2

Simplex Methods

The main idea of primal cost improvement is to start with a feasible flow vector x and to generate a sequence of other feasible flow vectors, each having a smaller primal cost than its predecessor. The main idea is that if the current flow vector is not optimal, an improved flow vector can be obtained by pushing flow along a simple cycle C with negative cost, that is,

$$\sum_{(i,j)\in C^+} a_{ij} - \sum_{(i,j)\in C^-} a_{ij} < 0,$$

where C^+ and C^- are the sets of forward and backward arcs of C, respectively (see Prop. 2.1 in Section 1.2).

There are several methods for finding negative cost cycles, but the most successful in practice are specialized versions of the simplex method for linear programming. This chapter focuses on methods of this type.

Simplex methods are not only useful for algorithmic solution of the problem; they also provide constructive proofs of some important analytical results. Chief among these are duality theorems asserting the equality of the primal and the dual optimal values, and the existence of optimal primal and dual solutions which are integer if the problem data are integer (see Prop. 2.3 in Section 2.2 and Prop. 3.2 in Section 2.3).

2.1 MAIN IDEAS IN SIMPLEX METHODS

To simplify the presentation, we first consider the version of the minimum cost flow problem with only nonnegativity constraints on the flows:

$$\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \tag{MCF–N}$$

subject to

$$\sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} = s_i, \qquad \forall\ i \in \mathcal{N}, \tag{1.1}$$

$$0 \le x_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}, \tag{1.2}$$

where a_{ij}, and s_i are given scalars.

We saw in Section 1.1.3 that the general minimum cost flow problem with upper and lower bounds on the arc flows can be converted to one with nonnegativity constraints. Thus, once we develop the main method for the simpler problem above, its extension to the more general problem will be straightforward.

The most important difference between the minimum cost flow problem with nonnegativity constraints and the one with upper and lower bounds is that the former can be *unbounded*. By this we mean that feasible flows may take arbitrarily large values, while the corresponding cost takes arbitrarily small (i.e., large negative) values. In particular, *the problem is unbounded if it is feasible and there exists a simple forward cycle with negative cost*, since then we can reduce the cost to arbitrarily large negative values by adding arbitrarily large flow along the negative cost cycle to any feasible flow vector. [In fact, we have seen an instance of this result in connection with the shortest path problem; cf. the corresponding minimum cost flow problem (3.3) in Section 1.3.] The converse is also true: *if the problem is unbounded, there must exist a simple forward cycle with negative cost*. This follows from Prop. 3.5 in Section 1.3, which implies that if the cost of every simple forward cycle is nonnegative, then the cost function of the problem is bounded from below by some constant.

Spanning Trees and Basic Flow Vectors

The main idea of the simplex method is to generate negative cost cycles by using a *spanning tree* of the given graph. Recall from Section 1.1 that a tree is an acyclic connected graph, and that a spanning tree of a given graph is a subgraph that is a tree and includes all nodes of the given graph. A *leaf node* of a tree is defined to be a node with a single incident arc. Figure 1.1 illustrates a spanning tree and a leaf node. The following lemma proves some important properties.

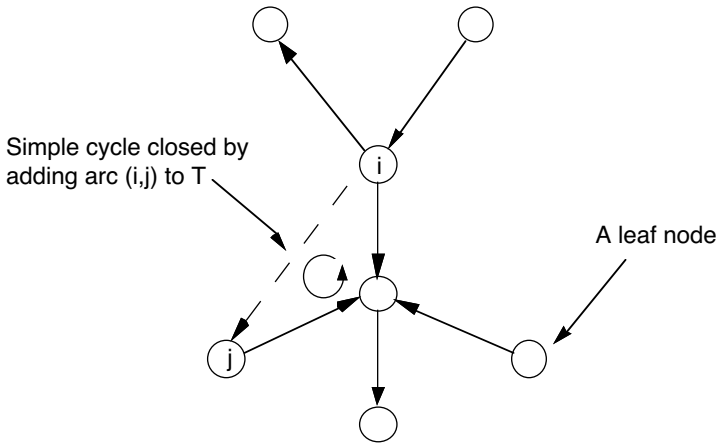


Figure 1.1 Illustration of a spanning tree T. Note that that there is a unique simple path of T connecting any pair of nodes. Furthermore, the addition of any arc to T [arc (i, j) in the figure] creates a unique simple cycle in which (i, j) is a forward arc.

Lemma 1.1: Let T be a subgraph of a graph with N nodes.

(a) If T is acyclic and has at least one arc, then it must have at least one leaf node.

(b) T is a spanning tree if and only if it is connected and has N nodes and $N - 1$ arcs.

(c) If T is a tree, for any two nodes i and j of T there is a unique simple path of T starting at i and ending at j. Furthermore, any arc e that is not in T, when added to T, creates a unique simple cycle in which e is a forward arc.

(d) If T is a tree and an arc (i, j) of T is deleted, the remaining arcs of T form two disjoint trees, one containing i and the other containing j.

Proof: (a) Choose a node n_1 of T with at least one incident arc e_1 and let n_2 be the opposite node of that arc. If n_2 is a leaf node, the result is proved; else choose an arc $e_2 \neq e_1$ that is incident to n_2, and let n_3 be the opposite end node. If n_3 is a leaf node, the result is proved; else continue similarly. Eventually a leaf node will be found, for otherwise some node will be repeated in the sequence, which is impossible since T is acyclic.

(b) Let T be a spanning tree. Then T has N nodes, and since it is connected and acyclic, it must have a leaf node n_1. (We assume without loss of generality that $N \geq 2$.) Delete n_1 and its unique incident arc from T, thereby obtaining

a connected graph T_1, which has $N-1$ nodes and is acyclic. Repeat the process with T_1 in place of T, obtaining T_2, T_3, and so on. After $N-1$ steps and $N-1$ arc deletions, we will obtain T_{N-1}, which consists of a single node. This proves that T has $N-1$ arcs.

Suppose now that T is connected and has N nodes and $N-1$ arcs. If T had a simple cycle, by deleting any arc of the cycle, we would obtain a graph T_1 that would have $N-2$ arcs and would still be connected. Continuing similarly if necessary, we obtain for some $k \geq 1$ a graph T_k, which has $N-k-1$ arcs, and is connected and acyclic (i.e., it is a spanning tree). This is a contradiction, because we proved earlier that a spanning tree has exacly $N-1$ arcs. Hence, T has no simple cycle and must be a spanning tree.

(c) There is at least one simple path starting at a node i and ending at a node j because T is connected. If there were a second path starting at i and ending at j, by reversing this path so that it starts at j and ends at i, and by concatenating it to the first path, we would form a cycle. It can be seen that this cycle must contain a simple cycle, since otherwise the two paths would be identical. This contradicts the hypothesis that T is a tree.

If arc e is added to T, it will form a simple cycle together with any simple path that lies in T and connects its end nodes. Since there is only one such path, it follows that e, together with the arcs of T, forms a unique simple cycle in which e is a forward arc.

(d) It can be seen that removal of a single arc from any connected graph either leaves the graph connected or else creates exactly two connected components. The unique simple path of T connecting i to j consists of arc (i,j); with the removal of this arc, no path connecting i to j remains, and the graph cannot stay connected. Hence, removal of (i,j) must create exactly two connected components, which must be trees since, being subgraphs of T, they must be acyclic. **Q.E.D.**

Suppose that we have a feasible problem and we are given a spanning tree T. A key property for our purposes is that there is a flow vector x, satisfying the conservation of flow constraints (1.1), with the property that only arcs of T can have a nonzero flow. Such a flow vector is called *basic* and is uniquely determined by T, as the following proposition shows.

Proposition 1.1: Assume that $\sum_{i \in \mathcal{N}} s_i = 0$. Then, for any spanning tree T, there exists a unique flow vector x that satisfies the conservation of flow constraints (1.1) and is such that all arcs not in T have zero flow. In particular, if an arc (i,j) of T separates T into two components T_i and T_j, containing i and j respectively, we have

$$x_{ij} = \sum_{n \in T_i} s_n.$$

Proof: To show uniqueness, note that for any flow vector x and arc $(i,j) \in T$ the flux across the cut $[T_i, \mathcal{N} - T_i]$ is equal to the sum of divergences of the nodes of T_i [see Eq. (2.5) in Section 1.2.2]. Thus, if x satisfies the conservation of flow constraints, the flux across the cut must be $\sum_{n \in T_i} s_n$. If in addition all arcs of the cut carry zero flow except for (i,j), this flux is just x_{ij}, so we must have

$$x_{ij} = \begin{cases} \sum_{n \in T_i} s_n & \text{if } (i,j) \in T \\ 0 & \text{if } (i,j) \notin T. \end{cases}$$

Thus, if a flow vector has the required properties, it must be equal to the vector x defined by the preceding formula.

To show existence, i.e. that the flow vector x, defined by the preceding formula, satisfies the conservation of flow constraints, we use a constructive proof based on the algorithm of Fig. 1.2. (An alternative algorithm is outlined in Exercise 1.4.) **Q.E.D.**

Note that a basic flow vector need not be feasible; some of the arc flows may be negative, violating the lower bound constraints (see the example of Fig. 1.2). If the corresponding basic flow vector is feasible, the spanning tree will be called (with slight abuse of terminology) a *feasible tree*.

Overview of the Simplex Method

The simplex method starts with a feasible tree and proceeds in iterations, generating another feasible tree and a corresponding feasible basic flow vector at each iteration. The cost of each basic flow vector is no worse than the cost of its predecessor. At each iteration (also called a *pivot* in the standard terminology of linear programming), the method operates roughly as follows:

(a) It uses a convenient method to add one arc to the tree so as to generate a simple cycle with negative cost.

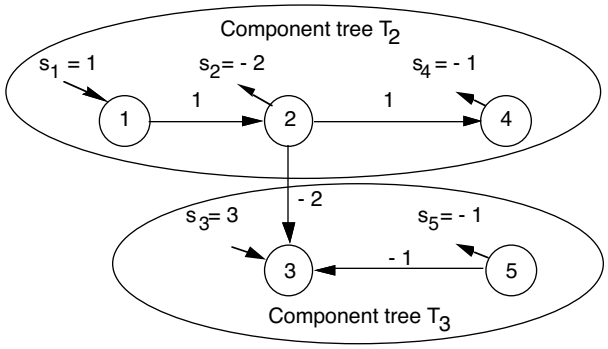
(b) It pushes along the cycle as much flow as possible without violating feasibility.

(c) It discards one arc of the cycle, thereby obtaining another feasible tree to be used at the next iteration.

Thus, each tree $\overline{T}$ in the sequence generated by the simplex method differs from its predecessor T by two arcs: the *out-arc* e, which belongs to T but not to $\overline{T}$, and the *in-arc* $\overline{e}$, which belongs to $\overline{T}$ but not to T; see Fig. 1.3. We will use the notation

$$\overline{T} = T + \overline{e} - e$$

to express this relation. The arc $\overline{e}$ when added to T closes a unique simple cycle in which $\overline{e}$ is a forward arc. This is the cycle along which we try to push



Iteration #	Leaf Node Selected	Arc Flow Computed
1	1	$x_{12} = 1$
2	5	$x_{53} = -1$
3	3	$x_{23} = -2$
4	2	$x_{24} = 1$

Figure 1.2 Method for constructing the flow vector corresponding to T, starting from the arc incident to some leaf node and proceeding "inward." The algorithm maintains a tree R, a flow vector x, and scalars $w_1, \ldots, w_N$. Upon termination, x is the desired flow vector. Initially, $R = T$, $x = 0$, and $w_i = s_i$ for all $i \in \mathcal{N}$.

Step 1: Choose a leaf node $i \in R$. If (i, j) is the unique incident arc of i, set

$$x_{ij} := w_i, \qquad w_j := w_j + w_i;$$

if (j, i) is the unique incident arc of i, set

$$x_{ji} := -w_i, \qquad w_j := w_j - w_i.$$

Step 2: Delete i and its incident arc from R. If R now consists of a single node, terminate; else, go to Step 1.

