Chapter 3. Linear Programming

Linear programming is a tool for solving certain optimization problems. In 1947, George Dantzig developed a fast algorithm, the simplex algorithm, for solving linear programming problems. Since the development of the simplex algorithm, linear programming has been used to solve optimization problems in industries as diverse as banking, forestry, petroleum, and trucking.

Section 3.1. The Diet Problem

The next section has been adapted from the book: *Linear Programming* by V. Chvatal (1983), printed by W.H. Freeman and Company, New York / San Francisco.

We introduce linear programming problems by means of an example. Consider the following diet problem.

Polly wonders how much money she must spend on food in order to get all the energy (2000 kcal), protein (55 g), and calcium (800 mg) that she needs every day. (For iron and vitamins, she will depend on pills. Nutritionists would disapprove, but our example ought to be simple.) She chooses six foods that seem to be cheap sources of the nutrients; her data are collected in Table 1.1.

Food	Serving size	Energy	Protein	Calcium	Price per serving
Oatmeal	28 g	110	4	2	3
Chicken	100 g	205	32	12	24
Eggs	2 large	160	13	54	13
Whole milk	237 cc	160	8	285	9
Cherry pie	170 g	420	4	22	20
Pork with beans	260 g	260	14	80	19

Table 1.1 Nutritive value per serving.

Then she begins to think about her menu. For example, 10 servings of pork with beans would take care of all her needs for only 1.90 per day. On the other hand, 10 servings of pork with beans is a lot of pork with beans; she would not be able to stomach more than 2 servings a day. She decides to impose servings-per-day limits on all six foods:

Oatmeal	at most 4 servings per day.
Chicken	at most 3 servings per day.
Eggs	at most 2 servings per day.
Whole milk	at most 8 servings per day.
Cherry pie	at most 2 servings per day.
Pork with beans	at most 2 servings per day.

Now, another look at her data shows Polly that 8 servings of milk and 2 servings of cherry pie every day will satisfy the requirements nicely and at a cost of only 1.12. In fact, she could cut down a little on the pie or milk or perhaps try a different combination. But so many combinations seem promising that one could go on and on, looking for the best one. Trial and error is not particularly helpful here. To be systematic, we may speculate about some as of yet unspecified menu consisting of x_1 servings of oatmeal, x_2 servings of chicken, x_3 servings of eggs, and so on. In order to stay below the upper limits, the menu must satisfy:

$$\begin{aligned}
0 \le x_1 &\le 4\\
0 \le x_2 &\le 3\\
0 \le x_3 &\le 2\\
0 \le x_4 &\le 8\\
0 \le x_5 &\le 2\\
0 \le x_6 &\le 2.
\end{aligned}$$

And of course, there are the requirements for energy, protein, and calcium; they lead to the inequalities

$$\begin{aligned}
&110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \ge 2000\\
&4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \ge 55\\
&2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \ge 800.
\end{aligned}$$

If some numbers $x_1, x_2, \ldots, x_6$ satisfy all these inequalities, then they describe a satisfactory menu; such a menu will cost, in cents per day

$$3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6.$$

In designing the most economical menu, Polly wants to find numbers $x_1, x_2, \ldots, x_6$ that satisfy all the constraints and makes the cost as small as possible. As a mathematician would put it

$$\text{minimize } x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

subject to

$$\begin{aligned}
&110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \ge 2000\\
&4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \ge 55\\
&2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \ge 800
\end{aligned}$$

$$\begin{aligned}
0 \le x_1 \le 4\\
0 \le x_2 \le 3\\
0 \le x_3 \le 2\\
0 \le x_4 \le 8\\
0 \le x_5 \le 2\\
0 \le x_6 \le 2.
\end{aligned}$$

Problems of this kind are called linear programming problems.

Section 3.2. Easily solvable linear programs

In general, a linear programming problem is the problem of maximizing (or minimizing) a linear function subject to a finite number of linear constraints, that is,

$$\max \sum_{1 \le j \le n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad (i = 1, \ldots, N_1)$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad (i = N_1 + 1, \ldots, N_2)$$

$$\sum_{j=1}^{n} a_{ij} x_j \ge b_i \quad (i = N_2 + 1, \ldots, N_3).$$

As indicated in the introduction, the simplex method solves linear programming problems fast. There are, however, two types of linear programming problems that can be solved without having to resort to the simplex method.

The first type of easily solvable linear programming problem concerns a linear programming problem in which there is only one constraint that involves more than one variable. Consider for instance the following *continuous knapsack problem*.

$$\max \sum_{1 \le j \le n} c_j x_j$$

subject to

$$\sum_{1 \le j \le n} a_j x_j \le b$$

$$0 \le x_j \le 1 \quad (j = 1, \ldots, n).$$

The knapsack problem refers to a situation in which there are n different items that can be put into a knapsack with capacity b; each item j has a certain weight a_j associated with it as well as a value c_j. The problem is to determine how much of each item to place in the knapsack in order to maximize the total value subject to the constraint that the total weight should not exceed the capacity b of the knapsack.

