfying linear constraints expressed in the form of Lebesgue integrals. Dantzig later published his "homework" as a thesis to earn his doctorate. The column geometry used in this thesis gave Dantzig insight that made him believe that the Simplex method would be very efficient.[15]

Standard form [edit]

The transformation of a linear program to one in standard form may be accomplished as follows.[16] First, for each variable with a lower bound other than 0, a new variable is introduced representing the difference between the variable and bound. The original variable can then be eliminated by substitution. For example, given the constraint

$$x_1 \geq 5$$

a new variable, y_1, is introduced with

$$\begin{aligned} y_1 &= x_1 - 5 \\ x_1 &= y_1 + 5 \end{aligned}$$

The second equation may be used to eliminate x_1 from the linear program. In this way, all lower bound constraints may be changed to non-negativity restrictions.

Second, for each remaining inequality constraint, a new variable, called a *slack variable*, is introduced to change the constraint to an equality constraint. This variable represents the difference between the two sides of the inequality and is assumed to be non-negative. For example, the inequalities

$$\begin{aligned} x_2 + 2x_3 &\leq 3 \\ -x_4 + 3x_5 &\geq 2 \end{aligned}$$

are replaced with

$$\begin{aligned} x_2 + 2x_3 + s_1 &= 3 \\ -x_4 + 3x_5 - s_2 &= 2 \\ s_1,\, s_2 &\geq 0 \end{aligned}$$

It is much easier to perform algebraic manipulation on inequalities in this form. In inequalities where ≥ appears such as the second one, some authors refer to the variable introduced as a *surplus variable*.

Third, each unrestricted variable is eliminated from the linear program. This can be done in two ways, one is by solving for the variable in one of the equations in which it appears and then eliminating the variable by substitution. The other is to replace the variable with the difference of two restricted variables. For example, if z_1 is unrestricted then write

$$\begin{aligned} &z_1 = z_1^+ - z_1^- \\ &z_1^+,\, z_1^- \geq 0 \end{aligned}$$

The equation may be used to eliminate z_1 from the linear program.

When this process is complete the feasible region will be in the form

$$\mathbf{A}\mathbf{x} = \mathbf{b},\, \forall i\ x_i \geq 0$$

It is also useful to assume that the rank of $\mathbf{A}$ is the number of rows. This results in no loss of generality since otherwise either the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ has redundant equations which can be dropped, or the system is inconsistent and the linear program has no solution.[17]

Simplex tableau [edit]

A linear program in standard form can be represented as a *tableau* of the form

$$\begin{bmatrix} 1 & -\mathbf{c}^T & 0 \\ 0 & \mathbf{A} & \mathbf{b} \end{bmatrix}$$

The first row defines the objective function and the remaining rows specify the constraints. The zero in the first column represents the zero vector of the same dimension as vector *b*.(Different authors use different conventions as to the exact layout.) If the columns of A can be rearranged so that it contains the identity matrix of order *p* (the number of rows in A) then the tableau is said to be in *canonical form*.[18] The variables corresponding to the columns of the identity matrix are called *basic variables* while the remaining variables are called *nonbasic* or *free variables*. If the values of the nonbasic variables are set to 0, then the values of the basic variables are easily obtained as entries in *b* and this solution is a basic feasible solution. The algebraic interpretation here is that the coefficients of the linear equation represented by each row are either $\mathbf{0}$, $\mathbf{1}$, or some other number. Each row will have $\mathbf{1}$ column with value $\mathbf{1}$, $\boldsymbol{p-1}$ columns with coefficients $\mathbf{0}$, and the remaining columns with some other coefficients (these other variables represent our non-basic variables). By setting the values of the non-basic variables to zero we ensure in each row that the value of the variable represented by a $\mathbf{1}$ in its column is equal to the $\boldsymbol{b}$ value at that row.

