Network Methods.S1
Pure Minimum Cost Flow

Networks are especially convenient for modeling because of their simple nonmathematical structure that can be easily depicted with a graph. This simplicity also reaps benefits with regard to algorithmic efficiency. Although the simplex method of linear programming is one of the primary solution techniques, we have already seen that its implementation for network problems allows many procedural simplifications. Small instances can be solved by hand and computer programs are available for solving very large instances much faster than standard simplex codes. This section provides a solution algorithm for the pure network flow programming problem on a directed graph. The procedure for solving a generalized model that includes arc gains is similar but more complicated.

Problem Statement

A pure network flow minimum cost flow problem is defined by a given set of arcs and a given set of nodes, where each arc has a known capacity and unit cost and each node has a fixed external flow. The optimization problem is to determine the minimum cost plan for sending flow through the network to satisfy supply and demand requirements. The arc flows must be nonegative and be no greater than the arc capacities, and they must satisfy conservation of flow at the nodes. For this section, it is assumed that all arc lower bounds on flow are zero and all arc gains are unity. The algorithms of this section can be extended to handle generalized networks, where not all gains are unity, but the complexity is increased and the discussion is beyond the scope of this book.

Formulation

We now formulate the problem as a linear program for a directed, connected graph with *m* nodes and *n* arcs. It is assumed that there is at least one supply node and one demand node. Regarding notation, recall that in the formulation of the transportation problem in Section 7.1, the arc connecting nodes *i* and *j* was denoted by (i, j) and the decision variable associated with that arc was x_{ij}. Here we adopt the notation $k(i,j)$, or simply *k*, to denote the arc between nodes *i* and *j*. This is done primarily to simplify the algorithmic presentation and the labeling of arcs in the figures to follow. Accordingly, the decision variables are

$$x_k = \text{flow through arc } k(i, j) \text{ from node } i \text{ to node } j$$

and the given data are

$$c_k = \text{unit cost of flow through arc } k(i, j)$$

b_i = net supply (arc flow out – arc flow in) at node *i*

u_k = capacity of arc $k(i,j)$

The value of b_i is determined by the nature of node *i*. In particular,

$b_i > 0$ if *i* is a supply node;

$b_i < 0$ if *i* is a demand node;

$b_i = 0$ if *i* is a transshipment node.

Also, let

$\boldsymbol{K}_{O_i}$ = set of arcs leaving node *i*

$\boldsymbol{K}_{T_i}$ = set of arcs terminating at node *i*.

The mathematical model is

$$\text{Minimize } \sum_{k=1}^{n} c_k x_k \qquad (4a)$$

$$\text{subject to } \sum_{k \;\; \boldsymbol{K}_{O_i}} x_k \; - \sum_{k \;\; \boldsymbol{K}_{T_i}} x_k \; = b_i \;\; \text{for all } i = 1\ldots m \qquad (4b)$$

$$0 \le x_k \le u_k \text{ for all } k = 1\ldots n \qquad (4c)$$

The objective function (4a) sums the arc costs over all arcs in the network. Equation (4b) describes the flow balance or conservation of flow constraints. The first summation represents the flow out of node *i*, and the second is the flow in. The difference between the two represents the net flow generated at node *i*. The right side values b_i are positive if node *i* supplies flow, negative if it consumes flow and zero otherwise. In some applications, the lower bound in (4c) is not zero but some arbitrary value l_k. As we saw in the network modeling chapter, it is always possible to transform such a lower bound to zero by introducing the variable

$$\hat{x}_k = x_k - l_k \text{ and substituting } \hat{x}_k + l_k \text{ for } x_k$$

throughout the model. The upper bound must also be changed to

$$\hat{u}_k = u_k - l_k.$$

In the chapter on network modeling, models could have arcs with only originating or terminating nodes. Arcs 6, 7 and 8 in Fig. 24 are examples. The arcs represent variable external flows at nodes 3, 2 and 1 respectively. For theoretical and algorithmic purposes we introduce an additional node into the network called the slack node. Arcs representing the variable external flows originate or terminate at the slack node as shown in Fig. 25. Now all arcs have both originating and terminating nodes. Generally we assign the index *m* to the slack node.

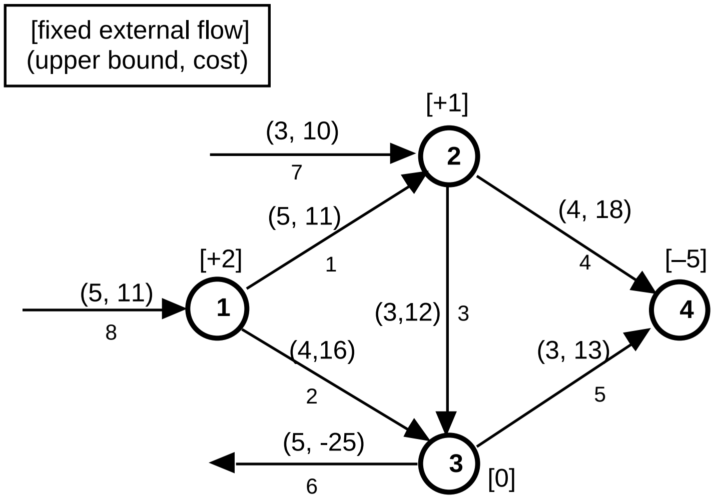


Figure 24. A model with variable external flows

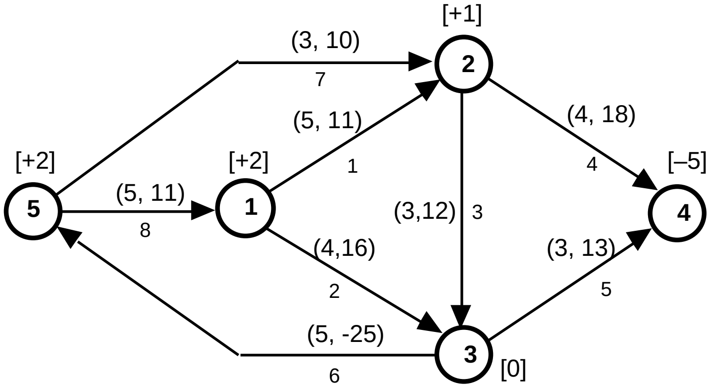


Figure 25. A model with a slack node

For a pure network problem, flows are conserved in the arcs, so for a feasible solution the external flows at the nodes must sum to zero.

Feasibility property: A necessary condition for a pure minimum cost flow problem to have a feasible solution is

$$\sum_{i=1}^{m} b_i = 0.$$

In other words, the total flow being supplied at the nodes in the network must equal the total demand being absorbed by the nodes in the network.

The feasibility property will hold if we assign the external flow of the slack node to be

$$b_m = -\sum_{i=1}^{m-1} b_i \, .$$

The slack node external flow has been assigned in Fig. 25.

Because of the feasibility property, when we add the *m* conservation of flow equations in 4b, we obtain 0 on both sides of the equal sign. This means that one of the equations is redundant. In fact, there are exactly $m - 1$ independent equations, and one of the equations may be arbitrarily discarded without changing the solution to the problem.

The feasibility property is necessary, but not sufficient. Whether or not a particular instance of model (4a) - (4c) has a feasible solution depends on the network structure, arc capacities, and supply and demand values.

In many applications, the flow on the arcs must take integer values. This is implicitly the case for the assignment problem where one worker, for example, must be assigned to a single job. In all the network examples worked out earlier in the chapter we saw that the solutions were always integral. This was not a coincidence but a direct result of the following property.

> **Integrality property**: For the pure minimum cost flow problem (4), when all supply and demand values b_i are integer and all upper bounds on arc flows u_k are integer, all basic solutions are integral.

The algebraic implication of this statement is that every basis inverse matrix has components that are either 0, 1 or –1. If the standard simplex algorithm were applied to problem (4) we would see that every pivot element has a value of ±1 so fractions never appear in the tableau. In our adaptation of the simplex method for solving problem (4), we will see that when the integrality property holds the decision variables similarly never become fractional.

Example

We use the network in Fig. 25 as an example. Upper bounds and costs are given for each arc as shown in the figure. It is convenient to identify vectors that represent the parameters of the problem. They are respectively, **x**, for arc flows, **u**, for arc flow capacities or upper bounds, **c**, for arc unit costs, and **b**, for node external flow. For the vectors for the example are below.

$$\begin{array}{rcl}
\text{Arc flows: } \mathbf{x} & = & [\ x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8\ ] \\
\text{Upper bounds: } \mathbf{u} & = & [\ 5,\ 4,\ 3,\ 4,\ 3,\ 5,\ 3,\ 5\ ] \\
\text{Unit costs: } \mathbf{c} & = & [\ 11,\ 16,\ 12,\ 18,\ 13,\ -25,\ 10,\ 11\ ] \\
\text{External flows: } \mathbf{b} & = & [\ 2,\ 1,\ 0,\ -5,\ 2\ ]
\end{array}$$

Basic Solutions

Because an optimal solution must be among the finite set of bases, the simplex algorithm only examines basic solutions as it iterates. When applied to the minimum cost flow problem, the algorithm maintains the needed information more efficiently and hence is able to access it more quickly than when applied to a general linear programming problem. This section describes basic solutions for the network flow programming problem and provides procedures for computing the primal and dual solutions associated with a given basis.

Basis Tree

We know that an optimal solution to the linear program, and to the network flow problem by extension, is a basic solution. The basis is defined by a selection of independent variables equal in number to the number of linearly independent constraints. Since the network model contains $m - 1$ independent conservation of flow constraints and the variables are the arc flows, a basis is determined by selecting $m - 1$ independent arcs. These are identified by the arc indices of the basic variables. Call the corresponding set $\mathbf{n}_B$. For the pure minimum cost flow problem, we have the interesting characteristic that every basis defines a spanning tree subnetwork. To illustrate, let $\mathbf{n}_B = [2, 4, 7, 8]$ for the network of Fig. 25. Drawing only the selected arcs forms the subnetwork shown in Fig. 26. Node 5 is defined as the root node of the tree.