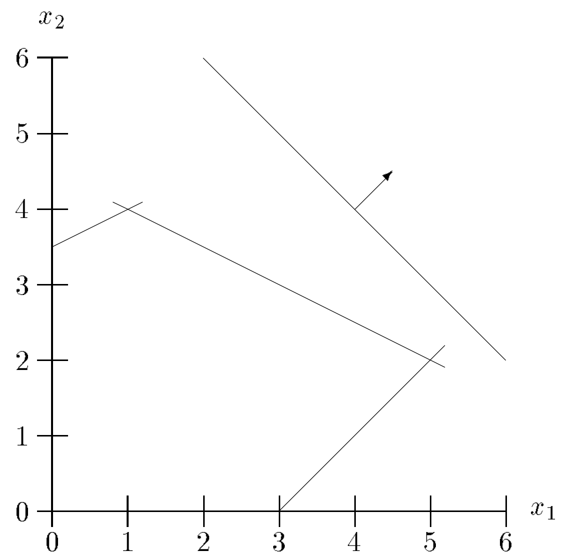
The problem is solved by sorting the items in order of decreasing ratios c_j/a_j and putting them in the knapsack in this order, always taking as much as possible, until the weight limit is reached. The knapsack problem is discussed in full detail in Chapter 4.

The second type of easily solvable linear programming problem concerns a linear programming problem with only two variables. In that case, the problem can be solved graphically. The set of feasible solutions is a polyhedron defined by the linear inequalities. Each feasible solution is on a profit line, that is, a set of feasible solution with the same solution value. There are many such profit lines, all parallel to each other. Using these profit lines it is easy find an optimal solution. Consider for instance the linear program

$$\max\{x_1 + x_2\}$$

subject to

$$\begin{aligned} -x_1 + 2x_2 &\leq 14 \\ x_1 + 2x_2 &\leq 18 \\ x_1 - x_2 &\leq 6 \\ x_1, x_2 &\geq 0. \end{aligned}$$



Section 3.3. Special cases of linear programs
So far, we have not addressed the following questions:

- Does there always exist a solution?
- If the solution space is not empty, does there exist a unique optimal solution?

Infeasible linear programs
It is possible that the feasible region of a linear program, that is, the set of points satisfying all constraints, is empty. Since the optimal solution to a linear program is the best point in the feasible region, such a linear program has no optimal solution. We say that the linear program is infeasible.

Consider for example the following linear program

$$\max \ 4x_1 + 2x_2$$

subject to

$$\begin{aligned} &3x_1 + 2x_2 \leq 12 \\ &x_1 \geq 3 \\ &x_2 \geq 2. \end{aligned}$$

The feasible region of this linear program is empty: the combination of $x_1 \geq 3$ and $x_2 \geq 2$ implies that $3x_1 + 2x_2 \geq 13$, which is in conflict with the constraint $3x_1 + 2x_2 \leq 12$.

Unbounded linear programs
For a maximization problem, an unbounded linear program occurs if it is possible to find points in the feasible region with arbitrarily large objective function values. Analogously, for a minimization problem, an unbounded linear program occurs if it is possible to find points in the feasible region with arbitrarily small objective function values.

Consider for example the following linear program

$$\max \; 2x_1 - x_2$$

subject to

$$\begin{aligned} &x_1 - x_2 \leq 1 \\ &2x_1 + x_2 \geq 8 \\ &x_1, x_2 \geq 0. \end{aligned}$$

Starting with a feasible solution, $x_1 = 0$ and $x_2 = 8$ for example, we are always able to increase the objective function by one unit by simply increasing x_1 and x_2 by one unit.

Alternative or multiple optimal solutions
It is possible for a linear program to have multiple or alternative optimal solutions. This is indicated by the fact that the vector of objective function coefficients is a positive multiple of one or more of the vectors of constraint coefficients.
Consider for example the following linear program

$$\max \; 3x_1 + 2x_2$$

subject to

$$\begin{aligned} &x_1 + x_2 \leq 5 \\ &3x_1 + 2x_2 \leq 12 \\ &x_1, x_2 \geq 0. \end{aligned}$$

Each point (x_1, x_2) with $3x_1 + 2x_2 = 12$ that satisfies the other constraints is an optimal solution for this linear program.

Section 3.4. An example of the simplex algorithm

Consider any feasible bounded linear program with two variables. The constraints of the linear program define a polyhedron that contains all feasible solutions to this linear program. Suppose that the optimal solution to the linear program does not correspond to a vertex of the polyhedron. If the optimal solution lies within the interior of the polyhedron, then we can increase or decrease the value of x_1 without decreasing the value of the objection function until we hit the boundary of the polyhedron, that is, the new optimal solution is on an edge of the polyhedron. Similarly, we can move along this edge without decreasing the value of the objective function until we end in a vertex of the polyhedron. Hence, we have that, in case of a feasible bounded linear problem with two variables, we only have to search over all edges of the polyhedron to find an optimal solution. This can be generalized to the case of a feasible bounded linear program with n variables.