Conversely, given a basic feasible solution, the columns corresponding to the nonzero variables can be expanded to a nonsingular matrix. If the corresponding tableau is multiplied by the inverse of this matrix then the result is a tableau in canonical form.[19]

Let

$$\begin{bmatrix} 1 & -\mathbf{c}_B^T & -\mathbf{c}_D^T & 0 \\ 0 & I & \mathbf{D} & \mathbf{b} \end{bmatrix}$$

be a tableau in canonical form. Additional row-addition transformations can be applied to remove the coefficients $\mathbf{c}_B^T$ from the objective function. This process is called *pricing out* and results in a canonical tableau

$$\begin{bmatrix} 1 & 0 & -\bar{\mathbf{c}}_D^T & z_B \\ 0 & I & \mathbf{D} & \mathbf{b} \end{bmatrix}$$

where z_B is the value of the objective function at the corresponding basic feasible solution. The updated coefficients, also known as *relative cost coefficients*, are the rates of change of the objective function with respect to the nonbasic variables.[11]

Pivot operations [edit]

The geometrical operation of moving from a basic feasible solution to an adjacent basic feasible solution is implemented as a *pivot operation*. First, a nonzero *pivot element* is selected in a nonbasic column. The row containing this element is multiplied by its reciprocal to change this element to 1, and then multiples of the row are added to the other rows to change the other entries in the column to 0. The result is that, if the pivot element is in row *r*, then the column becomes the *r*-th column of the identity matrix. The variable for this column is now a basic variable, replacing the variable which corresponded to the *r*-th column of the identity matrix before the operation. In effect, the variable corresponding to the pivot column enters the set of basic variables and is called the *entering variable*, and the variable being replaced leaves the set of basic variables and is called the *leaving variable*. The tableau is still in canonical form but with the set of basic variables changed by one element.[10][11]

Algorithm [edit]

Let a linear program be given by a canonical tableau. The simplex algorithm proceeds by performing successive pivot operations each of which give an improved basic feasible solution; the choice of pivot element at each step is largely determined by the requirement that this pivot improves the solution.

Entering variable selection [edit]

Since the entering variable will, in general, increase from 0 to a positive number, the value of the objective function will decrease if the derivative of the objective function with respect to this variable is negative. Equivalently, the value of the objective function is decreased if the pivot column is selected so that the corresponding entry in the objective row of the tableau is positive.

If there is more than one column so that the entry in the objective row is positive then the choice of which one to add to the set of basic variables is somewhat arbitrary and several *entering variable choice rules*[20] such as Devex algorithm[21] have been developed.

If all the entries in the objective row are less than or equal to 0 then no choice of entering variable can be made and the solution is in fact optimal. It is easily seen to be optimal since the objective row now corresponds to an equation of the form

$$z(\mathbf{x}) = z_B + \textbf{nonnegative terms corresponding to nonbasic variables}$$

By changing the entering variable choice rule so that it selects a column where the entry in the objective row is negative, the algorithm is changed so that it finds the maximum of the objective function rather than the minimum.

Leaving variable selection [edit]

Once the pivot column has been selected, the choice of pivot row is largely determined by the requirement that the resulting solution be feasible. First, only positive entries in the pivot column are considered since this guarantees that the value of the entering variable will be nonnegative. If there are no positive entries in the pivot column then the entering variable can take any nonnegative value with the solution remaining feasible. In this case the objective function is unbounded below and there is no minimum.

Next, the pivot row must be selected so that all the other basic variables remain positive. A calculation shows that this occurs when the resulting value of the entering variable is at a minimum. In other words, if the pivot column is *c*, then the pivot row *r* is chosen so that

$$b_r / a_{rc}$$

is the minimum over all *r* so that $a_{rc} > 0$. This is called the *minimum ratio test*.[20] If there is more than one row for which the minimum is achieved then a *dropping variable choice rule*[22] can be used to make the determination.