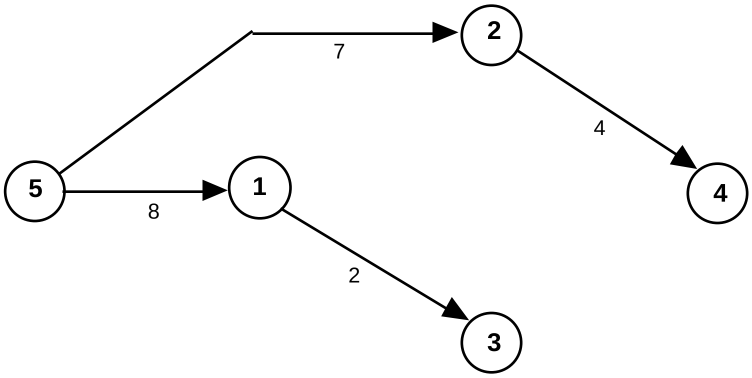


Figure 26. Basis tree for $\mathbf{n}_B = [2, 4, 7, 8]$

Nonbasic Arcs

The arcs not selected as basic are, by definition, the nonbasic arcs. In the *bounded variable simplex method*. of general linear programming, nonbasic variables take the value 0 or the value of their upper bound when defining a basic solution. We adapt this method to the network flow problem.

In a basic solution, each nonbasic arc *k*, has its flow at 0 or u_k, the upper bound. Let $\mathbf{n}_0$ denoted the set of nonbasic arcs with 0 flow, and let $\mathbf{n}_1$ denoted the set of arcs with upper bound flow. To represent a specific case graphically we add the members of $\mathbf{n}_1$ to the basis tree as dotted lines. To illustrate, Fig. 27 shows the basis tree representing $\mathbf{n}_B$ = [2, 4, 7, 8], $\mathbf{n}_1$ = [5] and $\mathbf{n}_0$ = [1, 3, 6].

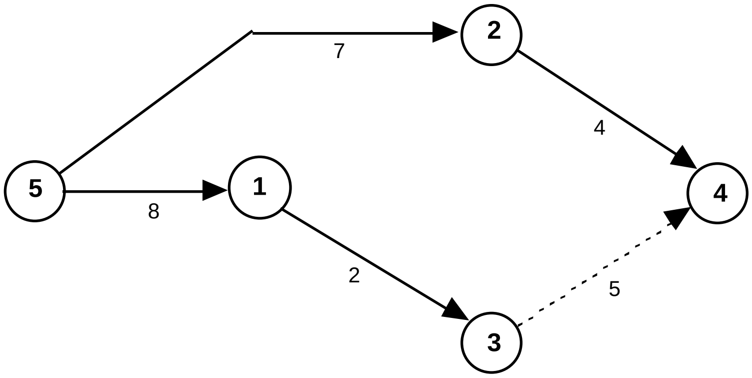


Figure 27. Basis tree with a nonbasic arc with a flow at its bound

Primal Basic Solution

Given a selection of basic arcs and an assignment of nonbasic arcs to either $\mathbf{n}_0$ or $\mathbf{n}_1$, there is a unique assignment of flows to the basic arcs that satisfies the conservation of flow requirement at the nodes. Let $\mathbf{x}_B$ be the vector of flows on the basic arcs and $\mathbf{x}_1$ be the flows on the arcs in $\mathbf{n}_1$. The flows on the arcs in $\mathbf{n}_0$ are zero.

To solve for the basic arc flows we must first adjust the external flows for the flows in the nonbasic arcs at their upper bounds. One way to do this is to take each member of $\mathbf{n}_1$ in turn. Say arc $k(i, j)$ is at its upper bound. To accommodate the flow on arc $k(i, j)$ for the following calculations, we reduce the external flow at its origin node *i* by u_k and increase the external flow at its terminal node *j* by u_k. For the network in Fig. 25, we the external flow vector:

$$\mathbf{b} = [2,\ 1\ ,\ 0, -5\ , 2].$$

Arc 5 is in the set $\mathbf{n}_1$, so we adjust the external flows at both ends of the arc: $b_3' = b_3 - u_5$, $b_4' = b_4 + u_5$. Since $u_5 = 3$, the adjusted external flows are

$$\mathbf{b}' = [2,\ 1\ ,\ -3, -2\ \ , 2]$$

Equation (5) provides a general expression for the adjusted external flow for node i, b_i'; that is, the original external flow reduced by the flow of the upper bound arcs leaving the node and increased by the flow on the upper bound arcs entering the node. Once again, $\boldsymbol{K}_{O_i}$ is the set of arcs leaving node i and $\boldsymbol{K}_{T_i}$ is the set of arcs terminating at node i.

$$b_i' = b_i - \sum_{k \in (\boldsymbol{K}_{O_i} \cap \mathbf{n}_1)} u_k + \sum_{k \in (\boldsymbol{K}_{T_i} \cap \mathbf{n}_1)} u_k \qquad (5)$$

Given the adjusted external flows, there is a unique assignment of flows to the basic arcs that satisfies conservation of flow at the nodes. The solution for the basis in Fig. 27 is shown in Fig. 28. Comparison of the arc flows to the upper and lower bounds for the basic arcs will show that the flows fall within these limits. This is a feasible, primal basic solution.

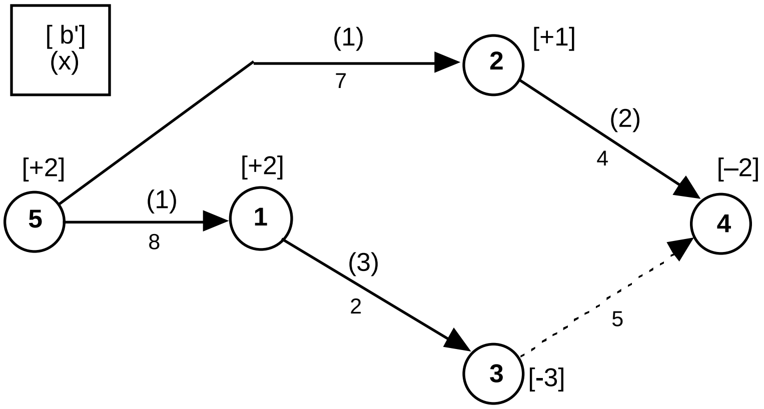


Figure 28. Solution for the basic flows

$\mathbf{b} = [2,\ 1\ ,\ 0, -5\ , 2].$

$\mathbf{b}' = [2,\ 1\ ,\ -3, -2\ \ , 2]$

$\mathbf{n}_B = [2, 4, 7, 8]$

$\mathbf{n}_1 = [5]$

$\mathbf{n}_0 = [1, 3, 6]$

$\mathbf{x}_B = [3, 2, 1, 1]$

$\mathbf{x}_1 = [3]$

$\mathbf{x}_0 = [0, 0, 0]$

Dual Basic Solution

Dual variables, or alternatively the dual values, are used explicitly in the solution algorithm for network flow programming. There is a dual variable for each node i, denoted by π_i and interpreted as the cost of bringing one unit of flow to node i from the slack node. Given a basis tree, the dual values are assigned using the requirement of complementary slackness. For every basic arc $k(i, j)$,

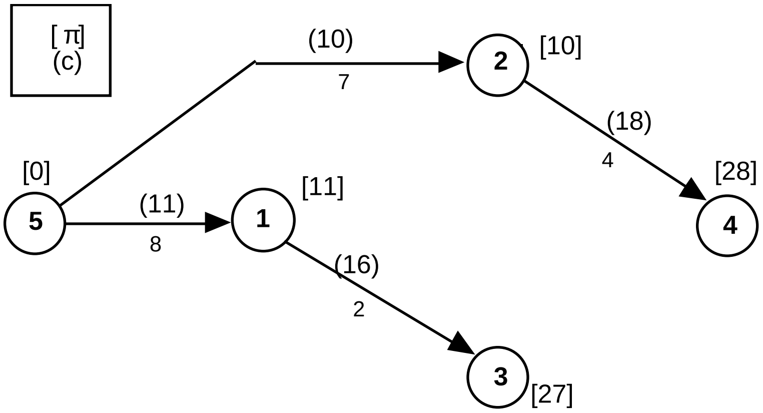
$$c_k + \pi_i - \pi_j = 0$$

Since one of the conservation of flow constraints the linear programming model is redundant, one of the dual variables is arbitrary. We choose to set the dual value for the slack variable, node 5 for the example to 0. The other dual variables are set to values that satisfy the complementary slackness conditions. The values are computed in the order shown below.

$$\pi_5 = 0, \pi_1 = \pi_5 + 11 = 11, \pi_3 = \pi_1 + 16 = 27,$$

$$\pi_2 = \pi_5 + 10 = 10, \pi_4 = \pi_2 + 18 = 27.$$

There are a number of different orders that can be followed in this computation, but they all must start at the slack node and work out. Fig. 29 shows the basis tree previously considered in Fig. 27 with costs shown on the arcs and dual values shown adjacent to the nodes. The assignment of nonbasic arcs to $\mathbf{n}_0$ and $\mathbf{n}_1$ does not affect the value of π.



$\mathbf{n}_B = [2, 4, 7, 8]$

$\pi = [\pi_1, \pi_2, \pi_3, \pi_4, \pi_5]$
$= [11, 10, 27, 28, 0]$

Figure 29. Computation of dual variables

Solution Characteristics

Every selection of arcs, $\mathbf{n}_B$, that forms a spanning tree together with a specification of $\mathbf{n}_1$ determines primal and dual basic solutions for the network problem. The spanning tree determines a unique assignment of dual variables (π) that satisfies complementary slackness. The spanning tree and $\mathbf{n}_1$ determine in a unique primal solution ($\mathbf{x}$) that satisfies conservation of flow at each node.