We now show that if $\sum_{n \in \mathcal{N}} s_n = 0$, the flow vector thus constructed satisfies the conservation of flow equations. Consider the typical iteration where the leaf node i of R is selected in Step 1. Suppose that (i, j) is the unique incident arc of R [the proof is similar if (j, i) is the incident arc]. Then just before this iteration, w_i is equal by construction to $s_i - \sum_{\{k \neq j \mid (i,k) \in \mathcal{A}\}} x_{ik} + \sum_{\{k \mid (k,i) \in \mathcal{A}\}} x_{ki}$, so by setting x_{ij} to w_i, the conservation of flow constraint is satisfied at node i. Upon termination, it is seen that for the last node i of R, w_i is equal to both $\sum_{n \in \mathcal{N}} s_n$ and $s_i - \sum_{\{k \mid (i,k) \in \mathcal{A}\}} x_{ik} + \sum_{\{k \mid (k,i) \in \mathcal{A}\}} x_{ki}$. Since $\sum_{n \in \mathcal{N}} s_n = 0$, the conservation of flow constraint is satisfied at this last node as well.

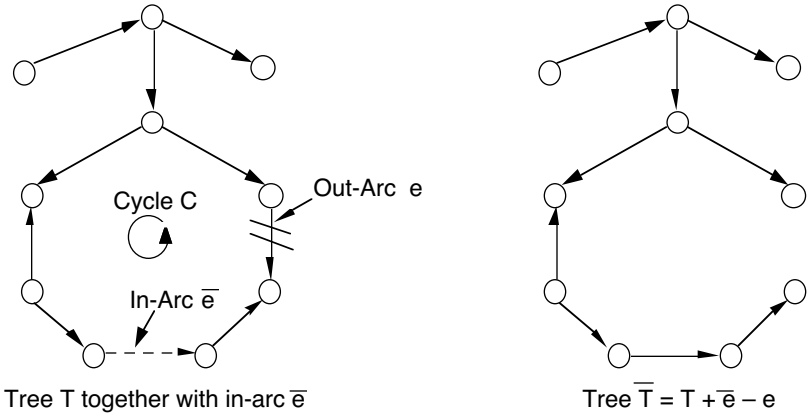


Figure 1.3 Successive trees T and $\overline{T}$ generated by the simplex method.

flow. (By convention, we require that the orientation of the cycle is the same as the orientation of the arc $\overline{e}$.)

Leaving aside for the moment the issue of how to select an initial feasible tree, the main questions now are:

(1) How to select the in-arc so as to close a cycle with negative cost or else detect that the current flow is optimal.

(2) How to select the out-arc so as to obtain a new feasible tree and associated flow vector.

(3) How to ensure that the method makes progress, eventually improving the primal cost. (The problem here is that even if a negative cost cycle is known, it may not be possible to push a positive amount of flow along the cycle because some backward arc on the cycle has zero flow. Thus, the flow vector may not change and the primal cost may not decrease strictly at any one pivot; in linear programming terminology, such a pivot is known as *degenerate*. Having to deal with degeneracy is the price for simplifying the search for a negative cost cycle.)

We take up these questions in sequence.

2.1.1 Using Prices to Obtain the In-Arc

Despite the fact that the simplex method is a primal cost improvement algorithm, it makes essential use of price vectors and duality ideas. In particular, the complementary slackness (CS) conditions

$$p_i - p_j \leq a_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}, \tag{1.3a}$$

$$p_i - p_j = a_{ij}, \qquad \text{for all } (i,j) \in \mathcal{A} \text{ with } 0 < x_{ij} \tag{1.3b}$$

[see Eqs. (2.11c) and (2.11d) in Section 1.2] will play an important role. If x is feasible and together with p satisfies these CS conditions, then x is an optimal solution of the problem (MCF-N) and p is an optimal solution of its dual problem

$$\begin{aligned} &\text{maximize } \sum_{i \in \mathcal{N}} s_i p_i \\ &\text{subject to } \quad p_i - p_j \le a_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}; \end{aligned}$$

the proof of this closely parallels the proof of Prop. 2.4 in Section 1.2 and is outlined in Exercise 2.11 in Section 1.2.

Along with a feasible tree T, the simplex method maintains a *price vector* $p = (p_1, \ldots, p_N)$ such that

$$p_i - p_j = a_{ij}, \qquad \forall\ (i,j) \in T.$$

This is obtained as follows: Fix a node r, called the *root of the tree*, and set p_r to some arbitrary scalar value; for any node i, let P_i be the unique simple path of T starting at the root node r and ending at i, and define p_i by

$$p_i = p_r - \sum_{(m,n) \in P_i^+} a_{mn} + \sum_{(m,n) \in P_i^-} a_{mn}, \tag{1.4}$$

where P_i^+ and P_i^- are the sets of forward and backward arcs of P_i, respectively. To see that with this definition of p_i we have $p_i - p_j = a_{ij}$ for all $(i,j) \in T$, write Eq. (1.4) for nodes i and j, subtract the two equations, and note that the paths P_i and P_j differ by just the arc (i,j).

For an equivalent construction method, select p_r arbitrarily, set the prices of the outward neighbors j of r with $(r,j) \in T$ to $p_j = p_r - a_{rj}$ and the prices of the inward neighbors j of r with $(j,r) \in T$ to $p_j = p_r + a_{jr}$, and then repeat the process with the neighbors j replacing r. Figure 1.4 gives an example.

It can be seen from Eq. (1.4), that *for each pair of nodes i and j, the price difference $(p_i - p_j)$ is independent of the arbitrarily chosen root node price p_r*; write Eq. (1.4) for node i and for node j, and subtract. Therefore, for each arc (i,j), the scalar

$$r_{ij} = a_{ij} + p_j - p_i, \tag{1.5}$$

called the *reduced cost* of the arc, is uniquely defined by the spanning tree T. By the definition of p, we have

$$r_{ij} = 0, \qquad \forall\ (i,j) \in T,$$

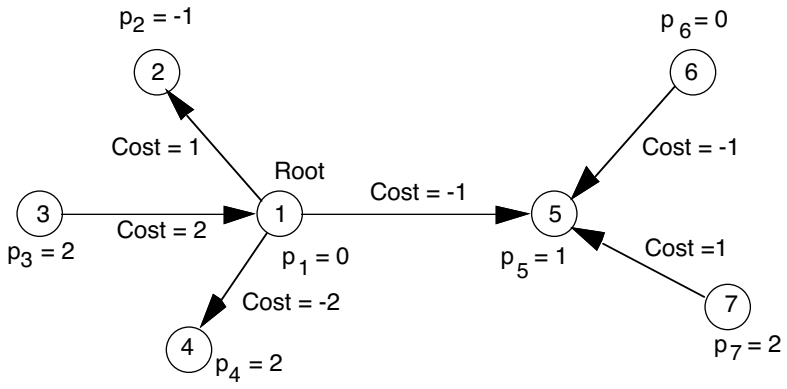


Figure 1.4 Illustration of the prices associated with a spanning tree. The root is chosen to be node 1, and its price is arbitrarily chosen to be 0. The other node prices are then uniquely determined by the requirement $p_i - p_j = a_{ij}$ for all arcs (i,j) of the spanning tree.

so if in addition we have

$$r_{ij} \geq 0, \qquad \forall\ (i,j) \notin T,$$

then the pair (x,p) satisfies the CS conditions (1.3a) and (1.3b). It then follows from Prop. 2.5 of Section 1.2 (more precisely, from the version of that proposition that applies to the problem with only nonnegativity constraints) that x is an optimal primal solution and p is an optimal dual solution.

If on the other hand, we have

$$r_{\overline{i}\,\overline{j}} < 0 \tag{1.6}$$

for some arc $\overline{e} = (\overline{i}, \overline{j})$ not in T, then we claim that the unique simple cycle C formed by T and the arc $(\overline{i}, \overline{j})$ has negative cost. Indeed, the cost of C can be written in terms of the reduced costs of its arcs as

$$\begin{aligned}\sum_{(i,j)\in C^+} a_{ij} - \sum_{(i,j)\in C^-} a_{ij} &= \sum_{(i,j)\in C^+} (a_{ij} + p_j - p_i) - \sum_{(i,j)\in C^-} (a_{ij} + p_j - p_i) \\ &= \sum_{(i,j)\in C^+} r_{ij} - \sum_{(i,j)\in C^-} r_{ij}.\end{aligned} \tag{1.7}$$

Since $r_{ij} = 0$ for all $(i,j) \in T$ [see Eq. (1.5)], and $\overline{e}$ is a forward arc of C by convention, we have

$$\text{Cost of } C\ = r_{\overline{i}\,\overline{j}},$$

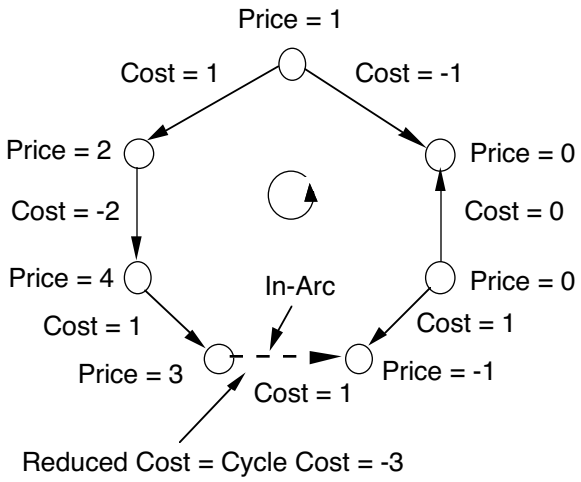


Figure 1.5 Obtaining a negative cost cycle in the simplex method. All arcs of the cycle have zero reduced cost, so the reduced cost of the in-arc is also the cost of the cycle, based on the calculation of Eq. (1.7). Thus, if the in-arc is chosen to have negative reduced cost, the cost of the cycle is also negative.

which is negative by Eq. (1.6); see Fig. 1.5.

The role of the price vector p associated with a feasible tree now becomes clear. By checking the sign of the reduced cost

$$r_{ij} = a_{ij} + p_j - p_i,$$

of all arcs (i, j) not in T, we will either verify optimality if r_{ij} is nonnegative for all (i, j), or else we will obtain a negative cost cycle by discovering an arc (i, j) for which r_{ij} is negative. The latter arc is the in-arc that will enter the tree of the next iteration.

There is a great deal of flexibility for selecting the in-arc. For example, one may search for an in-arc with *most negative* reduced cost; this rule requires a lot of computation – a comparison of r_{ij} for all arcs (i, j) not in the current tree. A simpler alternative is to search the list of arcs not in the tree and to select the *first* arc with negative reduced cost. Most practical simplex codes use an intermediate strategy. They maintain a *candidate list of arcs*, and at each iteration they search through this list for an arc with most negative reduced cost; in the process, arcs with nonnegative reduced cost are deleted from the list. If no arc in the candidate list has a negative reduced cost, a new candidate list is constructed. One way to do this is to scan the full arc list and enter in the candidate list all arcs with negative reduced cost, up to the point where the candidate list reaches a maximum size, which is

chosen heuristically. This procedure can also be used to construct the initial candidate list.

2.1.2 Obtaining the Out-Arc

Let T be a feasible tree generated by the simplex method with corresponding flow vector x and price vector p which are nonoptimal. Suppose that we have chosen the in-arc $\overline{e}$ and we have obtained the corresponding negative cost cycle C formed by T and $\overline{e}$. There are two possibilities:

(a) All arcs of C are oriented like $\overline{e}$, that is, C^- is empty. Then C is a forward cycle with negative cost, indicating that the problem is unbounded. Indeed, since C^- is empty, we can increase the flows of the arcs of C by an arbitrarily large common increment, while maintaining feasibility of x. The primal cost function changes by an amount equal to the cost of C for each unit flow change along C. Since C has negative cost, we see that the primal cost can be decreased to arbitrarily small (i.e. large negative) values.

(b) The set C^- of arcs of C with orientation opposite to that of $\overline{e}$ is nonempty. Then

$$\delta = \min_{(i,j)\in C^-} x_{ij} \tag{1.8}$$

is the maximum increment by which the flow of all arcs of C^+ can be increased and the flow of all arcs of C^- can be decreased, while still maintaining feasibility. The simplex method computes δ and changes the flow vector from x to $\overline{x}$, where

$$\overline{x}_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \notin C \\ x_{ij} + \delta & \text{if } (i,j) \in C^+ \\ x_{ij} - \delta & \text{if } (i,j) \in C^-. \end{cases} \tag{1.9}$$

Any arc $e = (i,j) \in C^-$ that attains the minimum in the equation $\delta = \min_{(i,j)\in C^-} x_{ij}$ satisfies $\overline{x}_{ij} = 0$ and can serve as the out-arc; see Fig. 1.6. (A more specific rule for selecting the out-arc will be given later.) The new tree is

$$\overline{T} = T + \overline{e} - e \tag{1.10}$$

and its associated basic flow vector is $\overline{x}$, given by Eq. (1.9).