Example [edit]

See also: Revised simplex algorithm § Numerical example

Consider the linear program

⇒ Maximize $Z = 2x + 3y + 4Z$
$3x + 2y + Z + s = 10$
$2x + 5y + 3Z + t = 15$
$x, y, Z, s, t \geq 0$

Minimize

$$Z = -2x - 3y - 4z$$

Subject to

$$\begin{aligned} 3x + 2y + z &\leq 10 \\ 2x + 5y + 3z &\leq 15 \\ x,\, y,\, z &\geq 0 \end{aligned}$$

With the addition of slack variables *s* and *t*, this is represented by the canonical tableau

$$\begin{bmatrix} 1 & 2 & 3 & 4 & \boxed{0} & \boxed{0} & 0 \\ 0 & 3 & 2 & 1 & \boxed{1} & \boxed{0} & 10 \\ 0 & 2 & 5 & 3 & \boxed{0} & \boxed{1} & 15 \end{bmatrix} \times\left(-\frac{1}{3}\right)$$

s t

where columns 5 and 6 represent the basic variables *s* and *t* and the corresponding basic feasible solution is

$$x = y = z = 0,\, s = 10,\, t = 15.$$

Z

Columns 2, 3, and 4 can be selected as pivot columns, for this example column 4 is selected. The values of *z* resulting from the choice of rows 2 and 3 as pivot rows are 10/1 = 10 and 15/3 = 5 respectively. Of these the minimum is 5, so row 3 must be the pivot row. Performing the pivot produces

$$\begin{bmatrix} 3 & -2 & -11 & 0 & 0 & -4 & -60 \\ 0 & 7 & 1 & 0 & 3 & -1 & 15 \\ 0 & 2 & 5 & 3 & 0 & 1 & 15 \end{bmatrix}$$

$\frac{2}{3}x + \frac{5}{3}y + Z - \frac{1}{3}t = 5$

x y z

Now columns 4 and 5 represent the basic variables *z* and *s* and the corresponding basic feasible solution is

$$x = y = t = 0,\, z = 5,\, s = 5.$$

For the next step, there are no positive entries in the objective row and in fact

$$Z = \frac{-60 + 2x + 11y + 4t}{3} = -20 + \frac{2x + 11y + 4t}{3}$$

so the minimum value of *Z* is −20.

Finding an initial canonical tableau [edit]

In general, a linear program will not be given in canonical form and an equivalent canonical tableau must be found before the simplex algorithm can start. This can be accomplished by the introduction of *artificial variables*. Columns of the identity matrix are added as column vectors for these variables. If the b value for a constraint equation is negative, the equation is negated before adding the identity matrix columns. This does not change the set of feasible solutions or the optimal solution, and it ensures that the slack variables will constitute an initial feasible solution. The new tableau is in canonical form but it is not equivalent to the original problem. So a new objective function, equal to the sum of the artificial variables, is introduced and the simplex algorithm is applied to find the minimum; the modified linear program is called the *Phase I* problem.[23]

The simplex algorithm applied to the Phase I problem must terminate with a minimum value for the new objective function since, being the sum of nonnegative variables, its value is bounded below by 0. If the minimum is 0 then the artificial variables can be eliminated from the resulting canonical tableau producing a canonical tableau equivalent to the original problem. The simplex algorithm can then be applied to find the solution; this step is called *Phase II*. If the minimum is positive then there is no feasible solution for the Phase I problem where the artificial variables are all zero. This implies that the feasible region for the original problem is empty, and so the original problem has no solution.[10][11][24]

Example [edit]

Consider the linear program

Minimize

$$Z = -2x - 3y - 4z$$

Subject to

$$\begin{aligned} 3x + 2y + z &= 10 \\ 2x + 5y + 3z &= 15 \\ x,\, y,\, z &\geq 0 \end{aligned}$$

This is represented by the (non-canonical) tableau

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 0 \\ 0 & 3 & 2 & 1 & 10 \\ 0 & 2 & 5 & 3 & 15 \end{bmatrix}$$

Introduce artificial variables *u* and *v* and objective function *W* = *u* + *v*, giving a new tableau

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 1 & 2 & 3 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 2 & 1 & 1 & 0 & 10 \\ 0 & 0 & 2 & 5 & 3 & 0 & 1 & 15 \end{bmatrix}$$

The equation defining the original objective function is retained in anticipation of Phase II.