The requirement of conservation of flow assures that conservation of flow is satisfied at each node.

$$\sum_{k \quad \boldsymbol{K}_{O_i}} x_k \; - \sum_{k \quad \boldsymbol{K}_{T_i}} x_k \; = b_i \quad \text{for all } i = 1 \ldots m.$$

The process of assigning arc flows does not assure that the flow solution is feasible, $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$. A basic solution may be feasible or infeasible. A solution for which either $x_k = 0$ or $x_k = u_k$ for some basic arcs is called *primal degenerate*. The solution of Fig. 28 is feasible because all the arc flows are nonnegative and no greater than the arc capacities. The solution is not primal degenerate.

There is a reduced cost, d_k, that can be computed for each arc in the network using the formula

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

We will use the reduced costs for determining the optimality of a basic solution.

The condition of complementary slackness assures that

$$d_k = 0 \quad \text{for all } k(i, j) \quad \mathbf{n}_{\mathrm{B}}$$

A solution for which $d_k = 0$ for some nonbasic arcs is called *dual degenerate*. For Fig. 29, the nonbasic arcs are 1, 3, 4, 6. Computing the values of d_k for these arcs we find: $d_1 = 12$, $d_3 = -5$, $d_5 = 12$, $d_6 = 2$. The solution is not dual degenerate.

Simplified Computation of Primal Solutions

The tree structure of the basis subnetwork makes possible simplified procedures for finding the primal and the dual basic solutions. Observe again the example basis in Fig. 30 and note that there is a directed path from node 5 to every other node in the tree. This is called a directed spanning tree rooted at node 5. Node 5 is called the root node.

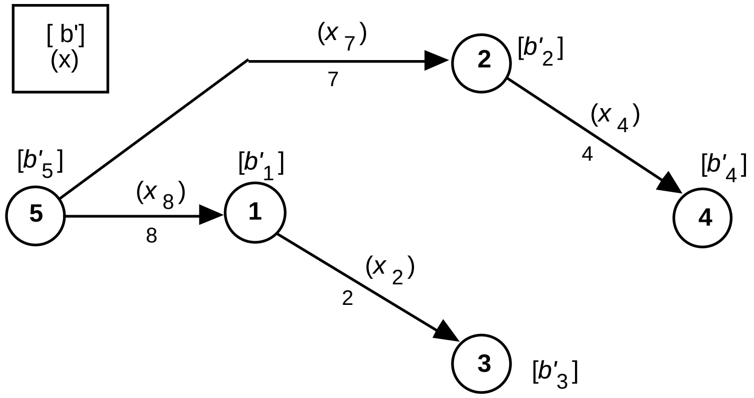


Figure 30. A directed spanning tree used to compute the primal solution

The basic flows are easily determined by assigning flows to the basic arcs in a sequential manner. At any time in the sequence there is enough information to assign one or more basic flows. The process starts at nodes at the extremes of the tree (the nodes incident to only one arc are called the *leaves* of the tree). We assign flow to the arcs incident to the leaves and work backward through the tree toward the root. The order in which arc flows are assigned is not unique, but for a given basis tree and set $\mathbf{n}_1$, there is a unique solution for the basic flows. For the example, we find two leaves of the tree in Fig. 30, nodes 3 and 4. We assign the flows to arcs 2 and 4 such that

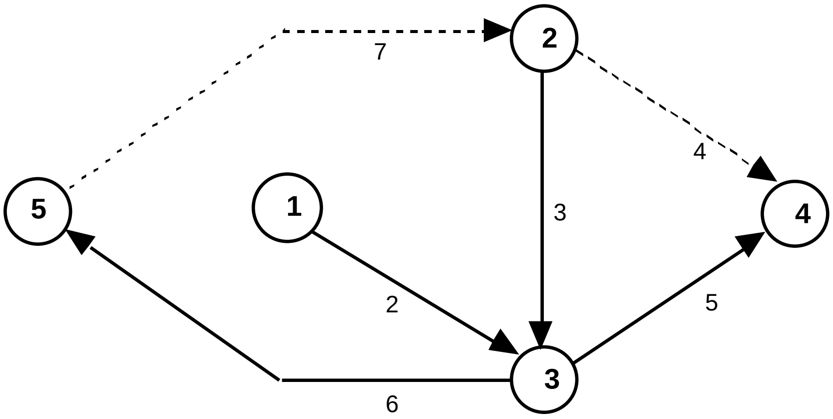
$$x_2 = -b'_3 = -(-3) = 3,\ x_4 = -b'_4 = -(-2) = 2.$$

Now the flows for arcs 7 and 8 can be assigned.

$$x_7 = -b'_2 + x_4 = -(1) + 2 = 1,\ x_8 = -b'_1 + x_2 = -(2) + 3 = 1.$$

This completes the determination of the basic flows.

The arcs selected for the basis of Fig. 30 naturally forms a directed spanning tree. Other selections may not. For instance consider the basis consisting of arcs 2, 3, 5 and 6 as shown in Fig. 31. Although the subnetwork defines a tree, it does not form a directed spanning tree. We accomplish this result by reversing the direction of some of the arcs as in Fig. 32.



$\mathbf{n}_B = [2, 3, 5, 6]$
$\mathbf{n}_1 = [4, 7]$
$\mathbf{n}_0 = [1, 8]$

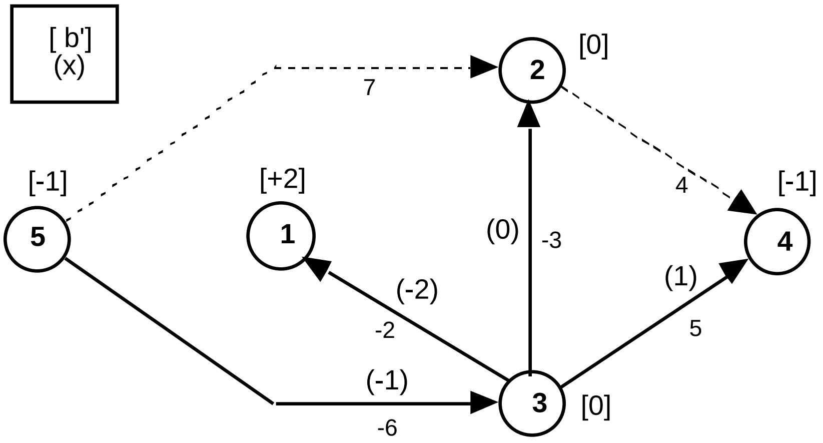
Figure 31. Solution for the basic flows

The reversed arcs are called mirror arcs, and they play a considerable roll in the minimum cost flow algorithm. We call the arc with a positive index a forward arc, and an arc with a negative index a mirror arc. The forward arc has the direction and parameters given in the problem statement. The parameters of the mirror arc are derived from the parameters of the forward arc as follows:

$$l_{-k} = -u_k,\ u_{-k} = -l_k,\ c_{-k} = -c_k, \text{ and } f_{-k} = -f_k.$$

It is always possible to construct a directed spanning tree by replacing some forward arcs with mirror arcs. The tree is shown in Fig. 32. The basic flows for this tree is easily obtained by following the tree backward from its leaves. The flows in the forward arcs are found by negating the flows in the mirror arcs.

For $\mathbf{n}_B = [2, 3, 5, 6]$, $\mathbf{x}_B = [2, 0, 1, 1]$.



$\mathbf{b} = [2,\ 1\ ,\ 0, -5\ , 2].$

$\mathbf{b}' = [2,\ 0\ ,\ 0, -1\ \ , -1]$

$\mathbf{n}_B = [-2, -3, 5, -6]$

$\mathbf{n}_1 = [4, 7]$

$\mathbf{n}_0 = [1, 8]$

$\mathbf{x}_B = [-2, 0, 1, -1]$

$\mathbf{x}_1 = [4, 3]$

$\mathbf{x}_0 = [0, 0]$

Figure 32. Solution for the basic flows

Simplified Computation of Dual Solutions

The directed spanning tree can also be used to compute the dual variables, but here we start at the root and work outward toward the leaves. We use basis of Fig. 33 to illustrate the approach. We continue to use the parameters of the example problem.

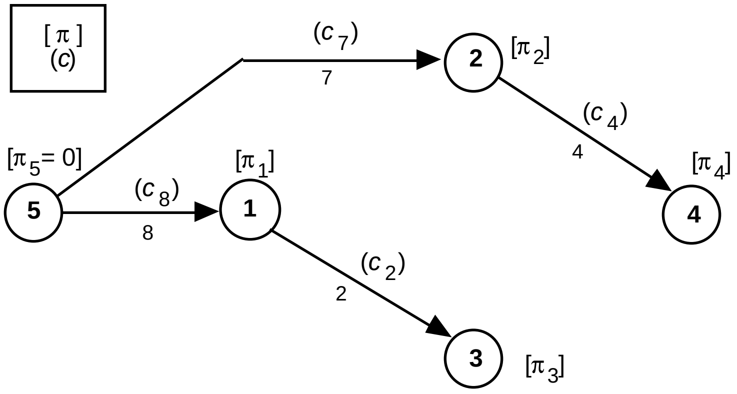


Figure 33. A directed spanning tree used to compute the dual solution

We begin by assigning the value of π_5 to 0. When the dual variable for a node on one end of a basic arc is known, the value for the node on the other end can be computed using the complementary slackness condition. Now we can compute the values of π_1 and π_2 using complementary slackness.

$$\pi_1 = \pi_5 + c_8 = 0 + 11 = 11, \pi_2 = \pi_7 + c_8 = 0 + 10 = 10.$$

Now the values of π_3 and π_4 can be calculated.

$$\pi_3 = \pi_1 + c_2 = 11 + 16 = 27, \pi_4 = \pi_2 + c_4 = 10 + 18 = 38.$$

The process of assigning the dual variables begins at the root node of the tree and progresses out toward the branches. At each step one more dual variable is assigned. The direction of the assignment process is the reverse of the procedure for assigning the primal variables. The nature of the spanning tree assures that the values of all the dual variables can be computed in this manner.