Theorem 3.1.
For any feasible bounded linear program there exists an optimal solution that corresponds to a vertex of the polyhedron that describes the set of feasible solutions. □

The simplex algorithm heavily depends on Theorem 3.1. It proceeds by hopping from one vertex of the polyhedron to another, in such a way that the value of the objective function never decreases. Before describing all ins and outs of the simplex algorithm, we show the iterative step through a simple example. Consider the following linear program

$$\max\ z = 2x_1 - 3x_2 + 4x_3$$

subject to

$$\begin{aligned}
&x_1 - x_2 + 2x_3 \leq 7\\
&2x_1 + x_2 + 3x_3 \leq 22\\
&x_1 - x_2 + x_3 \leq 2\\
&x_1, x_2, x_3 \geq 0.
\end{aligned}$$

It is not too hard to deal with the nonnegativity inequalities $x_i \geq 0$ $(i = 1, 2, 3)$; it it is the other inequalities that make the situation unclear. Therefore, we rewrite these inequalities by adding *slack* variables; the inequality $x_1 - x_2 + 2x_3 \leq 7$ is rewritten as $x_4 \geq 0$, where x_4 is defined as $x_4 = 7 - x_1 + x_2 - 2x_3$. In this way, we rewrite the whole linear program as a set of nonnegativity constraints together with a set of equalities. This yields the following linear program

$$\max\ z = 2x_1 - 3x_2 + 4x_3$$

subject to

$$\begin{aligned}
&x_4 = 7 - x_1 + x_2 - 2x_3\\
&x_5 = 22 - 2x_1 - x_2 - 3x_3\\
&x_6 = 2 - x_1 + x_2 - x_3\\
&x_1, x_2, x_3, x_4, x_5, x_6 \geq 0.
\end{aligned}$$

The essential information in the linear program above consists of the objective function and the set of equalities; we will just write these down, while keeping in mind that z has to be maximized and that the values of the variables should be nonnegative. The compressed version of a linear program is called a *dictionary.* The dictionary in our example reads:

$$\begin{aligned}
&x_4 = 7 - x_1 + x_2 - 2x_3\\
&x_5 = 22 - 2x_1 - x_2 - 3x_3\\
&x_6 = 2 - x_1 + x_2 - x_3\\
&z = 2x_1 - 3x_2 + 4x_3.
\end{aligned}$$

Almost everything is ready now to put the simplex method to work on this example. All we need now is an initial vertex of the polyhedron. From geometry, we know that a vertex of a polyhedron that is defined as a set of m equalities in $(n + m)$ variables has the form that there are at least n variables with value equal to zero; the values of the other variables should be positive to make it a feasible solution to the linear program. Hence, we have that $(x_1, x_2, x_3, x_4, x_5, x_6) = (0, 0, 0, 7, 22, 2)$ serves as an initial solution. The simplex algorithm hops from one vertex of the polyhedron to another. Since we have just stated that in a vertex of the polyhedron at most m variables have positive value, we should increase the value of one of the zero-valued variables (these variables are called the *non-basic variables*) and decrease the value of one of the variables with positive value (these variables are called the *basic variables*) to zero. The question is of course: which non-basic variable should replace which basic variable?

As to the choice of the non-basic variable, it should be such that the value of the objective function does not decrease. The objective function is stated in terms of non-basic variables only, so we can make sure that the value of the objective function does not decrease by selecting a non-basic variable whose coefficient in the objective function is positive. In our example, this rules out x_2; let us choose x_1. It is easy to compute by how much we should increase x_1: it is the smallest value that makes the value of one of the basic variables equal to zero. Since x_2 and x_3 remain equal to zero, we have that $x_4 = 7 - x_1$, $x_5 = 22 - 2x_1$, and $x_6 = 2 - x_1$. This yields that our new vertex, which we from now on call a *basic feasible solution* will be $(x_1, x_2, x_3, x_4, x_5, x_6) = (2, 0, 0, 5, 18, 0)$; the corresponding value of the objective function amounts to 4.

If we take a closer look at the above step, then we see that we could easily select a non-basic variable whose increase did not decrease the objective function, because the objective function was stated in terms of non-basic variables only. Similarly, we could easily compute the new basic feasible solution, because all basic variables were expressed in terms of non-basic variables only. Hence, in order to conduct the next step, we rewrite the dictionary such that the above conditions are satisfied again. It is easy to express x_1 in terms of the non-basic variables: from the relation $x_6 = 2 - x_1 + x_2 - x_3$ it follows immediately that $x_1 = 2 + x_2 - x_3 - x_6$. Using this relation, we obtain the new dictionary

$$\begin{aligned}
x_1 &= 2 + x_2 - x_3 - x_6\\
x_4 &= 5 - x_3 + x_6\\
x_5 &= 18 - 3x_2 - x_3 + 2x_6\\
z &= 4 - x_2 + 2x_3 - 2x_6.
\end{aligned}$$