By construction, *u* and *v* are both non-basic variables since they are part of the initial identity matrix. However, the objective function *W* currently assumes that *u* and *v* are both *0*. In order to adjust the objective function to be the correct value where *u = 10* and *v = 15*, add the third and fourth rows to the first row giving

$$\begin{bmatrix} 1 & 0 & 5 & 7 & 4 & 0 & 0 & 25 \\ 0 & 1 & 2 & 3 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 2 & 1 & 1 & 0 & 10 \\ 0 & 0 & 2 & 5 & 3 & 0 & 1 & 15 \end{bmatrix}$$

Select column 5 as a pivot column, so the pivot row must be row 4, and the updated tableau is

$$\begin{bmatrix} 3 & 0 & 7 & 1 & 0 & 0 & -4 & 15 \\ 0 & 3 & -2 & -11 & 0 & 0 & -4 & -60 \\ 0 & 0 & 7 & 1 & 0 & 3 & -1 & 15 \\ 0 & 0 & 2 & 5 & 3 & 0 & 1 & 15 \end{bmatrix}$$

Now select column 3 as a pivot column, for which row 3 must be the pivot row, to get

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 7 & 0 & -25 & 0 & 2 & -10 & -130 \\ 0 & 0 & 7 & 1 & 0 & 3 & -1 & 15 \\ 0 & 0 & 0 & 11 & 7 & -2 & 3 & 25 \end{bmatrix}$$

The artificial variables are now 0 and they may be dropped giving a canonical tableau equivalent to the original problem:

$$\begin{bmatrix} 7 & 0 & -25 & 0 & -130 \\ 0 & 7 & 1 & 0 & 15 \\ 0 & 0 & 11 & 7 & 25 \end{bmatrix}$$

This is, fortuitously, already optimal and the optimum value for the original linear program is −130/7.

Advanced topics [edit]

Implementation [edit]

Main article: Revised simplex algorithm

The tableau form used above to describe the algorithm lends itself to an immediate implementation in which the tableau is maintained as a rectangular (*m* + 1)-by-(*m* + *n* + 1) array. It is straightforward to avoid storing the m explicit columns of the identity matrix that will occur within the tableau by virtue of **B** being a subset of the columns of [**A**, **I**]. This implementation is referred to as the "*standard* simplex algorithm". The storage and computation overhead are such that the standard simplex method is a prohibitively expensive approach to solving large linear programming problems.

In each simplex iteration, the only data required are the first row of the tableau, the (pivotal) column of the tableau corresponding to the entering variable and the right-hand-side. The latter can be updated using the pivotal column and the first row of the tableau can be updated using the (pivotal) row corresponding to the leaving variable. Both the pivotal column and pivotal row may be computed directly using the solutions of linear systems of equations involving the matrix **B** and a matrix-vector product using **A**. These observations motivate the "*revised* simplex algorithm", for which implementations are distinguished by their invertible representation of **B**.[25]

In large linear-programming problems **A** is typically a sparse matrix and, when the resulting sparsity of **B** is exploited when maintaining its invertible representation, the revised simplex algorithm is much more efficient than the standard simplex method. Commercial simplex solvers are based on the revised simplex algorithm.[24][25][26][27][28]

Degeneracy: stalling and cycling [edit]