When the basic arcs do not naturally define a spanning tree, mirror arcs are used to construct one. For $\mathbf{n}_B$ = [2, 3, 5, 6], we replace arcs 2, 3, and 6 with mirror arcs. The relation, $c_{-k} = -c_k$, provides the unit costs for the mirror arcs. The spanning tree and the computed dual variables are in Fig. 34.

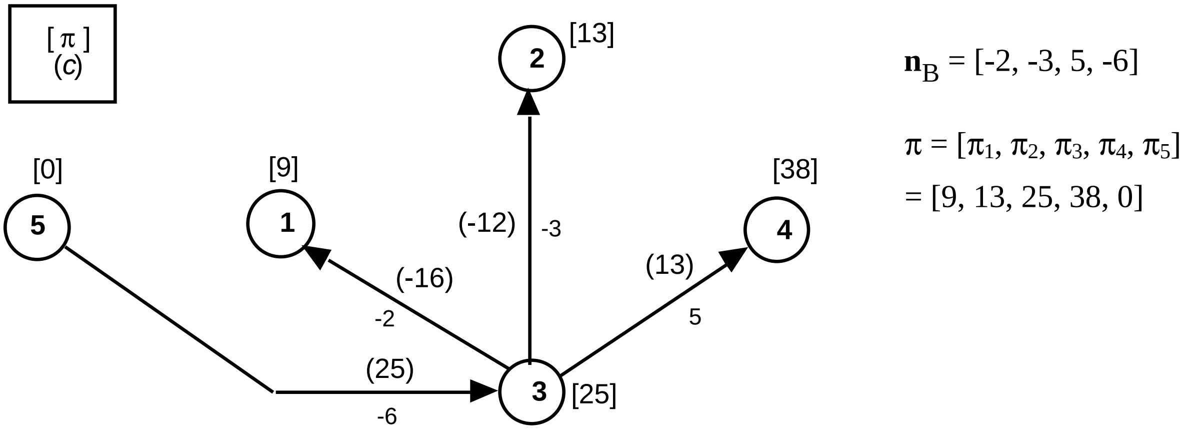


Figure 34. Solution for the dual variables

Optimality Conditions

The minimum cost flow problem is to find solutions **x** and π to the primal and dual problems, respectively, that satisfy the following optimality conditions.

1. *Primal feasibility*

   a. **x** provides conservation of flow at all nodes except the slack node and is basic;

   b. $0 \le x_k \le u_k$ for all arcs k.

2. *Complementary slackness*

   For each arc k, $d_k = c_k + \pi_i - \pi_j$ is the reduced cost, and

   a. if $0 < x_k < u_k$, then $d_k = 0$;

   b. if $x_k = 0$, then $d_k \ge 0$; or

c. if $x_k = u_k$, then $d_k \leq 0$.

There is no guarantee that every arc flow in a basic solution is within the range defined condition (1b). Primal solutions are termed feasible when all flows are within the bounds and infeasible when some flows are outside.

A primal solution is degenerate when a basic flow is either at zero or at the upper bound for the arc. Degeneracy may affect the progress of the solution algorithm toward the optimum.

Condition (2a) is only possible for basic variables; the method that we use for computing the values of π assures that it is satisfied. When one of the last two conditions (2b) or (2c) holds for every nonbasic arc, both primal and dual solutions are optimal. We describe presently an algorithm that constructively finds **x** and π satisfying both primal feasibility and complementary slackness.

Examples

We illustrate the optimality conditions using the two examples previously considered. Both are primal feasible and satisfy both parts of condition 1. Condition 2a is satisfied for the basic variables, so the solution is optimal if 2b or 2c is satisfied for each nonbasic arc. For the basic solution illustrated in Figs. 28 and 29 we compute the results below.

$\mathbf{n}_B = [2, 4, 7, 8]$	Nonbasic arcs with flows at the lower bound: $\mathbf{n}_0 = [1, 3, 6]$
$\mathbf{x}_B = [x_2, x_4, x_7, x_8]$	$d_1 = c_1 + \pi_1 - \pi_2 = 11 + 11 - 10 = 12$: Satisfies 2b
$= [3, 2, 1, 1]$	$d_3 = c_3 + \pi_2 - \pi_3 = 12 + 10 - 27 = -5$: Violates 2b
	$d_6 = c_6 + \pi_3 - \pi_5 = -25 + 27 - 0 = 2$: Satisfies 2b
$\pi = [\pi_1, \pi_2, \pi_3, \pi_4, \pi_5]$	Nonbasic arcs with flows at the upper bound: $\mathbf{n}_1 = [5]$
$= [11, 10, 27, 28, 0]$	$d_5 = c_5 + \pi_3 - \pi_4 = 13 + 27 - 28 = 12$: Violates 2c

The solution is evidently not optimum because arcs 3 and 6 violate the optimality conditions. In fact, these two arcs are candidates to enter the basis.

For the basic solution illustrated in Figs. 32 and 34 we compute the results below. Since all the optimality conditions are satisfied, this is the optimal solution.

For $\mathbf{n}_B = [2, 3, 5, 6]$	Nonbasic arcs with flows at the lower bound: $\mathbf{n}_0 = [1, 8]$

$\mathbf{x}_B = [x_2, x_3, x_5, x_6]$
$= [2, 0, 1, 1]$.

$\pi = [\pi_1, \pi_2, \pi_3, \pi_4, \pi_5]$
$= [9, 13, 25, 38, 0]$

$d_1 = c_1 + \pi_1 - \pi_2 = 11 + 9 - 13 = 7$: Satisfies 2b

$d_8 = c_8 + \pi_5 - \pi_1 = 11 + 0 - 9 = 2$: Satisfies 2b

Nonbasic arcs with flows at the upper bound: $\mathbf{n}_1 = [4, 7]$

$d_4 = c_4 + \pi_2 - \pi_4 = 18 + 13 - 38 = -7$: Satisfies 2c

$d_7 = c_7 + \pi_5 - \pi_2 = 10 + 0 - 13 = -3$: Satisfies 2c

Summary

This section has illustrated how the primal and dual basic solutions are computed directly from the graphical structure associated with the basis spanning tree. Because it is not necessary to store a representation of the basis inverse matrix as it is in general linear programming, a significant savings in both time and memory requirements is achieved. This is the source of the major computational advantage attending network flow programming. For any variation of the pure network flow problem, when the parameters are integral we also have the advantage of the integrality property. Thus all computations can be carried out in integer arithmetic, resulting in numerical stability and computational efficiency.

Primal Simplex Algorithm

This section describes the adaptation of the primal simplex algorithm for solving a pure network flow program; that is, the problem of finding the optimal flow distribution for the minimum cost flow network model given by (4a) - (4c). We begin with a statement of the algorithm and then provide an example to demonstrate how the computations are performed.

Algorithm

1. Start with a basis tree consisting of the arcs in the set $\mathbf{n}_B$ and the sets $\mathbf{n}_0$ and $\mathbf{n}_1$ corresponding to nonbasic arcs at their lower and upper bounds, respectively. We require that the initial basis be feasible for the primal problem. Compute the primal and dual solutions for the initial basis.
2. Compute the reduced costs, $d_k = c_k + \pi_i - \pi_j$,, for all nonbasic arcs $k(i, j)$.
3. If for each nonbasic arc k one of the conditions below holds, *stop* with the optimal solution.

   Optimality conditions: if $x_k = 0$, then $d_k \geq 0$

   if $x_k = u_k$, then $d_k \leq 0$

Otherwise, select some nonbasic arc that violates this condition and call it the entering arc.

4. Find the increase or decrease in the entering arc that will either drive it to its opposite bound or drive some basic arc to one of its bounds. If the entering arc is driven to its bound, go to Step 3. If a basic arc is driven to one of its bounds, let that be the leaving arc.

5. Change the basis by removing the leaving arc and adding the entering arc. Compute the primal and dual solutions associated with the new basis. Go to Step 2.

Initial Solution

As in general linear programming, we have the problem of finding an initial basic feasible solution for the primal simplex method. We use a two-phase approach where an artificial arc with unit cost is introduced for every node but the slack node. The artificial arcs and their initial flows are constructed in the following manner.

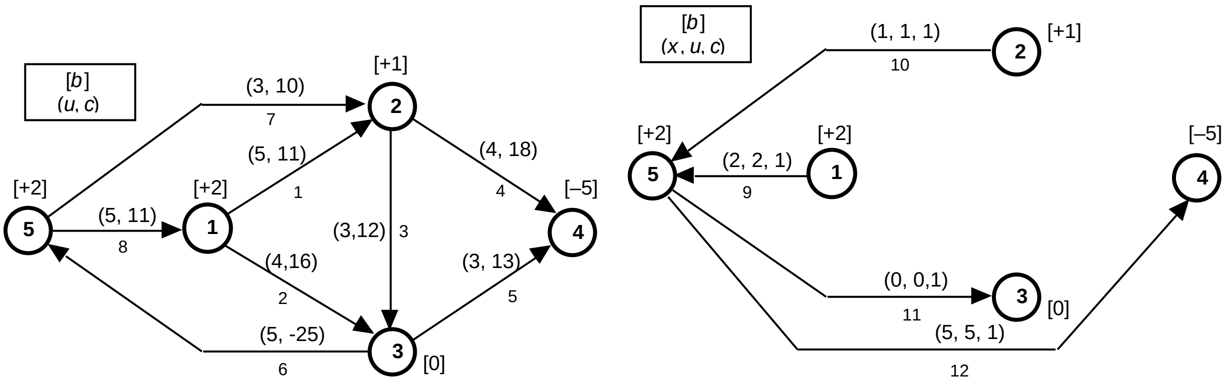
For each node $i = 1$ to $m - 1$.

a. If node i has $b_i > 0$, introduce an artificial arc from node i to the slack node and assign it the flow b_i. The upper bound on the artificial arc is also b_i.

b. If node i has $b_i \leq 0$, introduce an artificial arc from the slack node to node i and assign it the flow $-b_i$. The upper bound on the artificial arc is also $-b_i$.