There is only one choice for the non-basic variable that we should increase, since only an increase of x_3 can lead to an increase of the value of the objective function. As to the amount we should increase x_3 with, we have $x_1 = 2 - x_3$, $x_4 = 5 - x_3$, and $x_5 = 18 - x_3$, from which follows that the value of x_3 should become 2 and that x_1 becomes non-basic again. This leads to the basic feasible solution $(x_1, x_2, x_3, x_4, x_5, x_6) = (0, 0, 2, 3, 16, 0)$ with objective value 8, and the new dictionary reads

$$\begin{aligned}
x_3 &= 2 - x_1 + x_2 - x_6\\
x_4 &= 3 + x_1 - x_2 + 2x_6\\
x_5 &= 16 + x_1 - 4x_2 + 3x_6\\
z &= 8 - 2x_1 + x_2 - 4x_6.
\end{aligned}$$

We see that x_2 is the only candidate for being increased. We have that $x_3 = 2 + x_2$, $x_4 = 3 - x_2$, and $x_5 = 16 - 4x_2$, from which follows that the value of x_3 should become 3 and that x_4 becomes non-basic. This leads to the basic feasible solution $(x_1, x_2, x_3, x_4, x_5, x_6) = (0, 3, 5, 0, 4, 0)$ with objective value 11. The new dictionary is

$$\begin{aligned}
x_2 &= 3 + x_1 - x_4 + 2x_6 \\
x_3 &= 5 - x_4 + x_6 \\
x_5 &= 4 - 3x_1 + 4x_4 - 5x_6 \\
z &= 11 - x_1 - x_4 - 2x_6.
\end{aligned}$$

Since there are no candidates from among the set of non-basic variables to be increased in value, the simplex algorithm stops. Since all variables must be nonnegative, the last line of the dictionary implies that $z \leq 11$ for each feasible solution. Hence, we may conclude that the current basic feasible solution is optimal. This observation identifies the general rule that we have reached an optimal solution if the coefficients of all non-basic variables in the objective function are smaller than or equal to zero.

Section 3.5. The simplex algorithm

In the previous section, we have seen the simplex algorithm at work on a linear program that was selected to be bounded and feasible (together with some other nice features that will be pointed out later). The simplex algorithm cannot conclude beforehand that some linear program is bounded and feasible; in fact, showing that a linear program is feasible boils down to finding an optimal solution if we add the constraint $z \geq z^*$, where z^* denotes the optimum solution.

Suppose that we apply the simplex algorithm to a linear program that is unbounded; take for example the instance of Section 3.3:

$$\max\ 2x_1 - x_2$$

subject to

$$\begin{aligned}
&x_1 - x_2 \leq 1 \\
&-2x_1 - x_2 \leq -8 \\
&x_1, x_2 \geq 0.
\end{aligned}$$

We introduce slack variables $x_3 = 1 - x_1 + x_2$ and $x_4 = -8 + 2x_1 + x_2$, which we request to be nonnegative. Starting with the basic feasible solution $(x_1, x_2, x_3, x_4) = (0, 8, 9, 0)$, we obtain the following dictionary:

$$\begin{aligned}
x_2 &= 8 - 2x_1 + x_4 \\
x_3 &= 9 - 3x_1 + x_4 \\
z &= -8 + 4x_1 - x_4.
\end{aligned}$$

Clearly, x_1 is the non-basic variable that we want to increase as much as possible. As to the value of x_1, we are bounded by $x_2 = 8 - 2x_1 \geq 0$ and $x_3 = 9 - 3x_1 \geq 0$, from which we obtain our new basic feasible solution $(x_1, x_2, x_3, x_4) = (3, 2, 0, 0)$ and our new dictionary

$$
\begin{aligned}
x_1 &= 3 - \tfrac{1}{3}x_3 + \tfrac{1}{3}x_4 \\
x_2 &= 2 + \tfrac{2}{3}x_3 + \tfrac{1}{3}x_4 \\
z &= 4 - \tfrac{4}{3}x_3 + \tfrac{1}{3}x_4.
\end{aligned}
$$

The non-basic variable whose value we want to increase is x_4. As to the amount with which we are allowed to increase the value of x_4, we have that $x_1 = 3+\frac{1}{3}x_4 \geq 0$ and $x_2 = 2+\frac{1}{3}x_4 \geq 0$. Hence, we see that *there is no restriction to the amount with which we can increase the value of the variable* x_4. This is exactly the situation that signals the unboundedness of the linear program under consideration; whenever this happens, we know that the linear program is unbounded, and the simplex algorithm stops.