If the values of all basic variables are strictly positive, then a pivot must result in an improvement in the objective value. When this is always the case no set of basic variables occurs twice and the simplex algorithm must terminate after a finite number of steps. Basic feasible solutions where at least one of the *basic* variables is zero are called *degenerate* and may result in pivots for which there is no improvement in the objective value. In this case there is no actual change in the solution but only a change in the set of basic variables. When several such pivots occur in succession, there is no improvement; in large industrial applications, degeneracy is common and such "*stalling*" is notable. Worse than stalling is the possibility the same set of basic variables occurs twice, in which case, the deterministic pivoting rules of the simplex algorithm will produce an infinite loop, or "cycle". While degeneracy is the rule in practice and stalling is common, cycling is rare in practice. A discussion of an example of practical cycling occurs in Padberg.[24] Bland's rule prevents cycling and thus guarantees that the simplex algorithm always terminates.[24][29][30] Another pivoting algorithm, the criss-cross algorithm never cycles on linear programs.[31]

History-based pivot rules such as Zadeh's rule and Cunningham's rule also try to circumvent the issue of stalling and cycling by keeping track how often particular variables are being used, and then favor such variables that have been used least often.

Efficiency [edit]

The simplex method is remarkably efficient in practice and was a great improvement over earlier methods such as Fourier–Motzkin elimination. However, in 1972, Klee and Minty[32] gave an example, the Klee–Minty cube, showing that the worst-case complexity of simplex method as formulated by Dantzig is exponential time. Since then, for almost every variation on the method, it has been shown that there is a family of linear programs for which it performs badly. It is an open question if there is a variation with polynomial time, although sub-exponential pivot rules are known.[33]

In 2018, it was proved that a particular variant of the simplex method is NP-mighty, i.e., it can be used to solve, with polynomial overhead, any problem in NP implicitly during the algorithm's execution. Moreover, deciding whether a given variable ever enters the basis during the algorithm's execution on a given input, and determining the number of iterations needed for solving a given problem, are both NP-hard problems.[34] Computing the output of some other pivot rules was already known to be PSPACE-complete[35][36].

Analyzing and quantifying the observation that the simplex algorithm is efficient in practice despite its exponential worst-case complexity has led to the development of other measures of complexity. The simplex algorithm has polynomial-time average-case complexity under various probability distributions, with the precise average-case performance of the simplex algorithm depending on the choice of a probability distribution for the random matrices.[37][38] Another approach to studying "typical phenomena" uses Baire category theory from general topology, and to show that (topologically) "most" matrices can be solved by the simplex algorithm in a polynomial number of steps.[*citation needed*] Another method to analyze the performance of the simplex algorithm studies the behavior of worst-case scenarios under small perturbation – are worst-case scenarios stable under a small change (in the sense of structural stability), or do they become tractable? This area of research, called smoothed analysis, was introduced specifically to study the simplex method. Indeed, the running time of the simplex method on input with noise is polynomial in the number of variables and the magnitude of the perturbations.[39]

Other algorithms [edit]

Other algorithms for solving linear-programming problems are described in the linear-programming article. Another basis-exchange pivoting algorithm is the criss-cross algorithm.[40][41] There are polynomial-time algorithms for linear programming that use interior point methods: these include Khachiyan's ellipsoidal algorithm, Karmarkar's projective algorithm, and path-following algorithms.[12]

Linear-fractional programming [edit]

Main article: Linear-fractional programming

Linear–fractional programming (LFP) is a generalization of linear programming (LP). In LP the objective function is a linear function, while the objective function of a linear–fractional program is a ratio of two linear functions. In other words, a linear program is a fractional–linear program in which the denominator is the constant function having the value one everywhere. A linear–fractional program can be solved by a variant of the simplex algorithm[42][43][44][45] or by the criss-cross algorithm.[46]

See also [edit]

- Criss-cross algorithm
- Cutting-plane method
- Devex algorithm
- Fourier–Motzkin elimination
- Karmarkar's algorithm
- Nelder–Mead simplicial heuristic
- Pivoting rule of Bland, which avoids cycling

Simplex Details:

1. Is the feasible region non-empty?
2. Can you find a feasible basic solution.
3. How do you pick your variable to visit so that the objective function does increase.
4. When simplex gives an answer, is answer correct?