The costs on the artificial arcs depend on the phase of the solution algorithm.

Fig. 35a shows the example network; Fig. 35b shows the spanning tree created by introducing artificial arcs. The tree is rooted at slack node 5. During phase 1 the two networks in the figure are combined. The resultant network consists of 12 arcs but only the unit arc costs on the artificial arcs are used at this stage in the computations. The arcs in the original network carry a unit cost of zero in phase 1.



a. Example network b. Artificial arcs

Figure 35. Construction of artificial arcs for phase 1

As in general linear programming, we will apply the simplex algorithm in two phases as follows.

Phase 1

1. Assign an arc cost of +1 to each of the artificial arcs and a cost of 0 to each of the original arcs.
2. Using the artificial arcs as the initial basis, solve the minimum cost flow problem with the primal simplex algorithm. If the total cost at optimality is greater than zero, an artificial arc has nonzero flow; stop, there is no feasible solution. If the total cost at optimality is zero, all the artificial arcs have zero flow and a feasible solution has been found. Proceed with phase 2. If the total cost at optimality is greater than zero, stop, the problem is infeasible.

Phase 2

3. Assign the original arc costs to the original arcs. Delete nonbasic artificial arcs and assign 0 cost and 0 capacity to each artificial arc remaining in the basis. Starting with the basic solution found in phase 1, solve the network problem with the primal simplex algorithm.

The two-phase procedure applies the primal simplex algorithm twice. In phase 1, a basic feasible solution is found if one exists. Starting with this solution, phase 2 works towards optimality. In most cases all the artificial variables are driven from the basis during phase 1; however, it is conceivable that some artificial arcs remain. Setting their capacity to zero assures that the flows in these arcs remain at zero during phase 2.

Finding the Entering Arc

The reduced cost, d_k, for a nonbasic arc $k(i, j)$ is the cost of increasing the flow on that arc by one unit, and can be expressed as:

$$d_k = c_k + \pi_i - \pi_j \tag{6}$$

The three terms on the right-hand side of Eq. (6) include: the unit cost of flow on arc k, c_k, the cost, π_i, of bringing a unit of flow to node i from the slack node through a path defined by the basis tree, and the decrease in cost, $-\pi_j$, achieved by reducing the flow through the basic path to node j. For a nonbasic arc with flow at its upper bound, d_k is the savings in cost (marginal benefit) associated with reducing the flow on arc k by one unit.

If the solution is optimal, one of the following conditions must hold for each nonbasic arc $k(i,j)$:

$$\text{if } x_k = 0, \text{ then } d_k \geq 0 \tag{7a}$$

$$\text{if } x_k = u_k, \text{ then } d_k \leq 0 \tag{7b}$$

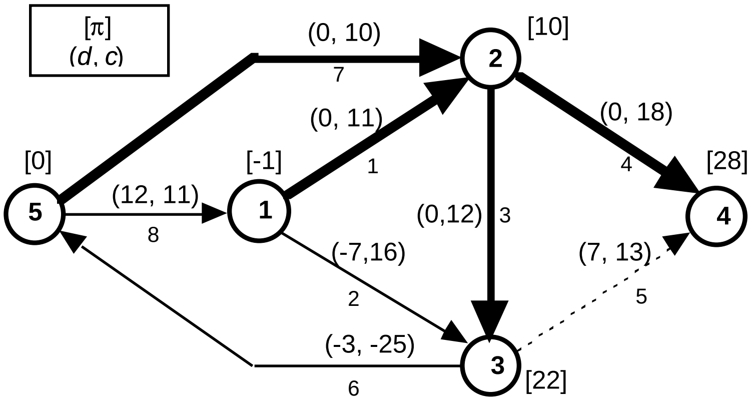
Condition (7a) indicates that if the nonbasic flow is at zero, the cost of increasing the flow on the arc must be nonnegative for the optimal solution; otherwise increasing the flow would improve the solution. Alternatively, if the flow on the nonbasic arc is at its upper bound as in (7b), the cost of decreasing the flow on the arc must be nonnegative (the cost of decreasing flow is $-d_k$) for an optimal solution; otherwise decreasing the flow would improve the solution. Values of $d_k = 0$ imply that changing the flow on the nonbasic arc will not change the objective function. In the case of an optimal solution, this indicates the existence of alternative optima.

When the simplex method is applied to the network flow problem, a nonbasic arc with flow at zero or its upper bound may enter the basis when it fails to satisfy the optimality condition. In our example, we select the entering arc with the largest reduced cost d_E. Thus we choose the entering arc k_E according to the rule

$$d_E = \max \begin{cases} \max\{-d_k : d_k < 0 \text{ for } k \quad \mathbf{n}_0\} \\ \max\{d_k : d_k > 0 \text{ for } k \quad \mathbf{n}_1\} \end{cases} \tag{8}$$

For large networks, this rule is not efficient because it requires a search over all nonbasic arcs at every iteration. Alternatives include choosing the first nonbasic nonoptimal arc encountered or selecting an arc from some list of candidates.

To illustrate this concept, consider the example network with the basis $\mathbf{n}_B = [1, 3, 4, 7]$ in Fig. 36. The arcs drawn with heavy lines depict the tree representation of this basis. We call this the *dual network* because, together with the original network structure, it shows the variables and parameters primarily associated with the dual of the min-cost flow problem, particularly the π variables for the nodes, the reduced arc costs, and the original arc costs. Arcs in $\mathbf{n}_0$ are shown with narrow solid lines, while arcs in $\mathbf{n}_1$ are shown with dashed lines. The structure and basis information on the figure is sufficient to compute the π vector and the reduced costs. These can then be used to test the solution for optimality and, if necessary, select an arc to enter the basis. From the figure we note that arcs 2, 5 and 6 fail the optimality test and are candidates to enter the basis. Using the rule stated in Eq. (8) either arc 2 or arc 5 would be selected to enter the basis. The choice is arbitrary and in the following we will see the effects of both selections.



Candidates to enter basis: [2, 5, 6]

Figure 36. Dual network with $\mathbf{n}_B$ = [1, 3, 4, 7].

Finding the Leaving Arc

After an arc has been selected to enter the basis, we move to an adjacent basic feasible solution by increasing or decreasing the flow in the entering arc. In this operation, the flows in the basic arcs in the cycle formed by the entering arc must also change in order to maintain conservation of flow at the nodes. The flow changes in the entering arc by an amount that just drives the flow on one of the basic arcs to zero or to its upper bound, or drives the flow on the entering arc to zero or to its upper bound.

Constructing the Cycle

Call the entering arc k_E and assume that it originates at node i_E and terminates at node j_E. We must now construct a signed list of arcs that describes the directed cycle formed by arc k_E and the basic arcs. There are two distinct cases.

a. Flow is increasing on the entering arc

   Find the directed path in the basis tree from node j_E to node i_E with arc signs indicated by the direction of arc traversal. Let $\boldsymbol{C}$ be the set of arcs formed by adding arc k_E to the path to create a cycle.

b. Flow is decreasing on the entering arc

Find the directed path in the basis tree from node i_E to node j_E with arc signs indicated by the direction of arc traversal. The addition of arc $-k_E$ to the path forms the set ***C***.

Figure 37 depicts another representation of the example problem called the *primal network*. The figure describes the network structure and information related to arc flows, particularly the external flow at the nodes, the current arc flows, and the upper bounds on flow. The oval on the figure shows the cycle formed when we select arc 2 as the entering arc. The information in the primal and dual networks could be combined in a single figure; however, the use of two separate diagrams simplifies the presentation.

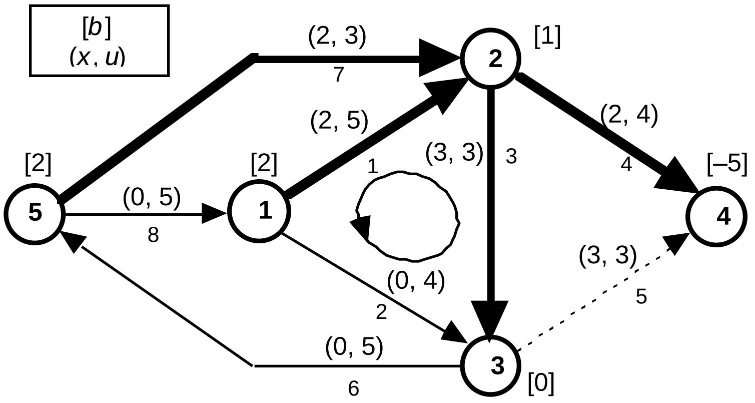


Figure 37. Primal network with $\mathbf{n}_B$ = [1, 3, 4, 7] with arc 2 selected to enter the basis

Because the current flow on arc 2 is 0, it must enter the basis with an increased flow. Thus k_E = 2(1,3) and the directed path from node 3 to node 1 is [-3, –1]. The directed cycle identified as ***C*** = [2, -3, –1]. The signs on the arc indices show the direction in which flow is to be changed on the arcs as arc 2 enters the basis.

