3

Dual Ascent Methods

3.1 DUAL ASCENT

In this chapter we focus on the minimum cost flow problem

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

subject to

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

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

Throughout the chapter we will assume that the scalars a_{ij}, b_{ij}, c_{ij}, and s_i are all integer. Usually, this is not an important practical restriction. However, there are extensions of the algorithms of this chapter that handle noninteger problem data, as will be discussed later.

The main idea of dual cost improvement (or dual ascent) algorithms is to start with a price vector and successively obtain new price vectors with improved dual cost value, with the aim of solving the dual problem. Recall from Section 1.2.2 that this problem is

$$\begin{aligned} &\text{maximize } q(p) \\ &\text{subject to no constraint on } p, \end{aligned} \tag{1.3}$$

where the dual functional q is given by

$$q(p) = \sum_{(i,j)\in\mathcal{A}} q_{ij}(p_i - p_j) + \sum_{i\in\mathcal{N}} s_i p_i, \tag{1.4}$$

with

$$\begin{aligned} q_{ij}(p_i - p_j) &= \min_{b_{ij}\le x_{ij}\le c_{ij}} \{(a_{ij} + p_j - p_i)x_{ij}\} \\ &= \begin{cases} (a_{ij} + p_j - p_i)b_{ij} & \text{if } p_i \le a_{ij} + p_j, \\ (a_{ij} + p_j - p_i)c_{ij} & \text{if } p_i > a_{ij} + p_j. \end{cases} \end{aligned} \tag{1.5}$$

It is helpful here to introduce some terminology. For any price vector p, we say that an arc (i, j) is

$$\text{inactive if} \quad p_i < a_{ij} + p_j,$$

$$\text{balanced if} \quad p_i = a_{ij} + p_j,$$

$$\text{active if} \quad p_i > a_{ij} + p_j.$$

The *complementary slackness* (CS) conditions for a flow–price vector pair (x, p), introduced in Section 1.2.2, can be restated as follows:

$$x_{ij} = b_{ij}, \qquad \text{for all inactive arcs } (i,j), \tag{1.6}$$

$$b_{ij} \le x_{ij} \le c_{ij}, \qquad \text{for all balanced arcs } (i,j), \tag{1.7}$$

$$x_{ij} = c_{ij}, \qquad \text{for all active arcs } (i,j), \tag{1.8}$$

(see Fig. 1.1).

We restate for convenience the following basic duality result, proved in Section 1.2.2.

Proposition 1.1: If a feasible flow vector x^* and a price vector p^* satisfy the complementary slackness conditions (1.6)–(1.8), then x^* is an optimal solution of the minimum cost flow problem and p^* is an optimal solution of the dual problem (1.3).

The major dual ascent algorithms select at each iteration a connected subset of nodes $\mathcal{S}$, and change the prices of these nodes by equal amounts while leaving the prices of all other nodes unchanged. In other words, each iteration involves a price vector change along a direction of the form $d_{\mathcal{S}} = (d_1, \ldots, d_N)$, where

$$d_i = \begin{cases} 1 & \text{if } i \in \mathcal{S} \\ 0 & \text{if } i \notin \mathcal{S} \end{cases} \tag{1.9}$$

and $\mathcal{S}$ is a connected subset of nodes. Such directions will be called *elementary*.

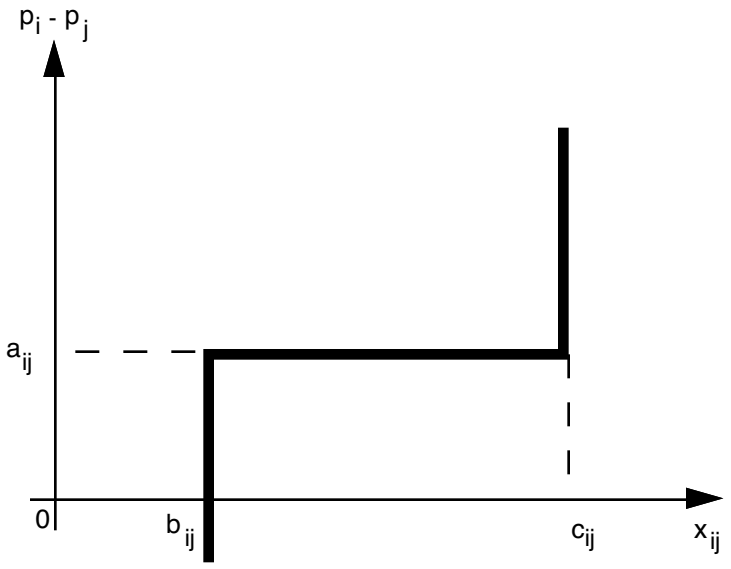


Figure 1.1 Illustration of the complementary slackness conditions. For each arc (i,j), the pair $(x_{ij}, p_i - p_j)$ should lie on the graph shown.

To check whether $d_{\mathcal{S}}$ is a direction of dual ascent, we need to calculate the corresponding directional derivative of the dual cost along $d_{\mathcal{S}}$ and check whether it is positive. From the dual cost expression (1.4)-(1.5), it is seen that this directional derivative is

$$\begin{aligned} q'(p;d_{\mathcal{S}}) &= \lim_{\alpha \downarrow 0} \frac{q(p+\alpha d_{\mathcal{S}}) - q(p)}{\alpha} \\ &= \sum_{(j,i)\,:\,\text{active},\ j\notin\mathcal{S},\, i\in\mathcal{S}} c_{ji} + \sum_{(j,i)\,:\,\text{inactive or balanced},\ j\notin\mathcal{S},\, i\in\mathcal{S}} b_{ji} \\ &\quad - \sum_{(i,j)\,:\,\text{active or balanced},\ i\in\mathcal{S},\, j\notin\mathcal{S}} c_{ij} - \sum_{(i,j)\,:\,\text{inactive},\ i\in\mathcal{S},\, j\notin\mathcal{S}} b_{ij} \\ &\quad + \sum_{i\in\mathcal{S}} s_i. \end{aligned} \tag{1.10}$$

In words, the directional derivative $q'(p;d_{\mathcal{S}})$ is the difference between inflow and outflow across the node set $\mathcal{S}$ when the flows of the inactive and active arcs are set at their lower and upper bounds, respectively, and the flow of each balanced arc incident to $\mathcal{S}$ is set to its lower or upper bound depending on whether the arc is incoming to $\mathcal{S}$ or outgoing from $\mathcal{S}$.

To obtain a suitable set $\mathcal{S}$, with positive directional derivative $q'\big(p,d_{\mathcal{S}}\big)$, it is convenient to maintain a flow vector x satisfying CS together with p. This

helps to organize the search for an ascent direction and to detect optimality, as will now be explained.

For a flow vector x, let us define the *surplus* g_i of node i as the difference between total inflow into i minus the total outflow from i, that is,

$$g_i = \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} - \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} + s_i. \tag{1.11}$$

We have

$$\sum_{i\in\mathcal{S}} g_i = \sum_{\{(j,i)\in\mathcal{A}|j\notin\mathcal{S},\, i\in\mathcal{S}\}} x_{ji} - \sum_{\{(i,j)\in\mathcal{A}|i\in\mathcal{S},\, j\notin\mathcal{S}\}} x_{ij} + \sum_{i\in\mathcal{S}} s_i, \tag{1.12}$$

and if x satisfies CS together with p, we obtain using Eqs. (1.10) and (1.12)

$$\begin{aligned}\sum_{i\in\mathcal{S}} g_i &= q'(p;d_{\mathcal{S}}) + \sum_{(j,i)\,:\ \text{balanced},\ j\notin\mathcal{S},\, i\in\mathcal{S}} (x_{ji}-b_{ji}) \\ &\qquad + \sum_{(i,j):\ \text{balanced},\ i\in\mathcal{S},\, j\notin\mathcal{S}} (c_{ij}-x_{ij}) \\ &\geq q'(p;d_{\mathcal{S}}).\end{aligned} \tag{1.13}$$

We see, therefore, that only a node set $\mathcal{S}$ that has positive total surplus is a candidate for generating a direction $d_{\mathcal{S}}$ of dual ascent. In particular, if there is no balanced arc (i,j) with $i\in\mathcal{S}$, $j\notin\mathcal{S}$, and $x_{ij}<c_{ij}$, and no balanced arc (j,i) with $j\notin\mathcal{S}$, $i\in\mathcal{S}$, and $b_{ij}<x_{ij}$, then

$$\sum_{i\in\mathcal{S}} g_i = q'(p;d_{\mathcal{S}}), \tag{1.14}$$

so if $\mathcal{S}$ has positive total surplus then $d_{\mathcal{S}}$ is an ascent direction. The following lemma expresses this idea and provides the basis for the subsequent algorithms.

Lemma 1.1: Suppose that x and p satisfy the CS conditions, and let $\mathcal{S}$ be a subset of nodes. Let $d_{\mathcal{S}}=(d_1,d_2,\ldots,d_N)$ be the vector with $d_i=1$ if $i\in\mathcal{S}$ and $d_i=0$ otherwise, and assume that

$$\sum_{i\in\mathcal{S}} g_i > 0.$$

Then either $d_{\mathcal{S}}$ is a dual ascent direction, that is,

$$q'(p;d_{\mathcal{S}})>0,$$

or else there exist nodes $i\in\mathcal{S}$ and $j\notin\mathcal{S}$ such that either (i,j) is a balanced arc with $x_{ij}<c_{ij}$ or (j,i) is a balanced arc with $b_{ji}<x_{ji}$.

Proof: Follows from Eq. (1.13). **Q.E.D.**

Overview of Dual Ascent Algorithms

The algorithms of this chapter start with an integer flow–price vector pair (x, p), satisfying CS, and operate in iterations. At the beginning of each iteration, we have a subset of nodes $\mathcal{S}$ such that

$$\sum_{i \in \mathcal{S}} g_i > 0;$$

initially $\mathcal{S}$ consists of one or more nodes with positive surplus. According to the preceding lemma, there are two possibilities:

(a) $\mathcal{S}$ defines a dual ascent direction $d_{\mathcal{S}} = (d_1, d_2, \ldots, d_N)$, where $d_i = 1$ if $i \in \mathcal{S}$, and $d_i = 0$ otherwise.

(b) $\mathcal{S}$ can be enlarged by adding a node $j \notin \mathcal{S}$ with the property described in Lemma 1.1, that is, for some $i \in \mathcal{S}$, either (i, j) is a balanced arc with $x_{ij} < c_{ij}$, or (j, i) is a balanced arc with $b_{ji} < x_{ji}$.

In case (b), there are two possibilities:

(1) $g_j \geq 0$, in which case,

$$\sum_{i \in \mathcal{S} \cup \{j\}} g_i > 0,$$

and the process can be continued with

$$\mathcal{S} \cup \{j\}$$

replacing $\mathcal{S}$.

(2) $g_j < 0$, in which case, it can be seen that there is a path originating at some node i of the starting set $\mathcal{S}$ and ending at node j that is *unblocked*, that is, all its arcs have room for a flow increase in the direction from i to j (see Fig. 1.2). Such a path is called an *augmenting path* (generalizing slightly the notion of an augmenting path used in the Ford-Fulkerson algorithm for the max-flow problem). By increasing the flow of the forward arcs (direction from i to j) of the path and by decreasing the flow of the backward arcs (direction from j to i) of the path, we can bring both surpluses g_i and g_j closer to zero by an integer amount while leaving the surplus of all other nodes unaffected and maintaining CS.