Simplex is average case in Polynomial case.

If you restrict LP to

ILP = Integer LP

ILP in NP-Complete.

Maximize $Z = 2x + 3y + 4z$

$$3x + 2y + z + s = 10$$

$$2x + 5y + 3z + t = 15$$

$$\begin{array}{ccccccc}
1 & 2 & 3 & 4 & 0 & 0 & 0 \\
0 & 3 & 2 & 1 & 1 & 0 & 10 \\
0 & 2 & 5 & 3 & 0 & 1 & 15 \\
Z & x & y & z & s & t & b
\end{array} \qquad \times\left(-\frac{1}{3}\right)$$

$$\Downarrow \quad \begin{aligned} row2 &= row2 + row3 \times \left(-\frac{1}{3}\right) \\ row1 &= row1 + row3 \times \left(-\frac{4}{3}\right) \end{aligned}$$

$$\begin{array}{ccccccc}
1 & -\frac{2}{3} & -\frac{11}{3} & 0 & 0 & -\frac{4}{3} & -20 \\
0 & \frac{7}{3} & \frac{1}{3} & 0 & 1 & -\frac{1}{3} & 5 \\
0 & 2 & 5 & 3 & 0 & 1 & 15
\end{array}$$

$$\Downarrow$$

$$\begin{array}{ccccccc}
3 & -2 & -11 & 0 & 0 & -4 & -60 \\
0 & 7 & 1 & 0 & 3 & -1 & 15 \\
0 & 2 & 5 & 3 & 0 & 1 & 15
\end{array}$$

$$3Z = -2x - 11y - 4t - 60$$

$$Z = -20 + \frac{-2x - 11y - 4t}{3}$$

$$\max(Z) = -20$$

$\min(-Z) = 20$

We now describe the main idea behind an iteration of the simplex algorithm. Associated with each iteration will be a "basic solution" that we can easily obtain from the slack form of the linear program: set each nonbasic variable to 0 and compute the values of the basic variables from the equality constraints. An iteration converts one slack form into an equivalent slack form. The objective value of the associated basic feasible solution will be no less than that at the previous iteration, and usually greater. To achieve this increase in the objective value, we choose a nonbasic variable such that if we were to increase that variable's value from 0, then the objective value would increase, too. The amount by which we can increase the variable is limited by the other constraints. In particular, we raise it until some basic variable becomes 0. We then rewrite the slack form, exchanging the roles of that basic variable and the chosen nonbasic variable. Although we have used a particular setting of the variables to guide the algorithm, and we shall use it in our proofs, the algorithm does not explicitly maintain this solution. It simply rewrites the linear program until an optimal solution becomes "obvious."

An example of the simplex algorithm

We begin with an extended example. Consider the following linear program in standard form:

$$
\begin{array}{llcl}
\text{maximize} & 3x_1 + x_2 + 2x_3 & & \qquad (29.53)\\
\text{subject to} & & & \\
& x_1 + x_2 + 3x_3 & \le 30 & \qquad (29.54)\\
& 2x_1 + 2x_2 + 5x_3 & \le 24 & \qquad (29.55)\\
& 4x_1 + x_2 + 2x_3 & \le 36 & \qquad (29.56)\\
& x_1, x_2, x_3 & \ge 0\,. & \qquad (29.57)
\end{array}
$$

In order to use the simplex algorithm, we must convert the linear program into slack form; we saw how to do so in Section 29.1. In addition to being an algebraic manipulation, slack is a useful algorithmic concept. Recalling from Section 29.1 that each variable has a corresponding nonnegativity constraint, we say that an equality constraint is ***tight*** for a particular setting of its nonbasic variables if they cause the constraint's basic variable to become 0. Similarly, a setting of the non-basic variables that would make a basic variable become negative ***violates*** that constraint. Thus, the slack variables explicitly maintain how far each constraint is from being tight, and so they help to determine how much we can increase values of nonbasic variables without violating any constraints.