Figures 1.7 and 1.8 illustrate the simplex method for some simple examples.

Note that the price vector $\overline{p}$ associated with the new tree $\overline{T}$ can be conveniently obtained from p as follows: Let $\overline{e} = (\overline{i}, \overline{j})$ be the in-arc and let e be the out-arc. If we remove e from T we obtain two trees, $T_{\overline{i}}$ and $T_{\overline{j}}$,

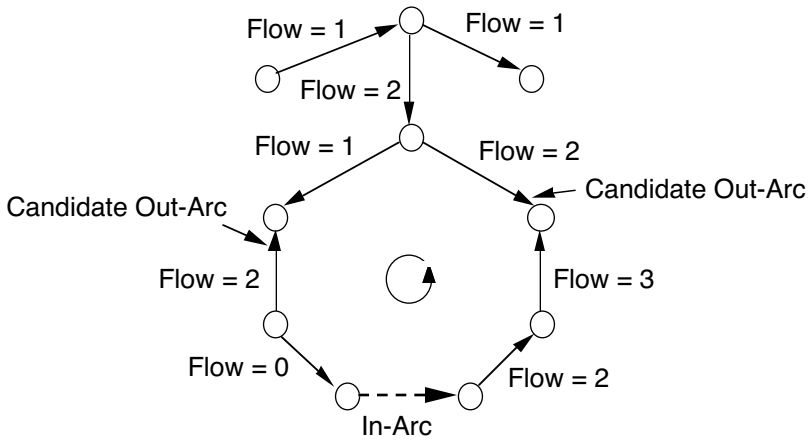


Figure 1.6 Choosing the out-arc in the simplex method. The in-arc $(4,5)$ closes a cycle C. The arcs of C^- are (3,2), (7,6) and (1,7), and define the flow increment $\delta = \min_{(i,j)\in C^-} x_{ij}$. Out of these arcs, the ones attaining the minimum are the candidates for out-arc, as shown.

containing the nodes $\overline{i}$ and $\overline{j}$, respectively; see Fig. 1.9. Then it is seen from the definition (1.4) that a price vector $\overline{p}$ associated with $\overline{T}$ is given by

$$\overline{p}_i = \begin{cases} p_i & \text{if } i \in T_{\overline{i}} \\ p_i - r_{\overline{i}\,\overline{j}} & \text{if } i \in T_{\overline{j}}, \end{cases} \tag{1.11}$$

where

$$r_{\overline{i}\,\overline{j}} = a_{\overline{i}\,\overline{j}} + p_{\overline{j}} - p_{\overline{i}}$$

is the reduced cost cost of the in-arc $(\overline{i}, \overline{j})$. Thus, to update the price vector, one needs to increase the prices of the nodes in $T_{\overline{j}}$ by the common increment $(-r_{\overline{i}\,\overline{j}})$. We may also use any other price vector, obtained by adding the same constant to all the prices $\overline{p}_i$ defined above; it will simply correspond to a different price for the root node. The formula

$$\overline{p}_i = \begin{cases} p_i + r_{\overline{i}\,\overline{j}} & \text{if } i \in T_{\overline{i}} \\ p_i & \text{if } i \in T_{\overline{j}}, \end{cases} \tag{1.12}$$

involving a decrease of the prices of the nodes in $T_{\overline{i}}$, is useful in some implementations.

Note that if the flow increment $\delta = \min_{(i,j)\in C^-} x_{ij}$ [cf. Eq. (1.8)] is positive, then the cost corresponding to $\overline{x}$ will be strictly smaller than the cost corresponding to x (by δ times the cost of the cycle C). Thus, when $\delta > 0$,

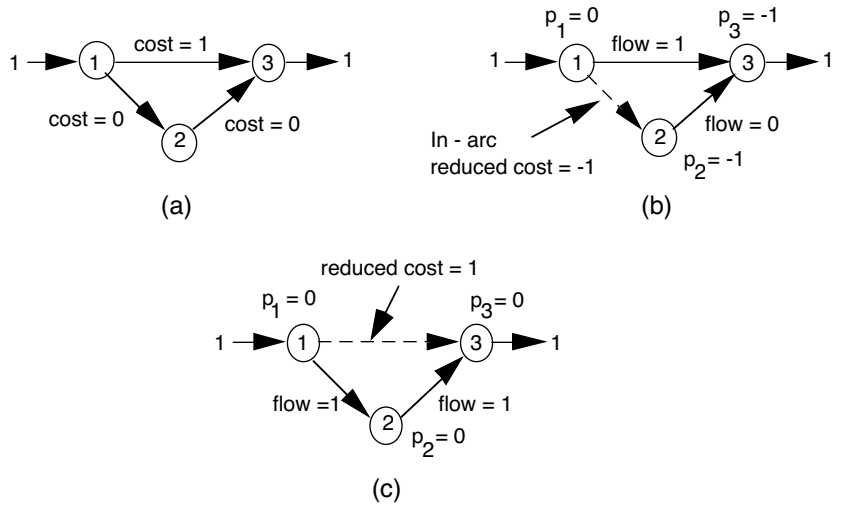


Figure 1.7 Illustration of the simplex method for the problem described in figure (a). The starting tree consists of arcs $(1, 3)$ and $(2, 3)$ and the corresponding flows and prices are as shown in figure (b). Arc $(1, 2)$ has negative reduced cost and is thus eligible to be an in-arc. Arc $(1, 3)$ is the only arc eligible to be the out-arc. The new tree is shown in figure (c). The corresponding flow is optimal because the reduced cost of arc $(1, 3)$ is positive.

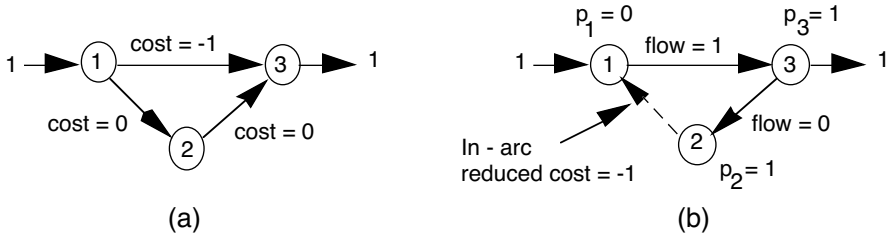


Figure 1.8 Illustration of the simplex method for the problem described in figure (a); this is an unbounded problem because the cycle $(1, 3, 2, 1)$ has negative cost. The starting tree consists of arcs $(1, 3)$ and $(2, 3)$ and the corresponding flows and prices are as shown in figure (b). Arc $(1, 2)$ has negative reduced cost and is thus eligible to be an in-arc. However, all the arcs of the corresponding cycle have the same orientation, so the problem is declared to be unbounded.

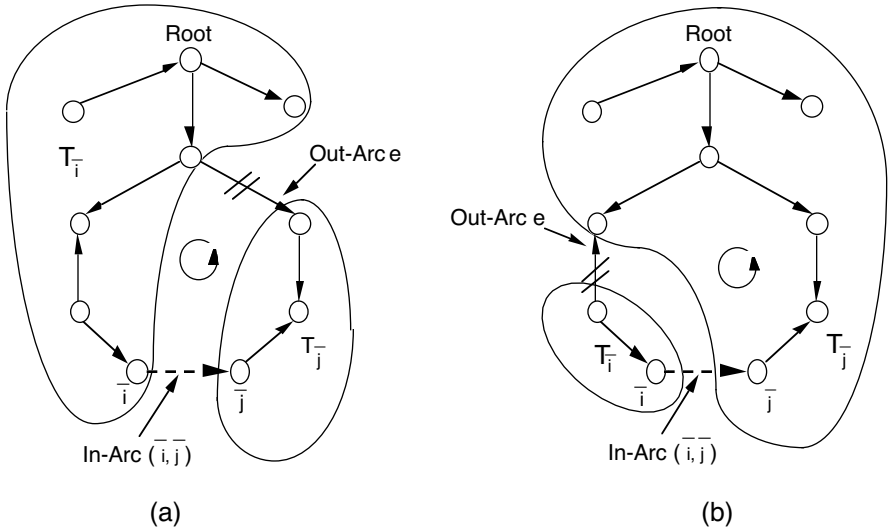


Figure 1.9 Component trees $T_{\overline{i}}$ and $T_{\overline{j}}$, obtained by deleting the out-arc e from T, where $\overline{e} = (\overline{i}, \overline{j})$ is the in-arc; these are the components that contain $\overline{i}$ and $\overline{j}$, respectively. Depending on the position of the out-arc e, the root node may be contained in $T_{\overline{i}}$ as in figure (a), or in $T_{\overline{j}}$ as in figure (b).

the simplex method will never reproduce x and the corresponding tree T in future iterations.

On the other hand, if $\delta = 0$, then $\overline{x} = x$, and the pivot is degenerate. In this case there is no guarantee that the tree T will not be repeated after several degenerate iterations with no interim improvement in the primal cost. We thus need to provide for a mechanism that precludes this from happening.

2.1.3 Dealing with Degeneracy

Suppose that the feasible trees generated by the simplex method are all distinct (which is true in particular when all pivots are nondegenerate). Then, since the number of distinct feasible trees is finite, the method will eventually terminate. Upon termination, there are two possibilities:

(a) The final flow and price vectors are primal and dual optimal, respectively.

(b) The problem is shown to be unbounded because at the final iteration, the cycle closed by the current tree and the in-arc $\overline{e}$ has no arc with orientation opposite to that of $\overline{e}$.

Unfortunately, if the tree sequence is not generated with some care, there is no guarantee that a tree will not be repeated an infinite number of times. To rule out this possibility, thereby ensuring termination of the method, we will use feasible trees with a special property called *strong feasibility*. We will make sure that the initial tree has this property, and we will choose the out-arc in a way that the property is maintained by the algorithm.

Let us fix the root node r used to compute the price vectors associated with feasible trees. Given a feasible tree T, we say that arc $(i, j) \in T$ is *oriented away* from the root if the unique simple path of T from the root to j passes through i. A feasible tree T with corresponding flow vector x is said to be *strongly feasible* if every arc (i, j) of T with $x_{ij} = 0$ is oriented away from the root. Figure 1.10 illustrates strongly feasible trees.

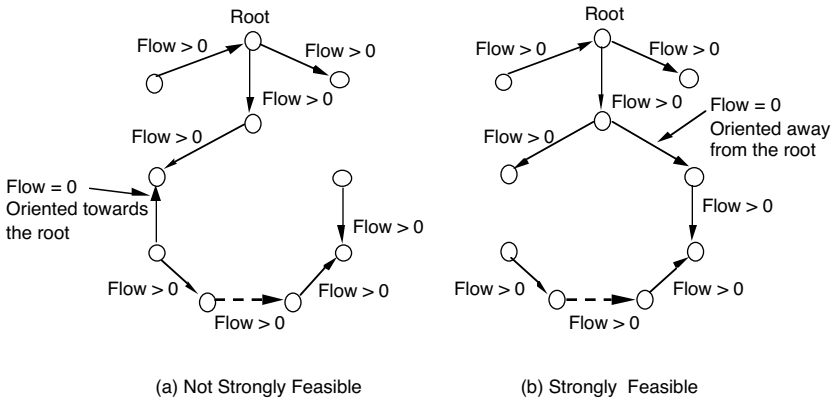


Figure 1.10 Illustration of a strongly feasible tree. The tree in (a) is not strongly feasible because the arc with zero flow on the tree is not oriented away from the root. The tree in (b) is strongly feasible. Note that these two trees are obtained from the strongly feasible tree in Fig. 1.6 by choosing a different out-arc.

The following proposition motivates the use of strongly feasible trees.

Proposition 1.2: If the feasible trees generated by the simplex method are all strongly feasible, then these trees are distinct.