Since the total absolute surplus $\sum_{i \in \mathcal{N}} |g_i|$ cannot be indefinitely reduced by integer amounts, it is seen that starting from an integer flow–price vector pair satisfying CS, after at most a finite number of iterations in which flow augmentations occur without finding an ascent direction, one of three things will happen:

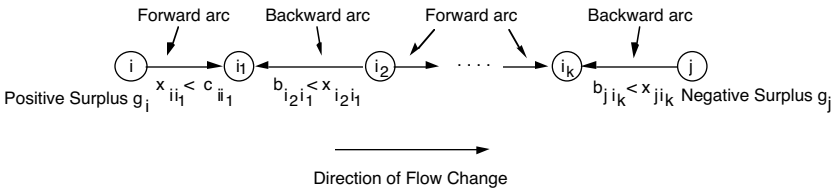


Figure 1.2 Illustration of an augmenting path. The initial node i and the final node j have positive and negative surplus, respectively. Furthermore, the path is unblocked, that is, each arc on the path has room for flow change in the direction from i to j. A flow change of magnitude $\delta > 0$ in this direction reduces the total absolute surplus $\sum_{m \in \mathcal{N}} |g_m|$ by 2δ provided $\delta \leq \min\{g_i, -g_j\}$.

(a) A dual ascent direction will be found; this direction can be used to improve the dual cost by an integer amount.

(b) $g_i = 0$ for all i; in this case the flow vector x is feasible, and since it satisfies CS together with p, by Prop. 1.1, x is primal-optimal and p is dual-optimal.

(c) $g_i \leq 0$ for all i but $g_i < 0$ for at least one i; since by adding Eq. (1.12) over all $i \in \mathcal{N}$ we have $\sum_{i \in \mathcal{N}} s_i = \sum_{i \in \mathcal{N}} g_i$ it follows that $\sum_{i \in \mathcal{N}} s_i < 0$, so the problem is infeasible.

Thus, for a feasible problem, the procedure just outlined can be used to find a dual ascent direction and improve the dual cost starting at any nonoptimal price vector. Figure 1.3 provides an illustration for a very simple problem.

In the next two sections, we discuss two different dual ascent methods. The first, known as *primal-dual*, in its classical form, tries at each iteration to use the *steepest ascent* direction, that is, the elementary direction with maximal directional derivative. This method can also be implemented by means of a shortest path computation. The second method, called *relaxation*, is usually faster in practice. It tries to use directions that are not necessarily steepest, but can be computed more quickly than the steepest ascent direction.

3.2 PRIMAL-DUAL (SEQUENTIAL SHORTEST PATH) METHODS

The primal-dual algorithm starts with any integer pair (x, p) satisfying CS. One possibility is to choose the integer vector p arbitrarily and to set $x_{ij} = b_{ij}$ if (i, j) is inactive or balanced, and $x_{ij} = c_{ij}$ otherwise. (Prior knowledge could be built into the initial choice of x and p using, for example, the results of an earlier optimization.) The algorithm preserves the integrality and CS property of the pair (x, p) throughout.

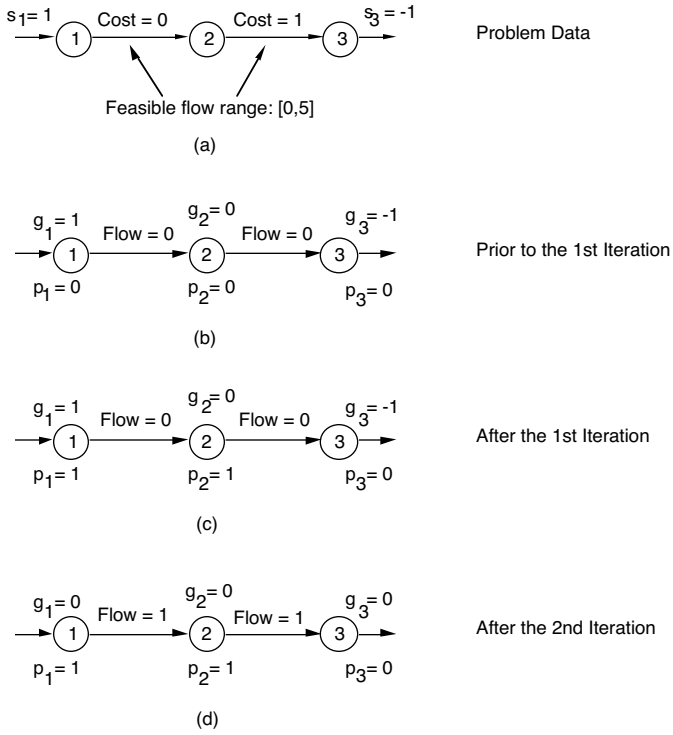


Figure 1.3 Illustration of a dual ascent method for the simple problem described in (a). Initially, $x = (0, 0)$ and $p = (0, 0, 0)$ as shown in (b).

The first iteration starts with $\mathcal{S} = \{1\}$. It can be seen using Eq. (1.13), that the directional derivative $q'(p; d_{\mathcal{S}})$ is -4, so $d_{\mathcal{S}} = (1, 0, 0)$ is not a direction of ascent. We thus enlarge $\mathcal{S}$ by adding node 2 using the balanced arc $(1, 2)$. Since there is no incident balanced arc to $\mathcal{S} = \{1, 2\}$, the direction $d_{\mathcal{S}} = (1, 1, 0)$ is a direction of ascent [using Eq. (1.13), $q'(p; d_{\mathcal{S}}) = 1$]. We thus increase the prices of the nodes in $\mathcal{S}$ by a common increment γ, and we choose $\gamma = 1$ because this is the increment that maximizes the dual function along the direction $d_{\mathcal{S}}$ starting from p; this can be seen by checking the directional derivative of q at the price vector $(\gamma, \gamma, 0)$ along the direction $d_{\mathcal{S}}$ and finding that it switches from positive $(= 1)$ to negative $(= -4)$ at $\gamma = 1$ where the arc $(2, 3)$ becomes balanced.

The second iteration starts again with $\mathcal{S} = \{1\}$. As in the first iteration, $\mathcal{S}$ is enlarged to $\mathcal{S} = \{1, 2\}$. Since the corresponding direction $d_{\mathcal{S}} = (1, 1, 0)$ is not a direction of ascent $[q'(p; d_{\mathcal{S}}) = -4]$, we explore the balanced incident arc $(2, 3)$ and we discover the negative surplus node 3. The augmenting path $(1, 2, 3)$ has now been obtained, and the corresponding augmentation sets the flows of the arcs $(1, 2)$ and $(2, 3)$ to 1. Since now all node surpluses become zero, the algorithm terminates; $x = (1, 1)$ is an optimal primal solution and $p = (1, 1, 0)$ is an optimal dual solution.

At the start of the typical iteration, we have an integer pair (x, p) satisfying CS. The iteration indicates that the primal problem is infeasible, or else indicates that (x, p) is optimal, or else transforms this pair into another pair satisfying CS. In particular, if $g_i \leq 0$ for all i, then in view of the fact $\sum_{i\in\mathcal{N}} g_i = \sum_{i\in\mathcal{N}} s_i$ [see Eq. (1.12) with $\mathcal{S} = \mathcal{N}$], there are two possibilities: (1) $g_i < 0$ for some i, in which case $\sum_{i\in\mathcal{N}} s_i < 0$ and the problem is infeasible, or (2) $g_i = 0$ for all i, in which case x is feasible and therefore also optimal, since it satisfies CS together with p. In either case, the algorithm terminates.

If on the other hand we have $g_i > 0$ for at least one node i, the iteration starts by selecting a nonempty subset I of nodes i with $g_i > 0$. The iteration maintains two sets of nodes $\mathcal{S}$ and $\mathcal{L}$, with $\mathcal{S} \subset \mathcal{L}$. Initially, $\mathcal{S}$ is empty and $\mathcal{L}$ consists of the subset I. We use the following terminology.

$\mathcal{S}$: Set of *scanned* nodes (these are the nodes whose incident arcs have been "examined" during the iteration).

$\mathcal{L}$: Set of *labeled* nodes (these are the nodes that have either been scanned during the iteration or are current candidates for scanning).

In the course of the iteration we continue to add nodes to $\mathcal{L}$ and $\mathcal{S}$ until either an augmenting path is found or $\mathcal{L} = \mathcal{S}$, in which case $d_{\mathcal{S}}$ will be shown to be an ascent direction. The iteration also maintains a *label* for every node $i \in \mathcal{L} - I$, which is an incident arc of i. The labels are useful for constructing augmenting paths (see Step 3 of the following iteration).

Typical Primal-Dual Iteration

Step 0 (Initialization): Select a set I of nodes i with $g_i > 0$. [If no such node can be found, terminate; the pair (x, p) is optimal if $g_i = 0$ for all i; otherwise the problem is infeasible.] Set $\mathcal{L} := I$ and $\mathcal{S} :=$ empty, and go to Step 1.

Step 1 (Choose a Node to Scan): If $\mathcal{S} = \mathcal{L}$, go to Step 4; else select a node $i \in \mathcal{L} - \mathcal{S}$, set $\mathcal{S} := \mathcal{S} \cup \{i\}$, and go to Step 2.

Step 2 (Label Neighbor Nodes of i): Add to $\mathcal{L}$ all nodes $j \notin \mathcal{L}$ such that either (j, i) is balanced and $b_{ji} < x_{ji}$ or (i, j) is balanced and $x_{ij} < c_{ij}$; also for every such j, give to j the label "(j, i)" if (j, i) is balanced and $b_{ji} < x_{ji}$, and otherwise give to j the label "(i, j)." If for all the nodes j just added to $\mathcal{L}$ we have $g_j \geq 0$, go to Step 1. Else select one of these nodes j with $g_j < 0$ and go to Step 3.

Step 3 (Flow Augmentation): An augmenting path P has been found that begins at a node i belonging to the initial set I and ends at the node j identified in Step 2. The path is constructed by tracing labels backward starting from j, and is such that we have

$$x_{mn} < c_{mn}, \qquad \forall\ (m, n) \in P^+$$

$$x_{mn} > b_{mn}, \qquad \forall\ (m,n) \in P^-$$

where P^+ and P^- are the sets of forward and backward arcs of P, respectively. Let

$$\delta = \min\big\{g_i, -g_j, \big\{c_{mn} - x_{mn} \mid (m,n) \in P^+\big\}, \big\{x_{mn} - b_{mn} \mid (m,n) \in P^-\big\}\big\}.$$

Increase by δ the flows of all arcs in P^+, decrease by δ the flows of all arcs in P^-, and go to the next iteration.