Associating the slack variables x_4, x_5, and x_6 with inequalities (29.54)–(29.56), respectively, and putting the linear program into slack form, we obtain

$$
\begin{array}{rclcrcrcrl}
z & = & & & 3x_1 & + & x_2 & + & 2x_3 & \quad (29.58)\\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 & \quad (29.59)\\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 & \quad (29.60)\\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3\,. & \quad (29.61)
\end{array}
$$

The system of constraints (29.59)–(29.61) has 3 equations and 6 variables. Any setting of the variables x_1, x_2, and x_3 defines values for x_4, x_5, and x_6; therefore, we have an infinite number of solutions to this system of equations. A solution is feasible if all of $x_1, x_2, \ldots, x_6$ are nonnegative, and there can be an infinite number of feasible solutions as well. The infinite number of possible solutions to a system such as this one will be useful in later proofs. We focus on the ***basic solution***: set all the (nonbasic) variables on the right-hand side to 0 and then compute the values of the (basic) variables on the left-hand side. In this example, the basic solution is $(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_6) = (0, 0, 0, 30, 24, 36)$ and it has objective value $z = (3 \cdot 0) + (1 \cdot 0) + (2 \cdot 0) = 0$. Observe that this basic solution sets $\bar{x}_i = b_i$ for each $i \in B$. An iteration of the simplex algorithm rewrites the set of equations and the objective function so as to put a different set of variables on the right-hand side. Thus, a different basic solution is associated with the rewritten problem. We emphasize that the rewrite does not in any way change the underlying linear-programming problem; the problem at one iteration has the identical set of feasible solutions as the problem at the previous iteration. The problem does, however, have a different basic solution than that of the previous iteration.

If a basic solution is also feasible, we call it a ***basic feasible solution***. As we run the simplex algorithm, the basic solution is almost always a basic feasible solution. We shall see in Section 29.5, however, that for the first few iterations of the simplex algorithm, the basic solution might not be feasible.

Our goal, in each iteration, is to reformulate the linear program so that the basic solution has a greater objective value. We select a nonbasic variable x_e whose coefficient in the objective function is positive, and we increase the value of x_e as much as possible without violating any of the constraints. The variable x_e becomes basic, and some other variable x_l becomes nonbasic. The values of other basic variables and of the objective function may also change.

To continue the example, let's think about increasing the value of x_1. As we increase x_1, the values of x_4, x_5, and x_6 all decrease. Because we have a nonnegativity constraint for each variable, we cannot allow any of them to become negative. If x_1 increases above 30, then x_4 becomes negative, and x_5 and x_6 become negative when x_1 increases above 12 and 9, respectively. The third constraint (29.61) is the tightest constraint, and it limits how much we can increase x_1. Therefore, we switch the roles of x_1 and x_6. We solve equation (29.61) for x_1 and obtain

$$x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}\,. \qquad (29.62)$$

To rewrite the other equations with x_6 on the right-hand side, we substitute for x_1 using equation (29.62). Doing so for equation (29.59), we obtain

$$\begin{aligned} x_4 &= 30 - x_1 - x_2 - 3x_3 \\ &= 30 - \left(9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}\right) - x_2 - 3x_3 \\ &= 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} . \end{aligned} \tag{29.63}$$

Similarly, we combine equation (29.62) with constraint (29.60) and with objective function (29.58) to rewrite our linear program in the following form:

$$z = 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4} \tag{29.64}$$

$$x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \tag{29.65}$$

$$x_4 = 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \tag{29.66}$$

$$x_5 = 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2} . \tag{29.67}$$

We call this operation a ***pivot***. As demonstrated above, a pivot chooses a nonbasic variable x_e, called the ***entering variable***, and a basic variable x_l, called the ***leaving variable***, and exchanges their roles.