Proof: With each feasible tree T, with corresponding basic feasible vector x and price vector p, we associate the two scalars

$$c(T) = \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \tag{1.13}$$

$$w(T) = \sum_{i\in\mathcal{N}} \big(p_r - p_i\big), \tag{1.14}$$

where r is the root node. [The price differences $p_r - p_i$ are uniquely determined by T according to

$$p_r - p_i = \sum_{(m,n)\in P_i^+} a_{mn} - \sum_{(m,n)\in P_i^-} a_{mn}$$

[see Eq. (1.4)], so $w(T)$ is uniquely determined by T. Note that, $w(T)$ may be viewed as the "aggregate length" of T; it is the sum of the lengths of the paths P_i from the root to the nodes i along the tree T, where the length of an arc (m,n) is a_{mn} or $-a_{mn}$ depending on whether (m,n) is or is not oriented away from the root, respectively.]

We will show that if T and $\overline{T} = T + \overline{e} - e$ are two successive feasible trees generated by the simplex method, then either $c(\overline{T}) < c(T)$ or else $c(\overline{T}) = c(T)$ and $w(\overline{T}) < w(T)$. This proves that no tree can be repeated.

Indeed, if the pivot that generates $\overline{T}$ from T is nondegenerate, we have $c(\overline{T}) < c(T)$, and if it is degenerate we have $c(\overline{T}) = c(T)$. In the former case the result is proved, so assume the latter case holds, and let $\overline{e} = (\overline{i}, \overline{j})$ be the in-arc. Then after the pivot, $\overline{e}$ still has zero flow, and since $\overline{T}$ is strongly feasible, $\overline{e}$ must be oriented away from the root node r. This implies that r belongs to the subtree $T_{\overline{i}}$, and by Eq. (1.11) we have

$$w(\overline{T}) = w(T) + |T_{\overline{j}}| r_{\overline{i}\overline{j}}, \tag{1.15}$$

where $r_{\overline{i}\overline{j}}$ is the reduced cost of $\overline{e}$, and $|T_{\overline{j}}|$ is the number of nodes in the subtree $T_{\overline{j}}$. Since $r_{\overline{i}\overline{j}} < 0$, it follows that $w(\overline{T}) < w(T)$. **Q.E.D.**

The next proposition shows how to select the out-arc in a simplex iteration so as to maintain strong feasibility of the generated trees.

Proposition 1.3: Let T be a strongly feasible tree generated by the simplex method, let $\overline{e} = (\overline{i}, \overline{j})$ be the in-arc, let C be the cycle formed by T and $\overline{e}$, suppose that C^- is nonempty, let $\delta = \min_{(i,j)\in C^-} x_{ij}$, and let $\hat{C}$ be the set of candidate out-arcs, that is, the set

$$\hat{C} = \{(i,j) \in C^- \mid x_{ij} = \delta\}.$$

Define the *join of* C as the first node of C that lies on the unique simple path of T that starts from the root and ends at $\overline{i}$ (see Fig. 1.11). Suppose that the out-arc e is chosen to be the arc of $\hat{C}$ encountered first as C is traversed in the forward direction (the direction of $\overline{e}$) starting from the join node. Then the next tree $\overline{T} = T + \overline{e} - e$ generated by the simplex method is strongly feasible.

Proof: Since the arcs of T which are not in C will not change their flow or orientation relative to the root, to check strong feasibility of $\overline{T}$, we need only be concerned with the arcs of $C + \overline{e} - e$ for which $\overline{x}_{ij} = 0$. These will

be the arcs of $\hat{C} - e$ and possibly arc $\overline{e}$ (in the case $\delta = 0$). By choosing e to be the first encountered arc of $\hat{C}$, all of the arcs of $\hat{C} - e$ will be encountered after e, and following the pivot, they will be oriented away from the join and therefore also from the root. If $\delta = 0$, the arcs (i, j) of $\hat{C}$ satisfy $x_{ij} = 0$, so by strong feasibility of T, all of them, including e, must be encountered *after* $\overline{e}$ as C is traversed in the direction of $\overline{e}$ starting from the join. Therefore, $\overline{e}$ will also be oriented away from the root following the pivot. **Q.E.D.**

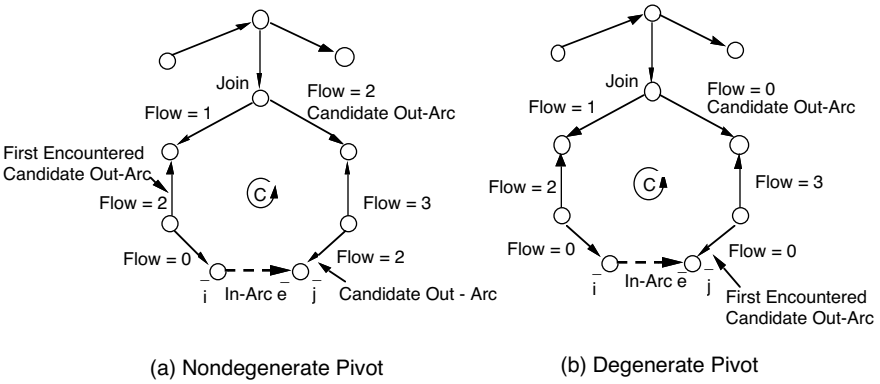


Figure 1.11 Maintaining a strongly feasible tree in the simplex method. Suppose that the in-arc $\overline{e} = (\overline{i}, \overline{j})$ is added to a strongly feasible T, closing the cycle C. Let $\hat{C}$ be the set of candidates for out-arc (the arcs of C^- attaining the minimum in $\delta = \min_{(i,j)\in C^-} x_{ij}$), and let e be the out-arc. The arcs of $\overline{T}$ with zero flow will be the arcs of $\hat{C} - e$ together with $\overline{e}$ if the pivot is degenerate. By choosing as out-arc the first encountered arc of $\hat{C}$ as C is traversed in the direction of $\overline{e}$ starting from the join, all of these arcs will be oriented away from the join and also from the root, so strong feasibility is maintained. Note that if the pivot is degenerate as in (b), then all arcs of $\hat{C}$ will be encountered after $\overline{e}$ (by strong feasibility of T), so the out-arc e must be encountered after $\overline{e}$. Thus, the in-arc $\overline{e}$ will be oriented away from the root in the case of a degenerate pivot, as required for strong feasibility of $\overline{T}$.

E X E R C I S E S

Exercise 1.1

Consider the tree of Fig. 1.11(a).

(a) Suppose that the in-arc is $(\overline{j}, \overline{i})$ [instead of $(\overline{i}, \overline{j})$]. Which arc should be the out-arc?

(b) Suppose that the in-arc is the arc starting at the join and ending at $\overline{j}$ [instead of $(\overline{i}, \overline{j})$]. Which arc should be the out-arc in order to preserve strong feasibility of the tree?

Exercise 1.2

Consider the minimum cost flow problem with nonnegativity constraints given in Fig. 1.12 (supplies are shown next to the nodes, arc costs are immaterial). Find all basic flow vectors and their associated trees. Specify which of these are feasible and which are strongly feasible (the root node is node 1).

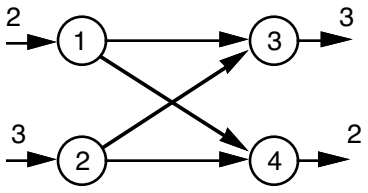


Figure 1.12 Graph for Exercise 1.2.

Exercise 1.3

Consider a feasible minimum cost flow problem such that the corresponding graph is connected. Suppose we are given a feasible flow vector x. Construct an algorithm that suitably modifies x to obtain a basic feasible flow vector and an associated spanning tree. *Hint:* For a feasible flow vector x there are two possibilities: (1) The subgraph S consisting of the set of arcs

$$\mathcal{A}_x = \{(i, j) \in \mathcal{A} \mid x_{ij} > 0\}$$

and the corresponding set of incident nodes is acyclic, in which case show that x is basic. (2) The subgraph S is not acyclic, in which case show how to construct a feasible flow vector x' differing from x by a simple cycle flow, and for which the arc set $\mathcal{A}_{x'}$ has at least one arc less than the set $\mathcal{A}_x$.

Exercise 1.4

Consider the following algorithm that tries to construct a flow vector that has a given divergence vector s, and is zero on arcs which are not in a given spanning tree T. For any vector x, define the surplus of each node i by

$$g_i = \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} - \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} + s_i.$$

The algorithm is initialized with $x = 0$. The typical iteration starts with a flow vector x and produces another flow vector $\overline{x}$ that differs from x along a simple path consisting of arcs of T. It operates as follows: a node i with $g_i > 0$ and a node j with $g_j < 0$ are selected, and the unique path P_{ij} that starts at i, ends at j, and has arcs in T is constructed (if no such nodes i and j can be found the algorithm stops). Then the flow of the forward arcs of P_{ij} are increased by δ and the flow of the backward arcs of P_{ij} are decreased by δ, where $\delta = \min\{g_i, -g_j\}$. Show that the algorithm terminates in a finite number of iterations, and that upon termination, we have $g_i = 0$ for all i if and only if $\sum_{i\in\mathcal{N}} s_i = 0$. *Hint*: Show that all the nodes with zero surplus with respect to x also have zero surplus with respect to $\overline{x}$. Furthermore, at least one node with nonzero surplus with respect to x has zero surplus with respect to $\overline{x}$.

Exercise 1.5

Consider a transportation problem involving the set of sources $\mathcal{S}$ and the set of sinks $\mathcal{T}$ (cf. Example 1.3 in Section 1.1). Suppose that there is no strict subset $\overline{\mathcal{S}}$ of $\mathcal{S}$ and strict subset $\overline{\mathcal{T}}$ of $\mathcal{T}$ such that

$$\sum_{i\in\overline{\mathcal{S}}} \alpha_i = \sum_{j\in\overline{\mathcal{T}}} \beta_j.$$

Show that for every feasible tree, the corresponding flow of every arc of the tree is positive. Conclude that if a feasible initial tree can be found, degeneracy never arises in the simplex method.

2.2 THE BASIC SIMPLEX ALGORITHM

We are now ready to state formally the simplex algorithm based on the ideas of the previous section.

At the beginning of each iteration we have a strongly feasible tree T and an associated basic flow vector x such that

$$x_{ij} = 0, \qquad \forall\ (i,j) \notin T \tag{2.1}$$

and a price vector p such that

$$r_{ij} = a_{ij} + p_j - p_i = 0, \qquad \forall\ (i,j) \in T. \tag{2.2}$$

The iteration has three possible outcomes:

(a) We will verify that x and p are primal and dual optimal, respectively.

(b) We will determine that the problem is unbounded.

(c) We will obtain by the method of Prop. 1.3 a strongly feasible tree $\overline{T} = T + \overline{e} - e$, differing from T by the in-arc $\overline{e}$ and the out-arc e.

Typical Simplex Iteration

Select an in-arc $\overline{e} = (\overline{i}, \overline{j}) \notin T$ such that

$$r_{\overline{i}\overline{j}} = a_{\overline{i}\overline{j}} + p_{\overline{j}} - p_{\overline{i}} < 0.$$

(If no such arc can be found, terminate; x is primal-optimal and p is dual-optimal.) Consider the cycle C formed by T and $\overline{e}$. If C^- is empty, terminate (the problem is unbounded); else, obtain the out-arc $e \in C^-$ as described in Prop. 1.3.

2.2.1 Justification of the Simplex Method

We now collect the facts already proved into a proposition that also deals with the integrality of the solutions obtained.

Proposition 2.1: Suppose that the simplex method is applied to the minimum cost flow problem with the nonnegativity constraints, starting with a strongly feasible tree.

(a) If the problem is not unbounded, the method terminates with an optimal primal solution x and an optimal dual solution p, and the optimal primal cost is equal to the optimal dual cost. Furthermore, if the supplies s_i are all integer, the optimal primal solution x is integer; if the starting price of the root node and the cost coefficients a_{ij} are all integer, the optimal dual solution p is integer.

(b) If the problem is unbounded, the method verifies this after a finite number of iterations.

Proof: (a) The trees generated by the method are strongly feasible, and by Prop. 1.2 these trees are all distinct, so the method terminates. Termination can only occur with either an optimal pair (x, p) or with the indication that the problem is unbounded. Thus, if the problem is not unbounded, the only possibility is termination with an optimal pair (x, p). Since upon termination x and p satisfy complementary slackness, the equality of the optimal primal and dual values follows from Prop. 2.3 in Section 1.2. Also, if the supplies s_i are all integer, from Prop. 1.1 it follows that all basic flow vectors are integer, including the one obtained at termination. If the starting price of the root node and the cost coefficients a_{ij} are all integer, it can be checked that all operations of the algorithm maintain the integrality of p.