Step 4 (Price Change): Let

$$\begin{aligned}\gamma = \min\big\{&\{p_j + a_{ij} - p_i \mid (i,j) \in \mathcal{A}, x_{ij} < c_{ij}, i \in \mathcal{S}, j \notin \mathcal{S}\},\\ &\{p_j - a_{ji} - p_i \mid (j,i) \in \mathcal{A}, b_{ji} < x_{ji}, i \in \mathcal{S}, j \notin \mathcal{S}\}\big\}.\end{aligned} \tag{2.1}$$

Set

$$p_i := \begin{cases} p_i + \gamma, & \text{if } i \in \mathcal{S}\\ p_i, & \text{otherwise.}\end{cases}$$

Add to $\mathcal{L}$ all nodes j for which the minimum in Eq. (2.1) is attained by an arc (i,j) or an arc (j,i); also for every such j, give to j the label "(i,j)" if the minimum in Eq. (2.1) is attained by an arc (i,j), and otherwise give to j the label "(j,i)." If for all the nodes j just added to $\mathcal{L}$ we have $g_j \geq 0$, go to Step 1. Else select one of these nodes j with $g_j < 0$ and go to Step 3. [*Note:* If there is no arc (i,j) with $x_{ij} < c_{ij}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$, or arc (j,i) with $b_{ji} < x_{ji}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$, the problem is infeasible and the algorithm terminates; see Prop. 2.1 that follows.]

Note the following regarding the primal-dual iteration:

(a) All operations of the iteration preserve the integrality of the flow–price vector pair.

(b) The iteration maintains CS of the flow–price vector pair. To see this, note that arcs with both ends in $\mathcal{S}$, which are balanced just before a price change, continue to be balanced after a price change. This means that a flow augmentation step, even if it occurs following several executions of Step 4, changes only flows of balanced arcs, so it cannot destroy CS. Also, a price change in Step 4 maintains CS because no arc flow is modified in this step and the price increment γ of Eq. (2.1) is such that no arc changes status from active to inactive or vice versa.

(c) At all times we have $\mathcal{S} \subset \mathcal{L}$. Furthermore, when Step 4 is entered, we have $\mathcal{S} = \mathcal{L}$ and $\mathcal{L}$ contains no node with negative surplus. Therefore, based on the logic of Step 2, there is no balanced arc (i,j) with $x_{ij} < c_{ij}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$, and no balanced arc (j,i) with $b_{ji} < x_{ji}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$. It follows from the discussion preceding Lemma 1.1 [cf. Eq. (1.14)] that $d_{\mathcal{S}}$ is an ascent direction.

(d) Only a finite number of price changes occur at each iteration, so each iteration executes to completion, either terminating with a flow augmentation in Step 3, or with an indication of infeasibility in Step 4. To see this, note that between two price changes, the set $\mathcal{L}$ is enlarged by at least one node, so there can be no more than N price changes per iteration.

(e) Only a finite number of flow augmentation steps are executed by the algorithm, since each of these reduces the total absolute surplus $\sum_{i\in\mathcal{N}}|g_i|$ by an integer amount [by (a) above], while price changes do not affect the total absolute surplus.

(f) The algorithm terminates. The reason is that each iteration will execute to completion [by (d) above], and will involve exactly one augmentation, while there will be only a finite number of augmentations [cf. (e) above].

The following proposition establishes the validity of the method.

Proposition 2.1: Consider the minimum cost flow problem and assume that a_{ij}, b_{ij}, c_{ij}, and s_i are all integer.

(a) If the problem is feasible, then the primal-dual method terminates with an integer optimal flow vector x and an integer optimal price vector p.

(b) If the problem is infeasible, then the primal-dual method terminates either because $g_i \leq 0$ for all i and $g_i < 0$ for at least one i or because there is no arc (i,j) with $x_{ij} < c_{ij}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$, or arc (j,i) with $b_{ji} < x_{ji}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$ in Step 4.

Proof: The algorithm terminates as argued earlier, and there are three possibilities:

(1) The algorithm terminates because all nodes have zero surplus. In this case the flow–price vector pair obtained upon termination is feasible and satisfies CS, so it is optimal.

(2) The algorithm terminates because $g_i \leq 0$ for all i and $g_i < 0$ for at least one i. In this case the problem is infeasible, since for a feasible problem we must have $\sum_{i\in\mathcal{N}} g_i = 0$.

(3) The algorithm terminates because there is no arc (i,j) with $x_{ij} < c_{ij}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$, or arc (j,i) with $b_{ji} < x_{ji}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$ in Step 4. Then the flux across the cut $Q = [\mathcal{S}, \mathcal{N} - \mathcal{S}]$ is equal to the capacity $C(Q)$ and is also equal to the sum of the divergences of the nodes of $\mathcal{S}$, which is $\sum_{i\in\mathcal{S}}(s_i - g_i)$ [cf. Eq. (1.11)]. Since $g_i \geq 0$ for all $i \in \mathcal{S}$, $g_i > 0$ for the nodes $i \in I$, and $I \subset \mathcal{S}$, we see that

$$C(Q) < \sum_{i\in\mathcal{S}} s_i.$$

This implies that the problem is infeasible, since for any feasible flow vector we must have

$$\sum_{i \in \mathcal{S}} s_i = F(Q) \leq C(Q),$$

where $F(Q)$ is the corresponding flux across Q. [Another way to show that the problem is infeasible in this case is to observe that $d_{\mathcal{S}}$ is a dual ascent direction, and if no arc (i, j) with the property stated exists, the rate of increase of the dual function remains unchanged as we move indefinitely along $d_{\mathcal{S}}$ starting from p. This implies that the dual optimal value is infinite or equivalently (by Prop. 3.2 in Section 2.3) that the primal problem is infeasible.]

Since termination can occur only under the above circumstances, the desired conclusion follows. **Q.E.D.**

There are a number of variations of the primal-dual method, using different choices of the initial set I of positive surplus nodes. The two most common possibilities are:

(1) I consists of a *single* node i with $g_i > 0$.

(2) I consists of *all* nodes i with $g_i > 0$.

The primal-dual method was originally proposed with the latter choice. In this case, whenever there is a price change, the set $\mathcal{S}$ contains all nodes with positive surplus, and from the directional derivative formulas (1.13) and (1.14), it follows that the ascent direction used in Step 4 has the *maximum* possible directional derivative among elementary directions. This leads to the interpretation of the primal-dual method as a steepest ascent method.

Figure 2.1 traces the steps of the primal-dual method for a simple example.

The Shortest Path Implementation

We will now provide an alternative implementation of the primal-dual method in terms of a shortest path computation. This is known as the *sequential shortest path method*; it will be seen to be mathematically equivalent with the primal-dual method given earlier in the sense that it produces the same sequence of flow–price vector pairs.

Given a pair (x, p) satisfying CS, define the *reduced cost* of an arc (i, j) by

$$r_{ij} = a_{ij} + p_j - p_i. \tag{2.2}$$

Recall that an unblocked path P with respect to x is a path such that $x_{ij} < c_{ij}$ for all forward arcs $(i, j) \in P^+$ and $b_{ij} < x_{ij}$ for all backward arcs $(i, j) \in P^-$.

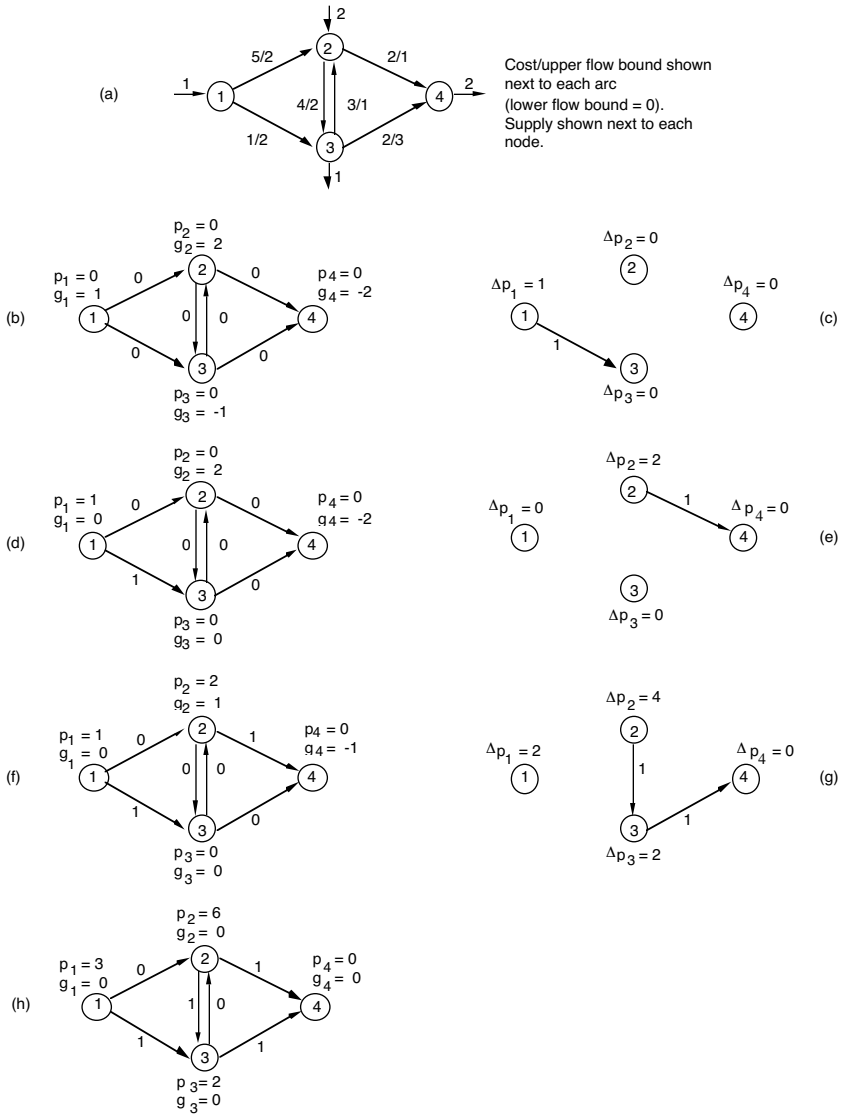


Figure 2.1 Example illustrating the primal-dual method, starting with zero prices.
(a) Problem data.
(b) Initial flows, prices, and surpluses.
(c) Augmenting path and price changes Δp_i of first iteration ($I = \{1\}$).
(d) Flows, prices, and surpluses after the first iteration.
(e) Augmenting path and price changes Δp_i of second iteration ($I = \{2\}$).
(f) Flows, prices, and surpluses after the second iteration.
(g) Augmenting path and price changes Δp_i of third iteration ($I = \{2\}$). There are two price changes here: first p_2 increases by 2, and then p_1, p_2, and p_3 increase by 2.
(h) Flows, prices, and surpluses after the third iteration. The algorithm terminates with an optimal flow–price pair, since all node surpluses are zero.

Furthermore, P is an augmenting path if its start and end nodes have positive and negative surplus, respectively. We define the *length* of an unblocked path P by

$$L_P = \sum_{(i,j)\in P^+} r_{ij} - \sum_{(i,j)\in P^-} r_{ij}. \tag{2.3}$$

Note that since (x, p) satisfies CS, all forward arcs of an unblocked path P must be inactive or balanced, while all backward arcs of P must be active or balanced [cf. Eqs. (1.6)-(1.8)], so we have

$$r_{ij} \geq 0, \qquad \forall\ (i,j) \in P^+, \tag{2.4}$$

$$r_{ij} \leq 0, \qquad \forall\ (i,j) \in P^-. \tag{2.5}$$

Thus, the length of P is nonnegative.

The sequential shortest path method starts each iteration with an integer pair (x, p) satisfying CS and with a set I of nodes i with $g_i > 0$, and proceeds as follows.