The linear program described in equations (29.64)–(29.67) is equivalent to the linear program described in equations (29.58)–(29.61). We perform two operations in the simplex algorithm: rewrite equations so that variables move between the left-hand side and the right-hand side, and substitute one equation into another. The first operation trivially creates an equivalent problem, and the second, by elementary linear algebra, also creates an equivalent problem. (See Exercise 29.3-3.)

To demonstrate this equivalence, observe that our original basic solution $(0, 0, 0, 30, 24, 36)$ satisfies the new equations (29.65)–(29.67) and has objective value $27 + (1/4) \cdot 0 + (1/2) \cdot 0 - (3/4) \cdot 36 = 0$. The basic solution associated with the new linear program sets the nonbasic values to 0 and is $(9, 0, 0, 21, 6, 0)$, with objective value $z = 27$. Simple arithmetic verifies that this solution also satisfies equations (29.59)–(29.61) and, when plugged into objective function (29.58), has objective value $(3 \cdot 9) + (1 \cdot 0) + (2 \cdot 0) = 27$.

Continuing the example, we wish to find a new variable whose value we wish to increase. We do not want to increase x_6, since as its value increases, the objective value decreases. We can attempt to increase either x_2 or x_3; let us choose x_3. How far can we increase x_3 without violating any of the constraints? Constraint (29.65) limits it to 18, constraint (29.66) limits it to $42/5$, and constraint (29.67) limits it to $3/2$. The third constraint is again the tightest one, and therefore we rewrite the third constraint so that x_3 is on the left-hand side and x_5 is on the right-hand side. We then substitute this new equation, $x_3 = 3/2 - 3x_2/8 - x_5/4 + x_6/8$, into equations (29.64)–(29.66) and obtain the new, but equivalent, system

$$z = \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \tag{29.68}$$

$$x_1 = \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \tag{29.69}$$

$$x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \tag{29.70}$$

$$x_4 = \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \,. \tag{29.71}$$

This system has the associated basic solution $(33/4, 0, 3/2, 69/4, 0, 0)$, with objective value $111/4$. Now the only way to increase the objective value is to increase x_2. The three constraints give upper bounds of 132, 4, and ∞, respectively. (We get an upper bound of ∞ from constraint (29.71) because, as we increase x_2, the value of the basic variable x_4 increases also. This constraint, therefore, places no restriction on how much we can increase x_2.) We increase x_2 to 4, and it becomes nonbasic. Then we solve equation (29.70) for x_2 and substitute in the other equations to obtain

$$z = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \tag{29.72}$$

$$x_1 = 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \tag{29.73}$$

$$x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \tag{29.74}$$

$$x_4 = 18 - \frac{x_3}{2} + \frac{x_5}{2} \,. \tag{29.75}$$

At this point, all coefficients in the objective function are negative. As we shall see later in this chapter, this situation occurs only when we have rewritten the linear program so that the basic solution is an optimal solution. Thus, for this problem, the solution $(8, 4, 0, 18, 0, 0)$, with objective value 28, is optimal. We can now return to our original linear program given in (29.53)–(29.57). The only variables in the original linear program are x_1, x_2, and x_3, and so our solution is $x_1 = 8$, $x_2 = 4$, and $x_3 = 0$, with objective value $(3 \cdot 8) + (1 \cdot 4) + (2 \cdot 0) = 28$. Note that the values of the slack variables in the final solution measure how much slack remains in each inequality. Slack variable x_4 is 18, and in inequality (29.54), the left-hand side, with value $8 + 4 + 0 = 12$, is 18 less than the right-hand side of 30. Slack variables x_5 and x_6 are 0 and indeed, in inequalities (29.55) and (29.56), the left-hand and right-hand sides are equal. Observe also that even though the coefficients in the original slack form are integral, the coefficients in the other linear programs are not necessarily integral, and the intermediate solutions are not