(b) If the problem is unbounded, there is no optimal primal solution, so the simplex method cannot terminate with an optimal pair (x, p). The only other possibility is for the method to terminate with an indication that the problem is unbounded. **Q.E.D.**

2.2.2 Choosing the Initial Strongly Feasible Tree – The Big-M Method

In the absence of an apparent choice for an initial strongly feasible tree, one may use the so called *big-M method*. In this method, some artificial variables are introduced to simplify the choice of an initial basic solution, but the cost coefficient M for these variables is chosen large enough so that the optimal solutions of the problem are not affected.

In particular, we modify the problem by introducing an extra node, labeled 0 and having zero supply $s_0 = 0$, together with a set of artificial arcs $\overline{\mathcal{A}}$ consisting of an arc $(i, 0)$ for each node i with $s_i > 0$, and an arc $(0, i)$ for each node i with $s_i \leq 0$. The cost coefficient of all these arcs is taken to be a scalar M, and its choice will be discussed shortly. We thus arrive at the following problem, referred to as the *big-M version* of the original problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} + M\left(\sum_{(i,0)\in\overline{\mathcal{A}}} x_{i0} + \sum_{(0,i)\in\overline{\mathcal{A}}} x_{0i}\right) \\ \text{subject to} \quad & \sum_{\{j|(i,j)\in\mathcal{A}\cup\overline{\mathcal{A}}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\cup\overline{\mathcal{A}}\}} x_{ji} = s_i, \quad \forall\ i \in \mathcal{N}\cup\{0\}, \\ & 0 \leq x_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}\cup\overline{\mathcal{A}}. \end{aligned} \tag{2.3}$$

The artificial arcs constitute a readily available initial spanning tree for the big-M version; see Fig. 2.1. It can be seen that the corresponding basic

flow vector is given by

$$x_{i0} = s_i, \qquad \text{for each } i \text{ with } s_i > 0$$

$$x_{0i} = -s_i, \qquad \text{for each } i \text{ with } s_i \leq 0$$

$$x_{ij} = 0, \qquad \forall\ (i,j) \in \mathcal{A}$$

and is therefore feasible. Let us choose the root to be the artificial node 0. The artificial arcs that carry zero flow are then oriented away from the root, so the tree is strongly feasible.

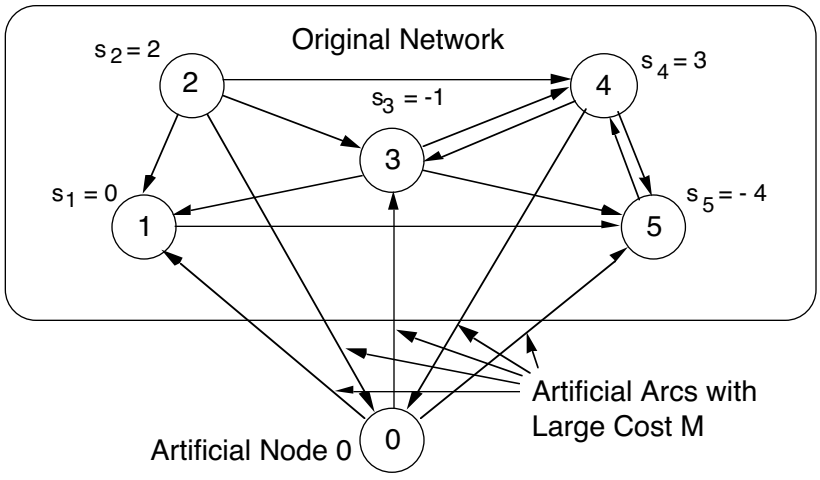


Figure 2.1 Artificial arcs used to modify the problem so as to facilitate the choice of an initial strongly feasible tree.

The cost M of the artificial arcs should be taken to be large enough so that these arcs will carry zero flow at every optimal solution of the big-M version. In this case, the flows of the nonartificial arcs define an optimal solution of the original problem. The following proposition quantifies the appropriate level of M for this to happen, and collects a number of related facts.

Proposition 2.2: Consider the minimum cost flow problem with nonnegativity constraints (referred to as the original problem), and consider also its big-M version. Suppose that

$$2M > \sum_{(i,j)\in P^+} a_{ij} - \sum_{(i,j)\in P^-} a_{ij} \tag{2.4}$$

for all simple paths P of the original problem graph.

(a) If the original problem is feasible but not unbounded, the big-M version has optimal solutions $\overline{x}$, and each of these solutions is of the form

$$\overline{x}_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in \mathcal{A} \\ 0 & \text{if } (i,j) \in \overline{\mathcal{A}}, \end{cases} \tag{2.5}$$

where x is an optimal solution of the original. Furthermore, every optimal solution x of the original problem gives rise to an optimal solution $\overline{x}$ of the big-M version via the preceding relation.

(b) If the original problem is unbounded, the big-M version is also unbounded.

(c) If the original problem is infeasible, then in every feasible solution of the big-M version some artificial arc carries positive flow.

Proof: (a) We first note that the big-M version cannot be unbounded unless the original problem is. To prove this, we argue by contradiction. If the big-M version is unbounded and the original problem is not, there would exist a simple forward cycle with negative cost in the big-M version. This cycle cannot consist of arcs of $\mathcal{A}$ exclusively, since the original is not unbounded. On the other hand, if the cycle consisted of the arcs $(m,0)$ and $(0,n)$, and a simple path of the original graph, then by the condition (2.4) the cycle would have positive cost, arriving at a contradiction.

Having proved that the big-M version is not unbounded, we now note that, by Prop. 2.1(a), the simplex method starting with the strongly feasible tree of all the artificial arcs will terminate with optimal primal and dual solutions of the big-M version. Thus, optimal solutions of the big-M version exist, and for every optimal solution $\overline{x}$ of the form (2.5), the corresponding vector $x = \{x_{ij} \mid (i,j) \in \mathcal{A}\}$ with $x_{ij} = \overline{x}_{ij}$ for all $(i,j) \in \mathcal{A}$ is an optimal solution of the original problem.

To prove that all optimal solutions $\overline{x}$ of the big-M version are of the form (2.5), we argue by contradiction. Suppose that $\overline{x}$ is an optimal solution such that some artificial arcs carry positive flow. Let

$$\mathcal{N}^+ = \{m \mid s_m > 0, \overline{x}_{m0} > 0\},$$

$$\mathcal{N}^- = \{n \mid s_n \leq 0, \overline{x}_{0n} > 0\}.$$

We observe that $\mathcal{N}^+$ and $\mathcal{N}^-$ must be nonempty and that there is no unblocked simple path P with respect to $\overline{x}$ that starts at some $m \in \mathcal{N}^+$ and ends at some $n \in \mathcal{N}^-$; such a path, together with arcs $(m,0)$ and $(0,n)$, would form an unblocked simple cycle, which would have negative cost in view of condition (2.4). Consider now the flow vector $x = \{x_{ij} \mid (i,j) \in \mathcal{A}\}$ with $x_{ij} = \overline{x}_{ij}$ for all $(i,j) \in \mathcal{A}$. Then, there is no path with respect to x of the original problem

graph $(\mathcal{N}, \mathcal{A})$, that is unblocked with respect to x and that starts at a node of $\mathcal{N}^+$ and ends at a node of $\mathcal{N}^-$. By using a very similar argument as in the proof of Prop. 2.2 of Section 1.2, we can show (see Exercise 2.14 in Section 1.2) that there must exist a saturated cut $[\mathcal{S}, \mathcal{N} - \mathcal{S}]$ such that $\mathcal{N}^+ \subset \mathcal{S}$, $\mathcal{N}^- \subset \mathcal{N} - \mathcal{S}$. The capacity of this cut is equal to the sum of the divergences of the nodes $i \in \mathcal{S}$,

$$\sum_{i \in \mathcal{S}} y_i = \sum_{i \in \mathcal{S}} \left(\sum_{\{j \mid (i,j) \in \mathcal{A}\}} x_{ij} - \sum_{\{j \mid (j,i) \in \mathcal{A}\}} x_{ji} \right),$$

which is also equal to

$$\sum_{i \in \mathcal{S}} \bigl(s_i - \overline{x}_{i0}\bigr) = \sum_{i \in \mathcal{S}} s_i - \sum_{i \in \mathcal{N}^+} \overline{x}_{i0} < \sum_{i \in \mathcal{S}} s_i.$$

On the other hand, if the original problem is feasible, the capacity of any cut $[\mathcal{S}, \mathcal{N} - \mathcal{S}]$ cannot be less than $\sum_{i \in \mathcal{S}} s_i$, so we obtain a contradiction.

Finally, let x be an optimal solution of the original problem, and let $\overline{x}$ be given by Eq. (2.5). We will show that $\overline{x}$ is optimal for the big-M version. Indeed, every simple cycle that is unblocked with respect to $\overline{x}$ in the big-M version either consists of arcs in $\mathcal{A}$ and is therefore unblocked with respect to x in the original, or else consists of the arcs $(m, 0)$ and $(0, n)$, and a simple path $\overline{P}$ that starts at n and ends at m. In the former case, the cost of the cycle is nonnegative, since x is optimal for the original problem; in the latter case, the cost of the cycle is positive by condition (2.4) (with the path P being the reverse of path $\overline{P}$). Hence, $\overline{x}$ is optimal for the big-M version.

(b) Note that every feasible solution x of the original problem defines a feasible solution $\overline{x}$ of equal cost in the big-M version via Eq. (2.5). Therefore, if the cost of the original can be made arbitrarily large negative, the same is true of the big-M version.

(c) Observe that any feasible solution of the big-M version having zero flow on the artificial arcs defines a feasible solution x of the original via Eq. (2.5). **Q.E.D.**

Note that to satisfy the condition (2.4), it is sufficient to take

$$M > \frac{(N-1)C}{2},$$

where C is the arc cost range $C = \max_{(i,j) \in \mathcal{A}} |a_{ij}|$. Note also that if M does not satisfy the condition (2.4), then the big-M version may be unbounded, even if the original problem has an optimal solution (Exercise 2.2). Many practical simplex codes use an adaptive strategy for selecting M, whereby a

moderate value of M is used initially, and this value is gradually increased if positive flows on the artificial arcs persist.

By combining the results of the preceding two propositions, we obtain the following proposition.

Proposition 2.3: Assume that the minimum cost flow problem with nonnegativity constraints is feasible and is not unbounded. Then there exists an optimal primal solution and an optimal dual solution, and the optimal primal cost is equal to the optimal dual cost. Furthermore, if the supplies s_i are all integer, there exists an optimal primal solution which is integer; if the cost coefficients a_{ij} are all integer, there exists an optimal dual solution which is integer.

Proof: Apply the simplex method to the big-M version with the initial strongly feasible tree of all the artificial arcs, and with M sufficiently large to satisfy condition (2.4). Then, by Prop. 2.2, the big-M version has optimal solutions, so by Prop. 2.1 the simplex method will provide an optimal pair $(\overline{x}, \overline{p})$, with $\overline{x}$ integer if the supplies are integer, and $\overline{p}$ integer if the cost coefficients are integer. By Prop. 2.2, the vector x defined by $x_{ij} = \overline{x}_{ij}$, for all $(i,j) \in \mathcal{A}$ will be an optimal solution of the original problem, while the price vector p defined by $p_i = \overline{p}_i$, for all $i \in \mathcal{N}$ will satisfy the CS conditions together with x. Hence, p will be an optimal dual solution. **Q.E.D.**

A Shortest Path Example

Consider a single origin/all destinations shortest path problem involving the graph of Fig. 2.2. We will use this example to illustrate the simplex method and some of its special properties when applied to shortest path problems. The corresponding minimum cost flow problem is

$$\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij}$$

subject to

$$\sum_{\{j|(1,j)\in\mathcal{A}\}} x_{1j} - \sum_{\{j|(j,1)\in\mathcal{A}\}} x_{j1} = 3,$$

$$\sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} = -1, \qquad i = 2,3,4,$$

$$0 \le x_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}.$$

We select as root the origin node 1. To deal with the problem of the initial choice of a strongly feasible tree, we use a variant of the big-M method. We introduce artificial arcs connecting the origin with each node $i \neq 1$ with very large cost M, and we use as an initial tree the set of artificial arcs with

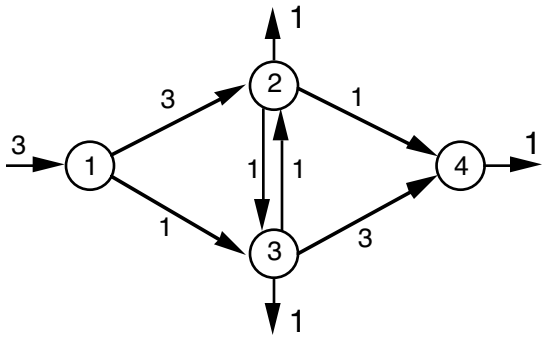


Figure 2.2 Example single origin/all destinations shortest path problem. The arc lengths are shown next to the arcs.

root node the origin (with this choice, there will be two arcs connecting the origin with each of its neighbors, but this should not cause any confusion). In the corresponding flow vector, every artificial arc carries unit flow, so the initial tree is strongly feasible (all arcs are oriented away from the root).