Now let us have a look at the special case with multiple optimal solutions. Consider the example of Section 3.3:

$$\max \; 3x_1 + 2x_2$$

subject to

$$
\begin{aligned}
& x_1 + x_2 \leq 5 \\
& 3x_1 + 2x_2 \leq 12 \\
& x_1, x_2 \geq 0.
\end{aligned}
$$

We introduce the slack variables $x_3 = 5 - x_1 - x_2$ and $x_4 = 12 - 3x_1 - 2x_2$ that have to be nonnegative. Starting with the basic feasible solution $(x_1, x_2, x_3, x_4) = (0, 0, 5, 12)$, we obtain the dictionary

$$
\begin{aligned}
x_3 &= 5 - x_1 - x_2 \\
x_4 &= 12 - 3x_1 - 2x_2 \\
z &= 3x_1 + 2x_2.
\end{aligned}
$$

We select x_1 to become basic; this yields the dictionary

$$
\begin{aligned}
x_1 &= 4 - \tfrac{2}{3}x_2 - \tfrac{1}{3}x_4 \\
x_3 &= 1 - \tfrac{1}{3}x_2 + \tfrac{1}{3}x_4 \\
z &= 12 - x_4.
\end{aligned}
$$

From the last dictionary, it follows that we should not increase the value of x_4, but there is nothing against increasing the value of x_2. We only have to take care that $x_1 = 4 - \frac{2}{3}x_2 \geq 0$ and $x_3 = 1-\frac{1}{3}x_2 \geq 0$. Hence, each increase of the value of x_2 with an amount at most 3 yields a feasible solution with equal cost. We see that the presence of multiple optimal solutions is signaled by the occurrence of a dictionary in which the coefficient of each of the non-basic variables is nonnegative with at least one coefficient equal to zero. Strictly speaking, this is not enough, since the constraints may prohibit an increase of this non-basic variable. If a non-basic variable cannot be increased in value because of the constraints, then we call the linear program *degenerate*.

Consider the following example of a degenerate linear program.

$$\max\ x_1 + x_2 - x_3$$

subject to

$$\begin{aligned}
&x_1 + 2x_2 - 3x_3 \leq 0\\
&-x_1 + x_2 + 4x_3 \leq 4\\
&x_1, x_2, x_3 \geq 0.
\end{aligned}$$

We introduce the slack variables $x_4 = -x_1 - 2x_2 + 3x_3$ and $x_5 = 4 + x_1 - x_2 - 4x_3$ that have to be nonnegative. Starting with the basic feasible solution $(x_1, x_2, x_3, x_4, x_5) = (0, 0, 0, 0, 4)$, we obtain the dictionary

$$\begin{aligned}
&x_4 = -x_1 - 2x_2 + 3x_3\\
&x_5 = 4 + x_1 - x_2 - 4x_3\\
&z = x_1 + x_2 - x_3.
\end{aligned}$$

The non-basic variables with positive coefficient in the objective function are x_1 and x_2; for both variables, however, we see that the constraint $x_4 \geq 0$ implies that we cannot increase their value. If we put x_1 in the basis at the expense of x_4, then we obtain the dictionary

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

which corresponds to the basic feasible solution $(x_1, x_2, x_3, x_4, x_5) = (0, 0, 0, 0, 4)$. On the one hand, you may conclude that this move did not get us any further; we are still in the same vertex. On the other hand, we changed the basis; perhaps we are luckier this time. So let us try another iterative step. There is only one possibility: we have to put x_3 in the basis. The constraint $x_5 \geq 0$ limits the increase to 4, and we put x_3 in the basis at the expense of x_5. Our new dictionary reads

$$\begin{aligned}
&x_1 = 12 - 11x_2 - 4x_4 - 3x_5\\
&x_3 = 4 - 3x_2 - x_4 - x_5\\
&z = 8 - 7x_2 - 3x_4 - 2x_5,
\end{aligned}$$

which corresponds to the optimal basic feasible solution $(x_1, x_2, x_3, x_4, x_5) = (12, 0, 4, 0, 0)$. It follows that the seemingly irrelevant change of basis in the previous iteration got us out of the vertex $(x_1, x_2, x_3, x_4, x_5) = (0, 0, 0, 0, 4)$ to which we seemed to be stuck.

This example of a degenerate linear program was not that horrible. The question that we should ask is of course: Will we always be so lucky to move out of a vertex by just changing the basis, or is it possible that we move from one basis to another without getting anywhere? Stated a bit more formally, the question is:

- Will the simplex algorithm always find an optimal solution, or can it cycle and get stuck in a non-optimal solution?

Unfortunately, the answer to this question is NO. Some pathological examples have been created in which the simplex algorithm keeps on returning to the same basis, which does not correspond to an optimal solution. There is a simple remedy against cycling, however. Until now, we have not stated any rule with respect to which non-basic variable we should choose to enter the basis, except that its coefficient in the objective function should be nonnegative. Similarly, we have not been concerned by the question which variable should leave the basis, except that it must be a variable whose value drops to zero when the new variable enters the basis. It turns out that if we prescribe that from among the possible variables we should choose the one with smallest index, then the simplex algorithm will never cycle; this is known as Bland's rule.

