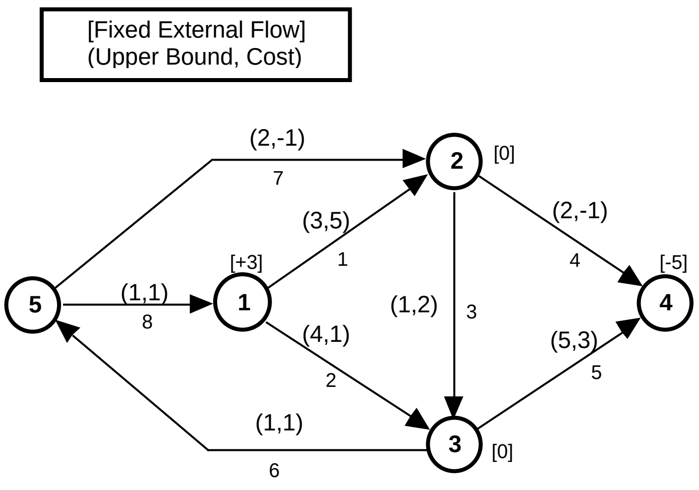
Relation of Pure Minimum Cost Flow Model to Linear Programming

The Network Model

The network pure minimum cost flow model has *m* nodes. The external flows given by the vector **b** with *m* -1 elements. The network has *n* arcs with parameter vectors **u** and **c**, and the flow variable **x**. The model is conveniently described by either the graphical form illustrated on the left of Fig. 1 or the vector description illustrated on the right. Implicit in both representations is the criterion for optimization, to minimize total cost, the requirement for conservation of flow at the nodes, and the restriction of flows between zero and the upper bounds.



I = {1, 2, 3, 4, 5}

b = [3, 0, 0, –5]

K= {1, 2, 3, 4, 5, 6, 7, 8}

o = [1, 1, 2, 2, 3, 3, 5, 5]

t = [2, 3, 3, 4, 4, 5, 2, 1]

u = [3, 4, 1, 2, 5, 1, 2, 1]

c = [5, 1, 2, –1, 3, 1, –1, 1]

Figure 1. Example of network model

Linear Programming Model

The minimum cost network flow problem is a special case of the linear programming problem. The solution algorithms described in this book are based on the primal simplex algorithm for linear programming. To determine optimality conditions it is necessary to provide both the primal and dual linear programming models for the network flow problem.

The Primal Model

The algebraic model expresses the objective function and constraints explicitly using linear functions of the flow variables. The result is a mathematical programming model.

Objective function

$$\text{Minimize } z = \sum_{k=1}^{n} c_k x_k \tag{1}$$

subject to

Conservation of flow

$$\sum_{k e \mathbf{K}_{Oi}} x_k - \sum_{k e \mathbf{K}_{Ti}} x_k = b_i, \quad i = 1, 2, \ldots, m-1 \tag{2}$$

Upper bounds on flow and nonnegativity

$$0 \le x_k \le u_k, \quad k = 1, 2 \ldots n \tag{3}$$

The model for the network of Fig. 1 provides a specific example.

$$\text{Min } x = 5x_1 \;+ 1x_2 \;+2x_3 \;-1x_4 \;+3x_5 \;+1x_6 \;-1x_7 \;+1x_8$$

subject to

$$\begin{array}{rrrrrrrrcr}
x_1 & +x_2 & & & & & & -x_8 & = & 3 \\
-x_1 & & x_3 & +x_4 & & & -x_7 & & = & 0 \\
 & -x_2 & -x_3 & & +x_5 & +x_6 & & & = & 0 \\
 & & & -x_4 & -x_5 & & & & = & -5 \\
x_1 & & & & & & & & \le & 3 \\
 & x_2 & & & & & & & \le & 4 \\
 & & x_3 & & & & & & \le & 1 \\
 & & & x_4 & & & & & \le & 2 \\
 & & & & x_5 & & & & \le & 5 \\
 & & & & & x_6 & & & \le & 1 \\
 & & & & & & x_7 & & \le & 2 \\
 & & & & & & & x_8 & \le & 1
\end{array}$$

$$x_k \ge 0 \quad \text{for } k = 1, \ldots, n$$

From the example and from the general expression, we see that, except for the arcs originating or terminating at the slack node, each arc has two nonzero coefficients in the conservation of flow constraints, a 1 in the row of its origin node, and –1 in the row of its terminal node. Arcs that originate at the slack node have a single –1 in their columns, while arcs that terminate at the slack node have a single +1. Let the matrix **A** represent the coefficients of the conservation of flow constraints. For the example