The corresponding price vector is $(0, -M, -M, -M)$ and the associated reduced costs of the nonartificial arcs are

$$r_{1j} = a_{1j} - M, \qquad \forall\ (1, j) \in \mathcal{A},$$

$$r_{ij} = a_{ij}, \qquad \forall\ (i, j) \in \mathcal{A},\ i \neq 1,\ j \neq 1.$$

One possible outcome of the first iteration is to select some arc $(1, j) \in \mathcal{A}$ as in-arc, and to select the artificial arc connecting 1 and j as out-arc. The process will then be continued, first obtaining the flow and price vectors corresponding to the new tree, then obtaining the out-arc, then the in-arc, etc.

Figures 2.3 and 2.4 show two possible sequences of pivots. The following can be noted:

(a) Each artificial arc eventually becomes the out-arc but never becomes the in-arc.

(b) In all trees, all the arcs are oriented away from the origin and carry unit flow.

(c) In Fig. 2.3, where the in-arc is selected to be the arc with minimum reduced cost, there are exactly $N - 1$ $(= 3)$ pivots, and each time the out-arc is an artificial arc. In fact, in this case the simplex method works exactly like Dijkstra's method, permanently setting the label of one additional node with every pivot; here, node labels should be identified with the negative of node prices.

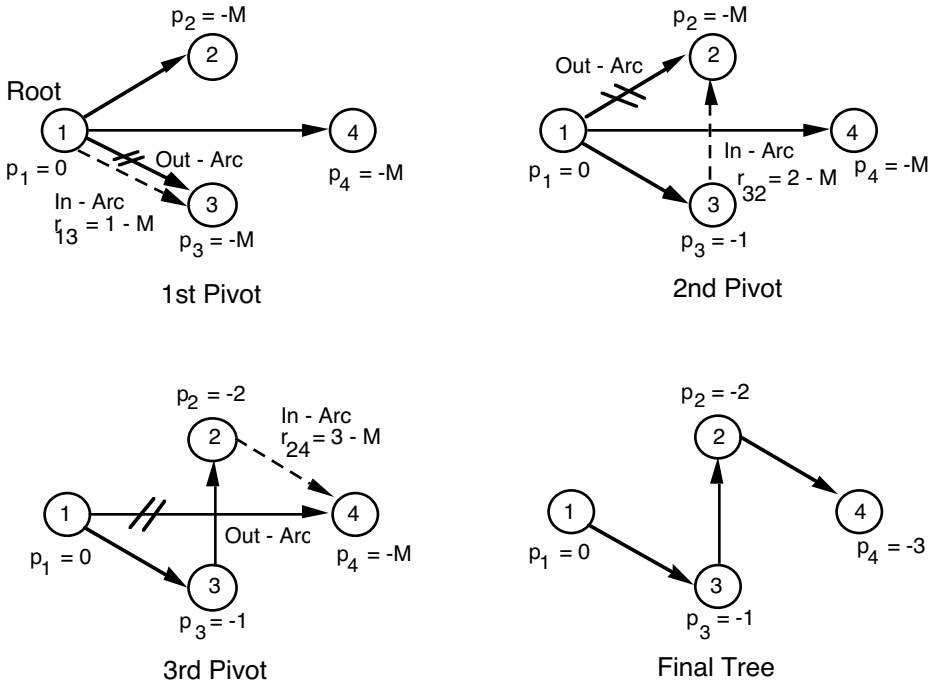


Figure 2.3 A possible sequence of pivots for the simplex method. The initial tree consists of the artificial arcs $(1,2)$, $(1,3)$, and $(1,4)$, each carrying one unit of flow. The in-arc is selected to be the arc with minimum reduced cost and the method behaves like Dijkstra's method, requiring only three $(= N-1)$ pivots.

It can be shown that observations (a) and (b) above hold in general for the simplex method applied to feasible shortest path problems, and observation (c) also holds in general provided $a_{ij} \geq 0$ for all arcs (i,j). The proof of this is left as Exercise 2.8 for the reader.

The simplex method can also be used effectively to solve the all-pairs shortest path problem. In particular, one may first use the simplex method to solve the shortest path problem for a single origin, say node 1, and then modify the final tree T_1 to obtain an initial tree T_2 for applying the simplex method with another origin, say node 2. This can be done by deleting the unique arc of T_1 that is incoming to node 2, and replacing it with an artificial arc from 2 to 1 that has a very large cost; see Fig. 2.5.

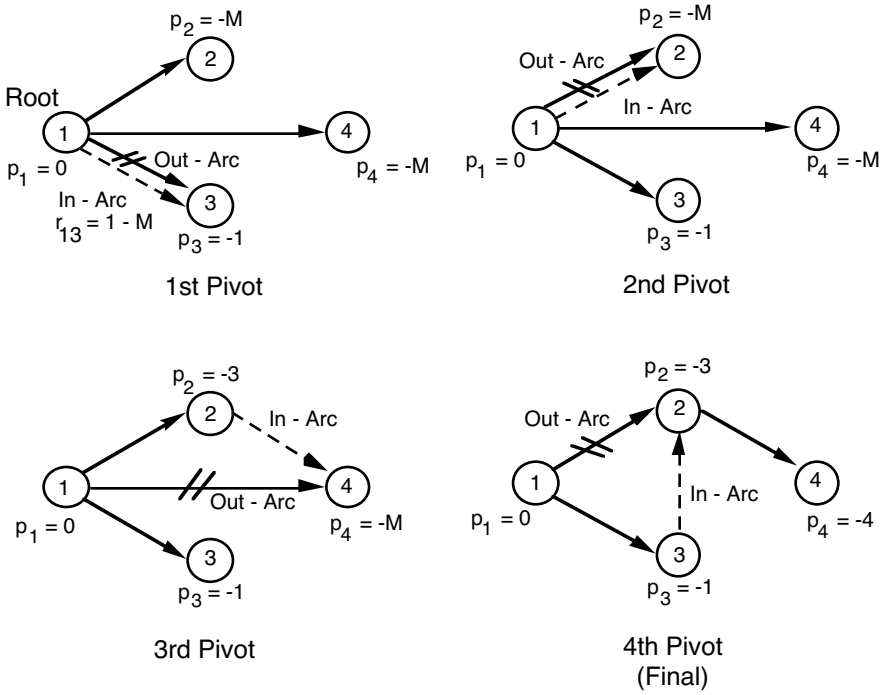


Figure 2.4 Another possible sequence of pivots for the simplex method. More than three pivots are required, in contrast with the sequence of Fig. 2.3.

E X E R C I S E S

Exercise 2.1

Use the simplex method with the big-M initialization to solve the problem in Fig. 2.6.

Exercise 2.2

Construct an example where M does not satisfy the condition (2.4), and the original problem has an optimal solution, while the big-M version is unbounded. *Hint:* It is sufficient to consider a graph with two nodes.

Exercise 2.3

Construct an example where M satisfies the condition (2.4), and the original problem is infeasible, while the big-M version is unbounded. *Hint:* Consider

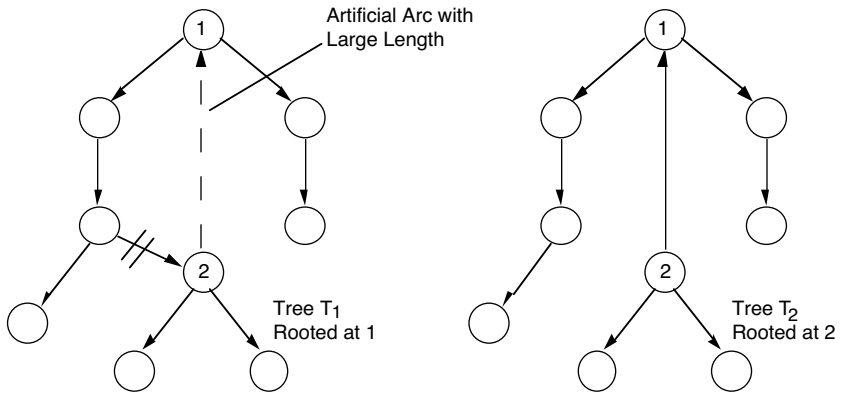


Figure 2.5 Obtaining an initial tree T_2 for the simplex method applied to the shortest path problem with origin 2, from the final tree T_1 of the simplex method applied for origin 1. We delete the unique arc of T_1 that is incoming to node 2, and replace it with an artificial arc from 2 to 1 that has a very large length.

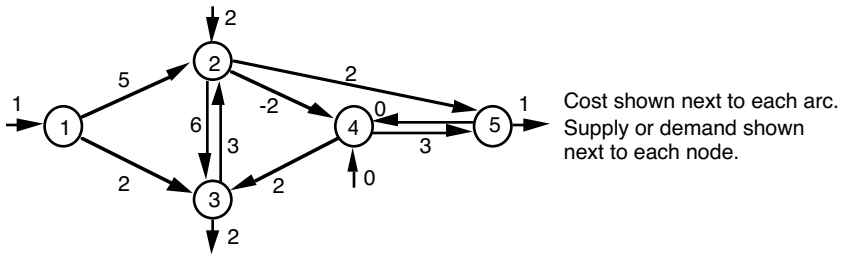


Figure 2.6 Minimum cost flow problem with nonnegativity constraints for Exercise 2.1.

problems that are infeasible and also contain a simple forward cycle of negative cost.

Exercise 2.4 (An Example of Cycling [Chv83])

Consider an assignment problem with sources 1, 2, 3, 4 and sinks 5, 6, 7, 8. There is an arc between each source and each sink. The arc costs are as follows:

$$a_{16} = a_{17} = a_{25} = a_{27} = a_{35} = a_{36} = a_{48} = 1, \qquad a_{ij} = 0 \text{ otherwise.}$$

Let the initial feasible tree consist of arcs (1,5), (1,6), (2,6), (2,8), (4,8), (4,7), (3,7), with corresponding arc flows

$$x_{15} = x_{26} = x_{37} = x_{48} = 1, \qquad x_{ij} = 0 \text{ otherwise.}$$

Suppose that the simplex method is applied without restriction on the choice of the out-arc (so the generated trees need not be strongly feasible). Verify that one possible sequence of in-arc/out-arc pairs is given by

$$\big((1,8),(2,8)\big), \big((3,6),(1,6)\big), \big((4,6),(4,7)\big),$$
$$\big((3,5),(3,6)\big), \big((3,8),(1,8)\big), \big((2,5),(3,5)\big),$$
$$\big((4,5),(4,6)\big), \big((2,7),(2,5)\big), \big((2,8),(3,8)\big),$$
$$\big((1,7),(2,7)\big), \big((4,7),(4,5)\big), \big((1,6),(1,7)\big),$$

and that after these twelve pivots we obtain the initial tree again.

Exercise 2.5 (Birchoff's Theorem for Doubly Stochastic Matrices)

A *doubly stochastic* $n \times n$ matrix $X = \{x_{ij}\}$ is a matrix such that the elements of each of its rows and columns are nonnegative, and add to one, that is, $x_{ij} \geq 0$ for all i and j, $\sum_{j=1}^{n} x_{ij} = 1$ for all i, and $\sum_{i=1}^{n} x_{ij} = 1$ for all j. A *permutation matrix* is a doubly stochastic matrix whose elements are either one or zero, so that there is a single one in each row and each column, with all other elements being zero.