Computing the Flow Change and Selecting the Leaving Arc

In general terms, let the flow change in the entering arc by an amount Δ. Arcs with positive signs in the set ***C*** experience a flow increase of Δ ,while arcs with negative signs in ***C*** experience a flow decrease of Δ. The largest flow change that results in a feasible solution is the smallest value that will drive the flow on one of the cycle arcs to zero or to its upper bound.

$$\Delta = \min \begin{cases} \min\{u_k - x_k : k > 0 \text{ for } k \quad \boldsymbol{C}\} \\ \min\{x_{-k} : k < 0 \text{ for } k \quad \boldsymbol{C}\} \end{cases} \qquad (9)$$

The leaving arc, k_L, is the arc that provides the minimum value for Eq. (9). In the event of ties, the selection is arbitrary. When k_L ***C***, the arc will leave the basis with its flow at its upper bound. When $-k_L$ ***C***, the arc will leave the basis with its flow at zero.

In the example, when arc 2 enters the basis we have ***C*** = [2, -3, –1] so the flow change is computed as follows:

$$\Delta = \min \begin{cases} \min\{u_2 - x_2\} \\ \min\{x_1, x_3\} \end{cases} = \min \begin{cases} \min\{4 - 0\} \\ \min\{2,3\} \end{cases} = 2$$

Because the minimum is obtained for arc 1, it must leave the basis. Arc 2 enters and arc 1 leaves with flow a flow change around the cycle of 2.

Changing the Basis

Changing the Basis Tree and Computing the Dual Values

The new basis tree is constructed by deleting k_L from the previous tree and adding k_E. The dual values can be computed directly from the tree representation using the condition that $\pi_j = c_k + \pi_i$ for all arcs in the basis. A more efficient computational procedure is recommended, however, based on the observation that prior to adding k_E, the deletion of k_L from the basis breaks the tree into two parts, one of which necessarily containing the slack node. This is illustrated in Fig. 38. Call $\boldsymbol{I}_L$ the set of nodes separated from the slack node by deleting arc k_L. In the figure, $\boldsymbol{I}_L$ consists of node 1.

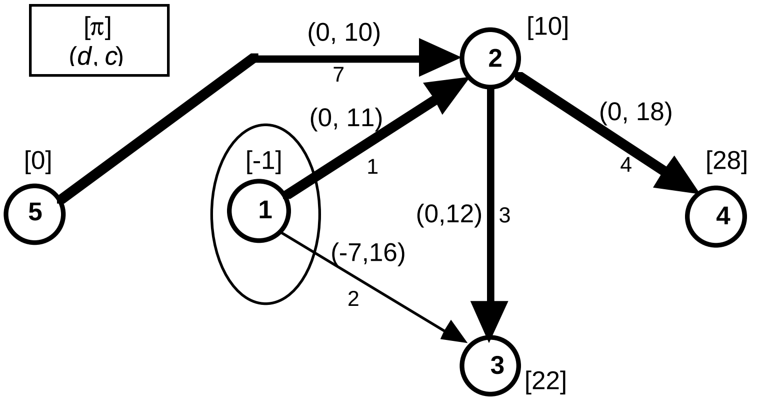


Figure 38. Subtree formed by deleting the leaving arc

The values of π will change for the nodes in $\boldsymbol{I}_L$ by the reduced cost of the entering arc. When the entering arc is $k_E(i_E, j_E)$, its reduced cost is

$$d_E = c_{k_E} + \pi_{i_E} - \pi_{j_E}$$

The revised dual costs, π' for the nodes *i* in $\boldsymbol{I}_L$ depend on whether the origin or the terminal node of k_E is a member of $\boldsymbol{I}_L$.

$$\pi_i' = \begin{cases} \pi_i + d_E : \text{for } j_E \quad \boldsymbol{I}_L \\ \pi_i - d_E : \text{for } i_E \quad \boldsymbol{I}_L \end{cases} \tag{10}$$

For the example when $k_E = 2$ and $k_L = 1$, the basis tree with dual values assigned is shown in Fig. 39. Because the origin node of arc 2 is in the set $\boldsymbol{I}_L$, we subtract –7 to the dual variables in $\boldsymbol{I}_L$. This changes π_1 to 6 in the new basis tree.

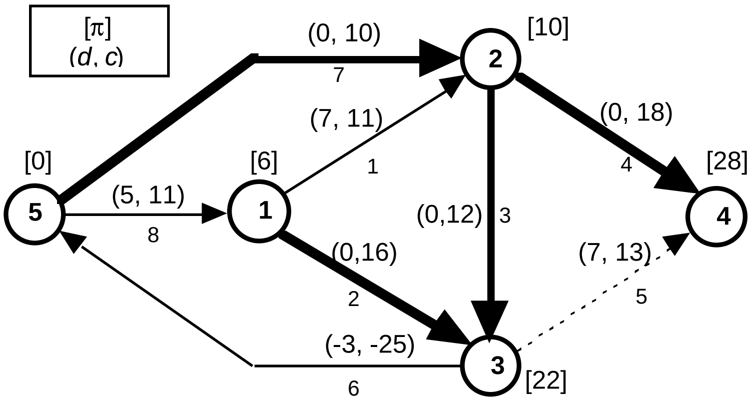


Figure 39. Dual network with $\mathbf{n}_B$ = [2, 3, 4, 7]

Flow Change

Flow for the new basic solution is easily determined by adjusting the flows on the cycle arcs — increasing the flow on the arcs that have positive indices in ***C*** and decreasing the flow on the arcs with negative indices in ***C***. The new flows on these arcs are computed with Eq. (11).

$$x_k' = \begin{cases} x_k + \Delta : k \quad \boldsymbol{C} \text{ and } k > 0 \\ x_{-k} - \Delta : k \quad \boldsymbol{C} \text{ and } k < 0 \end{cases} \tag{11}$$

The values of x_k' become the basic flows in the next iteration. Fig. 40 illustrates the new flows after the basis change.

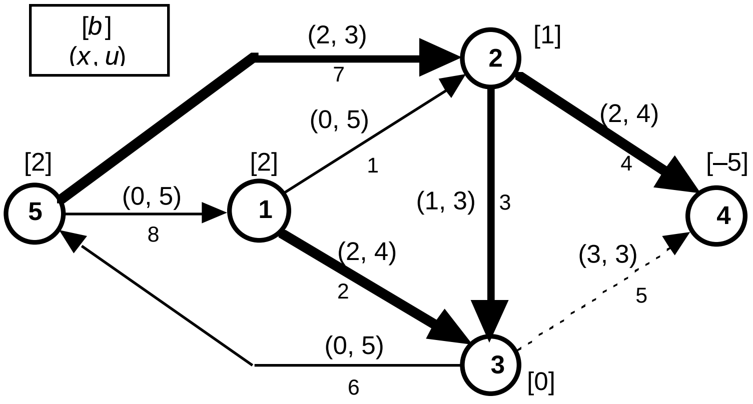


Figure 40. Primal network with $\mathbf{n}_B$ = [2, 3, 4, 7]

Continuing the Algorithm

After changing the dual values, arc flows and basis tree, the algorithm continues by returning to Step 2 and computing the reduced costs for the nonbasic arcs. From Fig. 39, we see that arcs 5 and 6 are candidates to enter the basis, with arc 5 as most violating the optimality condition. Since arc 5 has flow at the upper bound, it must enter the basis with flow decreasing. The cycle formed be arc 5 and the basis arcs is ***C*** = [–5, –3, 4]. The maximum flow change on the cycle is 1 with arc 3 leaving the basis. Deleting arc 3 from the basis forms the subtree consisting of nodes $\boldsymbol{I}_L$ = {1, 3}. According to Eq. 10, the dual values for these nodes will decrease by 7. The resulting dual and primal networks appear in Figs. 41 and 42.

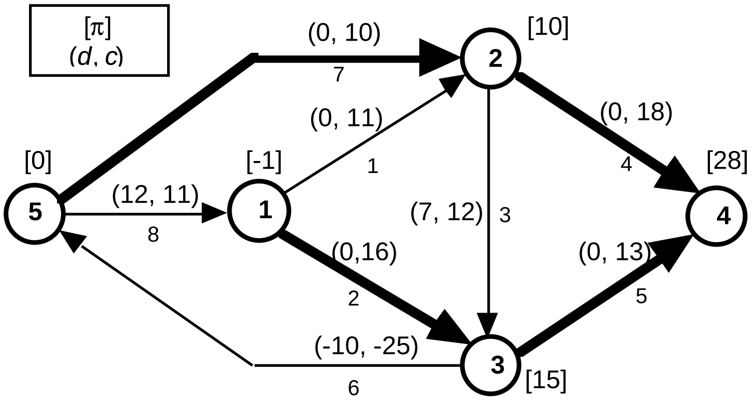


Figure 41. The dual network with $\mathbf{n}_B$ = [2, 4, 5, 7]

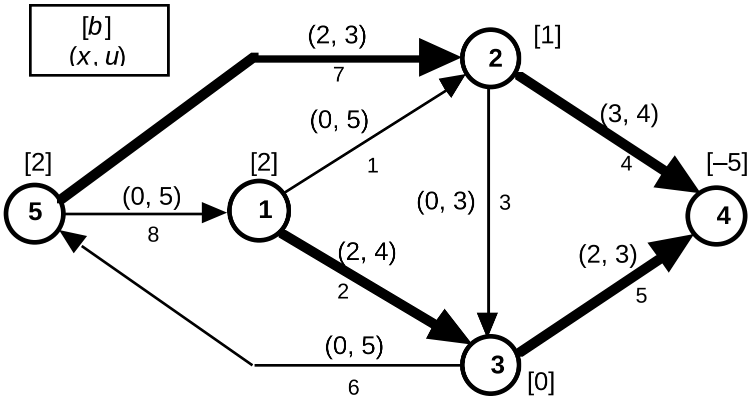


Figure 42. The primal network with $\mathbf{n}_B$ = [2, 4, 5, 7]

The solution is still not optimal because arc 6 violates the optimality condition. The process will eventually terminate with the optimal solution. At each iteration, the basis changes with the deletion of one arc and the addition of another. Except when degeneracy causes a 0 flow change, the objective function decreases by the product of the reduced cost and the flow change. When an iteration begins with a primal feasible solution, the rules for changing the flow assure that the solution remains primal feasible.