Sequential Shortest Path Iteration

Construct an augmenting path P with respect to x that has minimum length over all augmenting paths with respect to x that start at some node $i \in I$. Then, carry out an augmentation along P (cf. Step 3 of the primal-dual iteration) and modify the node prices as follows: let $\overline{d}$ be the length of P and for each node $m \in \mathcal{N}$, let d_m be the minimum of the lengths of the unblocked paths with respect to x that start at some node in I and end at m ($d_m = \infty$ if no such path exists). The new price vector $\overline{p}$ is given by

$$\overline{p}_m = p_m + \max\{0, \overline{d} - d_m\}, \qquad \forall\ m \in \mathcal{N}. \tag{2.6}$$

The method terminates under the following circumstances:

(a) All nodes i have zero surplus; in this case it will be seen that the current pair (x, p) is primal and dual optimal.

(b) $g_i \leq 0$ for all i and $g_i < 0$ for at least one i; in this case the problem is infeasible, since $\sum_{i\in\mathcal{N}} s_i = \sum_{i\in\mathcal{N}} g_i < 0$.

(c) There is no augmenting path with respect to x that starts at some node in I; in this case it will be seen that the problem is infeasible.

We will show shortly that the method preserves the integrality and the CS property of the pair (x, p), and that it terminates.

It is important to note that the shortest path computation can be executed using the standard shortest path algorithms described in Section 1.3.

The idea is to use r_{ij} as the length of each forward arc (i,j) of an unblocked path, and to reverse the direction of each backward arc (i,j) of an unblocked path and to use $-r_{ij}$ as its length [cf. the unblocked path length formula (2.3)]. In particular, the iteration can be executed using the following procedure.

Consider the *residual graph*, which has the same node set $\mathcal{N}$ of the original problem graph, and has

an arc (i,j) with length r_{ij} for every arc $(i,j) \in \mathcal{A}$ with $x_{ij} < c_{ij}$,

an arc (j,i) with length $-r_{ij}$ for every arc $(i,j) \in \mathcal{A}$ with $b_{ij} < x_{ij}$.

[If this creates two arcs in the same direction between two nodes, discard the arc with the larger length (in case of a tie, discard either arc).] Find a path P that is shortest among paths of the residual graph that start at some node in I and end at some node with negative surplus. Find also the shortest distances d_m from nodes of I to all other nodes m [or at least to those nodes m with d_m less than the length of P; cf. Eq. (2.6)].

Figure 2.2 illustrates the sequential shortest path method and shows the sequence of residual graphs for the example worked out earlier (cf. Fig. 2.1).

Note here that by Eqs. (2.4) and (2.5), the arc lengths of the residual graph are nonnegative, so Dijkstra's method can be used for the shortest path computation. Since all forward paths in the residual graph correspond to unblocked paths in the original problem graph, and corresponding paths have the same length, it is seen that the shortest path P is an augmenting path as required and that the shortest distances d_m yield the vector $\overline{p}$ defined by Eq. (2.6). We now prove the validity of the method.

Proposition 2.2: Consider the minimum cost flow problem and assume that a_{ij}, b_{ij}, c_{ij}, and s_i are all integer. Then, for the sequential shortest path method, the following hold:

(a) Each iteration maintains the integrality and the CS property of the pair (x,p).

(b) If the problem is feasible, then the method terminates with an integer optimal flow vector x and an integer optimal price vector p.

(c) If the problem is infeasible, then the method terminates either because $g_i \leq 0$ for all i and $g_i < 0$ for at least one i, or because there is no augmenting path starting at some node of the set I and ending at some node with negative surplus.

Proof: (a) We will show that if the starting pair (x,p) of an iteration is integer and satisfies CS, the same is true for a pair $(\overline{x},\overline{p})$ produced by the iteration. Indeed, a flow augmentation maintains the integrality of the flows, since the upper and lower flow bounds are assumed integer. Furthermore, the arc lengths of the residual graph are integer, so by Eq. (2.6), $\overline{p}$ is integer.

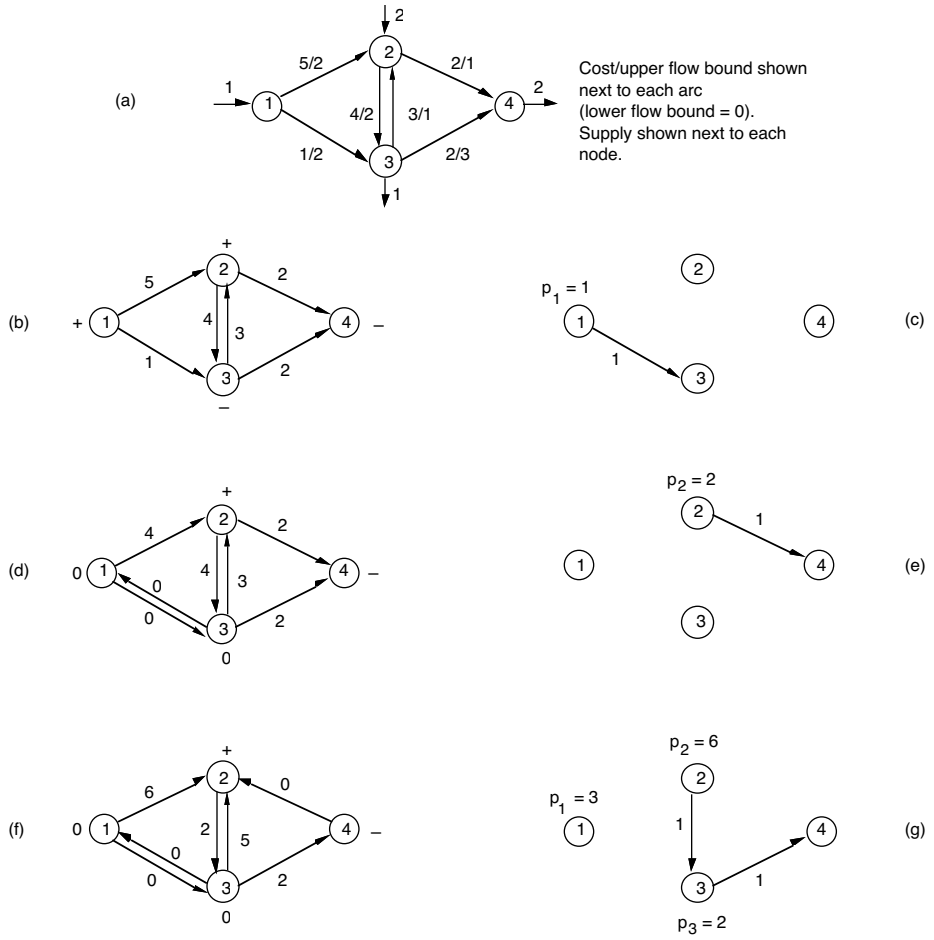


Figure 2.2 The sequential shortest path method applied to the problem of Fig. 2.1, starting with all zero prices. The sequences of flows, prices, and surpluses are the same as those generated by the primal-dual method.

(a) Problem data.

(b) Initial residual graph with the arc lengths shown next to the arcs. The nodes with positive, zero, and negative surplus are indicated by "+", "0", and "−", respectively.

(c) Shortest augmenting path and changed prices of first iteration ($I = \{1\}$).

(d) Residual graph with the arc lengths shown next to the arcs after the first iteration.

(e) Shortest augmenting path and changed prices of second iteration ($I = \{2\}$).

(f) Residual graph with the arc lengths shown next to the arcs after the second iteration.

(g) Shortest augmenting path and changed prices of third (and final) iteration ($I = \{2\}$).

To show that $(\overline{x}, \overline{p})$ satisfies CS, consider an arc (i, j) for which $\overline{x}_{ij} < c_{ij}$. We will show that $\overline{p}_i - \overline{p}_j \leq a_{ij}$. We distinguish two cases:

(1) $x_{ij} = c_{ij}$. In this case, we have $b_{ij} < x_{ij}$, the direction of (i, j) is reversed in the residual graph, and the reverse arc (j, i) lies on the shortest augmenting path P. Hence, we have

$$d_i \leq \overline{d}, \qquad d_j \leq \overline{d}, \qquad d_i = d_j - r_{ij}.$$

Using these equations, and Eqs. (2.2) and (2.6), we obtain

$$\begin{aligned}\overline{p}_i - \overline{p}_j &= p_i - p_j + \max\{0, \overline{d} - d_i\} - \max\{0, \overline{d} - d_j\} \\ &= p_i - p_j - (d_i - d_j) = p_i - p_j + r_{ij} = a_{ij}.\end{aligned}$$

(2) $x_{ij} < c_{ij}$. In this case we have

$$d_j \leq d_i + r_{ij},$$

since (i, j) is an arc of the residual graph with length r_{ij}. Using this relation and the nonnegativity of r_{ij}, we see that

$$\begin{aligned}\max\{0, \overline{d} - d_i\} &\leq \max\{0, \overline{d} - d_j + r_{ij}\} \\ &\leq \max\{r_{ij}, \overline{d} - d_j + r_{ij}\} = \max\{0, \overline{d} - d_j\} + r_{ij}.\end{aligned}$$

Hence, we have

$$\overline{p}_i - \overline{p}_j = p_i - p_j + \max\{0, \overline{d} - d_i\} - \max\{0, \overline{d} - d_j\} \leq p_i - p_j + r_{ij} = a_{ij}.$$

Thus, in both cases we have $\overline{p}_i - \overline{p}_j \leq a_{ij}$. We can similarly show that if $b_{ij} < \overline{x}_{ij}$, then $\overline{p}_i - \overline{p}_j \geq a_{ij}$, completing the proof of the CS property of the pair $(\overline{x}, \overline{p})$.

(b) and (c) Every completed iteration in which a shortest augmenting path is found reduces the total absolute surplus $\sum_{i \in \mathcal{N}} |g_i|$ by an integer amount, so termination must occur. Part (a) shows that at the start of each iteration, the pair (x, p) satisfies CS. There are two possibilities:

(1) $g_i \leq 0$ for all i. In this case, either $g_i = 0$ for all i in which case x is feasible, and x and p are primal and dual optimal, respectively, since they satisfy CS, or else $g_i < 0$ for some i, in which case the problem is infeasible.

(2) $g_i > 0$ for at least one i. In this case we can select a nonempty set I of nodes with positive surplus, form the residual graph, and attempt the corresponding shortest path computation. There are two possibilities: either a shortest augmenting path is found, in which case the iteration

will be completed with an attendant reduction of the total absolute surplus, or else there is no unblocked path with respect to x from a node of I to a node with negative surplus. In the latter case, we claim that the problem is infeasible. Indeed, by Prop. 2.2 in Section 1.2 (more accurately, the generalization given in Exercise 2.12 in Section 1.2), there exists a saturated cut $Q = [\mathcal{S}, \mathcal{N} - \mathcal{S}]$ such that all nodes of I belong to $\mathcal{S}$ and all nodes with negative surplus belong to $\mathcal{N} - \mathcal{S}$. The flux across Q is equal to the capacity $C(Q)$ of Q and is also equal to the sum of the divergences of the nodes of $\mathcal{S}$, which is $\sum_{i \in \mathcal{S}}(s_i - g_i)$ [cf. Eq. (1.11)]. Since $g_i \geq 0$ for all $i \in \mathcal{S}$, $g_i > 0$ for the nodes $i \in I$, and $I \subset \mathcal{S}$, we see that

$$C(Q) < \sum_{i \in \mathcal{S}} s_i.$$

This implies that the problem is infeasible, since for any feasible flow vector we must have $\sum_{i \in \mathcal{S}} s_i = F(Q) \leq C(Q)$, where $F(Q)$ is the corresponding flux across Q.