(a) Show that given a doubly stochastic matrix X, there exists a permutation matrix X^* such that, for all i and j, if $x^*_{ij} = 1$, then $x_{ij} > 0$. *Hint:* View X as a feasible solution of the minimum cost flow version of an assignment problem, and view X^* as a feasible assignment.

(b) Use part (a) to show constructively that every doubly stochastic matrix X can be written as $\sum_{i=1}^{k} \gamma_i X^*_i$, where X^*_i are permutation matrices and $\gamma_i \geq 0$, $\sum_{i=1}^{k} \gamma_i = 1$. *Hint:* Define a sequence of matrices $X_0, X_1, \ldots, X_k$, which are nonnegative multiples of doubly stochastic matrices, such that $X_0 = X$, $X_k = 0$, and for all i, $X_i - X_{i+1}$ is a positive multiple of a permutation matrix.

Exercise 2.6 (Hall's Theorem for Perfect Matrices)

A *perfect* matrix is a matrix with nonnegative integer elements such that the elements of each of its rows and each of its columns add to the same integer k. Show that such a perfect matrix can be written as the sum of k permutation matrices. *Hint:* Use the hints and constructions of the preceding exercise.

Exercise 2.7 (Dual Feasibility Theorem)

Show that the dual problem is feasible, that is, there exists a price vector p with

$$p_i - p_j \leq a_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}$$

if and only if all forward cycles have nonnegative cost. *Hint:* Assume without loss of generality that the primal is feasible (take $s_i = 0$ if necessary), and note that all forward cycles have nonnegative cost if and only if the primal problem is not unbounded (see the discussion near the beginning of Section 2.1).

Exercise 2.8 (Relation of Dijkstra and Simplex for Shortest Paths)

Consider the single origin/all destinations shortest path problem

$$\begin{aligned}
&\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \\
&\text{subject to} \\
&\qquad \sum_{\{j|(1,j)\in\mathcal{A}\}} x_{1j} - \sum_{\{j|(j,1)\in\mathcal{A}\}} x_{j1} = N-1, \\
&\qquad \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} = -1, \qquad \forall\ i \neq 1, \\
&\qquad 0 \leq x_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}.
\end{aligned}$$

Introduce an artificial arc $(1,i)$ for all $i \neq 1$ with very large cost M, and consider the simplex method starting with the strongly feasible tree of artificial arcs. Let the origin node 1 be the root node.

(a) Show that all the arcs of the trees generated by the simplex method are oriented away from the origin and carry unit flow.

(b) How can a negative length cycle be detected with the simplex method?

(c) Assume that $a_{ij} \geq 0$ for all $(i,j) \in \mathcal{A}$ and suppose that the in-arc is selected to have minimum reduced cost out of all arcs that are not in the tree. Use induction to show that after the kth pivot the tree consists of a shortest path tree from node 1 to the k closest nodes to node 1, together with the artificial arcs $(1,i)$ for all i that are not among the k closest nodes to node 1. Prove then that this implementation of the simplex method is equivalent to Dijkstra's method.

2.3 EXTENSION TO THE PROBLEM WITH UPPER AND LOWER BOUNDS

We now consider the extension of the simplex method of the previous section to the general minimum cost flow problem

$$\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \tag{MCF}$$

subject to

$$\sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} = s_i, \qquad \forall\ i \in \mathcal{N}, \tag{3.1}$$

$$b_{ij} \le x_{ij} \le c_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}. \tag{3.2}$$

To simplify the presentation, we assume that $b_{ij} < c_{ij}$ for all arcs (i,j); any arc (i,j) with $b_{ij} = c_{ij}$ can be eliminated, and its flow, which is equal to the common bound, can be incorporated into the supplies s_i and s_j. A nice aspect of this problem is that we need not worry about unboundedness, since all arc flows are constrained to lie in a bounded interval.

The extension of the simplex method to the problem with upper and lower bounds is straightforward, and we will simply state the algorithm and the corresponding results without much elaboration. In fact, one may derive the simplex method for this problem by converting it to the minimum cost flow problem with nonnegativity constraints (cf. Fig. 1.7 in Section 1.1.3), applying the simplex method of the preceding section, and appropriately streamlining the computations. We leave the verification of this as Exercise 3.2 for the reader.

The method uses at each iteration a spanning tree T. Only arcs of T can have flows that are neither at the upper bound nor at the lower bound. However, to uniquely associate a basic flow vector with T, we must also specify for each arc $(i,j) \notin T$ whether $x_{ij} = b_{ij}$ or $x_{ij} = c_{ij}$. Thus, the simplex method maintains a triplet

$$(T, L, U),$$

where

T is a spanning tree.

L is the set of arcs $(i,j) \notin T$ with $x_{ij} = b_{ij}$.

U is the set of arcs $(i,j) \notin T$ with $x_{ij} = c_{ij}$.

Such a triplet will be called a *basis*. It uniquely specifies a flow vector x, called the *basic flow vector* corresponding to (T, L, U). In particular, if the

arc (i, j) belongs to T and separates T into the subtrees T_i and T_j, we have

$$x_{ij} = \sum_{n \in T_i} s_n - \sum_{\{(m,n) \in L \mid m \in T_i, n \in T_j\}} b_{mn} - \sum_{\{(m,n) \in U \mid m \in T_i, n \in T_j\}} c_{mn} + \sum_{\{(m,n) \in L \mid m \in T_j, n \in T_i\}} b_{mn} + \sum_{\{(m,n) \in U \mid m \in T_j, n \in T_i\}} c_{mn}.$$

If x is feasible, then the basis (T, L, U) is called feasible.

Similar to the previous section, we fix a root node r throughout the algorithm. A basis (T, L, U) specifies a price vector p using the same formula as in the previous section:

$$p_i = p_r - \sum_{(m,n) \in P_i^+} a_{mn} + \sum_{(m,n) \in P_i^-} a_{mn}, \qquad \forall\ i \in \mathcal{N},$$

where P_i is the unique simple path of T starting at the root node r and ending at i, and P_i^+ and P_i^- are the sets of forward and backward arcs of P_i, respectively.

We say that the feasible basis (T, L, U) is *strongly feasible* if all arcs $(i, j) \in T$ with $x_{ij} = b_{ij}$ are oriented away from the root and if all arcs $(i, j) \in T$ with $x_{ij} = c_{ij}$ are oriented toward the root (that is, the unique simple path from the root to i passes through j).

Given the strongly feasible basis (T, L, U) with a corresponding flow vector x and price vector p, an iteration of the simplex method produces another strongly feasible basis $(\overline{T}, \overline{L}, \overline{U})$ as follows.

Typical Simplex Iteration

Find an in-arc $\overline{e} = (\overline{i}, \overline{j}) \notin T$ such that either

$$r_{\overline{ij}} < 0 \qquad \text{if} \qquad \overline{e} \in L$$

or

$$r_{\overline{ij}} > 0 \qquad \text{if} \qquad \overline{e} \in U.$$

(If no such arc can be found, x is primal-optimal and p is dual-optimal.) Let C be the cycle closed by T and $\overline{e}$. Define the forward direction of C to be the same as the one of $\overline{e}$ if $\overline{e} \in L$ and opposite to $\overline{e}$ if $\overline{e} \in U$ (that is, $\overline{e} \in C^+$ if $\overline{e} \in L$ and $\overline{e} \in C^-$ if $\overline{e} \in U$). Also let

$$\delta = \min \left\{ \min_{(i,j) \in C^-} \{x_{ij} - b_{ij}\}, \min_{(i,j) \in C^+} \{c_{ij} - x_{ij}\} \right\},$$

and let $\hat{C}$ be the set of arcs where this minimum is obtained:

$$\hat{C} = \big\{(i, j) \in C^- \mid x_{ij} - b_{ij} = \delta\big\} \cup \big\{(i, j) \in C^+ \mid c_{ij} - x_{ij} = \delta\big\}.$$

Define the *join of* C as the first node of C that lies on the unique simple path of T that starts from the root and ends at $\overline{i}$. Select as out-arc the arc e of $\hat{C}$ that is encountered first as C is traversed in the forward direction starting from the join node. The new tree is $\overline{T} = T + \overline{e} - e$, and the corresponding flow vector $\overline{x}$ is obtained from x by

$$\overline{x}_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \notin C \\ x_{ij} + \delta & \text{if } (i,j) \in C^+ \\ x_{ij} - \delta & \text{if } (i,j) \in C^-. \end{cases}$$

Note that it is possible that the in-arc is the same as the out-arc, in which case T is unchanged. In this case, the flow of this arc will simply move from one bound to the other, affecting the sets L and U, and thus affecting the basis. The proofs of the preceding section can be modified to show that the algorithm maintains a strongly feasible tree.

The following proposition admits a very similar proof of Prop. 2.1.

Proposition 3.1: Assume that the minimum cost flow problem (MCF) is feasible. The simplex method starting from a strongly feasible tree terminates with an optimal primal solution x and an optimal dual solution p. Furthermore, the optimal primal cost is equal to the optimal dual cost. If the supplies s_i and the flow bounds b_{ij}, c_{ij} are all integer, the optimal primal solution x is integer; if the starting price of the root node and the cost coefficients a_{ij} are all integer, the optimal dual solution p is integer.

If an initial strongly feasible tree is not readily available, we can solve instead a big-M version of the problem with suitably large value of M. This problem is

$$\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} + M\left(\sum_{(i,0)\in\overline{\mathcal{A}}} x_{i0} + \sum_{(0,i)\in\overline{\mathcal{A}}} x_{0i}\right)$$

subject to

$$\sum_{\{j|(i,j)\in\mathcal{A}\cup\overline{\mathcal{A}}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\cup\overline{\mathcal{A}}\}} x_{ji} = s_i, \qquad \forall\ i \in \mathcal{N}\cup\{0\},$$

$$b_{ij} \le x_{ij} \le c_{ij}, \qquad \forall\ (i,j) \in \mathcal{A},$$

$$0 \le x_{i0} \le \overline{s}_i, \qquad \forall\ i \text{ with } s_i > 0,$$

$$0 \le x_{0i} \le \underline{s}_i, \qquad \forall\ i \text{ with } s_i \le 0,$$

where

$$\overline{s}_i = s_i - \sum_{\{j|(i,j)\in\mathcal{A}\}} b_{ij} + \sum_{\{j|(j,i)\in\mathcal{A}\}} b_{ji},$$

$$\underline{s}_i = -s_i + \sum_{\{j|(i,j)\in\mathcal{A}\}} b_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} b_{ji}.$$

The initial strongly feasible tree consists of the artificial arcs. The corresponding basic flow vector x is given by $x_{ij} = b_{ij}$ for all $(i,j) \in \mathcal{A}$, $x_{i0} = s_i$, for all i with $s_i > 0$, and $x_{0i} = -s_i$, for all i with $s_i \leq 0$.

Similar to the case of the problem with nonnegativity constraints, we obtain the following.

Proposition 3.2: If the minimum cost flow problem (MCF) is feasible, then it has at least one optimal solution, and its dual problem also has at least one optimal solution. Furthermore, if the supplies s_i and the flow bounds b_{ij}, c_{ij} are all integer, there exists an optimal primal solution which is integer; if the cost coefficients a_{ij} are all integer, there exists an optimal dual solution which is integer.

E X E R C I S E S

Exercise 3.1

Use the simplex method to solve the minimum cost flow problem with the data of Fig. 2.6, and with the arc flow bounds $0 \leq x_{ij} \leq 1$ for all $(i,j) \in \mathcal{A}$.

Exercise 3.2

Suppose that the problem of this section is transformed to a minimum cost flow problem with nonnegativity constraints as in Fig. 1.7 of Section 1.1.3. Show that the simplex method of the previous section, when applied to the latter problem, is equivalent to the simplex method of the present section. In particular, relate feasible trees, basic flow vectors, and price vectors generated by the two methods, and show that they are in one-to-one correspondence.

2.4 IMPLEMENTATION ISSUES

To implement a network optimization algorithm efficiently it is essential to exploit the graph nature of the problem using appropriate data structures. There are two main issues here:

(a) Representing the problem in a way that facilitates the application of the algorithm.

(b) Using additional data structures that are well suited to the operations of the algorithm.