Theorem 3.2. (Bland)
The simplex algorithm terminates as long as the entering and leaving variables are selected by the smallest subscript rule in each iteration. □

Theorem 3.2 implies that, if we have succeeded in determining an initial basic feasible solution of the polyhedron, the simplex algorithm will find an optimal solution. Hence, it has become time to describe a systematical way to determine an initial basic feasible solution to the linear program or to conclude that the linear program is infeasible.

Consider any linear program that is stated in the standard form

$$\max \sum_{1 \le j \le n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad (i = 1, \dots, m)$$

$$x_1, x_2, \dots, x_n \ge 0.$$

If all values b_i are nonnegative, then we can use the basic feasible solution in which all variables x_i $(i = 1, \dots, n)$ have value zero and all slack variables x_{n+i} have value b_i $(i = 1, \dots, m)$ to initialize the simplex algorithm with. Hence, suppose that not all values b_i are nonnegative; let b_m denote the most negative one. We now create another linear program, called the *auxiliary* problem, to determine whether the above linear program is feasible, and if the answer is yes, to find an initial basic feasible solution.

We relax each constraint $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$, to $\sum_{j=1}^{n} a_{ij}x_j - x_0 \leq b_i$, where x_0 is an auxiliary variable that we want to be nonnegative. This move enlarges the feasible region; we have that the feasible region of the original problem coincides with the set of feasible solutions with $x_0 = 0$. Since we are interested in finding a basic feasible solution that does not contain x_0 in the basis, we choose as our objective to minimize x_0, that is, our objective function will be to maximize $z = -x_0$.

Theorem 3.3.
The original linear program is feasible if and only the objective value of the optimal solution amounts to zero. □

We solve the auxiliary problem through the simplex algorithm. To start the simplex method, we use $(x_0, x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m-1}, x_{n+m}) = (-b_m, 0, \ldots, 0, b_1 - b_m, \ldots, b_{m-1} - b_m, 0)$ as our initial basic feasible solution; since b_m is minimal from among the set of b_i values, the nonnegativity constraints are satisfied.

Theorem 3.3 provides us with a tool to check whether the original linear program is feasible or not. Consider the case that the original linear program is feasible, that is, we have determined a basic feasible solution with $x_0 = 0$. This is not necessarily a basic feasible solution for the original problem, since x_0 might be a basic variable. To circumvent this problem, we run the simplex algorithm with the additional rule that we choose x_0 to leave the basis whenever this is possible, that is, whenever its value drops to zero, which must happen sooner or later. This optimal basic feasible solution is used in our initialization of the simplex algorithm when solving the original problem.

Section 3.6. Duality

In many practical situations, we may not be interested in finding the optimal solution: for instance, if we just want to know whether there exists a feasible solution with an objective function value that falls within some given interval. To solve such a problem, we want to obtain a quick estimate of the optimal value z^* of the objective function. Consider the example of Section 3.4

$$\max\ z = 2x_1 - 3x_2 + 4x_3$$

subject to

$$\begin{aligned} &x_1 - x_2 + 2x_3 \leq 7 \\ &2x_1 + x_2 + 3x_3 \leq 22 \\ &x_1 - x_2 + x_3 \leq 2 \\ &x_1, x_2, x_3 \geq 0. \end{aligned}$$

To get a lower bound on z^*, we need only come up with a feasible solution. For example $(x_1, x_2, x_3) = (2, 0, 0)$ provides a lower bound of 4, $(x_1, x_2, x_3) = (0, 0, 2)$ provides a lower bound of 8, and $(x_1, x_2, x_3) = (0, 3, 5)$ provides a lower bound of 11.

To get an upper bound, observe that

$$2x_1 - 3x_2 + 4x_3 \leq 2x_1 - 2x_2 + 4x_3 = 2(x_1 - x_2 + 2x_3) \leq 2 \times 7 = 14.$$

Hence, every feasible solution (x_1, x_2, x_3) has a value less than 14. In particular, this inequality holds for the optimal solution, which implies that $z^* \leq 14$. Similarly,

$$2x_1 - 3x_2 + 4x_3 \leq 3x_1 - 3x_2 + 4x_3 = (x_1 - x_2 + 2x_3) + 2(x_1 - x_2 + x_3) \leq 11.$$

Therefore, we have that $z^* \leq 11$. Note that at this point we have proved that $(x_1, x_2, x_3) = (0, 3, 5)$ is an optimal solution, since the objective function value for this solution is equal to the upper bound 11.