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \end{bmatrix}.$$

A is called the incidence matrix of the network. The linear programming model for the minimum cost flow problem can now be written in a matrix format using the vectors previously defined.

$$\text{Minimize } z = \mathbf{c\,x}$$
$$\text{subject to } \quad \mathbf{A\,x = b}$$
$$\mathbf{0 \le x \le u}$$

Since the network flow problem has a linear programming model, it may be solved by general linear programming algorithms. The fact that the **A** matrix always has the form shown above allows significant simplifications of the linear programming algorithms and thus much greater efficiencies of solution.

For the pure network flow model, the matrix **A** consists of only +1, –1, and 0. Because of the arrangement of the nonzero coefficients in the matrix, it can be shown that the pure network flow problem with integer parameters will always have an integer optimum solution when solved as a linear program. This problem has the characteristic of total unimodularity.

The Dual Model

The dual solution of the network flow problem plays an important role in the solution algorithm. In order to form the dual, assign the dual variable π_i to the conservation of flow constraint for node *i*, and the dual variable δ_k to the upper bound constraint for arc *k*. The dual model for the network flow programming problem becomes

Dual Problem:

$$\text{Minimize } z_D = \sum_{i=1}^{m1} \pi_i b_i + \sum_{k=1}^{n} \delta_k u_k \tag{4}$$

$$\text{subject to} \quad \pi_i - \pi_j + \delta_k \ge -c_k \quad \text{for all } k(i, j) \quad K \tag{5}$$

$$\pi_m = 0, \ \pi_i \text{ unrestricted for all } i \quad I \tag{6}$$

$$\delta_k \ge 0 \text{ for all } k(i, j) \quad K. \tag{7}$$

Conditions for Optimality

From the duality theory of linear programming, we have conditions for optimality for the primal and dual solutions. Given solutions to the primal and dual problems, we are assured that they are optimal for their respective problems if they are feasible and satisfy the complementary slackness conditions. We will not derive the optimality conditions, however, the form that we use in this chapter is shown below. Assume we have solution **x** for the primal problem and solution π for the dual problem.

For simplicity define for the arcs:

$$d_k = \pi_i - \pi_j + c_k \text{ for all } k(i, j) \quad K$$

The solutions **x** and π are optimal if the following conditions are satisfied.

1. *Primal feasibility*

   a. **x** provides conservation of flow at all nodes except slack node (8)

   b. $0 \le x_k \le u_k$ for all arcs (9)

2. *Complementary Slackness*

   For each arc k:

   a. if $0 < x_k < u_k$, then $d_k = 0$ (10)

   b. if $x_k = 0$ then $d_k \ge 0$ (11)

   c. if $x_k = u_k$ then $d_k \le 0$ (12)

These conditions are fundamental for the solution algorithms. The conditions do not use the dual variables, δ_k. These values are determined by

$$\delta_k = \max\{0, -d_k\} \text{ for each arc } k \qquad (13)$$

With this definition, we assure that for any value of π, the dual solution is feasible.

Basic Solutions and the Primal Simplex

Using the notation of the general linear programming, the matrices defining the example problem are

$\mathbf{x} = [\ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\ ]^T$

$\mathbf{u} = [\ 3,\ 4, 1,\ 2, 5, 1\ 2, 1\ ]^T$

$\mathbf{c} = [\ 5, 1, 2, -1, 3,\ 1, -1, 1\ ]$

$\mathbf{b} = [\ 3,\ 0,\ 0, -5\ ]^T$

$$\mathbf{A} = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}\begin{array}{cccccccc} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 & \mathbf{a}_5 & \mathbf{a}_6 & \mathbf{a}_7 & \mathbf{a}_8 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \end{array}$$

The Basis and Basis Inverse

For linear programming the basis defines the basis matrix **B** formed by selecting the columns of **A** associated with the elements of the basis vector. For the basis: $\mathbf{n}_B = [8, 7, 2, 5]$

$$\mathbf{B} = \begin{array}{cccc} \mathbf{a}_8 & \mathbf{a}_7 & \mathbf{a}_2 & \mathbf{a}_5 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{array}$$

We have ordered the columns of the basis so that the matrix has an upper diagonal form. This is always possible to do when the basis forms a spanning tree.