Complete Example

We now provide a complete solution to the example problem introduced in Fig. 23 starting with phase 1 and a basis consisting of artificial arcs. Fig. 43 repeats the example for easy reference. Each iteration of the primal simplex algorithm is illustrated in Fig. 44. We have combined the primal and dual networks for a more compact representation.

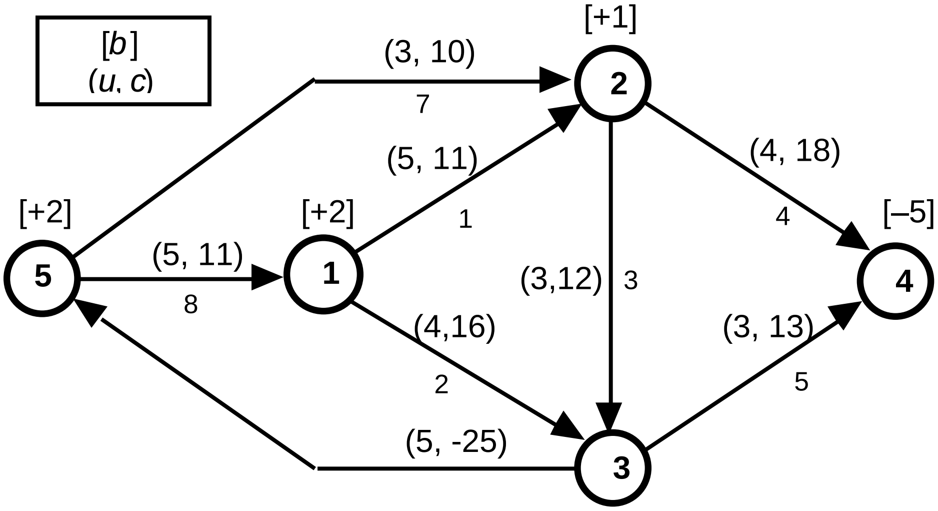
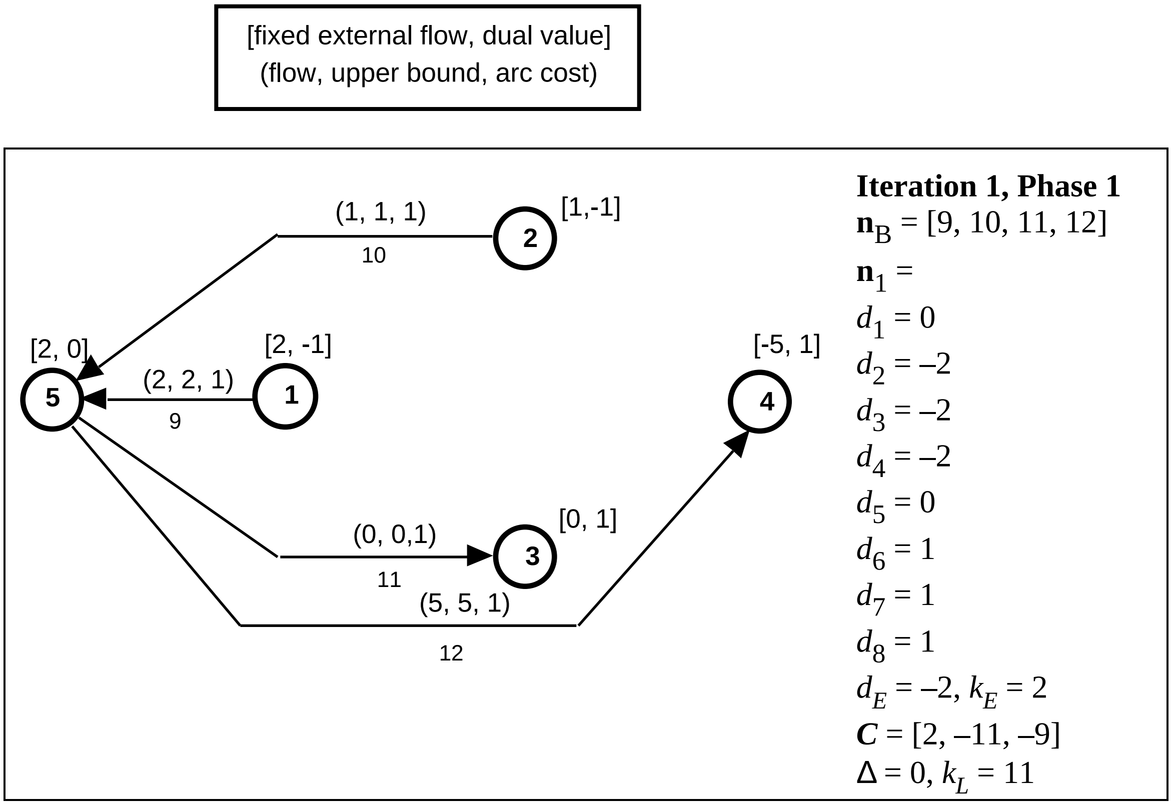


Figure 43. Example network.



Iteration 1, Phase 1

$\mathbf{n}_B$ = [9, 10, 11, 12]

$\mathbf{n}_1$ =

$d_1 = 0$

$d_2 = -2$

$d_3 = -2$

$d_4 = -2$

$d_5 = 0$

$d_6 = 1$

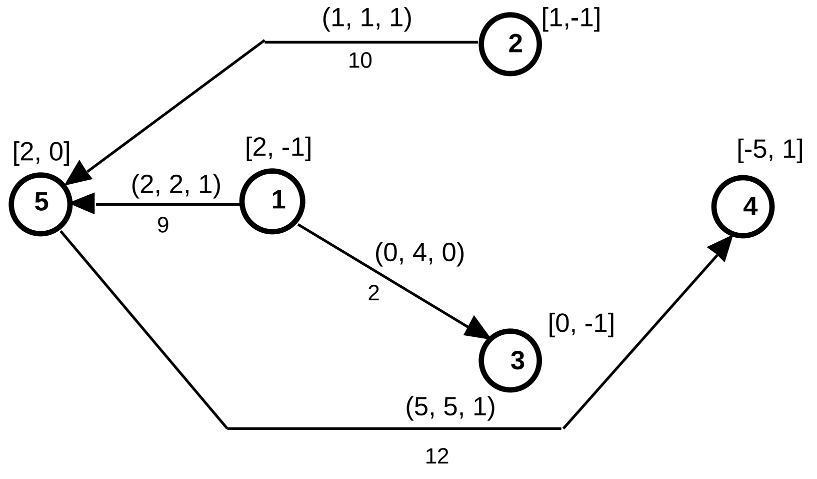
$d_7 = 1$

$d_8 = 1$

$d_E = -2$, $k_E = 2$

$\boldsymbol{C} = [2, -11, -9]$

$\Delta = 0$, $k_L = 11$



Iteration 2, Phase 1

$\mathbf{n}_B$ = [2, 9, 11, 12]

$\mathbf{n}_1$ =

$d_1 = 0$

$d_3 = 0$

$d_4 = –2$

$d_5 = –2$

$d_6 = –1$

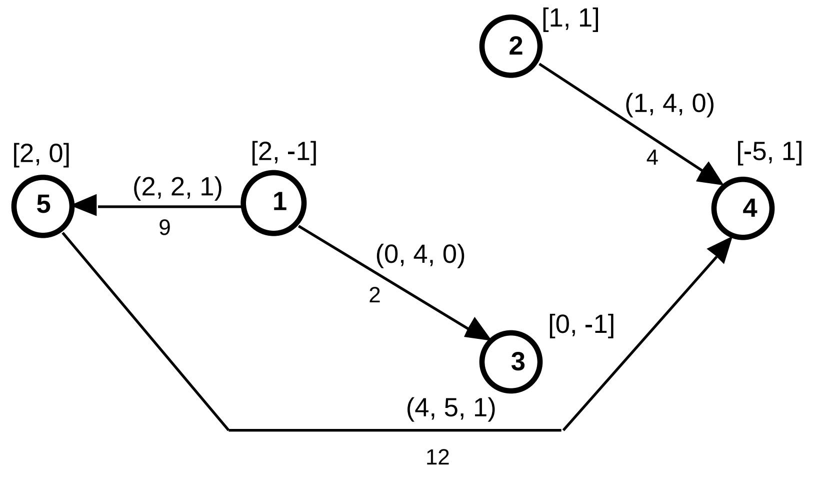
$d_7 = 1$

$d_8 = 1$

$d_E = 4$, $k_E = 2$

$\boldsymbol{C}$ = [4,–10, –12]

$\Delta = 1$, $k_L = 10$



Iteration 3, Phase 1

$\mathbf{n}_B$ = [2, 4, 9, 12]