Thus, termination of the algorithm must occur in the manner stated in the proposition. **Q.E.D.**

By appropriately adapting the shortest path algorithms of Section 1.3, one can obtain a variety of implementations of the sequential shortest path iteration. Here is an example, which adapts the generic single origin/single destination algorithm of Section 1.3.4 and supplements it with a labeling procedure that constructs the augmenting path. We introduce a candidate list V, a label d_i for each node i, a shortest distance estimate $\overline{d}$, and a node $\overline{j}$ whose initial choice is immaterial. Given a pair (x, p) satisfying CS and a set I of nodes with positive surplus, we set initially

$$V = I, \qquad \overline{d} = \infty,$$

$$d_i = 0, \qquad \forall\ i \in I, \qquad d_i = \infty, \qquad \forall\ i \notin I.$$

The shortest path computation proceeds in steps and terminates when V is empty. The typical step (assuming V is nonempty) is as follows:

Typical Shortest Path Step in a Sequential Shortest Path Iteration

Remove a node i from V. For each outgoing arc $(i, j) \in \mathcal{A}$, with $x_{ij} < c_{ij}$, if

$$d_i + r_{ij} < \min\{d_j, \overline{d}\},$$

give the label "(i, j)" to j, set

$$d_j := d_i + r_{ij},$$

add j to V if it does not already belong to V, and if $g_j < 0$, set $\overline{d} = d_i + r_{ij}$ and $\overline{j} = j$. Also, for each incoming arc $(j,i) \in \mathcal{A}$, with $b_{ji} < x_{ji}$, if

$$d_i - r_{ji} < \min\{d_j, \overline{d}\},$$

give the label "(j,i)" to j, set

$$d_j := d_i - r_{ji},$$

add j to V if it does not already belong to V, and if $g_j < 0$, set $\overline{d} = d_i - r_{ji}$ and $\overline{j} = j$.

When the shortest path computation terminates, an augmenting path of length $\overline{d}$ can be obtained by tracing labels backward from the node $\overline{j}$ to some node $i \in I$. The new price vector $\overline{p}$ is obtained via the equation $\overline{p}_m = p_m + \max\{0, \overline{d} - d_m\}$ for all $m \in \mathcal{N}$ [cf. Eq. (2.6)]. Note that if the node i removed from V has the minimum label property

$$d_i = \min_{j \in V} d_j,$$

the preceding algorithm corresponds to Dijkstra's method.

We finally note that the primal-dual method discussed earlier and the sequential shortest path method are mathematically equivalent in that they produce identical sequences of pairs (x,p), as shown by the following proposition (for an example, compare the calculations of Figs. 2.1 and 2.2). In fact with some thought, it can be seen that the primal-dual iteration amounts to the use of a form of Dijkstra's algorithm to calculate the shortest augmenting path and the corresponding distances.

Proposition 2.3: Suppose that a primal-dual iteration starts with a pair (x,p), and let I be the initial set of nodes i with $g_i > 0$. Then:

(a) An augmenting path P may be generated in the augmentation Step 3 of the iteration (through some order of operations in Steps 1 and 2) if and only if P has minimum length over all augmenting paths with respect to x that start at some node in I.

(b) If $\overline{p}$ is the price vector produced by the iteration, then

$$\overline{p}_m = p_m + \max\{0, \overline{d} - d_m\}, \qquad \forall\ m \in \mathcal{N}, \tag{2.7}$$

where $\overline{d}$ is the length of the augmenting path P of the iteration and for each $m \in \mathcal{N}$, d_m is the minimum of the lengths of the unblocked paths with respect to x that start at some node in I and end at m.

Proof: Let $\overline{k} \geq 0$ be the number of price changes of the iteration. If $\overline{k} = 0$, i.e., no price change occurs, then any augmenting path P that can be produced

by the iteration consists of balanced arcs, so its length is zero. Hence P has minimum length as stated in part (a). Furthermore, $\overline{p} = p$, which verifies Eq. (2.7).

Assume that $\overline{k} \geq 1$, let $\mathcal{S}_k$, $k = 1, \ldots, \overline{k}$, be the set of scanned nodes $\mathcal{S}$ when the kth price change occurs, and let γ_k, $k = 1, \ldots, \overline{k}$, be the corresponding price increment [cf. Eq. (2.1)]. Let also $\mathcal{S}_{\overline{k}+1}$ be the set $\mathcal{S}$ at the end of the iteration. We note that the sets $\mathcal{S}_k$ (and hence also γ_k) depend only on (x, p) and the set I, and are independent of the order of operations in Steps 1 and 2. In particular, $\mathcal{S}_1 - I$ is the set of all nodes j such that there exists an unblocked path of balanced arcs [with respect to (x, p)] that starts at some node $i \in I$ and ends at j. Thus, $\mathcal{S}_1$ and also γ_1, is uniquely defined by I and (x, p). Proceeding inductively, it is seen that $\mathcal{S}_{k+1} - \mathcal{S}_k$ is the set of all nodes j such that there exists an unblocked path of balanced arcs [with respect to (x, p^k), where p^k is the price vector after k price changes] that starts at some node $i \in \mathcal{S}_k$ and ends at j. Thus, $\mathcal{S}_{k+1}$ and γ_{k+1} are uniquely defined by I and (x, p) if $\mathcal{S}_1, \ldots, \mathcal{S}_k$ and $\gamma_1, \ldots, \gamma_k$ are.

It can be seen from Eq. (2.1) that for all k,

γ_k =minimum over the lengths of all (single arc) unblocked paths starting at a node $i \in \mathcal{S}_k$ and ending at a node $j \notin \mathcal{S}_k$.

Using this property, and an induction argument (left for the reader), we can show that d_m, which is defined as the minimum over the lengths of all unblocked paths that start at some node $i \in I$ and end at node m, satisfies for all k,

$$d_m = \gamma_1 + \gamma_2 + \ldots + \gamma_k, \qquad \forall\ m \in \mathcal{S}_{k+1} - \mathcal{S}_k. \tag{2.8}$$

Furthermore, the length of any unblocked path that starts at some node $i \in I$ and ends at a node $m \notin \mathcal{S}_{k+1}$ is larger than $\gamma_1 + \gamma_2 + \ldots + \gamma_k$. In particular, the length of any augmenting path produced by the iteration is

$$\gamma_1 + \gamma_2 + \ldots + \gamma_{\overline{k}},$$

so it has the property stated in part (a). Also, the price vector $\overline{p}$ produced by the primal-dual iteration is given by

$$\overline{p}_m = \begin{cases} p_m + \gamma_1 + \gamma_2 + \ldots + \gamma_k & \text{if } m \in \mathcal{S}_{k+1} - \mathcal{S}_k,\ k = 1, \ldots, \overline{k}, \\ p_m & \text{otherwise,} \end{cases}$$

which in view of Eq. (2.8), agrees with Eq. (2.7). **Q.E.D.**

E X E R C I S E S

Exercise 2.1

Use the primal-dual method and the sequential shortest path method to solve the problem of Fig. 2.3. Verify that the two methods yield the same sequence of flows and prices (with identical initial data and appropriate choices of the initial sets I and augmenting paths).

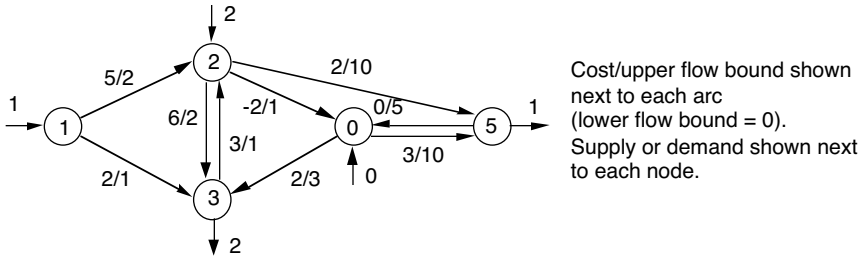


Figure 2.3 Minimum cost flow problem for Exercise 2.1.

Exercise 2.2 (Relation of Primal-Dual and Ford-Fulkerson)

Consider the Ford-Fulkerson algorithm for the max-flow problem, where $b_{ij} = 0$ for all $(i, j) \in \mathcal{A}$. Show that the method can be interpreted as an application of the primal-dual method to the minimum cost flow formulation of the max-flow problem of Example 1.2 in Section 1.1, starting with $p = 0$ and $x = 0$ [except for the flow of the artificial arc (t, s), which must be at its upper bound to satisfy CS]. Show in particular that all iterations of the primal-dual method start at s and terminate with an augmentation along a path ending at t. Furthermore, the method will execute only one price change, which will occur after a minimum cut is identified. The last iteration consists of an augmentation along the artificial arc (t, s).

Exercise 2.3 (Relation of Primal-Dual and Dijkstra)

Consider the shortest path problem with node 1 being the origin and all other nodes being destinations. Formulate this problem as a minimum cost flow problem with the origin having supply $N - 1$ and all destinations having supply -1. Assume that all arc lengths are nonnegative. Start with all flows and prices equal to zero, and apply the primal-dual method. Show that the

method is equivalent to Dijkstra's algorithm. In particular, each augmentation uses a shortest path from the origin to some destination, the augmentations are done in the order of the destinations' proximity to the origin, and upon termination, $p_1 - p_i$ gives the shortest distance from 1 to each destination i that can be reached from the origin via a forward path.

Exercise 2.4 (Noninteger Problem Data)

Verify that the primal-dual method terminates even when the arc costs are noninteger. (Note, however, that the arc flow bounds must still be integer; the max-flow example of Exercise 2.9 in Section 1.2 applies to the primal-dual method as well, in view of the relation described in Exercise 2.2.) Modify the primal-dual method so that augmenting paths have as few arcs as possible. Show that with this modification, the arc flow bounds need not be integer for the method to terminate. How should the sequential shortest path method be modified so that it terminates even when the problem data are not integer?

3.3 THE RELAXATION METHOD

This method admits a similar implementation as the primal-dual method but computes ascent directions much faster. In particular, while in the primal-dual method we continue to enlarge the scanned set $\mathcal{S}$ until it is equal to the labeled set $\mathcal{L}$ (in which case we are sure that $d_{\mathcal{S}}$ is an ascent direction), in the relaxation method we stop adding nodes to $\mathcal{S}$ immediately after $d_{\mathcal{S}}$ becomes an ascent direction [this is done by computing the directional derivative $q'(p; d_{\mathcal{S}})$ using an efficient incremental method and by checking its sign]. In practice, $\mathcal{S}$ often consists of a single node, in which case the ascent direction is a single price coordinate, leading to the interpretation of the method as a *coordinate ascent method*. Unlike the primal-dual method, the relaxation method cannot be implemented using a shortest path computation.

As in the primal-dual method, at the start of the typical iteration we have an integer pair (x, p) satisfying CS. The iteration indicates that the primal problem is infeasible, or else indicates that (x, p) is optimal, or else transforms this pair into another pair satisfying CS. In particular, if $g_i \leq 0$ for all i, then there are two possibilities: (1) $g_i < 0$ for some i, in which case $\sum_{i \in \mathcal{N}} s_i < 0$ and the problem is infeasible, or (2) $g_i = 0$ for all i, in which case x is feasible and therefore also optimal, since it satisfies CS together with p. In either case, the algorithm terminates.