Although generally it is fairly difficult to find the inverse of an arbitrary matrix, it is easy to find the inverse associated with the basis of a pure network flow problem. The inverse can be found by observation from the directed spanning tree defining the basis using the following rules.

a. Identify the rows of $\mathbf{B}^{-1}$ with the arcs in $\mathbf{n}_B$ and the columns with the nodes 1 to m–1 of the network.

b. For each node i, construct column i of $\mathbf{B}^{-1}$ by finding the path from the slack node to node i in the basis tree. Entry (k, i) of the basis inverse is –1 if arc k is on the path, +1 if arc -k is on the path, otherwise it is zero.

The information required from the basis inverse is immediately available from the paths in the tree defining the basis. The computational procedures of network flow programming will use the tree and not the inverse of the basis.

Observing the paths from the basis tree, we construct the basis inverse.

$$\mathbf{B}^{-1} = \begin{array}{cccc|c} \text{Node: } 1 & 2 & 3 & 4 & \text{Arc} \\ -1 & 0 & -1 & -1 & 8 \\ 0 & -1 & 0 & 0 & 7 \\ 0 & 0 & -1 & -1 & 2 \\ 0 & 0 & 0 & -1 & 5 \end{array}$$

Matrix multiplication verifies for the example that $\mathbf{BB}^{-1} = \mathbf{I}$.

Basic Solution

The collection of columns of the **A** matrix associated with the set $\mathbf{n}_0$ is the matrix $\mathbf{N}_0$, and the collection of columns of the **A** matrix associated with the set $\mathbf{n}_1$ is the matrix $\mathbf{N}_1$. The objective coefficients **c** are partitioned into the vectors $\mathbf{c}_B$, $\mathbf{c}_0$, and $\mathbf{c}_1$ according to the sets $\mathbf{n}_B$, $\mathbf{n}_0$, and $\mathbf{n}_1$. Similarly the upper bounds **u** are partitioned into the arrays $\mathbf{u}_B$, $\mathbf{u}_0$, and $\mathbf{u}_1$.

From the conservation of flow constraints we have $\mathbf{Ax} = \mathbf{b}$.

Partitioning the matrices into basic variables and the two kinds of nonbasic variables:

$$\mathbf{Bx}_B + \mathbf{N}_0\mathbf{x}_0 + \mathbf{N}_1\mathbf{x}_1 = \mathbf{b}.$$

Solving for the basic variables while substituting $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{x}_1 = \mathbf{u}_1$ we have

$$\mathbf{x}_B = \mathbf{B}^{-1}[\mathbf{b} - \mathbf{N}_1\mathbf{u}_1].$$

For the network model, the terms in brackets represent the vector of adjusted external flows equivalent to Eq. (1). An individual component of $\mathbf{x}_B$ multiplies this vector by a row of $\mathbf{B}^{-1}$. From the previous discussion this row represents an arc with entries –1 in column i if the arc is on the path to node i and 0 if the arc is not on the path. Thus the matrix multiplication sums the adjusted external flows for every node containing the arc in its path to the slack node .

For the example, $\mathbf{n}_B = [8, 7, 2, 5]$ so

$$\begin{bmatrix} x_8 \\ x_7 \\ x_2 \\ x_5 \end{bmatrix} = \mathbf{B}^{-1}[\mathbf{b} - \mathbf{N}_1\mathbf{u}_1] = \begin{bmatrix} -1 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ 0 \\ -3 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 3 \\ 3 \end{bmatrix}$$

For the general linear program, we compute the dual variables from the expression

$$\pi = -\mathbf{c}_B\mathbf{B}^{-1}$$

A negative sign appears in this expression because the network problem is stated as a minimization. For the network problem, the *i*th column of $\mathbf{B}^{-1}$ is a vector representing the arcs on the path from the slack node to node *i* (–1 indicates that the arc is on the path, +1 indicates that the mirror arc is on the path, and 0 indicates that the arc is not part of the path). Thus the equation for π_i simply sums the arc costs on the path to node *i*.