For simplex methods, the appropriate representations of the problem tend to be quite simple. However, additional fairly complex data structures are needed to implement efficiently the various operations related to flow and price computation, and tree manipulation. This is quite contrary to the situation with the methods that will be discussed in the next two chapters, where the appropriate problem representations are quite complex but the additional data structures are simple.

Problem Representation for Simplex Methods

For concreteness, consider the following problem with zero lower flow bounds

$$\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \tag{4.1}$$

subject to

$$\sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} = s_i, \qquad \forall\ i \in \mathcal{N},$$

$$0 \le x_{ij} \le c_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}.$$

This has become the standard form for commonly available minimum cost flow codes. As was mentioned in Section 1.1.3, a problem with nonzero lower arc flow bounds b_{ij} can be converted to one with nonnegativity constraints by using a flow translation (replacing x_{ij} by $x_{ij} - b_{ij}$ and appropriately adjusting c_{ij}, s_i, and s_j).

One way to represent this problem, which is the most common in simplex codes, is to use four arrays of length A and one array of length N:

$START(a)$: The start node of arc a.

$END(a)$: The end node of arc a.

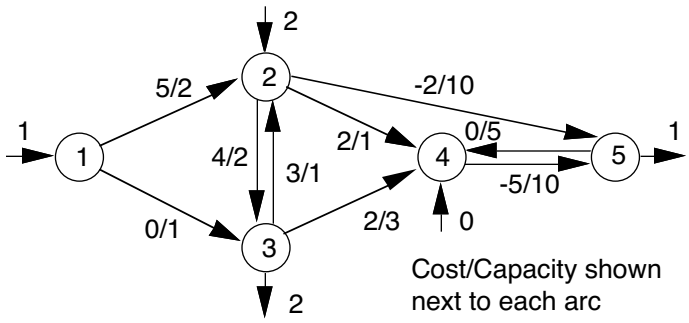
$COST(a)$: The cost coefficient of arc a.

$CAPACITY(a)$: The upper flow bound of arc a.

$SUPPLY(i)$: The supply of node i.

Figure 4.1 gives an example of a problem represented in this way.

An alternative representation is to store the costs a_{ij} and the upper flow bounds c_{ij} in two-dimensional $N \times N$ arrays (or in one-dimensional arrays of length N^2, with the elements of each row stored contiguously). This wastes memory and requires a lot of extra overhead when the problem is sparse ($A << N^2$), but it may be a good choice for dense problems since it avoids the storage of the start and end nodes of each arc.



ARC	START	END	COST	CAPACITY
1	1	2	5	2
2	1	3	0	1
3	2	3	4	2
4	3	2	3	1
5	2	5	-2	10
6	2	4	2	1
7	3	4	2	3
8	5	4	0	5
9	4	5	-5	10

NODE	SUPPLY
1	1
2	2
3	-2
4	0
5	-1

Figure 4.1 Representation of a minimum cost flow problem in terms of the five arrays *START*, *END*, *COST*, *CAPACITY*, and *SUPPLY*.

Data Structures for Tree Operations

Taking a closer look at the operations of the simplex method, we see that the main computational steps at each iteration are the following:

(a) Finding an in-arc with negative reduced cost.

(b) Identifying the cycle formed by the current tree and the in-arc.

(c) Modifying the flows along the cycle and obtaining the out-arc.

(d) Recalculating the node prices.

As mentioned in Section 2.1.1, most codes maintain a candidate list, that is, a subset of arcs with negative reduced cost; the arc with most negative reduced cost from this list is selected as the in-arc at each iteration. The maximum size of the candidate list is set at some reasonable level (chosen heuristically), thereby avoiding a costly search and comparison of the reduced costs of all the arcs.

To identify the cycle and the associated flow increment at each iteration, simplex codes commonly use the following two arrays of length N:

(a) $PRED(i)$: The arc preceding node i on the unique path from the root to i on the current tree, together with an indication (such as a plus or a minus sign) of whether this is an incoming or outgoing arc of i.

(b) $DEPTH(i)$: The number of arcs of the unique path from the root to i on the current tree.

The *PRED* array (together with the *START* and *END* arrays) is sufficient both to represent the current tree and to construct the unique path on the tree from any node i to any other node j. (Construct the paths from i to the root and from j to the root, and subtract out the common portion of these paths.) In particular, if (i, j) is the in-arc, the cycle formed by (i, j) and the current tree could be obtained by finding the path joining i and j in this way. By using the *DEPTH* array, however, the cycle can be constructed more quickly without having to go from i to j all the way to the root. In particular, one can start constructing the paths from i and j to the root simultaneously, adding a new node to the path whose current end node has greater *DEPTH* (ties are broken arbitrarily). The join of the cycle can then be identified as the first encountered common node in the two paths. The following procedure starting with the in-arc (i, j) accomplishes this. In this procedure, $\overline{i}$ and $\overline{j}$ represent successive nodes of the paths starting at i and j, respectively, and ending at the join of the cycle.

Identifying the Join of the Cycle Corresponding to the In-Arc (i, j)

Set $\overline{i} = i$, $\overline{j} = j$.

Step 1: If $DEPTH(\overline{i}) \geq DEPTH(\overline{j})$, go to Step 2; else go to Step 3.

Step 2: Set $\overline{i} := START(PRED(\overline{i}))$ if $PRED(\overline{i})$ is an incoming arc to $\overline{i}$, and set $\overline{i} := END(PRED(\overline{i}))$ if $PRED(\overline{i})$ is an outgoing arc from $\overline{i}$. Go to Step 4.

Step 3: Set $\overline{j} := START(PRED(\overline{j}))$ if $PRED(\overline{j})$ is an incoming arc to $\overline{j}$, and set $\overline{i} := END(PRED(\overline{j}))$ if $PRED(\overline{j})$ is an outgoing arc from $\overline{j}$. Go to Step 4.

Step 4: If $\overline{i} = \overline{j}$, terminate; $\overline{i}$ is the join of the cycle corresponding to the in-arc (i, j). Else go to Step 1.

The cycle corresponding to the in-arc consists of the arcs $PRED(\overline{i})$ and $PRED(\overline{j})$ encountered during this procedure. With a simple modification of the procedure, we can simultaneously obtain the out-arc and calculate the flow increment. With little additional work, we can also change the flow along the cycle and update the *PRED* and *DEPTH* arrays consistently with the new tree.

We must still provide for a mechanism to calculate efficiently the prices corresponding to a given tree. This can be done iteratively, using the prices of the preceding tree as shown in Section 1.1; cf. Eqs. (1.11) and (1.12). To apply these equations, it is necessary to change the prices of the descendants of the endnode of the out-arc that has the larger value of *DEPTH*; cf. Fig. 4.2. Thus, it is sufficient to be able to calculate the descendants of a given node i in the current tree (the nodes whose unique path to the root passes through i). For this it is convenient to use one more array, called *THREAD*. It defines a traversal order of the nodes of the tree in depth-first fashion. To understand this order, it is useful to think of the tree laid out in a plane, and to consider visiting all nodes starting from the root, and going "top to bottom" and "left to right". An example is given in Fig. 4.3. It can be seen that every node i appears in the traversal order immediately before all of its descendants. Hence the descendants of i are all the nodes immediately following node i in the traversal order up to the first node j with $DEPTH(j) \leq DEPTH(i)$. The array *THREAD* encodes the traversal order by storing in $THREAD(i)$ the node following node i; cf. Fig. 4.3. An important fact is that when the tree changes, the *THREAD* array can be updated quite efficiently [with $O(N)$ operations]. The details, however, are too tedious and complicated to be included here; for a clear presentation, see [Chv83], p. 314.

2.5 NOTES AND SOURCES

2.1. The first specialized version of the simplex method for the transportation problem problem was given in [Dan51]. This method was also described and extended to the minimum cost flow problem in [Dan63]. A general primal cost improvement algorithm involving flow changes along negative cost

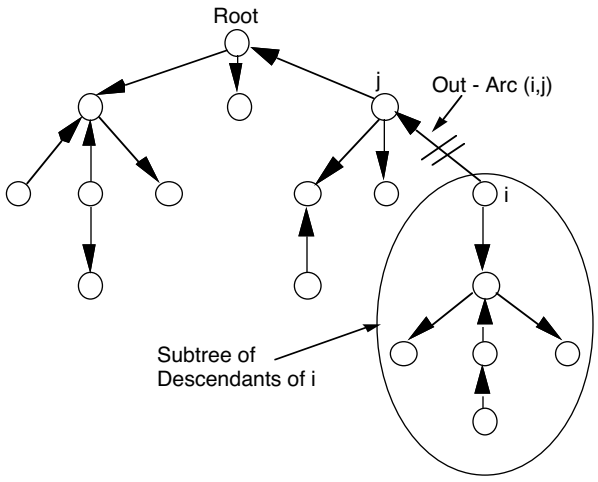
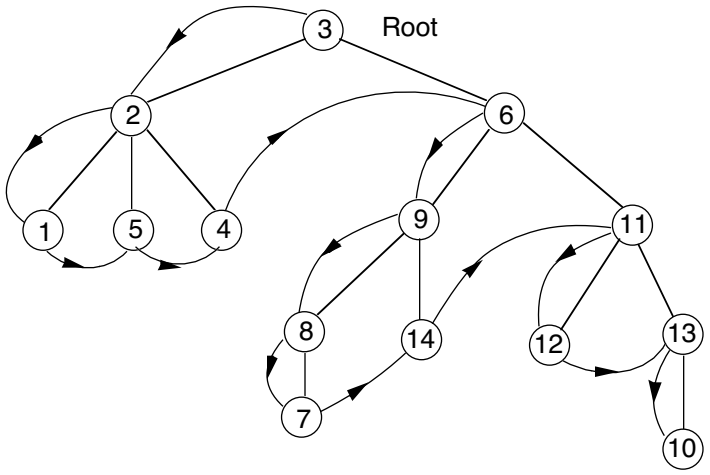


Figure 4.2 The two subtrees obtained when the out-arc is deleted from the current tree. The subtree containing the endnode of the out-arc with larger *DEPTH* (node i in the example of the figure) consists of all the descendants of that endnode.

cycles was given in [Kle67]. Strongly feasible trees and their use in resolving degeneracy were introduced in [Cun76].

The subject of pivot selection has received considerable attention in the literature. Examples of poor performance of the simplex method are given in [Zad73a] and [Zad73b]. The performance of various pivot rules was studied empirically in [GSS77], [GoR77], [BBG77], [BGK77], [BGK78], [Mul78a], [Mul78b], and [GGK83]. Generally, even with the use of strongly feasible trees, it is possible that the number of successive degenerate pivots is not polynomial. Pivot rules with guaranteed polynomial upper bounds on the lengths of sequences of degenerate pivots are given in [Cun79] and [GHK87]. One of the simplest such rules maintains a strongly feasible tree and operates as follows: if the in-arc at some iteration has start node i, the in-arc at the next iteration must be the outgoing arc from node $(i+k)$ modulo N that has minimum reduced cost, where k is the smallest nonnegative integer such that node $(i+k)$ modulo N has at least one outgoing arc with negative reduced cost. For a textbook discussion of a variety of pivot rules under which the simplex method has polynomial running time, see [BJS90].

2.3. Specialized simplex methods have been developed for the assignment problem; see [BGK77], [Hun83], [Akg86], [Bal85], [Gol85a], [Bal86]. For analysis and application of simplex methods in shortest path and max-flow problems, see [FuD55], [FNP81], [GKM84], [GHK86], and [GoH88].



Traversal Order: 3, 2, 1, 5, 4, 6, 9, 8, 7, 14, 11, 12, 13, 10

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$THREAD(i)$	5	1	2	6	4	9	14	7	8	0	12	13	10	11

Figure 4.3 Illustration of the *THREAD* array, which defines a depth-first traversal order of the nodes in the tree. Given the set S of already traversed nodes, the next node traversed is an immediate descendant of one of the nodes in S, which has maximum value of *DEPTH*. For each node i, $THREAD(i)$ defines the successor of node i in this order (for the last node, *THREAD* is equal to 0).

2.4. The development of good implementation techniques played a crucial role in the efficient use of the simplex method. Important contributions in this area include [Joh66], [SrT73], [GKK74a], [GKK74b], [BBG77], and [BGK79]. Textbook presentations of these techniques that supplement ours are given in [KeH80], [Chv83], and [BJS90].