If on the other hand we have $g_i > 0$ for at least one node i, the iteration starts by selecting a node $\bar{i}$ with $g_{\bar{i}} > 0$. As in the primal-dual method, the iteration maintains two sets of nodes $\mathcal{S}$ and $\mathcal{L}$, with $\mathcal{S} \subset \mathcal{L}$. At the start of the

iteration, $\mathcal{S}$ is empty and $\mathcal{L}$ consists of the node $\bar{i}$ with $g_{\bar{i}} > 0$. The iteration also maintains a *label* for every node $i \in \mathcal{L}$ except for the starting node $\bar{i}$; the label is an incident arc of i.

Typical Relaxation Iteration

Step 0 (Initialization): Select a node $\bar{i}$ with $g_{\bar{i}} > 0$. [If no such node can be found, terminate; the pair (x, p) is optimal if $g_i = 0$ for all i; otherwise the problem is infeasible.] Set $\mathcal{L} := \{\bar{i}\}$ and $\mathcal{S} :=$ empty, and go to Step 1.

Step 1 (Choose a Node to Scan): If $\mathcal{S} = \mathcal{L}$, go to Step 4; else select a node $i \in \mathcal{L} - \mathcal{S}$, set $\mathcal{S} := \mathcal{S} \cup \{i\}$, and go to Step 2.

Step 2 (Label Neighbor Nodes of i): If

$$q'(p; d_{\mathcal{S}}) > 0, \tag{3.1}$$

go to Step 4; else add to $\mathcal{L}$ all nodes $j \notin \mathcal{L}$ such that either (j, i) is balanced and $b_{ji} < x_{ji}$ or (i, j) is balanced and $x_{ij} < c_{ij}$; also for every such j, give to j the label "(j, i)" if (j, i) is balanced and $b_{ji} < x_{ji}$, and otherwise give to j the label "(i, j)." If for every node j just added to $\mathcal{L}$, we have $g_j \geq 0$, go to Step 1; else select one of these nodes j with $g_j < 0$ and go to Step 3.

Step 3 (Flow Augmentation): An augmenting path P has been found that begins at the starting node $\bar{i}$ and ends at the node j identified in Step 2. The path is constructed by tracing labels backward starting from j, and is such that we have

$$x_{mn} < c_{mn}, \qquad \forall\ (m, n) \in P^+, \tag{3.2}$$

$$x_{mn} > b_{mn}, \qquad \forall\ (m, n) \in P^-, \tag{3.3}$$

where P^+ and P^- are the sets of forward and backward arcs of P, respectively. Let

$$\delta = \min\big\{g_i, -g_j, \{c_{mn} - x_{mn} \mid (m, n) \in P^+\}, \{x_{mn} - b_{mn} \mid (m, n) \in P^-\}\big\}.$$

Increase by δ the flows of all arcs in P^+, decrease by δ the flows of all arcs in P^-, and go to the next iteration.

Step 4 (Price Change): Set

$$x_{ij} = c_{ij}, \qquad \forall \text{ balanced arcs } (i, j) \text{ with } i \in \mathcal{S},\ j \notin \mathcal{S}, \tag{3.4}$$

$$x_{ji} = b_{ji}, \qquad \forall \text{ balanced arcs } (j, i) \text{ with } i \in \mathcal{S},\ j \notin \mathcal{S}. \tag{3.5}$$

Let

$$\gamma = \min\Big\{\{p_j + a_{ij} - p_i \mid (i,j) \in \mathcal{A}, x_{ij} < c_{ij}, i \in \mathcal{S}, j \notin \mathcal{S}\}, \\ \{p_j - a_{ji} - p_i \mid (j,i) \in \mathcal{A}, b_{ji} < x_{ji}, i \in \mathcal{S}, j \notin \mathcal{S}\}\Big\}. \tag{3.6}$$

Set

$$p_i := \begin{cases} p_i + \gamma, & \text{if } i \in \mathcal{S} \\ p_i, & \text{otherwise.} \end{cases} \tag{3.7}$$

Go to the next iteration. [*Note:* As in the case of the primal-dual iteration, if after the flow adjustments of Eqs. (3.4) and (3.5) there is no arc (i,j) with $x_{ij} < c_{ij}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$, or arc (j,i) with $b_{ji} < x_{ji}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$, the problem is infeasible and the algorithm terminates.]

It can be seen that the relaxation iteration is quite similar to the primal-dual iteration. However, there are two important differences. First, in the relaxation iteration, after a price change in Step 4, we do not return to Step 1 to continue the search for an augmenting path like we do in the primal-dual method. Thus, the relaxation iteration terminates either with an augmentation as in Step 3 or with a price change as in Step 4, in contrast with the primal-dual iteration, which can only terminate with an augmentation. The second and more important difference is that in the relaxation iteration, a price change may be performed in Step 4 even if $\mathcal{S} \neq \mathcal{L}$ [cf. Eq. (3.1)]. It is because of this feature that the relaxation method identifies ascent directions faster than the primal-dual method. Note that in contrast with the primal-dual method, the total absolute surplus $\sum_{i \in \mathcal{N}} |g_i|$ may increase as a result of a relaxation iteration.

An important property of the method is that each time we enter Step 4, $d_{\mathcal{S}}$ is an ascent direction. To see this note that there are two possibilities: (1) we have $\mathcal{S} = \mathcal{L}$ (cf. Step 1) in which case $d_{\mathcal{S}}$ is an ascent direction similar to the corresponding situation in the primal-dual method, or (2) we have $\mathcal{S} \neq \mathcal{L}$ (cf. Step 2) in which case by Eq. (3.1) $d_{\mathcal{S}}$ is an ascent direction.

It is possible to "combine" several iterations of the relaxation method into a single iteration in order to save computation time, and this is done judiciously in the RELAX codes, which are public domain implementations of the relaxation method [BeT88], [BeT90]. Figure 3.1 traces the steps of the method for a simple example.

The following proposition establishes the validity of the method.

Proposition 3.1: Consider the minimum cost flow problem and assume that a_{ij}, b_{ij}, c_{ij}, and s_i are all integer. If the problem is feasible, then the relaxation method terminates with an integer optimal flow vector x and an integer optimal price vector p.

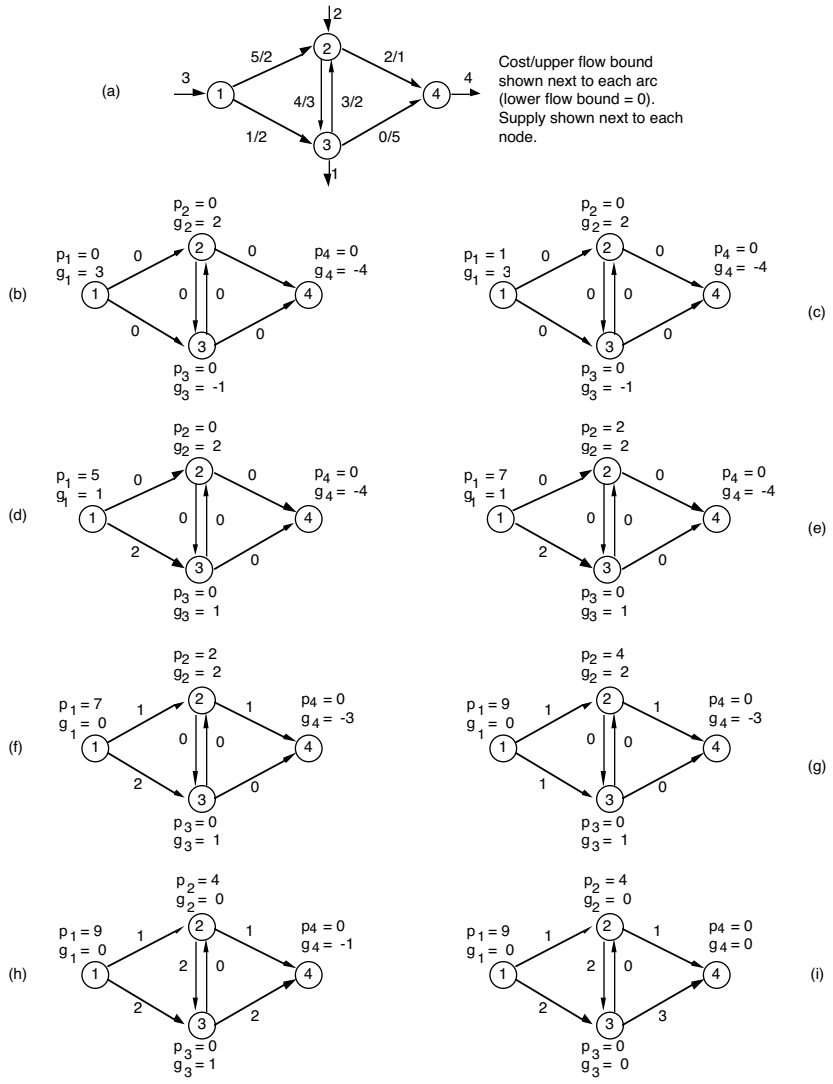


Figure 3.1 An illustration of the relaxation method, starting with all zero prices.

(a) Problem data.

(b) Initial flows, prices, and surpluses.

(c) After the first iteration, which consists of a price change of node 1.

(d) After the second iteration, which consists of another price change of node 1 [note the flow change of arc (1,3); cf. Eq. (3.4)].

(e) After the third iteration, which consists of a price change of nodes 1 and 2.

(f) After the fourth iteration, which consists of an augmentation along the path $(1, 2, 4)$.

(g) After the fifth iteration, which consists of a price change of nodes 1 and 2.

(h) After the sixth iteration, which consists of an augmentation along the path $(2, 3, 4)$.

(i) After the seventh iteration, which consists of an augmentation along the path $(3, 4)$.

Proof: The proof is similar to the corresponding proof for the primal-dual method (cf. Prop. 2.1). We first note that all operations of the iteration preserve the integrality of the flow–price vector pair. To see that CS is also maintained, note that a flow augmentation step changes only flows of balanced arcs and therefore cannot destroy CS. Furthermore, the flow changes of Eqs. (3.4) and (3.5), and the price changes of Eqs. (3.6) and (3.7) maintain CS, because they set the flows of the balanced arcs that the price change renders active (or inactive) to the corresponding upper (or lower) bounds.

Every time there is a price change in Step 4, there is a strict improvement in the dual cost by the integer amount $\gamma q'(p; d_{\mathcal{S}})$ [using the CS property, it can be seen that $\gamma > 0$, and as argued earlier, $d_{\mathcal{S}}$ is an ascent direction so $q'(p; d_{\mathcal{S}}) > 0$]. Thus, for a feasible problem, we cannot have an infinite number of price changes. On the other hand, it is impossible to have an infinite number of flow augmentations between two successive price changes, since each of these reduces the total absolute surplus by an integer amount. It follows that the algorithm can execute only a finite number of iterations, and must terminate. Since upon termination x is feasible and satisfies CS together with p, it follows that x is primal-optimal and p is dual-optimal. **Q.E.D.**

If the problem is infeasible, the method may terminate because $g_i \leq 0$ for all i and $g_i < 0$ for at least one i, or because after the flow adjustments of Eqs. (3.4) and (3.5) in Step 4, there is no arc (i, j) with $x_{ij} < c_{ij}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$, or arc (j, i) with $b_{ji} < x_{ji}$, $i \in \mathcal{S}$, and $j \notin \mathcal{S}$. However, there is also the possibility that the method will execute an infinite number of iterations and price changes, with the prices of some of the nodes increasing to ∞. Exercise 3.2 shows that, when the problem is feasible, the node prices stay below a certain precomputable bound in the course of the algorithm. This fact can be used as an additional test to detect infeasibility.