$\mathbf{n}_1$ =

$d_1 = –2$

$d_3 = 2$

$d_5 = –2$

$d_6 = –1$

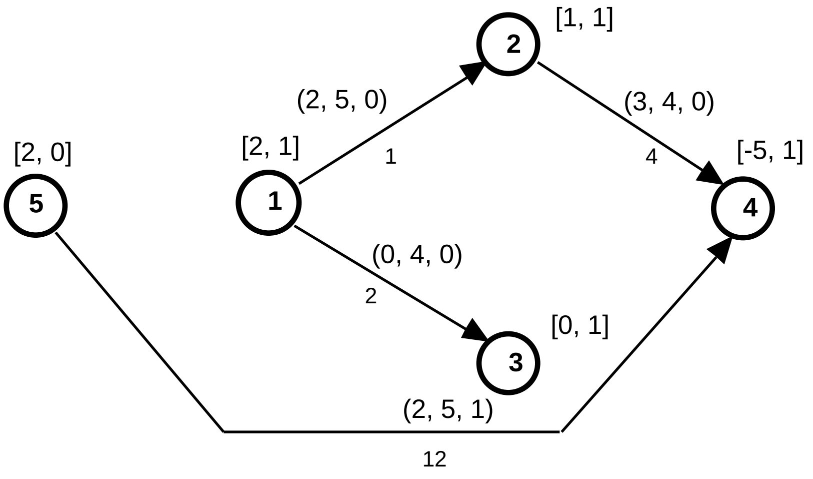
$d_7 = –1$

$d_8 = 1$

$d_E = –2$, $k_E = 1$

$\boldsymbol{C}$ = [1, 4, –12, –9]

$\Delta = 2$, $k_L = 9$



Iteration 4, Phase 1

$\mathbf{n}_B$ = [1, 2, 4, 12]

$\mathbf{n}_1$ =

d_3 = 0

d_5 = 0

d_6 = 1

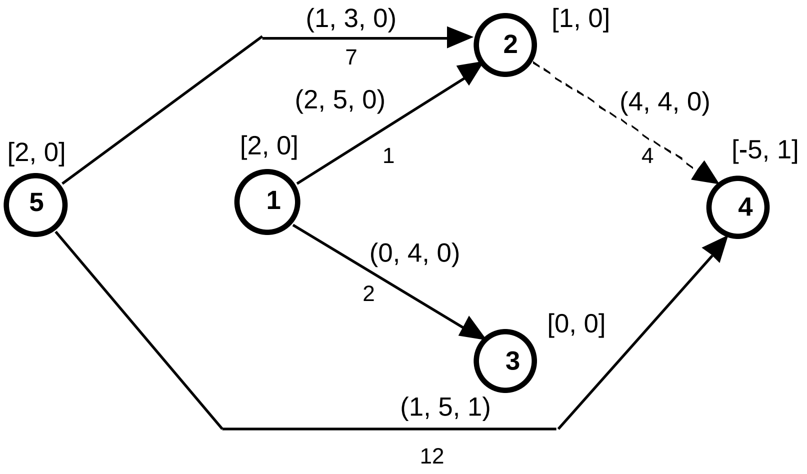
d_7 = –1

d_8 = –1

d_E = –1, k_E = 7

C = [7, 4, –12]

Δ = 1, k_L = 4



Iteration 5, Phase 1

$\mathbf{n}_B$ = [1, 2, 7, 12]

$\mathbf{n}_1$ = [4]

d_3 = 0

d_4 = –1

d_5 = –1

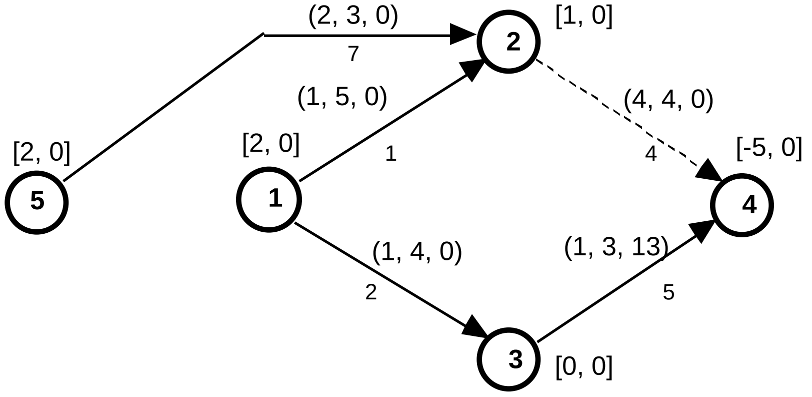
d6 = 0

d8 = 0

dE = 5, kE = 7

C = [5, –12, 7, –1, 2]

_ = 1, kL = 12



Iteration 6, Phase 1

$\mathbf{n}_B$ = [1, 2, 5, 7]

$\mathbf{n}_1$ = [4]

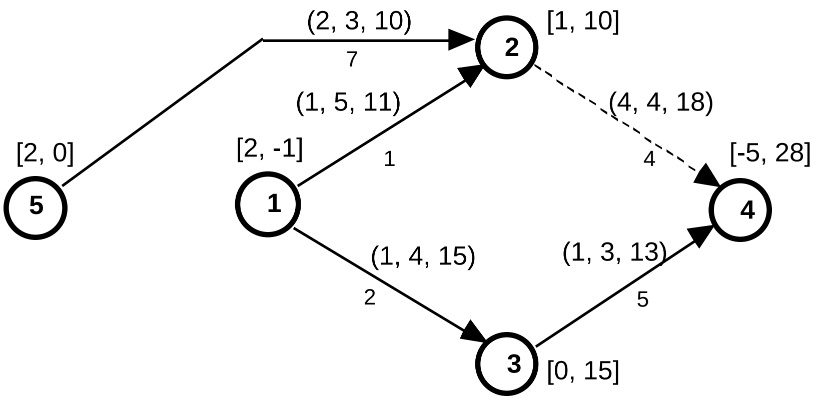
d_3 = 0

d_4 = 0

d_6 = 0

d_8 = 0

Optimal for Phase 1
Switch to Phase 2



Iteration 7, Phase 2

$\mathbf{n}_B = [1, 2, 5, 7]$

$\mathbf{n}_1 = [4]$

$d_3 = 7$

$d_4 = 0$

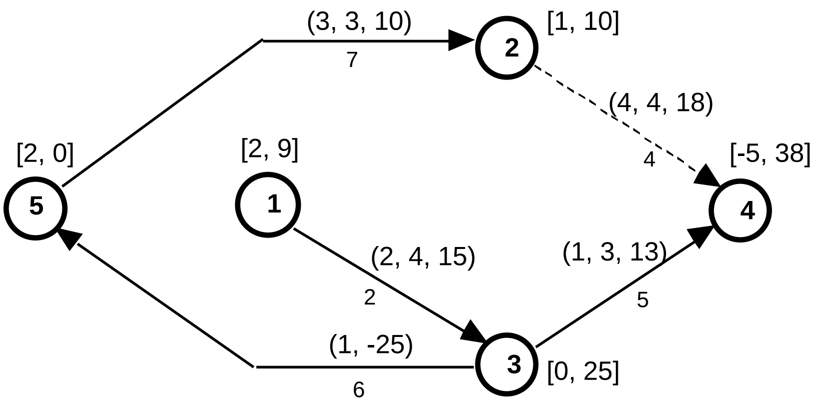
$d_6 = -10$

$d_8 = 12$

$d_E = -10$, $k_E = 6$

$\boldsymbol{C} = [6, 7, -1, 2]$

$\Delta = 1$, $k_L = 1$



Iteration 8, Phase 2

$\mathbf{n}_B = [2, 5, 6, 7]$

$\mathbf{n}_1 = [4]$

$d_1 = 10$

$d_3 = -3$

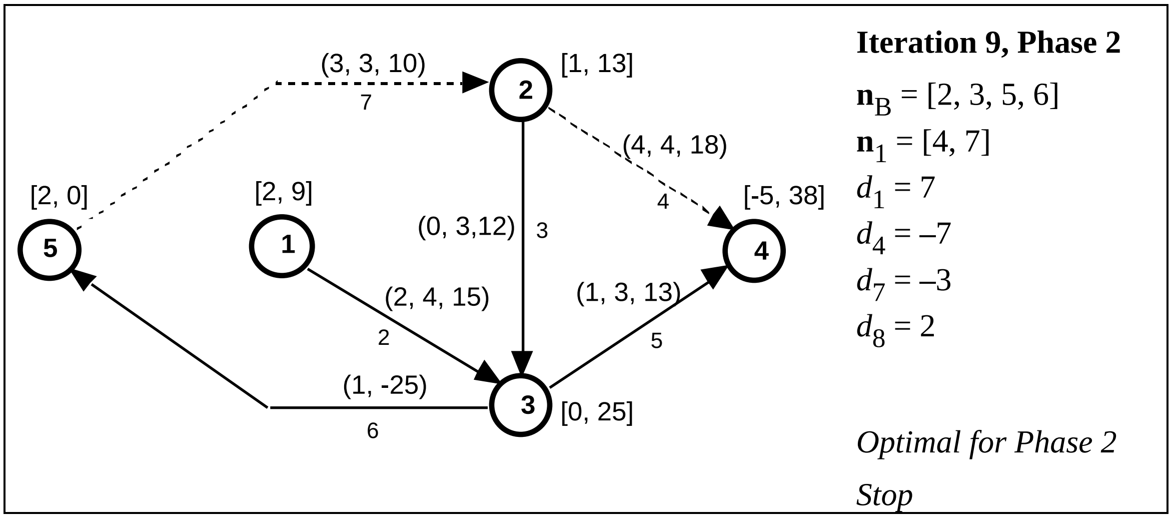
$d_4 = -10$

$d_8 = 2$

$d_E = -3$, $k_E = 3$

$\boldsymbol{C} = [3, 6, 7]$

$\Delta = 0$, $k_L = 7$



Iteration 9, Phase 2

$\mathbf{n}_B = [2, 3, 5, 6]$

$\mathbf{n}_1 = [4, 7]$

$d_1 = 7$

$d_4 = -7$

$d_7 = -3$

$d_8 = 2$

Optimal for Phase 2

Stop

Figure 44. Simplex iterations for example network