To obtain upper bounds, we have constructed linear combinations of constraints. That is, we multiply the first constraint by some non-negative number y_1, the second by y_2, the third by y_3, after which we add them up (in the first case, we had $(y_1, y_2, y_3) = (2, 0, 0)$; in the second case, we had $(y_1, y_2, y_3) = (1, 0, 2)$). The resulting inequality reads

$$\begin{aligned}
&(y_1 + 2y_2 + y_3)x_1 + (-y_1 + y_2 - y_3)x_2 + (2y_1 + 3y_2 + y_3) \leq \\
&7y_1 + 22y_2 + 2y_3.
\end{aligned}$$

Of course, each of these multipliers must be nonnegative: otherwise the corresponding inequality would reverse its direction. Now, we want to use this inequality as an upper bound, that is,

$$\begin{aligned}
&2x_1 - 3x_2 + 4x_3 \leq (y_1 + 2y_2 + y_3)x_1 + (-y_1 + y_2 - y_3)x_2 + (2y_1 + 3y_2 + y_3) \leq \\
&7y_1 + 22y_2 + 2y_3,
\end{aligned}$$

which implies that

$$\begin{aligned}
&y_1 + 2y_2 + y_3 \geq 2 \\
&-y_1 + y_2 - y_3 \geq -3 \\
&2y_1 + 3y_2 + y_3 \geq 4.
\end{aligned}$$

Of course, we want the upper bound to be as small as possible. Thus, we are led to the following problem

$$\min 7y_1 + 22y_2 + 2y_3$$

subject to

$$\begin{aligned}
&y_1 + 2y_2 + y_3 \geq 2 \\
&-y_1 + y_2 - y_3 \geq -3 \\
&2y_1 + 3y_2 + y_3 \geq 4 \\
&y_1, y_2, y_3 \geq 0,
\end{aligned}$$

which is a linear program again. This linear program is called the *dual* of the original linear program.

Summarizing the above, we see that every linear programming problem of the form

$$\max \sum_{1\le j\le n} c_j x_j$$

subject to

$$\sum_{1\le j\le n} a_{ij}x_j \le b_i \quad \text{for all } i\in\{1,\ldots,m\}$$

$$x_j \ge 0, \quad \text{for all } j\in\{1,\ldots,n\}$$

has an associated linear programming problem

$$\min \sum_{1\le i\le m} b_i y_i$$

subject to

$$\sum_{1\le i\le m} a_{ij}y_i \ge c_j \quad \text{for all } j\in\{1,\ldots,n\}$$

$$y_i \ge 0, \quad \text{for all } i\in\{1,\ldots,m\}\ .$$

In this context, the original problem is called the *primal* problem and the associated problem is called the *dual* problem. As in our example, every feasible solution of the dual yields an upper bound on the optimal value of the primal. To state it more explicitly, for every primal feasible solution $(x_1,\ldots,x_n)$ and for every dual feasible solution $(y_1,\ldots,y_m)$, we have that

$$\sum_{1\le j\le n} c_j x_j \le \sum_{1\le j\le n}\left(\sum_{1\le i\le m} a_{ij}y_i\right)x_j = \sum_{1\le i\le m}\left(\sum_{1\le j\le n} a_{ij}x_j\right)y_i \le \sum_{1\le i\le m} b_i y_i.$$

The above inequality is extremely useful: if we happen to stumble upon a primal feasible solution $(x_1^*,\ldots,x_n^*)$ and a dual feasible solution $(y_1^*,\ldots,y_m^*)$ such that

$$\sum_{1\le j\le n} c_j x_j^* = \sum_{1\le i\le n} b_i y_i^*,$$

then we may conclude that both these solutions are optimal.

The question that arises immediately in case of a linear program with finite objective function value is:

- Under which conditions can we guarantee that the optimal solution values of the primal and the dual problem are equal?

Since we are concerned with linear programs with a finite objective function value, we only consider feasible bounded linear programs. For such linear programs, we can determine the optimal solution by means of a dictionary. In case of the general example of Section 3.4, the final dictionary read:

$$\begin{aligned}
x_2 &= 3 + x_1 - x_4 + 2x_6 \\
x_3 &= 5 - x_4 - x_6 \\
x_5 &= 4 - 3x_1 + 4x_4 - 5x_6 \\
z &= 11 - x_1 - x_4 - 2x_6,
\end{aligned}$$

and we showed that this dictionary corresponded to the optimal solution, because the last line implied that the value 11, which was assumed by the current basic feasible solution, was an upper bound on z. If we look more closely at the last line of the dictionary, then we see that $z \leq 11 - x_4 - 2x_6$. Remembering that x_4 and x_6 were the slack variables defined by $x_4 = 7 - x_1 + x_2 - 2x_3$ and $x_6 = 2 - x_1 + x_2 - x_3$, we can transform the inequality into $z \leq (x_1 - x_2 + 2x_3) + 2(x_1 - x_2 + x_3) \leq 11$, which corresponds to the feasible solution $(y_1, y_2, y_3) = (1, 0, 2)$ of the dual problem. Since the value of this solution of the dual problem is equal to the value of a solution of the primal problem, this solution is optimal. Hence, in case of a feasible bounded linear program, we conclude from the last line of the final dictionary that we can always determine an optimal solution to the dual problem with value that is equal to the value of the optimal solution of the primal problem; this solution is such that y_i is equal to the absolute value of the coefficient of x_{n+i}, that is, the slack variable that corresponds to the ith constraint.