It is important to note that the directional derivative $q'(p; d_{\mathcal{S}})$ needed for the ascent test (3.1) in Step 2 can be calculated *incrementally* (as new nodes are added one-by-one to $\mathcal{S}$) using the equation

$$q'(p; d_{\mathcal{S}}) = \sum_{i \in \mathcal{S}} g_i - \sum_{(j,i):\ \text{balanced},\ j \notin \mathcal{S},\ i \in \mathcal{S}} (x_{ji} - b_{ji}) - \sum_{(i,j):\ \text{balanced},\ i \in \mathcal{S},\ j \notin \mathcal{S}} (c_{ij} - x_{ij}); \tag{3.8}$$

cf. Eq. (1.13). Indeed, it follows from this equation that, given $q'(p; d_{\mathcal{S}})$ and a node $i \notin \mathcal{S}$, one can calculate the directional derivative corresponding to the

enlarged set $\mathcal{S} \cup \{i\}$ using the formula

$$\begin{aligned} q'(p; d_{\mathcal{S}\cup\{i\}}) = q'(p; d_{\mathcal{S}}) &+ \sum_{\{j|(i,j):\ \text{balanced},\ j\in\mathcal{S}\}} (x_{ij} - b_{ij}) \\ &+ \sum_{\{j|(j,i):\ \text{balanced},\ j\in\mathcal{S}\}} (c_{ji} - x_{ji}) \\ &- \sum_{\{j|(j,i):\ \text{balanced},\ j\notin\mathcal{S}\}} (x_{ji} - b_{ji}) \\ &- \sum_{\{j|(i,j):\ \text{balanced},\ j\notin\mathcal{S}\}} (c_{ij} - x_{ij}). \end{aligned} \tag{3.9}$$

This formula is convenient because it involves only the incident balanced arcs of the new node i, which must be examined anyway while executing Step 2.

In practice, the method is implemented using iterations that start from both positive and negative surplus nodes. This seems to improve substantially the performance of the method. It can be shown that for a feasible problem, the algorithm terminates properly under these circumstances (Exercise 3.3). Another important practical issue has to do with the initial choice of flows and prices. One possibility is to try to choose an initial price vector that is as close to optimal as possible (for example, using the results of some earlier optimization); one can then choose the arc flows to satisfy the CS conditions.

Line Search and Coordinate Ascent Iterations

The stepsize γ of Eq. (3.6) corresponds to the first break point of the piecewise linear dual function along the ascent direction $d_{\mathcal{S}}$. It is also possible to calculate through a line search an optimal stepsize that maximizes the dual function along $d_{\mathcal{S}}$. We leave it for the reader to verify that this computation can be done quite economically, using Eq. (1.10) or Eq. (1.13) to test the sign of the directional derivative of the dual function at successive break points along $d_{\mathcal{S}}$. Computational experience shows that a line search is beneficial in practice. For this reason, it has been used in the RELAX codes [BeT88], [BeT90].

Consider now the case where there is a price change via Step 4 and the set $\mathcal{S}$ consists of just the starting node, say node i. This happens when the iteration scans the incident arcs of i at the first time Step 2 is entered and finds that the corresponding coordinate direction leads to a dual cost improvement $[q'\big(p; d_{\{i\}}\big) > 0]$. If line search of the type just described is performed, the price p_i is changed to a break point where the right derivative is nonpositive and the left derivative is nonnegative (cf. Fig. 3.2).

A precise description of this single-node relaxation iteration with line search, starting from a pair (x, p) satisfying CS, is as follows:

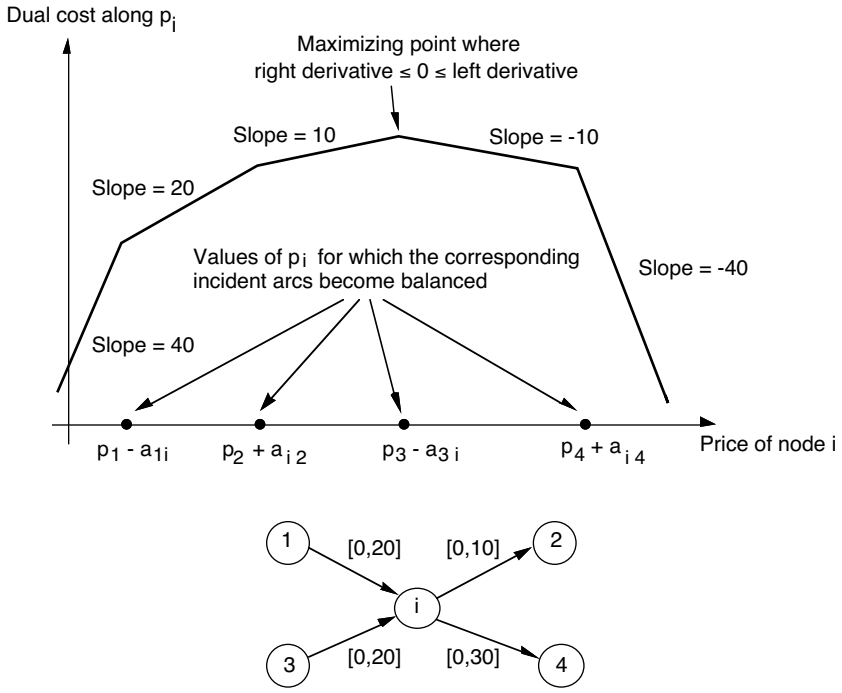


Figure 3.2 Illustration of single-node relaxation iteration. Here, node i has four incident arcs $(1,i)$, $(3,i)$, $(i,2)$, and $(i,4)$ with flow ranges $[0,20]$, $[0,20]$, $[0,10]$, and $[0,30]$, respectively, and supply $s_i = 0$. The arc costs and current prices are such that

$$p_1 - a_{1i} \leq p_2 + a_{i2} \leq p_3 - a_{3i} \leq p_4 + a_{i4},$$

as shown in the figure. The break points of the dual cost along the price p_i correspond to the values of p_i at which one or more incident arcs to node i become balanced. For values between two successive break points, there are no balanced arcs. For any price p_i to the left of the maximizing point, the surplus g_i must be positive to satisfy CS. A single-node iteration with line search increases p_i to the maximizing point.

Single-Node Relaxation Iteration

Choose a node i with $g_i > 0$. Let

$$B_i^+ = \{j \mid (i,j) : \text{ balanced, } x_{ij} < c_{ij}\}, \tag{3.10}$$

$$B_i^- = \{j \mid (j,i) : \text{ balanced, } b_{ji} < x_{ji}\}. \tag{3.11}$$

Step 1: If

$$g_i \geq \sum_{j \in B_i^+} (c_{ij} - x_{ij}) + \sum_{j \in B_i^-} (x_{ji} - b_{ji}),$$

go to Step 4. Otherwise, if $g_i > 0$, choose a node $j \in B_i^+$ with $g_j < 0$ and go to Step 2, or choose a node $j \in B_i^-$ with $g_j < 0$ and go to Step 3; if no such node can be found, or if $g_i = 0$, go to the next iteration.

Step 2 (Flow Adjustment on Outgoing Arc): Let

$$\delta = \min\{g_i, -g_j, c_{ij} - x_{ij}\}.$$

Set

$$x_{ij} := x_{ij} + \delta, \qquad g_i := g_i - \delta, \qquad g_j := g_j + \delta$$

and if $x_{ij} = c_{ij}$, delete j from B_i^+; go to Step 1.

Step 3 (Flow Adjustment on Incoming Arc): Let

$$\delta = \min\{g_i, -g_j, x_{ji} - b_{ji}\}.$$

Set

$$x_{ji} := x_{ji} - \delta, \qquad g_i := g_i - \delta, \qquad g_j := g_j + \delta$$

and if $x_{ji} = b_{ji}$, delete j from B_i^-; go to Step 1.

Step 4 (Increase Price of i): Set

$$g_i := g_i - \sum_{j \in B_i^+} (c_{ij} - x_{ij}) - \sum_{j \in B_i^-} (x_{ji} - b_{ji}), \tag{3.12}$$

$$x_{ij} = c_{ij}, \qquad \forall\ j \in B_i^+, \tag{3.13}$$

$$x_{ji} = b_{ji}, \qquad \forall\ j \in B_i^-, \tag{3.14}$$

$$\begin{aligned} p_i := \min\big\{ &\{p_j + a_{ij} \mid (i,j) \in \mathcal{A},\ p_i < p_j + a_{ij}\}, \\ &\{p_j - a_{ji} \mid (j,i) \in \mathcal{A},\ p_i < p_j - a_{ji}\}\big\}. \end{aligned} \tag{3.15}$$

If after these changes $g_i > 0$, recalculate the sets B_i^+ and B_i^+ using Eqs. (3.10) and (3.11), and go to Step 1; else, go to the next iteration. [*Note:* If the set of arcs over which the minimum in Eq. (3.15) is calculated is empty, there are two possibilities: (a) $g_i > 0$, in which case it can be shown that the dual cost increases without bound along p_i and the primal problem is infeasible, or (b) $g_i = 0$, in which case the cost stays constant along p_i; in this case we leave p unchanged and go to the next iteration.]

Note that the single-node iteration may be unsuccessful in that it may fail to change either x or p. In this case, it should be followed by a regular relaxation iteration that labels the appropriate neighbors of node i, etc. Experience has shown that the most efficient way to implement the relaxation iteration is to first attempt its single-node version; if this fails to change x or p, then we proceed with the multiple node version, while salvaging whatever computation is possible from the single-node attempt. The RELAX codes [BeT88], [BeT90] make use of this idea. Experience shows that single-node iterations are very frequent in the early stages of the relaxation algorithm and account for most of the total dual cost improvement, but become much less frequent near the algorithm's termination.

A careful examination of the single-node iteration logic shows that in Step 4, after the surplus change of Eq. (3.12), the surplus g_i may be equal to zero; this will happen if $g_i = 0$ and simultaneously there is no balanced arc (i, j) with $x_{ij} < c_{ij}$, or balanced arc (j, i) with $b_{ji} < x_{ji}$. In this case, it can be shown (see also Fig. 3.2) that the price change of Eq. (3.15) leaves the dual cost unchanged, corresponding to movement of p_i along a flat segment to the next breakpoint of the dual cost, as shown in Fig. 3.3. This is known as a *degenerate ascent iteration*. Computational experience has shown that it is generally preferable to allow such iterations whenever possible. For special types of problems such as assignment, the use of degenerate ascent iterations can reduce dramatically the overall computation time.

We finally note that single-node relaxation iterations may be used to initialize the primal-dual method. In particular, one may start with several cycles of single-node iterations, where each node with nonzero surplus is taken up for relaxation once in each cycle. The resulting pair (x, p) is then used as a starting pair for the primal-dual method. Experience has shown that this initialization procedure is very effective.

E X E R C I S E S

Exercise 3.1

Use the relaxation method to solve the problem of Fig. 2.3.