Using the matrices for the example we have

$$\begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{bmatrix} = -\mathbf{c}_B\mathbf{B}^{-1} = -(1, -1, 1, 3)\begin{bmatrix} -1 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

$$\pi = [\ 1\ \ -1\ \ 2\ \ 5\ ]\ .$$

The computation of the reduced cost, d_k, comes directly from the operation for computing the marginal costs for the general linear program.

$$d_k = c_k + \pi\mathbf{a}_k$$

where $\mathbf{a}_k$ is the *k*th column of **A**, shown again for the example below

$$\mathbf{A} = \begin{array}{c} \begin{array}{cccccccc} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 & \mathbf{a}_5 & \mathbf{a}_6 & \mathbf{a}_7 & \mathbf{a}_8 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

The Primal Simplex

The first step of the primal simplex is to compute the reduced cost of the nonbasic variables. The computation of the reduced cost, d_k, comes directly from the operation for computing the marginal costs for the general linear program.

$$d_k = c_k + \pi\ \mathbf{a}_k$$

where $\mathbf{a}_k$ is the *k*th column of **A**, shown again for the example below

$$\begin{array}{cc} & \begin{array}{cccccccc} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 & \mathbf{a}_5 & \mathbf{a}_6 & \mathbf{a}_7 & \mathbf{a}_8 \end{array} \\ \mathbf{A} = & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

For the special case of the network, the column $\mathbf{a}_k$, representing the arc $k(i, j)$, has at most two entries a +1 in row i and a –1 in row j. The matrix multiplication $\pi\mathbf{a}_k$ involves only two terms of the vector of dual variables: $\pi_i - \pi_j$. The general case then specializes for the network problem into the simpler form

$$d_k = c_k + \pi_i - \pi_j.$$

For the general linear program, the arc to leave the basis is found by computing the vector $\mathbf{y}_k = \mathbf{B}^{-1}\mathbf{a}_k$, where $\mathbf{a}_k$ is the column of **A** for the entering variable. For the network model $\mathbf{a}_k$, representing arc $k(i, j)$, has only two nonzero entries, +1 in row i and –1 in row j. Recall the ith column of $\mathbf{B}^{-1}$ describes the path from the source node to node i in the basis tree with an element equal to +1 if the arc is traversed in the mirror direction and –1 if it is traversed in the forward direction. Let $\mathbf{p}_i$ be the column of $\mathbf{B}^{-1}$ describing this path for node i. Similarly $\mathbf{p}_j$ describes the path to node j. Then the computation

$$\mathbf{y}_k = \mathbf{B}^{-1}\mathbf{a}_k = \mathbf{p}_i - \mathbf{p}_j.$$

When paths $\mathbf{p}_i$ and $\mathbf{p}_j$ have elements in common, they cancel out in the expression above and $\mathbf{y}_k$ simply indicates the arcs on the cycle formed by the entering arc. A 0 element indicates that the basic arc is not on the cycle, a –1 indicates that it is traversed in the forward direction, and a +1 indicates that it is in the mirror direction.

The vector $\mathbf{y}_k$ is used for the familiar ratio test that selects the variable to leave the basis for linear programming. When the arc flows and upper bounds are integer, $\mathbf{y}_k$ will always prescribe an integer value of Δ_f. When the initial flows are integer, the flows in subsequent iterations remain integer. This is further justification, that all basic solutions will be integer, an important result for many applications.

For general linear programming a basis change is accomplished by changing the contents of the set $\mathbf{n}_B$ and recomputing the inverse of the basis. It is unnecessary to compute the basis inverse for the network problem, because the information associated with the basis inverse is obtained directly from the basis tree.