Theorem 3.4.
In case of a feasible bounded linear program, the optimal solutions to the primal and the dual problem yield the same objective function value. □

Theorem 3.4 provides us with a good *certificate of optimality*: given a solution to the primal and dual problem, one can easily check whether these solutions are optimal by just checking if both objective solution values coincide. Another pro of Theorem 3.4 is the following. Obviously, the dual of the dual problem is the primal problem. Hence, instead of solving the primal problem, we can just as well solve the dual problem and read the optimal solution to the primal problem from the last line of the final dictionary. This may be useful if the number of constraints in the primal problem is larger than the number of variables, since empirical experiments have shown that the simplex algorithm runs in time proportional to the number of constraints, whereas the number of variables does not make a real difference.

Section 3.7. More examples of linear programming problems

Transportation problem
The transportation problem arises frequently in planning and distribution of goods and services from supply locations to demand locations. Usually a given quantity of goods is available at each supply location (origin), and we have to transport these goods to meet a specified need at each demand location (destination). With a variety of shipping routes that all have

different costs, the objective is to determine how many units should be shipped from each origin to each destination such that all demands are satisfied and the total cost is minimized. This problem can be formulated as a linear program as follows

$$\min \sum_{1\le i\le m} \sum_{1\le j\le n} c_{ij}x_{ij}$$

subject to

$$\sum_{1\le j\le n} x_{ij} \le s_i \quad (\text{for all } i = 1,\ldots,m),$$

$$\sum_{1\le i\le m} x_{ij} \ge d_j \quad (\text{for all } j = 1,\ldots,n),$$

$$x_{ij} \ge 0 \quad (\text{for all } i, j),$$

where c_{ij} is the cost of shipping one unit from origin i to destination j, s_i is a limit on the number of units that can be supplied from origin i, and d_j is the number of units of demand at destination j.

Maximum flow problem
The maximum flow problem, which was introduced in Section 2.3, can be formulated as a linear problem as follows. Recall that the maximum flow problem is the problem of transporting the maximum amount of flow from the source s to the sink t through a network with capacities on the arcs; the capacity of arc (i, j) is denoted by c_{ij}.

$$\max \sum_{(s,j)\in A} x_{sj}$$

subject to

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = 0 \quad (\text{for all } j \in V\backslash\{s,t\})$$

$$0 \le x_{ij} \le c_{ij} \quad (\text{for all } (i,j) \in A),$$

where the network is given by a graph $G = (V, A)$ with source s and sink t and c_{ij} is the capacity of an arc.

Section 3.8. 0-1 Linear programming

There exists a large class of combinatorial optimization problems that cannot be formulated as an ordinary linear program; to obtain a correct formulation, we need to introduce binary variables, that is, variables that can assume the values 0 and 1 only. In this section, we investigate the consequences of this binary nature of variables.

The weighted matching problem for bipartite and arbitrary graphs
Consider the following formulation of the weighted matching problem

$$\min \sum_{\{i,j\} \in E} w_{ij} x_{ij}$$

subject to

$$\sum_{j:\{i,j\} \in E} x_{ij} = 1 \qquad (\text{for all } i \in V),$$

$$x_{ij} \in \{0,1\} \qquad (\text{for all } \{i,j\} \in E).$$

We do not know how to solve this problem, since we do not know how to handle the condition that the variables should be binary. If we relax the condition $x_{ij} \in \{0,1\}$ to $0 \leq x_{ij} \leq 1$, however, then we obtain a linear program. If we solve this linear program and find a binary solution, then we have a solution to the original problem as well. It can be shown that for bipartite graphs the linear programming relaxation always produces integral solutions. This does not hold for arbitrary graphs, however. Consider for example the graph in Figure 3.1, that is, a triangle with equal edge weights. All variables in the optimal solution to the linear programming relaxation have value 0.5.

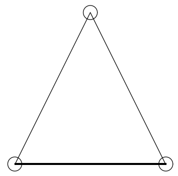


Figure 3.1

Now consider the following inequality

$$\sum_{\{i,j\} \in C} x_{ij} \leq (|C| - 1)/2,$$

where C is an odd cycle in the graph. It is not hard to see that this inequality is satisfied by all integral solutions. It is not satisfied by all solutions to the linear programming relaxation, however, as we can see in our triangle example. This shows that these inequalities are redundant in the 0-1 formulation, but necessary in the linear programming relaxation. It can be shown that if we add all these inequalities to the linear programming relaxation, then the solution to this extended linear programming relaxation will always be integral. Note that there is an exponential number of these inequalities. Therefore, they are handled implicitly in practice.

The 0-1 knapsack problem
The 0-1 knapsack problem is formulated as follows

$$\max \sum_{1\le j\le n} c_j x_j$$

subject to

$$\sum_{1\le j\le n} a_j x_j \le b$$

$$x_j \in \{0,1\}.$$

In this case, the linear programming relaxation does not always produce integral solutions, and we do not know the complete set of linear inequalities that need to be added to force solutions to be integral. In Chapter 4, we provide some solution methods for this problem.