Exercise 3.2 (An Infeasibility Test for the Relaxation Method)

Consider the relaxation method, let p_i^0 be the initial price of node i, and let $\mathcal{M}$ be the set of nodes that have negative surplus initially. For every simple path P that ends at a node $j \in \mathcal{M}$, let H_P be the sum of the costs of the forward arcs of the path minus the sum of the costs of the backward arcs of the path, and let $H = \max_P H_P$. Show that, if the problem is feasible, then during the

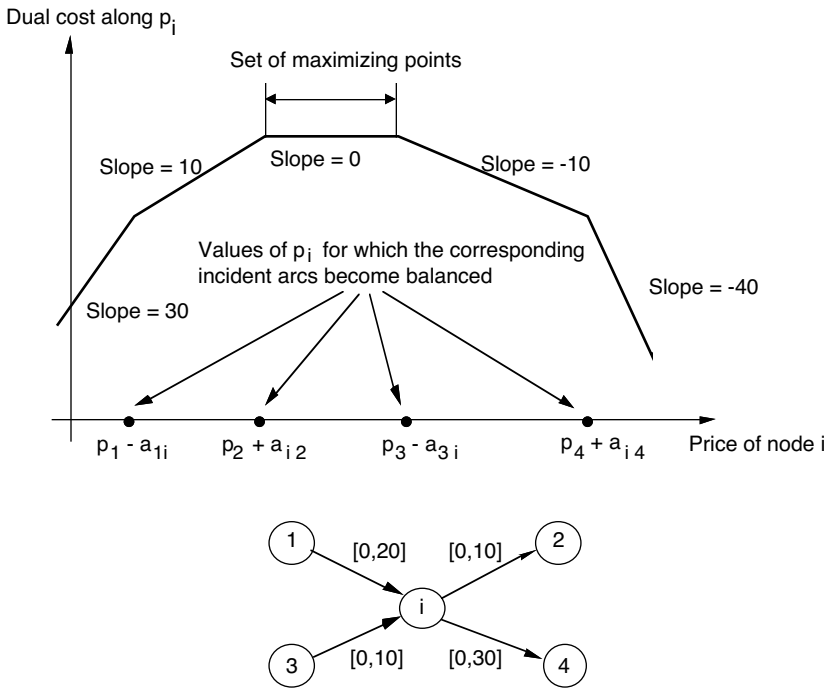


Figure 3.3 Illustration of a degenerate price increase. The difference between this example and the example of Fig. 3.2 is that the feasible flow range of arc $(3, i)$ is now $[0, 10]$ instead of $[0, 20]$. Here, there is a flat segment of the graph of the dual cost along p_i, corresponding to maximizing points. A degenerate price increase moves p_i from the extreme left maximizing point to the extreme right maximizing point.

course of the algorithm, the price of any positive surplus node cannot exceed its initial price by more than $H + \max_{j \in \mathcal{M}} p_j^0 - \min_{i \in \mathcal{N}} p_i^0$. Discuss how to use this bound to test for problem infeasibility in the relaxation method. *Hint:* Observe that at any point in the algorithm the prices of all nodes with negative surplus have not changed since the start of the algorithm. Show also that if i is a node with positive surplus, there must exist some node with negative surplus j and an unblocked path starting at i and ending at j.

Exercise 3.3

Write the form of the relaxation iteration starting from *both* positive and negative surplus nodes. Show that the method terminates at an optimal flow–price vector pair if a feasible solution exists. *Hint*: Show that each price

change improves the dual cost by an integer amount, while there can be only a finite number of flow augmentations between successive price changes.

3.4 IMPLEMENTATION ISSUES

For the application of the methods of this chapter, one can represent the problem using the five arrays *START*, *END*, *COST*, *CAPACITY*, and *SUPPLY*, as in simplex methods (cf. Section 2.4). For an efficient implementation, however, it is essential to provide additional data structures that facilitate the labeling operations, the ascent steps of Step 4, and the shortest path computations. In particular, it is necessary to have easy access to the set of all incident arcs of each node. This can be done with the help of the following four additional arrays.

FIRST_IN(i): The first arc incoming to node i ($= 0$ if i has no incoming arcs).

FIRST_OUT(i): The first arc outgoing from node i ($= 0$ if i has no outgoing arcs).

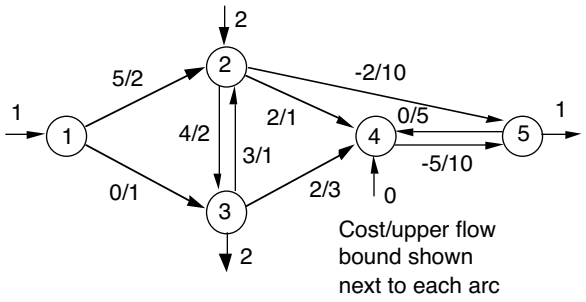
NEXT_IN(a): The arc following arc a with the same end node as a ($= 0$ if a is the last incoming arc of the end node of a).

NEXT_OUT(a): The arc following arc a with the same start node as a ($= 0$ if a is the last outgoing arc of the start node of a).

Figure 5.1 illustrates these arrays. As an example of their use, suppose that we want to scan all the incoming arcs of node i. We first obtain the arc $a_1 =$ *FIRST_IN*(i), then the arc $a_2 =$ *NEXT_IN*(a_1), then the arc $a_3 =$ *NEXT_IN*(a_2), etc., up to the arc a_k for which *NEXT_IN*$(a_k) = 0$.

It is possible to forgo the use of the array *NEXT_OUT* if the arcs are stored in the order of their starting node, that is, the arcs outgoing from each node i are arcs *FIRST_OUT*(i) to *FIRST_OUT*$(i + 1) - 1$. Then the array *FIRST_OUT* is sufficient to generate all arcs outgoing from any one node. Some codes (for example the assignment codes of Appendixes A.4 and A.5) use this device; they require that the arcs of the problem be ordered by starting node, thereby saving storage of one array (and usually some computation as well). The drawback to this idea is that it complicates sensitivity analysis. In particular, if the problem data are changed to add or remove some arcs, all the arrays describing the problem, except for *SUPPLY*, must be recompiled.

An additional data structure, useful primarily for the relaxation method, stores the *balanced* incident arcs of each node so as to facilitate the labeling step (Step 2). These arcs can be stored in two arrays of length N and two arrays of length A, much like the arrays *FIRST_IN*, *FIRST_OUT*, *NEXT_IN*,



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

NODE	SUPPLY	FIRST_IN	FIRST_OUT
1	1	0	1
2	2	1	3
3	-2	2	4
4	0	6	9
5	-1	9	8

Figure 4.1 Representation of the data of a minimum cost flow problem in terms of the nine arrays *START*, *END*, *COST*, *CAPACITY*, *SUPPLY*, *FIRST_IN*, *FIRST_OUT*, *NEXT_IN*, and *NEXT_OUT*.

and $NEXT\_OUT$. However, as the set of balanced arcs changes in the course of the algorithm, the arrays used to store this set must be updated. We will not go into further details, but the interested reader can study the RELAX codes [BeT88], [BeT90] to see how this can be done efficiently.

Overall it can be seen that dual ascent methods require more arrays of length A than simplex methods, and therefore also more storage space (roughly twice as much).

3.5 NOTES AND SOURCES

3.1. A dual ascent method that we did not cover here is the dual simplex method. This is a general linear programming method that has been specialized to the minimum cost flow problem by several authors (see e.g. [HeK77], [JeB80]) but has not achieved much popularity.

3.2. The primal-dual method was first proposed in [Kuh55] for assignment problems under the name "Hungarian method." The method was generalized to the minimum cost flow problem in [FoF56a] and [FoF57]. A further generalization, the *out-of-kilter* method, was proposed independently in [FoF62] and [Min60]; see [Law76], [Roc84], and [BJS90] for detailed discussions. The out-of-kilter method can get started with any flow–price vector pair, not necessarily one that satisfies CS. It appears, however, that there isn't much that can be gained in practice by this extra flexibility, since for any given flow–price vector pair one can modify very simply the arc flows to satisfy CS. A method that is closely related to the primal-dual method and emphasizes the shortest path implementation was given by [BuG61]. An extension of the primal-dual method to network problems with gains was given in [Jew62], and extensions of the primal-dual and out-of-kilter methods to network flow problems with separable convex cost functions are given in [Roc84]. Primal-dual methods for the assignment problem are discussed in [Eng82], [McG83], [Der85], [CaS86], [CMT88]. Combinations of naive auction and sequential shortest path methods are discussed in [Ber81], [JoV86], [JoV87]; the code of Appendix A.5 is based on these references. Variations of the Hungarian and the primal-dual methods that are well suited for parallel computation have been developed in [BMP89], [BeC90a], and [BeC90b].

One can show a pseudopolynomial worst-case bound on the running time of the primal-dual method. The (practical) average running time of the method, however, is much better than the one suggested by this bound. It is possible to convert the algorithm to a polynomial one by using scaling procedures; see [EdK72] and [BlJ85]. Unfortunately, these procedures do not seem to improve the algorithm's performance in practice.

Despite the fundamentally different principles underlying the simplex and primal-dual methods (primal cost versus dual cost improvement), these

methods are surprisingly related. It can be shown that the big-M version of the simplex method with a particular pivot selection rule is equivalent to the steepest ascent version of the primal-dual method [Zad79]. This suggests that the simplex method with the empirically best pivot selection rule should be more efficient in practice than the primal-dual method. Computational experience tends to agree with this conjecture. However, in many practical contexts, the primal-dual method has an advantage: it can easily use a good starting flow and price vector pair, obtained for example from the solution of a slightly different problem by modifying some of the arc flows to satisfy CS; this is true of all the methods of this chapter. Simplex methods are generally less capable of exploiting such prior knowledge; see also the discussion on sensitivity analysis in Section 5.5.

3.3. The relaxation method was first proposed in the context of the assignment problem by the author in [Ber81]. Its extension to the general minimum cost flow problem was given in [Ber82b]. References [BeT85] and [Tse86] consider the case where the problem data are noninteger. The relaxation method has been extended to network flow problems with gains ([BeT85] and [Tse86]), to general linear programs ([Tse86] and [TsB87a]), to network flow problems with convex arc cost functions [BHT87], and to monotropic programming problems [TsB87b]. When the arc cost functions are strictly convex, the method is particularly well suited for parallel implementation; see [BeE87a], [BHT87], [ElB89], [ChZ90], and [TBT90].

Extensive computational experience shows that the relaxation method typically outperforms primal-dual methods substantially for general minimum cost flow problems. In fact, primal-dual methods can often be speeded up considerably by initialization with a number of single-node relaxation iterations, although not to the point of challenging the relaxation method. The comparison between the relaxation method and simplex methods is less clear, although the relaxation method seems much faster for randomly generated problems. The relaxation method is also more capable of exploiting prior knowledge about an optimal solution; this advantage is shared with the primal-dual method. On the other hand, in contrast with the simplex method, the relaxation method requires that the problem data be integer; modified versions that can handle noninteger problem data ([BeT85] and [Tse86]), need not terminate, although they yield optimal solutions asymptotically.

3.4. The data structures for implementation of primal-dual methods briefly discussed in this section were proposed in [AaM76], and were used in the construction of an efficient out-of-kilter code. They are well suited for most types of dual ascent methods.